ONTOLOGY BASED INFORMATION RETRIEVAL SYSTEM FOR DIGITAL LIBRARY

THET THET AUNG

M.C.Sc. MAY 2021

ONTOLOGY BASED INFORMATION RETRIEVAL SYSTEM FOR DIGITAL LIBRARY

BY

THET THET AUNG

B.C.Sc.(Hons:)

A Dissertation Submitted in Partial Fulfillment of the Requirement for the Degree of

Master of Computer Science

(M.C.Sc.)

University of Computer Studies, Yangon
May 2021

ACKNOWLEDGEMENTS

To complete this thesis, many things are needed like my hard work and the supporting of many people who gave a lot of idea to me to complete this thesis.

Firstly, I would like to express my respectful thanks to **Prof. Dr. Mie Mie Thet Thwin**, Rector and **Dr. Yadanar Thein**, Pro- Rector of the University of Computer Studies, Yangon, for their kind permission to submit this thesis.

My sincere thanks and regards to **Dr. Thi Thi Soe Nyunt**, Professor, Faculty of Computer Science Department of the University of Computer Studies, Yangon, and **Dr. Soe Lae Phoo** and **Dr. Ei Phyoe Wai**, University of Computer Studies, Yangon, **Dr. Nilar Thein, Dr. Yaza Na,** Pro- Rectors and **Dr. May Phyoe Oo,** Pro- Rector of the University of Computer Studies, Hinthada, for their administrative supports and encouragement throughout the completion of this thesis.

I would like to express my sincere gratitude to **Daw Khin Lay Myint and Dr. Aye Aye Seint** for their helpful suggestions in the preparation of this thesis.

I would like to express my respectful thanks to my supervisor **Dr. Hlaing Htake Khaung Tin**, Professor and Head of Faculty of Information Science Department of the University of Computer Studies, Hinthada, for her valuable guidance, encouragement and helpful suggestions throughout the progress of the research and in the preparation of this thesis.

My gratitude also goes to **Daw Nyunt Nyunt Htwe,** Lecturer and Head of Department Language Development (English) of University of Computer studies, Hinthada, for her editing this thesis from the language point of view.

I especially give thanks to my parent, all of my colleagues, and friends, for their encouragement and help in writing this thesis.

Finally, I would like to express special thanks to all of the teachers, staffs from the University of Computer Studies, Hinthada for their support.

ABSTRACT

Using semantic web technology through Information Retrieval (IR) process is becoming an efficient way to enhance the accuracy of the search process and retrieve more relevant results in the web-based systems, especially in the Digital Library. In the Digital Library fields, Ontology can be used to organize bibliographic descriptions, represent and expose the contents of the document, and share knowledge between users.

Therefore, the IR model for digital libraries based on the adaptation of the Vector Space Model (VSM) combined with the Semantic Web technologies: Web Ontology Language (OWL) and SPARQL protocol is proposed in this research. The main concept of the proposed IR model is that metadata of resources are stored in Resource Description Framework (RDF) format and retrieved not only by the keywords contained in the user query but also by the contexts defined in Domain Ontology. In the proposed IR model, preprocessing, context matching, and calculating similarity values steps are included. The algorithm for the formatting of SPARQL query is developed in the context matching step of IR model.

Based on the proposed IR model, Ontology-based IR system for Digital Library is implemented in Service-Oriented Architecture (SOA) by using the XML Web Service technology and ASP.NET. The architecture of the proposed system consists of file storage for documents, one ontology dataset, and two programming components: Digital Library Web Service and Web Application. In this proposed system, Web Ontology Language (OWL) is used to design Ontology for Digital Library using Protégé v3.5 tool. Functions for publication and retrieving of documents are implemented as a web service by using the C# programming language. The user interface is designed and implemented as a web application in ASP.NET platform for consuming the functions of web service.

To show the performance of the proposed IR system, 415 training documents including various file types (.doc, .pdf, .txt) were tested and 33 queries for different properties of document were presented. To evaluate the performance of proposed IR system, the precision, recall, and F-values are measured and compared. According to the comparison results, the Ontology-based IR system is more accurate in searching for ObjectProperty type. As a result, the proposed system serves user-friendly, high-performance and scalable semantic search for information from the digital library.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Nowadays, the amount of available information in both printed media and electronic/digital mediums had increased dramatically. Moreover, the number of digital documents had rapidly increased and required easy and accessed mechanized methods. In the information retrieval systems, the information is usually searched by means of a full-text search; every term in the texts of the documents can function as a search key.

Digital libraries (DLs) had become the digital counterpart of the traditional library system. There are various ways to improve the search technology for accessing documents from DL. In this thesis, Ontology-based IR system is proposed for Digital Library. Ontologies have the potential to play an important role in DL, because ontology defines a common vocabulary for researchers who need to share information in a domain.

The proposed system intends to provide for students to retrieve the relevant information with their concept and to be able to search, read and download the textbooks, old questions (included tutorial, exam, multiple, assignments), journals, thesis papers, reference papers, novels efficiently in the short time.

1.1 Information Retrieval

Information retrieval is the study of helping users to find information that matches their information needs. Technically, IR studies the acquisition, organization, storage, retrieval, and distribution of information. Historically, IR is about document retrieval, emphasizing document as the basic unit [2]. General architecture of the IR system is shown in Figure 1.1.

In the general architecture of the IR system, the user with information needs issues a query (user query) to the retrieval system through the query operations module. The retrieval module uses the document index to retrieve those documents that contain some query terms (such documents are likely to be relevant to the query); compute relevance scores for them, and then rank the retrieved documents according to the scores. The ranked documents are then presented to the user. The document collection is also called the text database, which is indexed by the indexer for efficient retrieval [2].

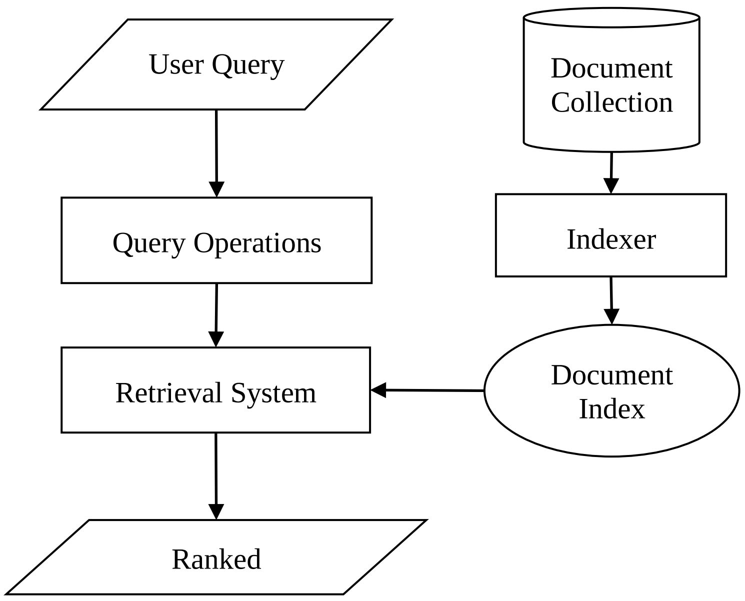


Figure 1.1 General Architecture of IR System

Information Retrieval (IR) systems provide populations of users with access to a large collection of stored information. These systems are concerned with the structure, analysis, organization, storage, and searching for such information. Dr. Glöckner [11] described a good IR system is able to accept a user query, understand from the user query what the user requires, search a database for relevant documents, retrieve the documents to the user, and rank the documents according to their relevance.

Information retrieval systems are everywhere: Web search engines, library catalogs, store catalogs, cookbook indexes, and so on. Information retrieval (IR), also called information storage and retrieval (ISR or ISAR) or information organization and retrieval, is the art and science of retrieving from a collection of items a subset that serves the user’s purpose [26]

1.2 Motivation of the Thesis

The Digital Libraries (DL) is a collection of documents organized in an electronic form [33]. There are some limitations facing his electronic process. Some of these limitations are:

- sorting materials by topic or by the material type.
- finding and using particular information's on web-based systems is a major challenge for most users.

- Retrieving huge amounts of data in a short time.
- Retrieving more relevant and specific results.

Due to these limitations, an extreme need for semantic digital library appeared to enhance the digital libraries performance and gives accurate results to fulfill users and needs, where Semantic web technologies, such as Ontology, can provide more functions for the digital library to improve the result of searching and retrieving processes.

1.3 Objectives of the Thesis

In the information retrieval systems, the information is usually searched by means of a full-text search; every term in the texts of the documents can function as a search key. There are various ways to improve the search technology for accessing documents from digital library. The objectives of the thesis are as follows:

(i) To implement a Digital Library which contains digital documents with the various formats, such as text, e-Book, MS Word document, or pdf.

(ii) To develop the domain ontology for Digital Library.

(iii) To study the semantic or context-based Information Retrieval (IR).

(iv) To develop an Ontology-based IR system for Digital Library.

(v) To improve the accuracy of IR results by combining semantic web technologies and vector space models.

(vi) To reduce the consuming time of searching for information.

1.4 Contributions of the Thesis

The system develops domain ontology for digital libraries and Ontology-based IR models. The proposed IR system is very useful in digital library domain area. The contributions of the thesis are as follows:

(i) Domain Ontology for the digital library is developed.

(ii) IR model for digital libraries based on the adaptation of the vector space model combined with the semantic web technologies (OWL and SPARQL) is proposed.

1.5 Organization of the Thesis

This thesis is organized into five chapters, abstract, acknowledgments and references. The Ontology-based Information Retrieval system is introduced for the digital libraries in the chapter one. This chapter also describes the motivation, contribution, aim, and objectives of the research work.

The features of the Digital Library, technologies of Semantic Web, and models of Information Retrieval (IR) are presented in the chapter two. Various types and models of IR, SPARQL query language, and Web Ontology Language (OWL) are briefly explained in this chapter.

The model of Information Retrieval based on Ontology is explained details in the chapter three. Design of the Domain Ontology for Digital Library is described, and then the detail explanation about context matching process, formatting of SPARQL query, and calculation of similarity by vector space model is finally presented.

Design and implementation of Ontology-based Information Retrieval System for Digital Library are presented in the chapter four. And then, the overview of system design, the architecture of the system, and the structure of Digital Library Ontology are described in this chapter. And then, the implementation of programming modules for the proposed system is explained with Graphical User Interfaces. Finally, the experimental results are shown by charts and tables.

The conclusion of the research work is drawn in the chapter five. In this chapter, further extensions that propose some improvements which could be made are presented. The limitations of the system are also described in this chapter.

CHAPTER 2
BACKGROUND THEORY

This chapter presents the features of Digital Library, technologies of Semantic Web, and models of Information Retrieval (IR). Firstly, this chapter describes the features and metadata of a Digital Library. Secondly, this chapter describes the overview of Ontology that is important to build a domain Ontology for Digital Library. And then, it explains Web Ontology Language (OWL) which is the most popular specification for defining domain Ontology and SPARQL query language which is the standard query language for Ontology. Finally, it presents various types and models of IR.

2.1 Digital Library

Digital libraries are a set of electronic resources and associated technical capabilities for creating, searching, and using information. They combine the structure and gathering of information, which libraries and archives have always done, with the digital representation that computers have made possible. The main purpose of a digital library is to collect, manage, and preserve in perpetuity digital content [6].

The Digital Libraries Federation in 1998 defines digital libraries as: "Digital libraries are organizations that provide the resources, including the specialized staff, to select, structure, offer intellectual access to, interpret, distribute, preserve the integrity of, and ensure the persistence over time of collections of digital works so that they are readily and economically available for use by a defined community or set of communities" [10].

2.1.1 Common Features of Digital Library

Common features of the digital library are as follows:

- providing round the clock services to users, within and without the library environment. Users can access the digital objects at any time and anywhere i.e. 24 hours and 7 days a week with only a computer and internet connection;
- providing a coherent view of all information contained within a library, no matter its form or format (e.g., text, audio, image and video);

- accessing a digital object by several users at the same time in different locations;
- requiring no large spaces, unlike traditional libraries where physical space is required for the construction and maintenance of the collections [14].

These Common features show the flexibility, portability, and accessibility of Digital libraries. Nevertheless, the principles underlying the functionality of digital libraries were simple, the premise that digital libraries dealing with traditional problems of searching for information delivery to users and to preserve it for posterity. Digital information takes up less space than information on paper and therefore, can help traditional libraries reduce costs is no longer enough anymore, so many more models are defined to meet specific needs that will be stumbling over time and with changing new technologies.

2.1.2 Metadata Creation

The word “metadata” means “data about data”. Metadata articulates a context for the object of interest, “resources” such as MP3 files, library books, or images, in the form of “resource description” [18]. It is machine-understandable information about web resources or other things [3]. Metadata serves many important purposes like data description, data browsing, data transfer, and metadata has an important role in digital resource management [12].

Traditional physical libraries employ metadata in the library catalogs. In digital libraries, metadata is obtained by cataloging resources such as books, periodicals, web pages, digital images, and DVDs, etc. The data is stored in the integrated system, using the MARC metadata standard. The purpose is to direct users to the location of the items and a detailed description of the items. Recently, standards for metadata in the digital libraries include Dublin Core, DDI [18]. Different metadata elements are needed to perform different tasks, for example, author, title and subject support the function of discovery. A DL may require many more forms of metadata than analog for management and use. According to the National Information Standard Organization’s (NISO) publication “Understanding Metadata”, there are three types of metadata [4].

- **Administrative Metadata**: Administrative metadata provides information to manage to resource e.g. when and how the resource has created.

- **Descriptive Metadata**: Descriptive metadata provides the source purpose e.g. title, abstract, author, etc.
- **Structural Metadata**: Information necessary to record the internal structure of an item so that it can be rendered to the user in a sensible form (for instance, a book must be delivered in its page order.) This type of metadata is necessary as an item may often be comprised of multiple (often thousands) of files. For example, the images of individual pages that make up a digitized book.

2.2 Ontology in Digital Library

The term ontology has been used for many years, to mean different things like glossaries and data dictionaries, thesauri and taxonomies, controlled vocabulary, schema and data models, and formal ontologies and inference. And also in many areas, such as philosophy, artificial intelligence, knowledge-based systems, it has been used to organize information. There are found in the literature several definitions of ontologies, several types proposed for application in different areas of knowledge, and proposals for building ontologies (methodologies, tools, and languages).

The philosophical field of ontology was not as successful as computer scientists, where they built some large and robust ontology, such as WordNet and Cyc [27]. Ontologies have aroused the interest of many researchers in Computer Science, being able to highlight main areas: Database, Software Engineering, Semantic Web, Information Architecture, Knowledge Engineering, Knowledge Representation, Qualitative Modeling, Language Engineering, Information Retrieval, and Extraction, Knowledge Management and Organization, and Artificial Intelligence as a form of knowledge representation about the world or some part this, describing: individuals, classes, attributes, relationships and events [20].

In the Digital Libraries fields, ontologies can be used to: organize bibliographic descriptions, represent and expose the contents of the document, and share knowledge between users. It’s important to note that the use of ontologies in digital libraries allows us to transfer the profile, the user's browsing behavior to other digital libraries and databases, so that when a user of a particular DL leaves service to connect to another DL, the user profile (including preferences and navigation behaviour) can be transferred from one base to another by using the appropriate semantic web services because all databases share a common domain of discourse that can be played by rules inference and application logic. For this we have a vast list of ontology languages that allow us to design ontologies according to our needs, however, when it comes to design ontology for digital libraries pertinent examples exist such as RDF (Resource Description Framework), in the family of W3C which is used for describing resources; XML (Extensible Markup Language), for describing data, information, and knowledge; OWL(Web Ontology Language), is becoming the standard for describing ontologies and accessing resources through the web [13].

2.3 Web Ontology Language (OWL)

The ontology describes the concepts in the domain and also the relationships that hold between those concepts. Different ontology languages provide different facilities. The most recent development in standard ontology languages is OWL from the World Wide Web Consortium (W3C).

Web Ontology Language (OWL) is a language for defining and instance ontologies in the Web. This includes descriptions of classes and their properties and their relationships. OWL was designed for use by applications that need to process the content of information, instead of just presenting it to humans. It further facilitates the possibility for interpretation by machines of Web content by providing additional vocabulary with formal semantics. OWL is a W3C recommendation [9].

OWL is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called ontology. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web [32]. OWL ontology consists of three components: Individuals, Properties, and Classes.

2.3.1 Classes

OWL classes are interpreted assets that contain individuals. They are described using formal (mathematical) descriptions that state precisely the requirements for membership of the class [17]. For example, the class Document would contain all the individuals that are documented in our domain of interest (Digital Library). Classes may be organized into a superclass-subclass hierarchy, which is also known as taxonomy. Subclasses specialize ('are subsumed by') their super-classes. For example, consider the classes Resource and Document - Document might be a subclass of Resource (so Resource is the superclass of Document). This says that "All documents are resources", "All members of the class Document are members of the class Resource", "Being a Document implies that it is a Resource", and "Document is subsumed by Resource". Figure 2.1 shows a representation of some classes containing individuals - classes are represented as circles or ovals, rather like sets in Venn diagrams.

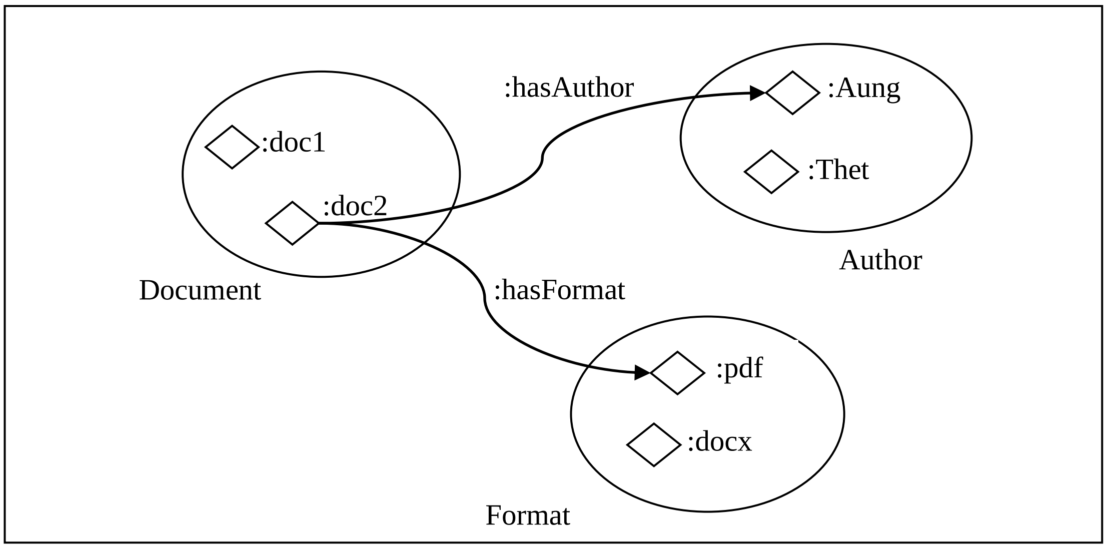


Figure 2.1 Representations of Classes (Containing Individuals)

2.3.2 Properties

Properties are binary relations on individuals - i.e. properties link two individuals together. There are two main types of properties, Object properties, and Datatype properties [17].

Object properties are relationships between two individuals [17]. For example, the property *hasAuthor* might link the individual "*Document_1*" to the individual "*Author_1*", Datatype properties link an individual to an XML Schema Datatype value or an RDF literal. In other words, they describe relationships between individual and data values. For example, A datatype property *numberOfPages* linking the individual “*Document_1*” to the data literal ‘*125*’, which has a type of an xsd: integer. Figure 2.2 shows a representation of some properties linking some individuals together.

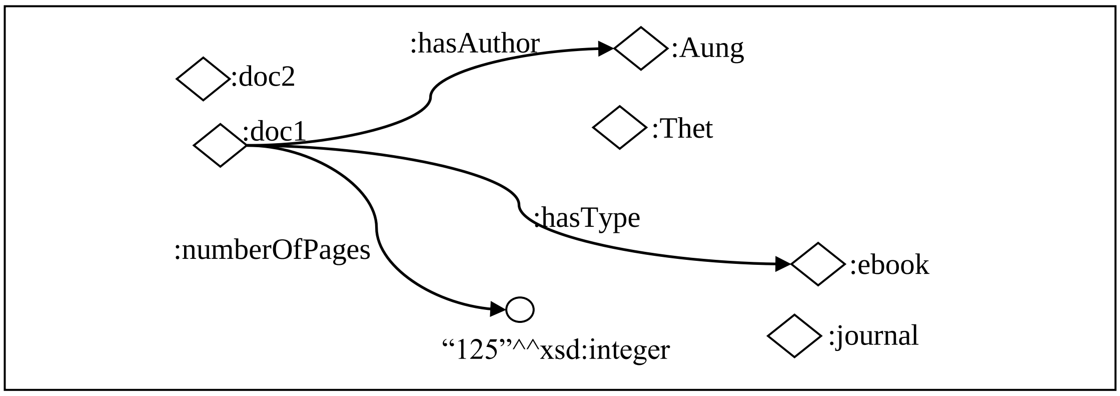
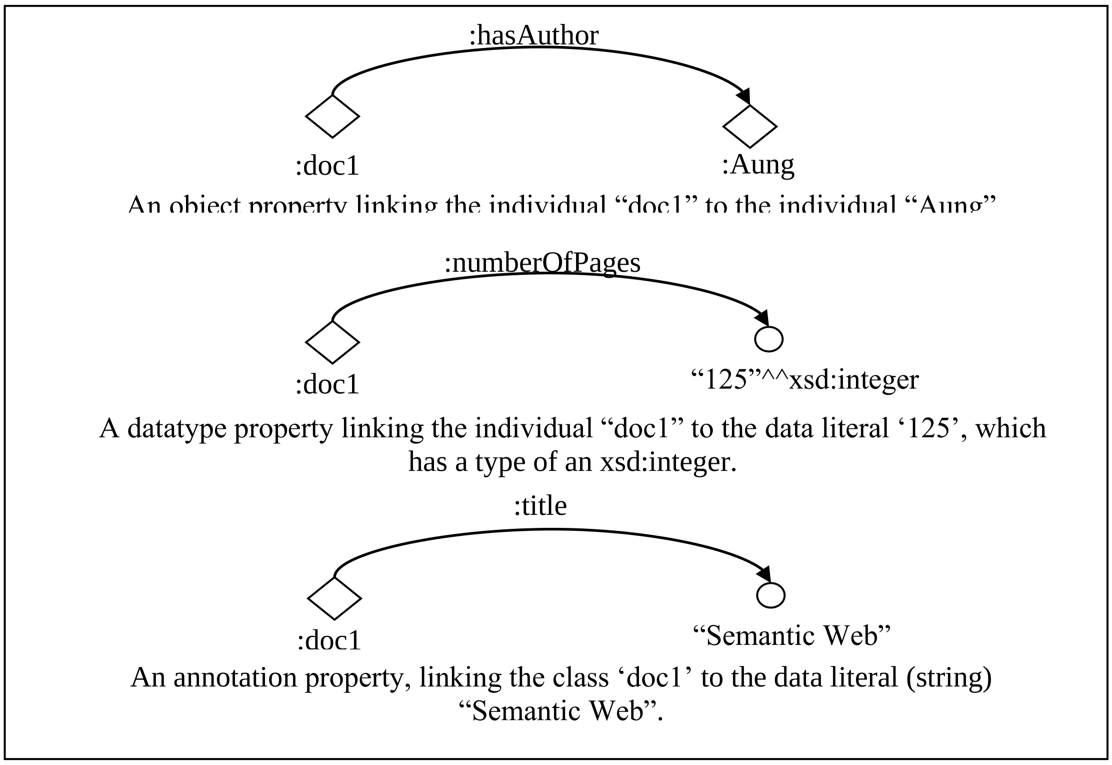


Figure 2.2: Representation of Properties

OWL also has a third type of property - Annotation properties. Annotation properties can be used to add information (metadata - data about data) to classes, individuals, and object/datatype properties. Figure 2.3 depicts an example of each type of property.



An object property linking the individual “doc1” to the individual “Aung”

A datatype property linking the individual “doc1” to the data literal ‘125’, which has a type of an xsd:integer.

An annotation property, linking the class ‘doc1’ to the data literal (string) “Semantic Web”.

Figure 2.3: The Different types of OWL Properties

Properties can have inverses. For example, the inverse of *hasAuthor* is *isAuthoredBy*. Properties can be limited to having a single value - i.e. to being functional. They can also be either transitive or symmetric [17].

2.3.3 Individuals

Individuals represent objects in the domain in which we are interested. OWL does not use the Unique Name Assumption (UNA) for individuals. This means that two different names could actually refer to the same individual. For example, "Queen Elizabeth", "The Queen" and "Elizabeth Windsor" might all refer to the same individual. In OWL, it must be explicitly stated that individuals are the same as each other, or different from each other otherwise they might be the same as each other, or they might be different from each other. Figure 2.4 shows a representation of some individuals in some domain represented as diamonds in diagrams.

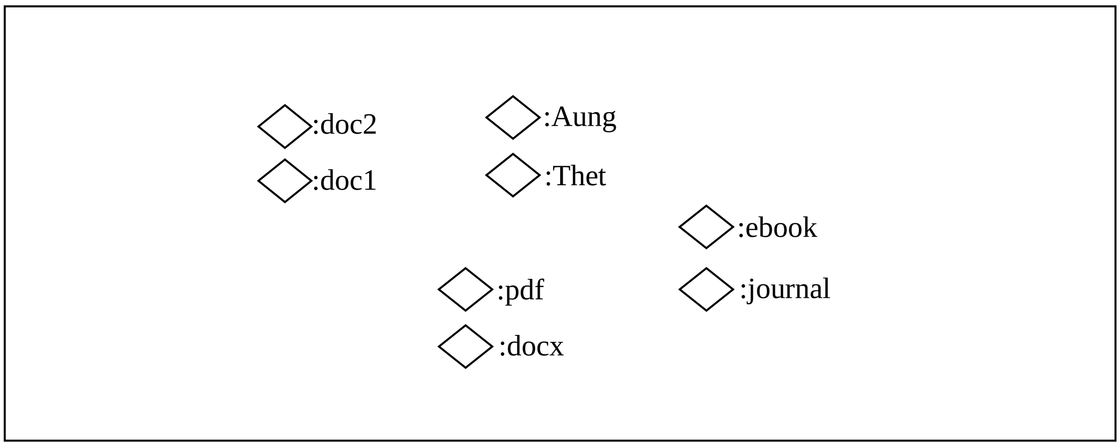


Figure 2.4: Representation of Individuals

2.4 SPARQL Query Language

SPARQL is a query language and a protocol for accessing RDF designed by the W3C RDF Data Access Working Group. As a query language, SPARQL is "data-oriented" in that it only queries the information held in the models; there is no inference in the query language itself [1].

RDF is a directed, labeled graph data format for representing information in the Web [22]. This specification defines the syntax and semantics of the SPARQL query language for RDF. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware [29]. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports aggregation, subqueries, negation, and creating values by expressions, extensible value testing, and constraining queries by source RDF graph. The results of SPARQL queries can be result sets or RDF graphs [31].

2.4.1 Forms of SPARQL Queries

SPARQL has four query forms. These query forms use the solutions from pattern matching to form result sets or RDF graphs [31]. The query forms are:

- SELECT: Returns all, or a subset of, the variables bound in a query pattern match.
- CONSTRUCT: Returns an RDF graph constructed by substituting variables in a set of triple templates.
- ASK: Returns a boolean indicating whether a query pattern matches or not.
- DESCRIBE: Returns an RDF graph that describes the resources found.

The SELECT form of results returns variables and their bindings directly. It combines the operations of projecting the required variables with introducing new variable bindings into a query solution.

Sample Data

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 42 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
```

SELECT Query

```
PREFIX dc: <http://purl.org/dc/elements/1.1/> .
PREFIX : <http://example.org/book/> .
PREFIX ns: <http://example.org/ns#> .
SELECT ?book ?title
WHERE
{
      ?book dc:title ?title .
      ?book ns:price ?price . FILTER ( ?price < 40 )
}
```

Result

Book	**title**
:book2	“The Semantic Web”

The CONSTRUCT query form returns a single RDF graph specified by a graph template. The result is an RDF graph formed by taking each query solution in the solution sequence, substituting for the variables in the graph template, and combining the triples into a single RDF graph by the set union.

If any such instantiation produces a triple containing an unbound variable or an illegal RDF construct, such as a literal in subject or predicate position, then that triple is not included in the output RDF graph. The graph template can contain triples with no variables (known as ground or explicit triples), and these also appear in the output RDF graph returned by the CONSTRUCT query form [31].

Sample Data

```
comp:A rov:haslegalName "Niké" .
comp:A org:hasRegisteredSite site:1234 .
comp:B rov:haslegalName "BARCO" .
site:1234 locn:fullAddress "Dahliastraat 24, 2160 Wommelgem" .
```

CONSTRUCT Query

```
PREFIX comp: < http://example/org/org/>
PREFIX org: < http://www.w3.org/TR/vocab-regorg/ >
PREFIC rdfs: <http://www.w3.org/TR/rdf-schema/>
CONSTRUCT {?comp rdfs:label ?name}
WHERE
{ ?comp org:haslegalName ?name. }
```

Resulting Graph

```
@prefix comp: <http://example/org/> .
@prefix rdfs: <http://www.w3.org/TR/rdf-schema/>
comp:a rdfs:label "Niké" .
comp:b rdfs:label "BARCO" .
```

Applications can use the ASK form to test whether or not a query pattern has a solution. No information is returned about the possible query solutions, just whether or not a solution exists [31]. The following example describes the ASK query "Are there any organizations having "1234" as their registered site?" for the above sample data.

ASK Query

```
PREFIX org: < http://www.w3.org/TR/vocab-regorg/
ASK
WHERE
{?organisation org:hasRegisteredSite site:1234}
```

Result

```
TRUE
```

The DESCRIBE form returns a single result RDF graph containing RDF data about resources. This data is not prescribed by a SPARQL query, where the query client would need to know the structure of the RDF in the data source, but, instead, is determined by the SPARQL query processor. The query pattern is used to create a result set. The DESCRIBE form takes each of the resources identified in a solution, together with any resources directly named by IRI, and assembles a single RDF graph by taking a "description" which can come from any information available including the target RDF Dataset. The description is determined by the query service. The syntax DESCRIBE * is an abbreviation that describes all of the variables in a query [31]. The following example DESCRIBE query return all triples associated with a particular resource (organization) for the above sample data.

DESCRIBE query

```
PREFIX comp: <http://example/org/>
DESCRIBE comp:A
```

Result

```
@prefix comp: <http://example/org/> .
@prefix org: <http://www.w3.org/TR/vocab-regorg/> .
comp:A rov:haslegalName “Niké” .
comp:A org:hasRegisteredSite site:1234 .
```

2.4.2 Filtering in SPARQL Queries

Filtering of query solutions is done within a FILTER expression. SPARQL FILTERs restrict the solutions of a graph pattern match according to a given constraint [31]. This section describes how the values in a solution can be restricted. There are many comparisons available - we just cover two cases here: string matching and testing values.

SPARQL provides an operation to test strings, based on regular expressions. This includes the ability to ask SQL “LIKE” style tests, although the syntax of the regular expression is different from SQL. The regular expression language is the same as the XQuery regular expression language which is a codified version of that found in Perl [30].

The syntax is:

```
    FILTER regex(?x, "pattern" [, "flags"])
```

The flags argument is optional. The flag “i” means a case-insensitive pattern match is done. The following sample data and example query find books by title with a case-insensitive pattern in them.

Sample Data

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price “42"^^xsd:integer .
:book2 dc:title "The Semantic Web" .
:book2 ns:price “23"^^xsd:integer .
```

SPARQL Query with String Filter

```
PREFIX dc: <http://purl.org/dc/elements/1.1/> .
PREFIX : <http://example.org/book/> .
PREFIX ns: <http://example.org/ns#> .
SELECT ?book ?title
WHERE{
      ?book dc:title ?title .
      FILTER REGEX( ?title, “semantic”, “i”)
}
```

Result

book	**title**
:book2	“The Semantic Web”

In the above data simple data file, we have added an extra field for numberOfpages. A query with a value filter to find the title of books that have more than 10 pages is shown below.

Sample Data

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dl: <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book1 dl:numberOfPages “15”^^xsd:integer .
:book2 dc:title "The Semantic Web" .
:book2 dl:numberOfPages “10”^^xsd:integer.
```

SPARQL Query with Value Filter

```
PREFIX dc: <http://purl.org/dc/elements/1.1/> .
PREFIX : <http://example.org/book/> .
PREFIX ns: <http://example.org/ns#> .
SELECT ?book ?pages
WHERE{
      ?book dc:numberOfPages ?pages .
      FILTER ( ?pages > “10”^^xsd:integer)
}
```

Result

book	title
:book1	“SPARQL Tutorial”

2.5 Models of Information Retrieval

The typical IR model of the search process consists of three essentials: query, documents, and search results. The goal of an IR system is to retrieve documents containing information that might be useful or relevant to the specific purpose it’s being used. Information retrieval systems can also be distinguished by the scale at which they operate. Tasks of information retrieval are as follows [25]:

- Routing and filtering: To direct documents to interested parties
- Multimedia retrieval: To retrieve e.g. images or speech data
- Cross-language Retrieval: To find documents in one language that is relevant to an information need expressed in another language
- Summarization: To capture the essence of a text in fewer words
- Translation: To express in one language the meaning of a document written in another language
- Question-answering: To find text that answers a particular question
- Topic detection: To identify stories that discuss the same topic
- Classification: To assign documents to known classes
- Clustering: To assign documents to previously unknown groupings
- Novelty detection: To determine when a new topic is introduced

There are two good reasons for having models of information retrieval. The first is that models guide to research and provide the means for academic discussion. The second reason is that models can serve as a blueprint to implement an actual retrieval system.

Mathematical models are used in many scientific areas with the objective to understand and reason about some behavior or phenomenon in the real world. A model of information retrieval predicts and explains what a user will find relevant given the user query. The correctness of the model’s predictions can be tested in a controlled experiment. In order to do predictions and reach a better understanding of information retrieval, models should be firmly grounded in intuitions, metaphors, and some branch of mathematics [24].

Intuitions are important because they help to get a model accepted as reasonable by the research community. Metaphors are important because they help to explain the implications of a model to a bigger audience. For instance, by comparing the earth's atmosphere with a greenhouse, non-experts will understand the implications of certain models of the atmosphere. Mathematics is essential to formalize a model, to ensure consistency, and to make sure that it can be implemented in a real system. As such, a model of information retrieval serves as a blueprint that is used to implement an actual information retrieval system [7].

An IR model governs how a document and a query are represented and how the relevance of a document to a user query is defined. There are four main IR models: Boolean model, vector space model, language model, and probabilistic model.

Although these models represent documents and queries differently, they used the same framework. They all treat each document or query as a "bag" of words or terms. Term sequence and position in a sentence or a document are ignored. That is, a document is described by a set of distinctive terms. A term is simply a word whose semantics helps remember the document's main themes. The term here may not be a natural language word in a dictionary. Each term is associated with a weight. Given a collection of documents D, let $V = \{t_1, t_2,..., t|V|\}$ be the set of distinctive terms in the collection, where it is a term. The set V is usually called the vocabulary of the collection, and |V| is its size, i.e., the number of terms in V. A weight $w_{ij} > 0$ is associated with each term t_i of a document $d_j \in D$. For a term that does not appear in document d_j, $w_{ij} = 0$. Each document d_j is thus represented with a term vector, $d_j = (w_{1j}, w_{2j},...,w|V|_j)$, where each weight w_{ij} corresponds to the term $t_i \in V$, and quantifies the level of importance of t_i in document d_j. The sequence of the components (or terms) in the vector is not significant. With this vector representation, a collection of documents is simply represented as a relational table (or a matrix). Each term is an attribute, and each weight is an attribute value. In different retrieval models, w_{ij} is computed differently [2].

2.5.1 Boolean Model

In Boolean retrieval, a document is represented as a set of terms $d_j = t_1,\ldots,t_k$, where each t_i is a term that appears in document d_j. A query is represented as a

Boolean expression of terms using the standard Boolean operators: and, or and not. A document matches the query if the set of terms associated with the document stratifies the Boolean expression representing the query. The result of the query is the set of matching documents [15].

2.5.2 Language Model

Statistical language models (or simply language models) are based on probability and have foundations in statistical theory [5]. The basic idea of this approach to retrieval is simple. It first estimates a language model for each document and then ranks documents by the likelihood of the query given the language model. Similar ideas have previously been used in natural language processing and speech recognition.

2.5.3 Probabilistic model

This family of IR models is based on the general principle that documents in a collection should be ranked by decreasing the probability of their relevance to a query. This is often called the Probabilistic Ranking Principle (PRP). Since true probabilities are not available to an IR system, probabilistic IR models estimate the probability of relevance of documents for a query. This estimation is the key part of the model, and this is where most probabilistic models differ from one another. The probabilistic model is based on probability theory. It can be estimated the relevance of a given document for a user based upon their query.

2.3.4 Vector Space Model

In the vector space model text is represented by a vector of terms. The definition of a term is not inherent in the model, but terms are typically words and phrases. If words are chosen as terms, then every word in the vocabulary becomes an independent dimension in a very high dimensional vector space. Any text can then be represented by a vector in this high-dimensional space. If a term belongs to a text, it gets a non-zero value in the text-vector along with the dimension corresponding to the term. A vector-based information retrieval method represents both documents and queries with high-dimensional vectors while computing their similarities by the vector inner product [8].

2.6 Summary

This chapter presents the detail of theory background about Digital Library, Ontology and Information Retrieval. Common features of DL and about metadata creation are explained. Metadata is machine-understandable information about the resources of DL. The role of Ontology in DL fields is described. In the Digital Libraries fields, ontologies can be used to: organize bibliographic descriptions, represent and expose the contents of the document. And then Web Ontology Language (OWL) is presented in detail for describing the Ontology of DL. SPARQL query language for manipulating of Ontology dataset is described with examples. The general architecture of the IR and its different models are also presented in this chapter.

CHAPTER 3
ONTOLOGY BASED INFORMATION RETRIEVAL

In this section, designing the Domain Ontology for Digital Library is described. Creating classes and properties of Digital Library is explained using Protégé Tool, which is a free, open-source platform to construct domain models and knowledge-based applications with ontologies. Ontology-based Information Retrieval Model is presented in detail in this chapter. Using Vector Space Model for ranking the IR results is described.

3.1 Building Ontology for Digital Library

In the Digital Library fields, ontologies can be used to organize bibliographic descriptions, represent and expose the contents of the document, and share knowledge between users. To design ontology for DL pertinent examples exist such as:

- RDF (Resource Description Framework), in the family of W3C which is used for describing resources;
- XML(Extensible Markup Language), for describing data, information, and knowledge;
- OWL(Web Ontology Language), is becoming the standard for describing ontologies and accessing resources through the web;
- SKOS (Simple Knowledge Organization System), recommended by the W3C, enables easy publication and use of such vocabularies as linked data; etc.

In the proposed system, Web Ontology Language (OWL) is used to design ontology for Digital Library. Many ontology editors have been developed to help domain experts to develop and manage ontology, for example, Protégé, OntoEdit, or TopBraid. Protégé [21] is a free, open-source platform to construct domain models and knowledge-based applications with ontologies. Protégé OWL editor: enables users to build an ontology for the Semantic Web, in particular to OWL:

- Classes (5 subclasses)
- Properties (16 properties)
- Instances

3.1.1 Defining Classes

The most basic concepts in a domain should correspond to classes that are the roots of various taxonomic trees. Every individual in the OWL world is a member of the class owl: Thing. Thus each user-defined class is implicitly a subclass of owl: Thing. Domain-specific root classes are defined by simply declaring a named class [9].

To create the ontology of the Digital Library, we need to start the Protégé tool. When Protégé starts the OWL Classes tab shown in Figure 3.1 will be visible. The initial class hierarchy tree view should resemble the picture shown in Figure 3.2. The empty ontology contains one class called owl: Thing. As mentioned previously, OWL classes are interpreted as sets of individuals (or sets of objects). The class owl: Thing is the class that represents the set containing all individuals. Because of this, all classes are subclasses of owl: Thing.

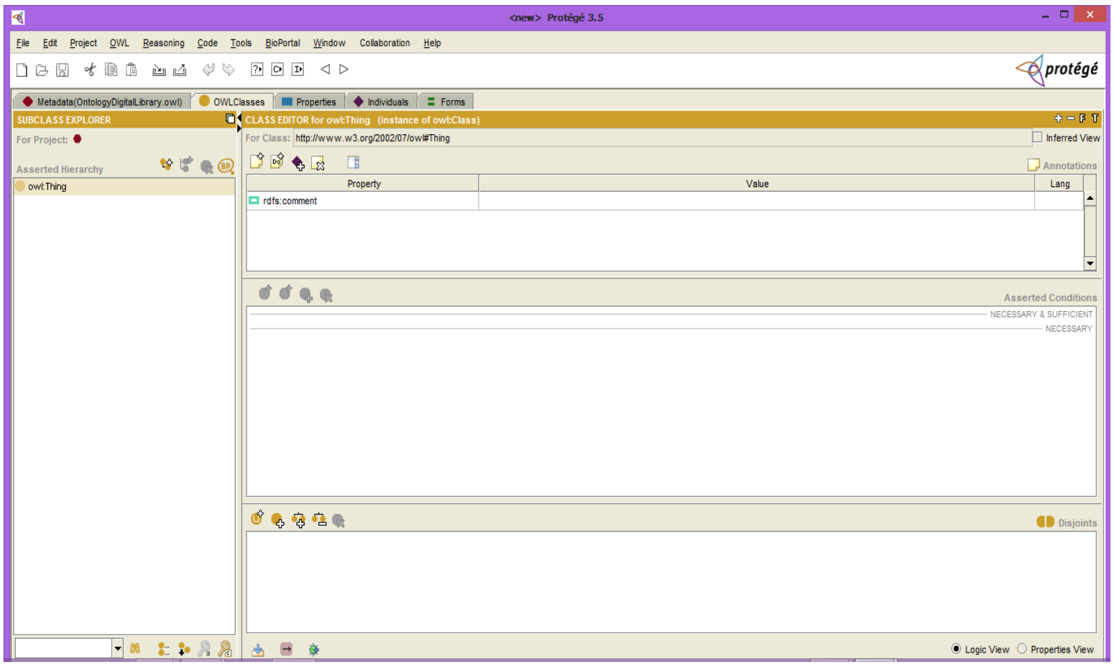


Figure 3.1 the Classes Tab

For our sample Digital Library domain, we define five root classes:

```
<owl:Class rdf:ID="Author"/>
<owl:Class rdf:ID="Category"/>
<owl:Class rdf:ID="Document"/>
<owl:Class rdf:ID="DocumentType"/>
<owl:Class rdf:ID="FileType"/>
```

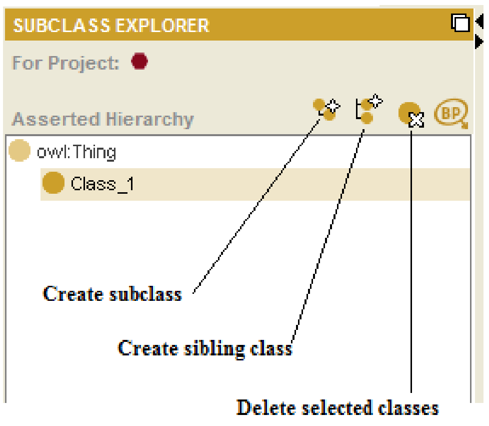
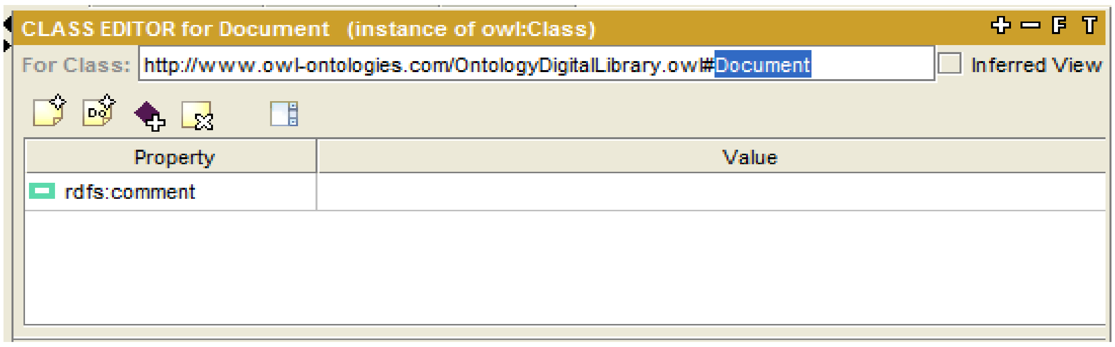


Figure 3.2 the Class Hierarchy Pane

To create these classes in Protégé, firstly we need to press the "Create subclass" button shown in Figure 3.2. This button creates a new class as a subclass of the selected class (in this case we want to create a subclass of owl: Thing). And then, the class is renamed to "Document" by using the "Class Editor Pane" which is located to the right of the class hierarchy shown in Figure 3.3. The remaining subclasses Author, Category, DocumentType, FileType are created by the above steps. All classes for Digital Library ontology created in Protégé are shown in Figure 3.4.


CLASS EDITOR for Document (instance of owl:Class)

For Class: http://www.owl-ontologies.com/OntologyDigitalLibrary.owl#Document

Inferred View

Property	Value
rdfs:comment	

Figure 3.3 the Class Editor Pane

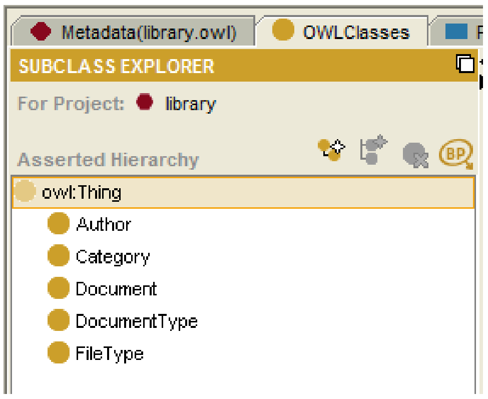


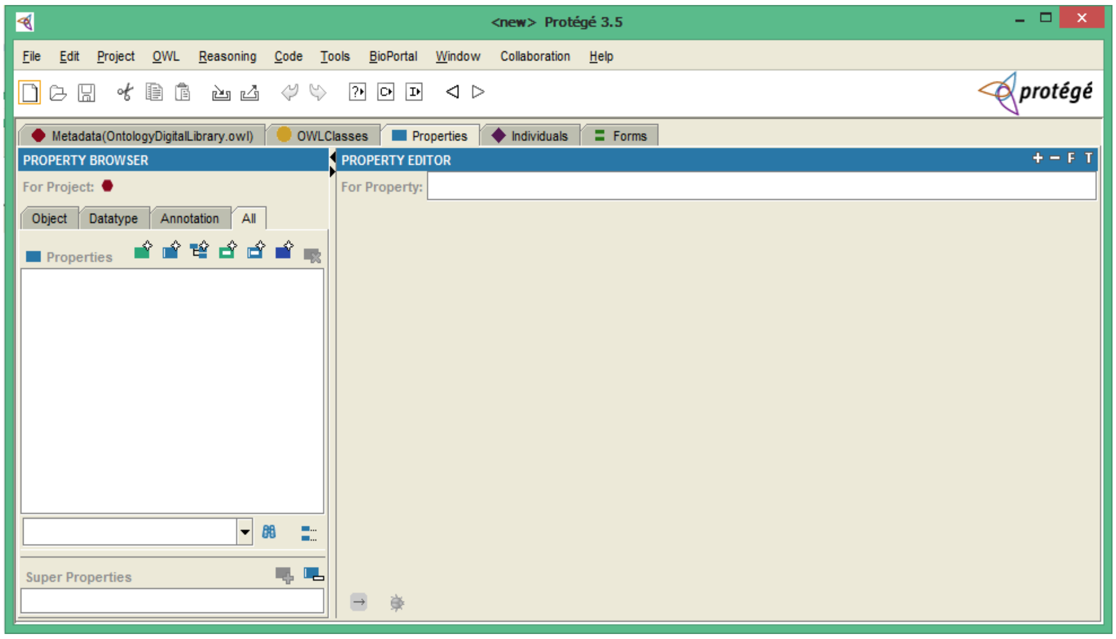
Figure 3.4 the Classes of Digital Library Ontology

3.1.2 Defining Properties

A property is a binary relation. Properties let us assert general facts about the members of classes and specific facts about individuals. Two types of properties are distinguished [9]:

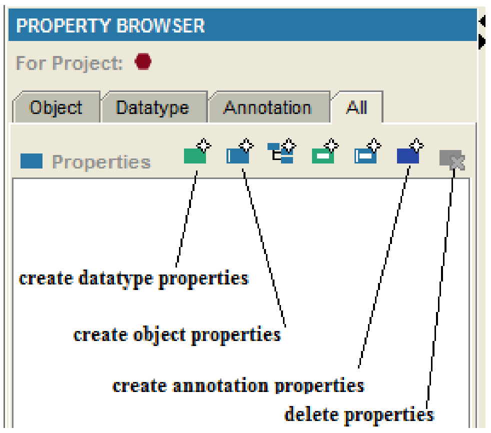
- datatype properties, relations between instances of classes and RDF literals and XML Schema datatypes
- object properties, relations between instances of two classes.

In the Protégé tool, properties may be created using the "Properties" tab shown in Figure 3.5. Figure 3.6 shows the buttons located in the top left-hand corner of the "Properties" tab that is used for creating OWL properties.



<new> Protégé 3.5
File Edit Project OWL Reasoning Code Tools BioPortal Window Collaboration Help
protégé
Metadata(OntologyDigitalLibrary.owl) OWLClasses Properties Individuals Forms
PROPERTY BROWSER
For Project:
Object Datatype Annotation All
Properties
Super Properties
PROPERTY EDITOR
For Property:

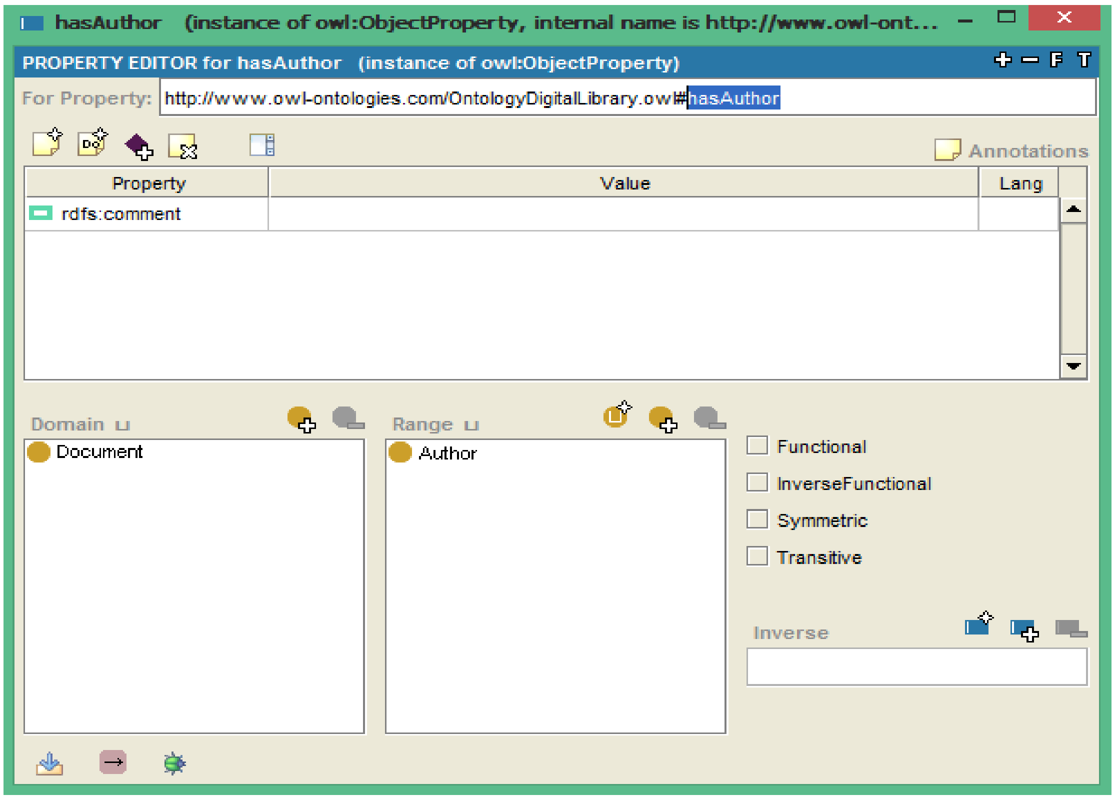
Figure 3.5 the Properties Tab



PROPERTY BROWSER
For Project:
Object Datatype Annotation All
Properties
create datatype properties
create object properties
create annotation properties
delete properties

Figure 3.6 the Properties Creation Buttons

As can be seen from Figure 3.6, there are buttons for creating Datatype properties, Object properties, and Annotation properties. Most properties created in our Digital Library will be Datatype properties. It is also possible to create properties using the "Properties Editor" shown in Figure 3.7 which is located on the "OWL Classes" tab.


hasAuthor (instance of owl:ObjectProperty, internal name is http://www.owl-ont...

PROPERTY EDITOR for hasAuthor (instance of owl:ObjectProperty)

For Property: http://www.owl-ontologies.com/OntologyDigitalLibrary.owl#hasAuthor

Annotations

Property	Value	Lang
rdfs:comment		

Domain: Document

Range: Author

- [ ] Functional
- [ ] InverseFunctional
- [ ] Symmetric
- [ ] Transitive

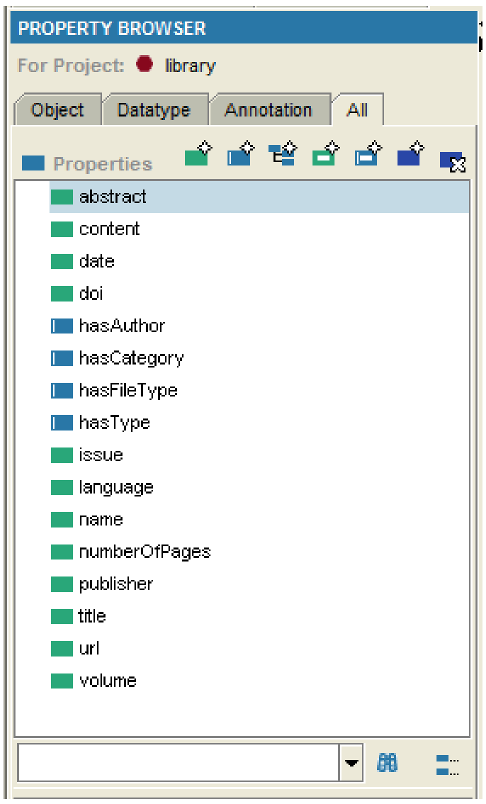
Inverse

Figure 3.7 the Properties Editor

We define a property there are a number of ways to restrict the relation. The domain and range can be specified. For example, the property "hasAuthor" has a domain of "Document" and a range of "Author". That is, it relates instances of the class "Document" to instances of the class "Author".

To create an Object property called "hasAuthor" we need to use the "Create Object Property" button shown in Figure 3.6 (second button on the left). An Object property with a generic name will be created. And then, the property is renamed to "hasAuthor", its domain and range is specified as shown in Figure 3.7 (the Properties Editor).

Four object type properties and twelve data type properties are defined in Digital Library Ontology. Object and Datatype properties are denoted by blue color and green color respectively. These properties are shown in Figure 3.8.



abstract
content
date
doi
hasAuthor
hasCategory
hasFileType
hasType
issue
language
name
numberOfPages
publisher
title
url
volume

Figure 3.8 the Properties of Digital Library Ontology

3.1.3 Defining Individuals

In addition to classes, we want to be able to describe their members. We normally think of these as individuals in our universe of things. An individual is minimally introduced by declaring it to be a member of a class [9] as follows.

```
<DocumentType rdf:ID="journal">
       <rdfs:label rdf:datatype="xsd:string">
             Journal
       </rdfs:label>
</DocumentType>
```

Note that the following is identical in meaning to the example above. The rdf:type is an RDF property that ties an individual to a class of which it is a member.

```
<owl:Thing rdf:ID="journal" />
<owl:Thing rdf:about="#journal">
       <rdf:type rdf:resource="#DocumentType"/>
</owl:Thing>
```

In the Protégé tool, OWL allows us to define individuals and to assert properties about them. Individuals can also be used in class descriptions, namely in “hasValue” restrictions and enumerated classes [16]. To create individuals in Protégé the “Individuals Tab” is used as shown in Figure 3.9.

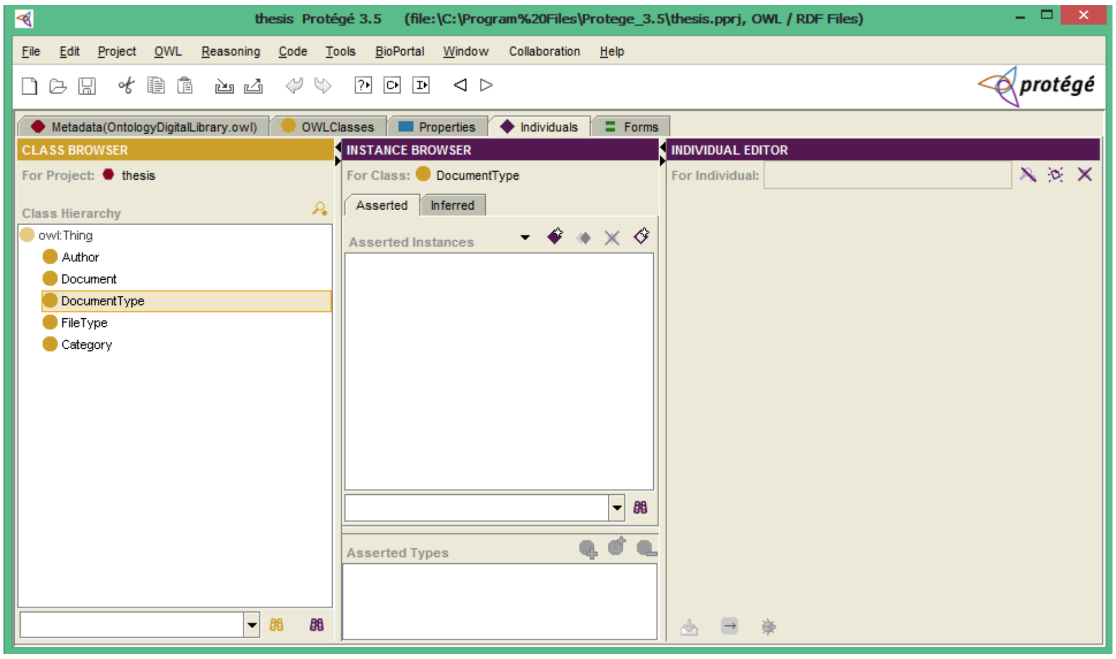

thesis Protégé 3.5 (file:\C:\Program%20Files\Protege_3.5\thesis.pprj, OWL / RDF Files)
File Edit Project OWL Reasoning Code Tools BioPortal Window Collaboration Help
protégé
Metadata(OntologyDigitalLibrary.owl) OWLClasses Properties Individuals Forms
CLASS BROWSER
For Project: thesis
Class Hierarchy
owl:Thing
Author
Document
DocumentType
FileType
Category
INSTANCE BROWSER
For Class: DocumentType
Asserted Inferred
Asserted Instances
Asserted Types
INDIVIDUAL EDITOR
For Individual:

Figure 3.9 the Individuals Tab

Suppose we wanted to describe the document types of various documents. We would first need to add various "DocumentType" to our ontology. Document types, for example, "eBook", "journal", "paper", "thesis", are typically thought of as being individuals.

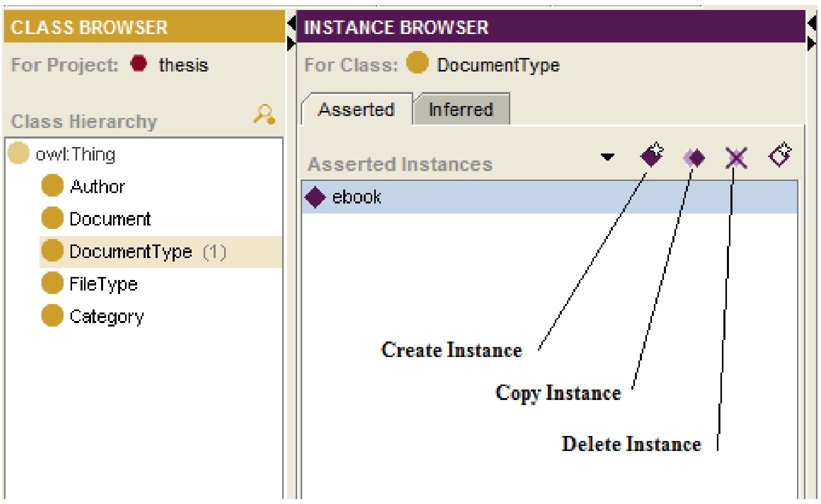


Figure 3.10 Instances Manipulation Buttons

To create the individuals for class DocumentType, we need to select the specific class from the "Class Browser" pane and then press the "Create Instance" button shown in Figure 3.10 from the "Instance Browser" pane. Finally, we have to rename the new individual to "ebook" using "Individual Editor" as shown in Figure 3.11.

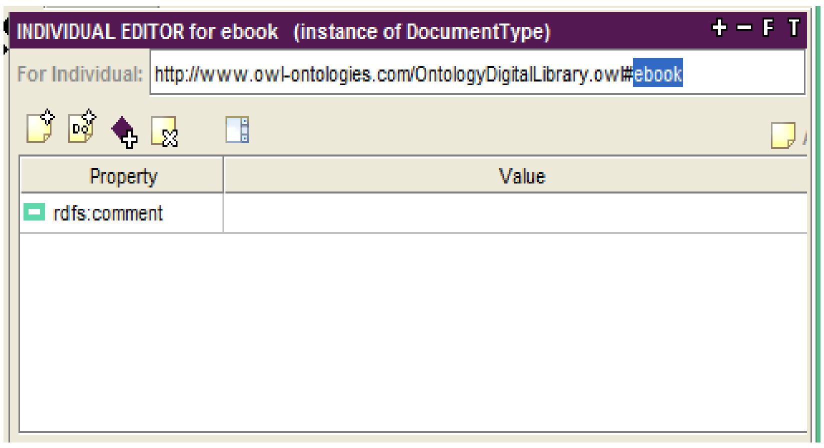

INDIVIDUAL EDITOR for ebook (instance of DocumentType)
For Individual:
http://www.owl-ontologies.com/OntologyDigitalLibrary.owl#ebook
Property
Value
rdfs:comment

Figure 3.11 the Individual Editor

In this system, forty-one individuals are created. Among them, thirty-four individuals are created for class "Category", four individuals for class "DocumentType" and three individuals for class "FileType" are defined in our Digital Library Ontology. Individuals for class "Category" are such as "accounting", "advanced database programming", "artificial intelligence", "business application area", etc. Individuals for class "FileType" are "doc", "pdf" and "txt". All individuals defined in the ontology are described in Figure 3.12, Figure 3.13, and Figure 3.14.

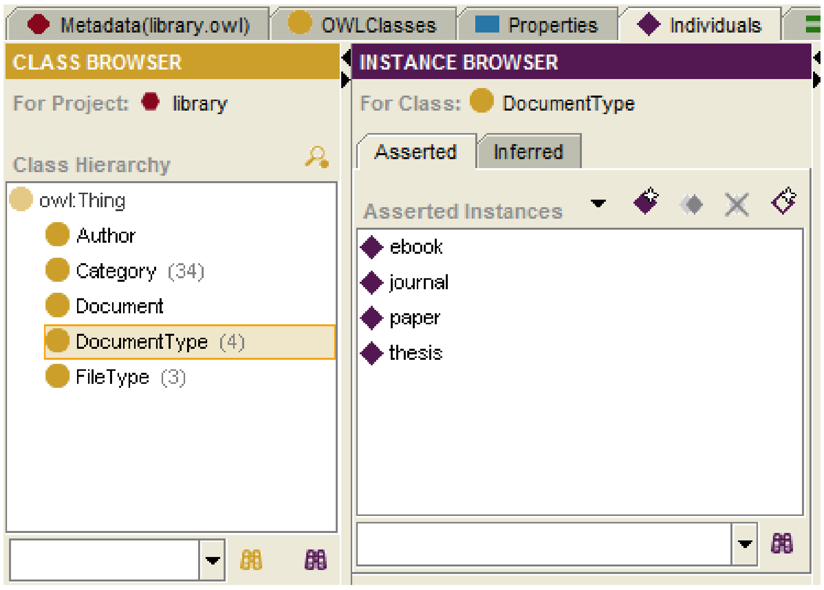
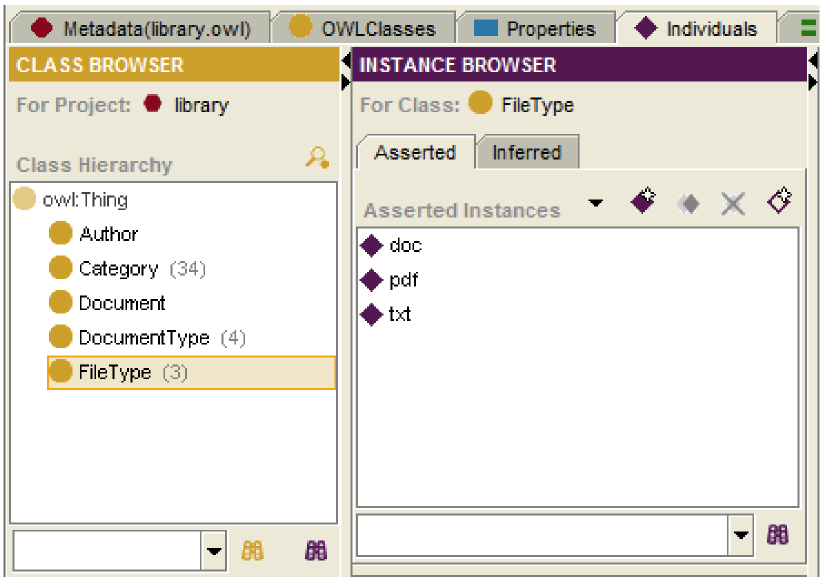


Figure 3.12 Individuals of "DocumentType" Class


Metadata(library.owl) | OWLClasses | Properties | Individuals

CLASS BROWSER

For Project: library

Class Hierarchy

- owl:Thing
  - Author
  - Category (34)
  - Document
  - DocumentType (4)
  - FileType (3)

INSTANCE BROWSER

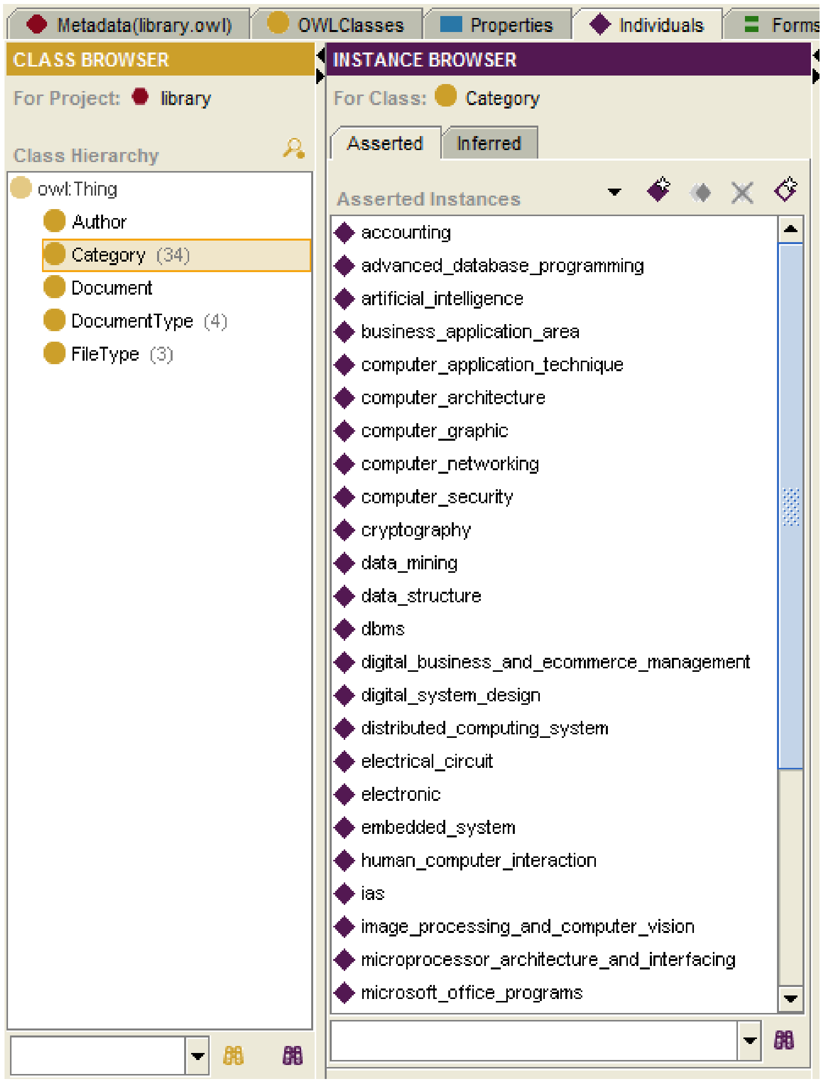
For Class: FileType

Asserted | Inferred

Asserted Instances

- doc
- pdf
- txt

Figure 3.13 Individuals of "FileType" Class


Metadata(library.owl) | OWLClasses | Properties | Individuals | Forms

CLASS BROWSER

For Project: library

Class Hierarchy

- owl:Thing
  - Author
  - Category (34)
  - Document
  - DocumentType (4)
  - FileType (3)

INSTANCE BROWSER

For Class: Category

Asserted | Inferred

Asserted Instances

- accounting
- advanced_database_programming
- artificial_intelligence
- business_application_area
- computer_application_technique
- computer_architecture
- computer_graphic
- computer_networking
- computer_security
- cryptography
- data_mining
- data_structure
- dbms
- digital_business_and_ecommerce_management
- digital_system_design
- distributed_computing_system
- electrical_circuit
- electronic
- embedded_system
- human_computer_interaction
- ias
- image_processing_and_computer_vision
- microprocessor_architecture_and_interfacing
- microsoft_office_programs

Figure 3.14 Individuals of "Category" Class

3.2 Ontology based IR Model

The main concept of Ontology-based Information Retrieval (IR) is that metadata of resources are stored in Resource Description Framework (RDF) format and retrieved not only by the keywords contained in the user query, but also by the contexts defined in Domain Ontology. Therefore, Ontology plays a significant role in this IR model to define the common vocabulary and structure for resources. In our proposed IR model, preprocessing, context matching, and calculating weight values steps are included.

3.2.1 Preprocessing Query

First of all, user-input query is preprocessed to get the keywords. In this process, tokenization and stopword removal processes are included. The tokenization process is performed by removing symbol characters such as ":", "!", "%", etc., and splitting the user query with delimiters such as white space, comma, semicolon, hyphen, and full stop characters which exist in the user query. After the tokenization process, the stopwords such as "a", "an", "the", "am", "is", "are", "about", "above", "in", "at", etc., are removed from the user query. Therefore, the preprocessing step provides for the next context matching process only the keywords contained in the user query. Example of tokenization and stopword removal process from input query is presented in below.

Input Query:

"Classification of query"

Tokenization:

1. classification
2. of
3. query

Stopword Removal:

1. classification
2. query

3.2.2 Context Matching

Context matching is the main process in Ontology-based IR model. The SPARQL query language is used to retrieve the resources in RDF format according to the keywords received by the previous step. Therefore, formatting of SPARQL query language by the keywords and property selected by the user is needed in this process. The simplified pseudo algorithm for the formatting of SPARQL query is described below. Algorithm for formatting of SPARQL is shown in Figure 3.15.

```
ALGORITHM :     Formatting of SPARQL query
INPUT :         query, property
OUTPUT :        sparql_query
BEGIN
        local variables : keywords, join_keywords, propertyName, propertyRange, propertyType,
                          prefix, sparql_query
        keywords ← tokenize(query)
        join_keywords← join(keywords, "|")
        prefix ← getPrefix()
        set propertyName = property.Name
        set propertyRange = property.Range
        set propertyType = property.Type
        set sparql_query = prefix +
                          "SELECT ?document ?i" +
                          "WHERE {?document " + propertyName + " ?i. "
        IF propertyType is owl:ObjectProperty THEN
                sparql_query += "?i a "  + propertyRange + ". "
                IF propertyRange is "dl:Author" THEN
                        sparql_query += "?i dl:name ?label "
                END IF
                sparql_query += "?i rdfs:label ?label
                                FILTER REGEX
                                (?label, '" +join_keywords + "', 'i')}";
        ELSE IF propertyType is owl:DatatypeProperty THEN
                IF propertyRange is xsd:integer THEN
                        sparql_query += "FILTER (?i='" +join_keywor s + "'^^xsd:integer)}"
                ELSE IF propertyRange is xsd:date THEN
                        sparql_query += "FILTER (?i='" + join_keywords + "'^^xsd:date)}"
                ELSE IF propertyRange is xsd:string THEN
                        sparql_query += "FILTER REGEX(?i, '" + join_keywords + "', 'i')}"
                END IF
        END IF
        RETURN sparql_query
END
```

Figure 3.15 Algorithm for formatting of SPARQL

The above algorithm takes two variables “query” and “property” given by the user, and returns the formatted SPARQL query for context matching with the RDF data source. The functions “tokenize” and “join” are called by the algorithm to preprocess the user input query. The function “getPrefix” gives the entire prefixes of our Ontology to use in execution of SPARQL query. The variable “property” selected by the user is divided into “propertyName”, “propertyRange” and “propertyType”. The main concept of the algorithm is that if the type of property is ObjectProperty then the querying process will perform by the appropriate range of this property. Otherwise, the querying process will perform by the appropriate data type of property.

Different filtering pattern for the keywords is used in this algorithm according to the range of property. The sample formatted SPARQL queries by the algorithm are described in follows:

Example 1:

Input Query: “classification of query”

Input Property: *name*-“title”, *type*-“owl:DatatypeProperty”, *range*-“xsd:string”

Output SPARQL:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dl: <http://www.owl-ontologies.com/library.owl#>
SELECT ?document ?i
WHERE {
     ?document dl:title ?i.
     FILTER REGEX(?i, '\\bclassification\\b|\\bquery\\b', 'i')
}
```

Example 2:

Input Query: “Aung Myint”

Input Property: *name*-“hasAuthor”, *type*-“owl:ObjectProperty”, *range*-“dl:Author”

Output SPARQL:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dl: <http://www.owl-ontologies.com/library.owl#>
SELECT ?document ?i
WHERE {
     ?document dl:hasAuthor ?i.
     ?i a dl:Author.
     ?i dl:name ?label
     FILTER REGEX(?label, '\\baung\\b|\\bmyint\\b', 'i')
}
```

Example 3:

Input Query: “2020”

Input Property: *name*-“date”, *type*-“owl:DatatypeProperty”, *range*-“xsd:date”

Output SPARQL:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dl: <http://www.owl-ontologies.com/library.owl#>
SELECT ?document ?i
WHERE {
     ?document dl:date ?i.
     FILTER (?i='2020'^^xsd:date)
}
```

3.2.3 Calculating TF-IDF and Similarity

In this step, the vector space model is used to retrieve more accurate data and rank it which semantically enhances the searching and retrieval process. In the vector space IR model, a document is represented as a weight vector, in which each component weight is computed based on some variation of TF or TF-IDF scheme.

The vector space model is a statistical model for representing text information for Information Retrieval. It is a simple, mathematically based approach that provides partial matching and ranked results. TF-IDF weighting is the most common term weighting approach for vector space model retrieval. The weight of the term in document vector can be determined using the method. The weight of the term is measured how often the term *j* occurs in document *i* (the Term Frequency) and IDF (the Inverse Document Frequency) as shown in Equation 3.1 and 3.2.

The weight equation for the term within document is as follows:

$$w_{ij} = tf_{ij} \times idf_i \tag{3.1}$$

where,

w_{ij} = weight of the term t_i in document d_j

tf_{ij} = the normalize term frequency (TF) of term t_i in document d_j

idf_i = the inverse document frequency (IDF) of term t_i

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots, f_{|v|j}\}} \tag{3.2}$$

where, f_{ij} = the raw frequency count of term t_i in document d_j

$$idf_i = \log \frac{N}{df_i} \tag{3.3}$$

where,

df_i = number of document in which term t_i appears at least once

N = the total number of document in the system

A query *q* is represented in exactly the same way as a document. The weight equation for the term within query is as follows:

$$w_{iq} = \left[0.5 + \frac{0.5 f_{iq}}{\max\{f_{1q}, f_{2q}, \ldots\ldots, f_{|v|q}\}} \right] \times \log \frac{N}{df_i} \tag{3.4}$$

where,

w_{iq} = weight of the term t_i in query vector q

f_{iq} = the raw frequency count of term t_i in query vector q

The similarity between query q and j^{th} document retrieved by context matching process is calculated by Dice similarity method as shown in Equation 3.5. This is quantified as the

$$Dice(d_j, q) = \frac{2\left|\sum_{i=1}^{|v|} w_{ij} \times w_{iq}\right|}{\sum_{i=1}^{|v|} w_{ij}^2 + \sum_{i=1}^{|v|} w_{iq}^2} \tag{3.5}$$

where,

Dice (d_j, q) = the dice similarity between document d_j and query q

w_{ij} = weight of the term i within document d_j

w_{iq} = weight of the term i within query

Example of TF-IDF and Dice Similarity calculation is explained with step by step in below.

Input Query:

Keyword:	"classification of query"
In Property:	"title"

Retrieved Documents by Context Matching Process:

Resource	Value
dl:Document1	"Web Query Classification System using NoSQL Graph Database"
dl:Document2	"Query Classification based Information Retrieval System"
dl:Document3	"Performance Comparison between Keyword based and Classification based Information Retrieval System"

STEP-1: Term Extraction (Tokenization and Stopword Removal) from resources. The extracted terms from resources are shown in Table 3.1. In total 19 terms from three documents are extracted. Among these, 7 terms with underline are duplicated, so 12 are received.

Table 3.1 Extracted Terms from Documents

Document1:	Document2:	Document3:
1. web	1. query	1. performance
2. query	2. classification	2. comparison
3. classification	3. information	3. keyword
4. system	4. retrieval	4. classification
5. nosql	5. system	5. information
6. graph		6. retrieval
7. database		7. system

STEP-2: Term Frequency (TF) Calculating for each Term in each Document. In this step, the Term Frequency for extracted terms from resources is calculated by Equation 3.2. These TF values are described in Table 3.2.

Table 3.2 Term Frequency Result for Extracted Terms

ID	Term	Document1	Document2	Document3
1	Web	1/1 = 1		
2	Query	1/1 = 1	1/1 = 1	
3	Classification	1/1 = 1	1/1 = 1	1/1 = 1
4	System	1/1 = 1	1/1 = 1	1/1 = 1
5	Nosql	1/1 = 1		
6	Graph	1/1 = 1		
7	Database	1/1 = 1		
8	Information		1/1 = 1	1/1 = 1
9	Retrieval		1/1 = 1	1/1 = 1
10	Performance			1/1 = 1
11	Comparison			1/1 = 1
12	Keyword			1/1 = 1

STEP-3: Inverse Document Frequency (IDF) Calculating for each Term in each Document. The IDF values for terms are calculated by Equation 3.3 and shown in Table 3.3.

Table 3.3 Inverse Document Frequency Result for Extracted Terms

ID	Term	Inverse Document Frequency (IDF)
1	Web	Log(3/1) = 0.47712
2	Query	Log(3/2) = 0.17609
3	Classification	Log(3/3) = 0
4	System	Log(3/3) = 0
5	Nosql	Log(3/1) = 0.47712
6	Graph	Log(3/1) = 0.47712
7	Database	Log(3/1) = 0.47712
8	Information	Log(3/2) = 0.17609
9	Retrieval	Log(3/2) = 0.17609
10	Performance	Log(3/1) = 0.47712
11	Comparison	Log(3/1) = 0.47712
12	Keyword	Log(3/1) = 0.47712

STEP-4: TF-IDF (weight) Calculating for each Term in each Document. The TF-IDF values for terms are calculated by Equation 3.1 and shown in Table 3.4.

Table 3.4 TF-IDF Result for Extracted Terms

ID	Term	Document1	Document2	Document3
1	Web	1 x 0.477 = 0.477		
2	Query	1 x 0.176 = 0.176	1 x 0.176 = 0.176	
3	Classification	1 x 0 = 0	1 x 0 = 0	1 x 0 = 0
4	System	1 x 0 = 0	1 x 0 = 0	1 x 0 = 0
5	Nosql	1 x 0.477 = 0.477		
6	Graph	1 x 0.477 = 0.477		
7	Database	1 x 0.477 = 0.477		
8	Information		1 x 0.176 = 0.176	1 x 0.176 = 0.176
9	Retrieval		1 x 0.176 = 0.176	1 x 0.176 = 0.176
10	Performance			1 x 0.477 = 0.477
11	Comparison			1 x 0.477 = 0.477
12	Keyword			1 x 0.477 = 0.477

STEP-5: TF-IDF (weight) Calculating for each Term in the keyword. The weight values for extracted terms from keyword are calculated by Equation 3.4 as follows.

weight $_{(\text{classification, keyword})}$ = (0.5 + (0.5 × (1/1))) × log 3/3 = **0**

weight $_{(\text{query, keyword})}$ = (0.5 + (0.5 × (1/1))) × log 3/2 = **0.17609**

STEP-6: Dice Similarity Calculating. The similarity between keywords and retrieved documents by context matching process is calculated by Dice Equation 3.5 as shown in below and the similarity results are shown in Table 3.5:

Similarity between keyword and Document1

$$Dice(document1, keyword) = \frac{2\left|\sum_{i=1}^{|v|} w_{i,document1} \times w_{i,keyword}\right|}{\sum_{i=1}^{|v|} w_{i,document1}^{2} + \sum_{i=1}^{|v|} w_{i,keyword}^{2}}$$

$$= \frac{2\left|\sum_{i=1}^{|2|} w_{i,document1} \times w_{i,keyword}\right|}{\sum_{i=1}^{|7|} w_{i,document1}^{2} + \sum_{i=1}^{|2|} w_{i,keyword}^{2}}$$

$$= \frac{2\left|(0 \times 0) + (0.17609 \times 0.17609)\right|}{\left[(0.477)^2 + (0.176)^2 + (0)^2 + (0)^2 + (0.477)^2 + (0.477)^2 + (0.477)^2\right] \quad + \quad \left[(0)^2 + (0.176)^2\right]}$$

$$= \frac{2\left|(0) + (0.03101)\right|}{\left[(0.22764) + (0.03101) + (0) + (0) + (0.22764) + (0.22764) + (0.22764)\right] \quad + \quad \left[(0) + (0.03101)\right]}$$

$$= \frac{2|0.03101|}{[0.94157] + [0.03101]}$$

$$= \frac{0.06202}{0.97258}$$

=0.06377

<u>Similarity between keyword and Document2</u>

$$Dice(document2, keyword) = \frac{2|\sum_{i=1}^{|v|} w_{i,document2} \times w_{i,keyword}|}{\sum_{i=1}^{|v|} w_{i,document2}^{2} + \sum_{i=1}^{|v|} w_{i,keyword}^{2}}$$

$$= \frac{2|\sum_{i=1}^{|2|} w_{i,document2} \times w_{i,keyword}|}{\sum_{i=1}^{|5|} w_{i,document2}^{2} + \sum_{i=1}^{|2|} w_{i,keyword}^{2}}$$

$$= \frac{2|(0 \times 0) + (0.17609 \times 0.17609)|}{\left[(0.17609)^2 + (0)^2 + (0)^2 + (0.17609)^2 + (0.17609)^2\right] + \left[(0)^2 + (0.17609)^2\right]}$$

$$= \frac{2|(0) + (0.03101)|}{[(0.03101) + (0) + (0) + (0.03101) + (0.03101)] + [(0) + (0.03101)]}$$

$$= \frac{2|0.03101|}{[0.09303] + [0.03101]}$$

$$= \frac{0.06202}{0.12404}$$

=0.5

<u>Similarity between keyword and Document2</u>

$$Dice(document3, keyword) = \frac{2|\sum_{i=1}^{|v|} w_{i,document3} \times w_{i,keyword}|}{\sum_{i=1}^{|v|} w_{i,document3}^{2} + \sum_{i=1}^{|v|} w_{i,keyword}^{2}}$$

$$= \frac{2|\sum_{i=1}^{|1|} w_{i,document3} \times w_{i,keyword}|}{\sum_{i=1}^{|7|} w_{i,document3}^{2} + \sum_{i=1}^{|2|} w_{i,keyword}^{2}}$$

$$= \frac{2|(0 \times 0)|}{\left[(0)^2 + (0)^2 + (0.176)^2 + (0.176)^2 + (0.477)^2 + (0.477)^2 + (0.477)^2\right] + \left[(0)^2 + (0.176)^2\right]}$$

$$= \frac{0}{[0.74494] + [0.03101]}$$

$$= \frac{0}{0.77595}$$

= 0

STEP-7: Ranking Documents by Similarity Score. After calculating the similarity values between documents and keywords, the retrieved documents are ranked by score. The ranked result is shown in Table 3.6.

Table 3.5 Similarity Results for Documents

Id	**similarity score**
Document1	0.06377
Document2	0.5
Document3	0

Table 3.6 Ranked Results for Documents

Id	**similarity score**	**Remark**
Document2	0.5	Most Relevant Document
Document1	0.06377	
Document3	0	

3.3 Summary

In this chapter, designing the Domain Ontology for Digital Library is described. Web Ontology Language and Protégé editor are used to design Ontology for Digital Library. Five subclasses: Document, Author, Category, DocumentType and FileType are defined in our Ontology. Four object type properties and twelve data type properties are also defined for the specific class. The designed Ontology plays a significant role in our IR model to define the common vocabulary and structure for resources. In our proposed IR model, preprocessing, context matching and calculating weight values steps are included. All the steps of proposed IR model are explained in detail in this chapter.

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

The detailed implementation of Ontology-based information retrieval system is presented in this chapter. Design and use case diagrams of the system, class structure of Ontology Web Language (OWL) is also included in this chapter. The logical architecture of the system and implementation of programming components are also explained in this chapter. Finally, it presents a graphical user interface of the system with step-by-step detailed explanation figures and experimental results of the system.

4.1 Overview Design of the System

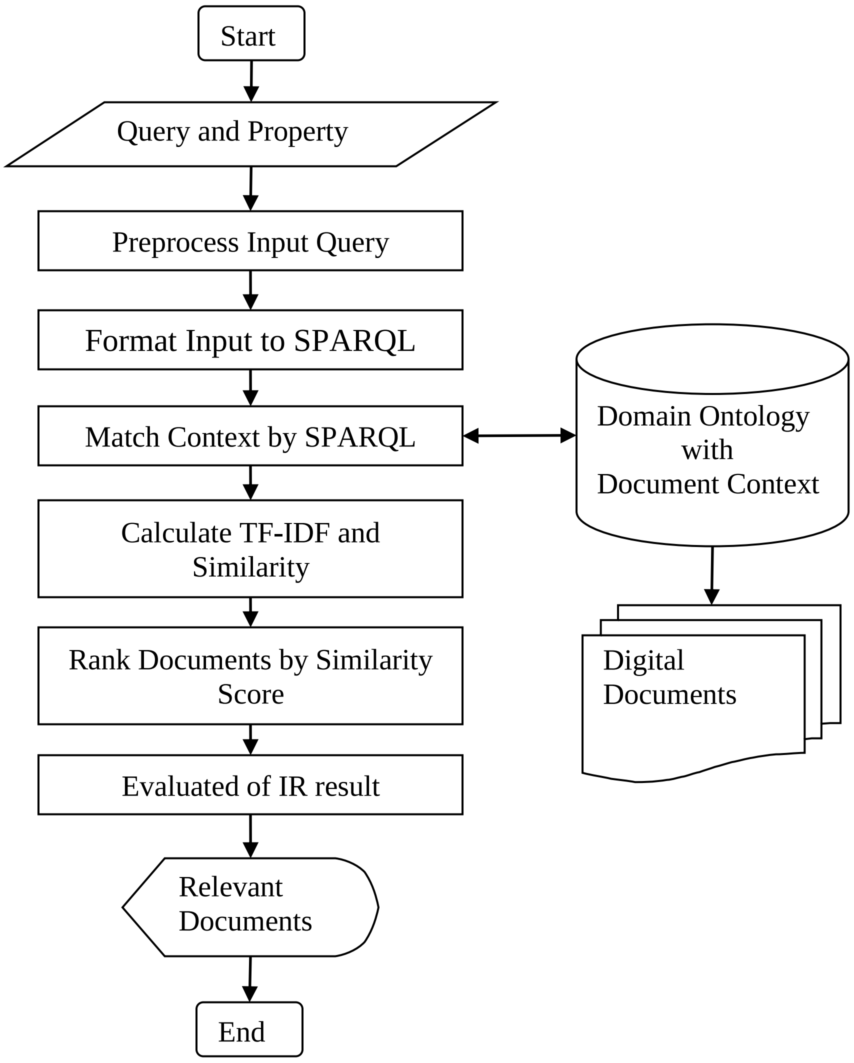


Figure 4.1 Overview Design of the System

The overview design diagram of the system is shown in Figure 4.1. The proposed system is implemented as the information retrieval system by using Domain Ontology. The main point of the proposed system is the formatting of the SPARQL query and context matching process by using the SPARQL query. In this system, there are six main steps.

In the first step, query preprocessing, which consists of the tokenization and stopwords removal process for the user query, is performed. This system accepts the query and property selected by the user to retrieve relevant documents from Digital Library.

In the second step, the tokenized keywords and selected property by the user are transformed to SPARQL query format by the algorithm for the formatting of SPARQL query which is described in the previous chapter Figure 3.15.

In the third step, the context matching process by formatted SPARQL query is performed. This process is used to match the context of documents from Domain Ontology with the formatted SPARQL query. The results of this process are relevant documents by the keywords and property of the document.

In the fourth step, relevant documents retrieved by context matching processes are calculated for TF-IDF values and similarity scores by using the Vector Space Model (VSM) and the Dice similarity method respectively.

In the next step, retrieved documents are ranked according to their similarity scores, and the whole process for retrieving documents is done here. Evaluation of the results of IR is performed in the final step by calculating its precision, recall, and f-measure values.

The relevant documents retrieved by SPARQL query are ranked and displayed as the result of our Ontology-based IR system.

4.2 Ontology Structure of the System

In this system, ontology is constructed for defining the metadata of documents and retrieving this using Protégé v 3.5. The structure of Digital Library Ontology is shown in Figure 4.2. There are six classes denoted by yellow color. The “Thing” class is the root class, and the “Document”, “Author”, “DocumentType”, “FileType”, and “Category” classes are the subclasses of “Thing”. All the relationships between root and subclasses are types of rdfs:subClassOf and shown in the diagram with purple color. The instances of these classes are designed in green color. The datatype properties of the classes: title and publisher, are shown in the box of the respective class. The Object properties of the classes: hasAuthor and hasCategory are described by the relationship between the boxes in black lines.

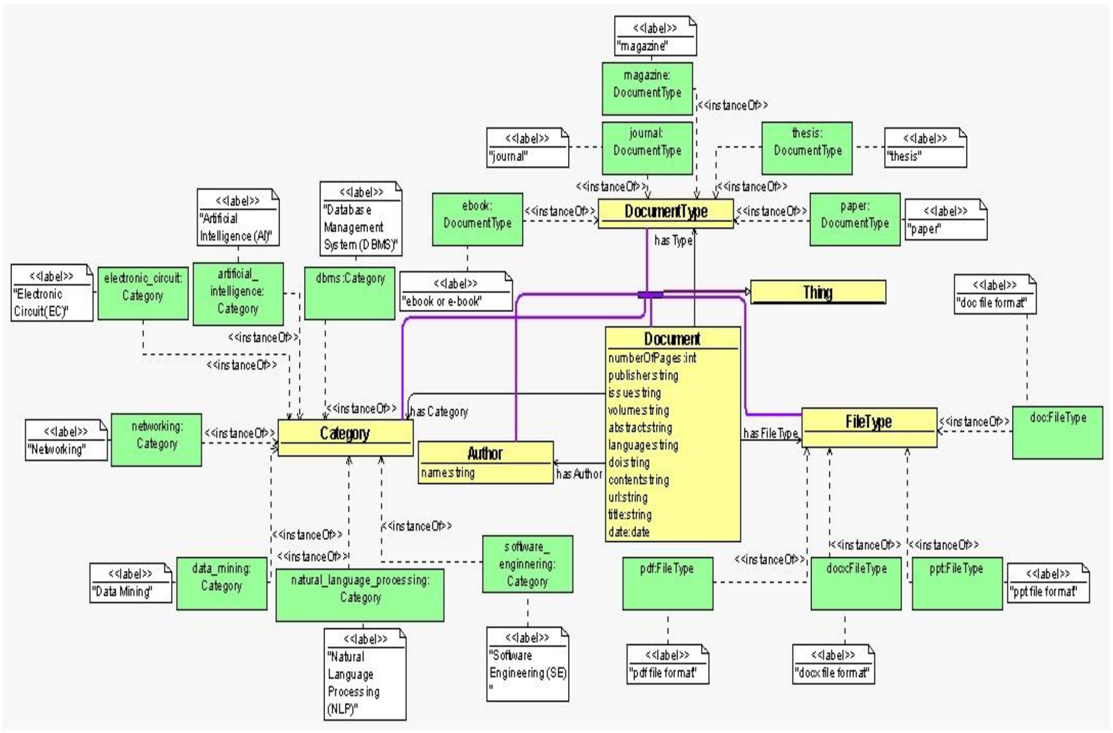


Figure 4.2 Ontology Structure of the System

4.3 Implementation of the system

The Ontology-based IR system for Digital Library is implemented based on Service-Oriented Architecture (SOA) by using the XML based web service technology and ASP.NET. The logical architecture of the system is shown in Figure 4.3.

The architecture of the proposed system consists of file storage for documents, one ontology dataset, and two programming components. All functions for the Digital Library web service can be grouped into two modules: Publication Module and Retrieval Module. The functions of the publication module are extracting contexts from documents and saving them to a dataset. The whole IR process of our proposed system is provided by the functions of the retrieval module. In our system architecture, the Digital Library web application just plays in the role of the user interface. Ontology dataset is used to store the extracted context of documents and file storage is used to save documents themselves.

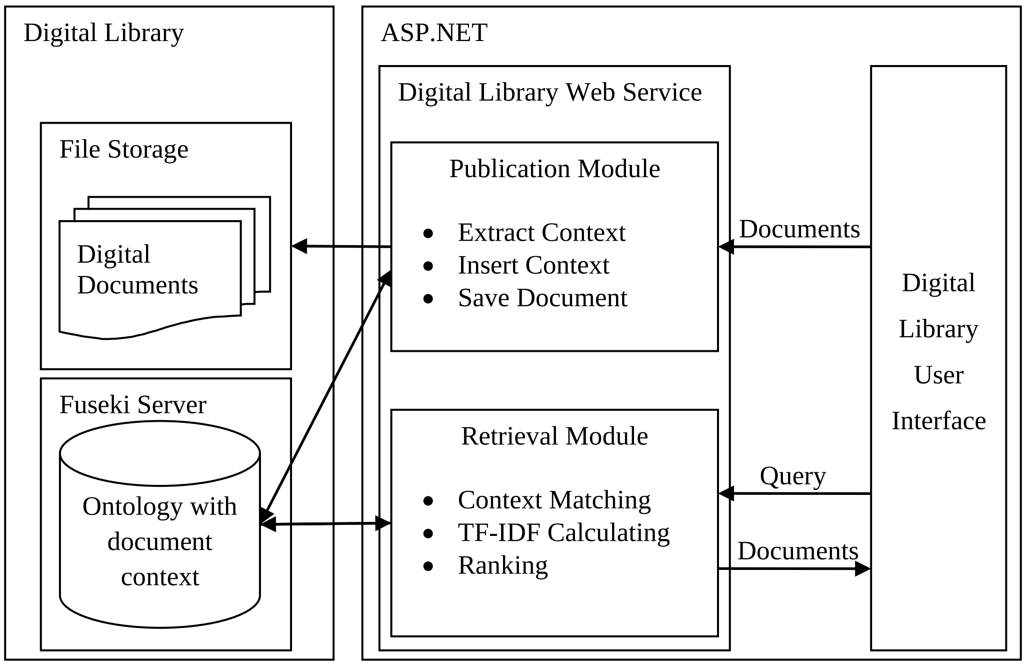


Figure 4.3 Architecture of the System

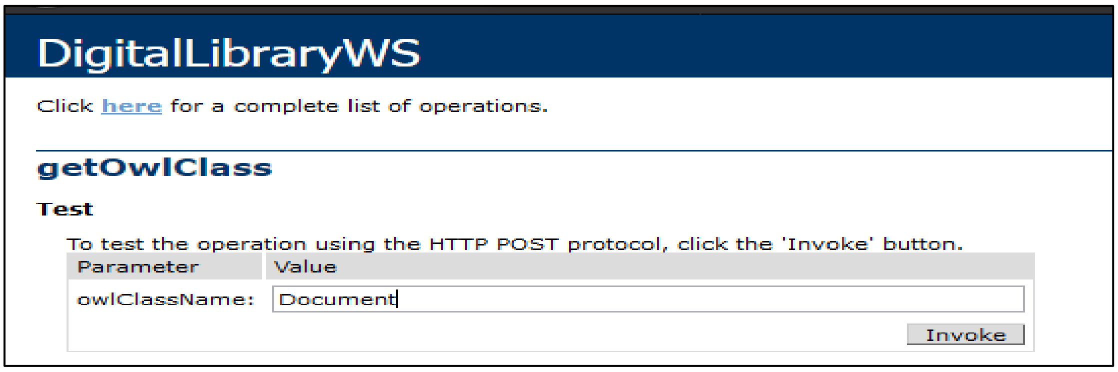
4.3.1 Implementation of Digital Library Web Service

The Digital Library web service is implemented by using C# programming language. This web service consists of functions for publication and retrieving of documents. Getting the class structure of Ontology and its instances, saving and manipulating the instances of the specific classes, and extracting the contents of documents are the main functions of the publication module. The functions of the publication module are performed by connecting with the ontology dataset on the Fuseki server. These functions are as follows:

- getOwlClass: getting the whole structure of a specific class including its datatype and object properties from Ontology dataset
- getIndividuals: getting all the instances of a specific class from Ontology dataset.
- getIndividualByName: getting an instance of a specific class by its name from the Ontology dataset.
- setIndividual: saving an instance of a specific class to the Ontology dataset. The name of the instance is programmatically defined by the last inserted ID for this class.
- setIndividualByName: saving an instance of a specific class to the Ontology dataset by a given name.

- updateIndividual: manipulating the properties of an instance of the specific class by name of this instance.
- deleteIndividual: deleting an instance of the specific class by its name from the Ontology dataset.
- isExist: checking the instance of a specific class is exist in our Ontology dataset or not.
- isDocumentExist: checking the specific instance of Document class is exist in our Ontology dataset or not.
- isAuthorExit: checking the specific instance of Author class is exist in our Ontology or not.
- getFileContent: extracting the content from various types of files such as “pdf”, “txt” and “docx”.

Testing the getOwlClass function of publication module of web service with a sample input parameter “Document” is shown in Figure 4.4. As a result, the structure of the Document class with fifteen properties is returned by getOwlClass function. The result in XML format returned by this function is shown in Figure 4.6. Testing the getIndividuals function and its result are shown in Figure 4.5 and Figure 4.7 respectively.


DigitalLibraryWS

Click here for a complete list of operations.

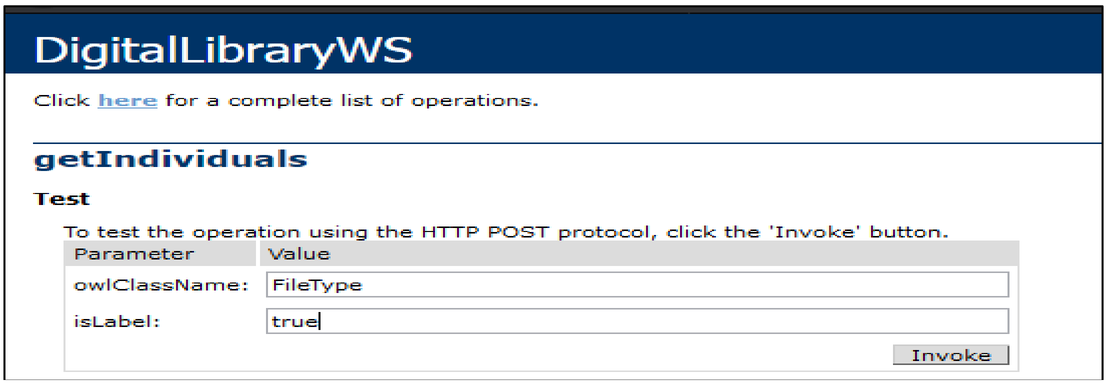
getOwlClass

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
owlClassName:	Document

Figure 4.4 Testing the getOwlClass Function of Web Service


DigitalLibraryWS

Click here for a complete list of operations.

getIndividuals

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
owlClassName:	FileType
isLabel:	true

Figure 4.5 Testing the getIndividuals Function of Web Service

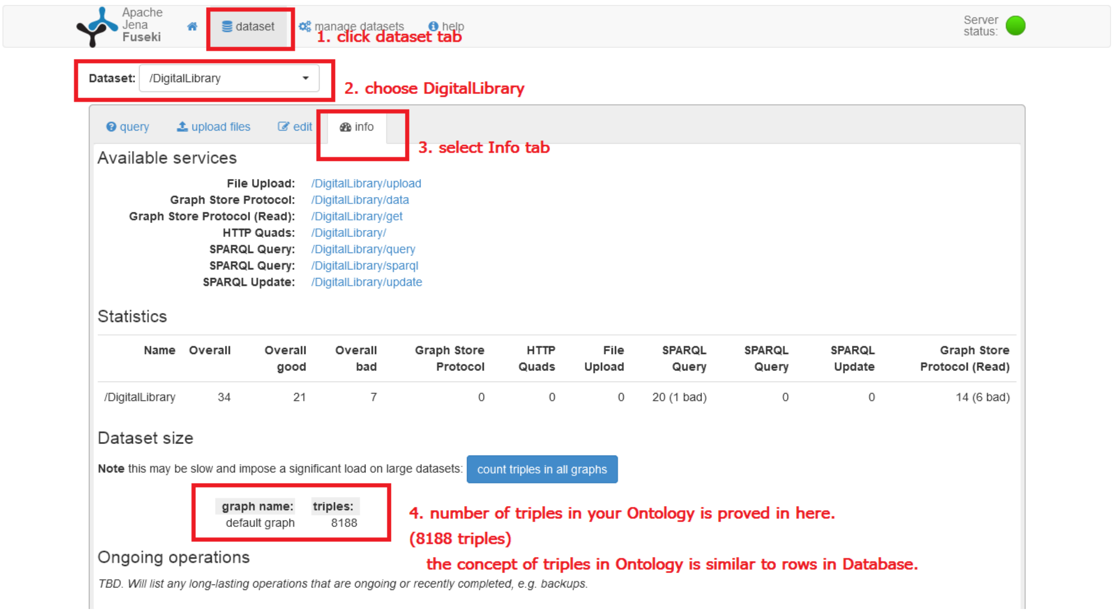
```
<OwlClass>
   <ClassName>dl:Document</ClassName>
   <Properties>
      <Property>
         <Id>1</Id>
         <Label>Title</Label>
         <Name>dl:title</Name>
         <Type>owl:DatatypeProperty</Type>
         <Range>xsd:string</Range>
      </Property>
      <Property>
         <Id>2</Id>
         <Label>Author(s)</Label>
         <Name>dl:hasAuthor</Name>
         <Type>owl:ObjectProperty</Type>
         <Range>dl:Author</Range>
      </Property>
      <Property>
        … <Label>Category</Label> …
      </Property>
      <Property>
        … <Label>Publisher or Journal</Label> …
      </Property>
      <Property>
        … <Label>Published Date</Label> …
      </Property>
      <Property>
        … <Label>Type</Label> …
      </Property>
      <Property>
        … <Label>Format</Label> …
      </Property>
      <Property>
        … <Label>Volume</Label> …
      </Property>
      <Property>
        … <Label>Issue</Label> …
      </Property>
      <Property>
        … <Label>Number of Pages</Label> …
      </Property>
      <Property>
        … <Label>Language</Label> …
      </Property>
      <Property>
        … <Label>Identifier (DOI, ISBN,
        ISSN)</Label> …
      </Property>
      <Property>
        … <Label>URL</Label> …
      </Property>
      <Property>
        … <Label>Abstract</Label> …
      </Property>
      <Property>
        … <Label>Content</Label> …
      </Property>
   </Properties>
</OwlClass>
```

Figure 4.6 Owl Class Structure schema

```
<ArrayOfOwlClass>
   <OwlClass>
      <Label>.doc</Label>
      <ClassName>dl:doc</ClassName>
      <SValue>0</SValue>
      <ID>0</ID>
      <Properties/>
   </OwlClass>
   <OwlClass>
      <Label>.pdf</Label>
      <ClassName>dl:pdf</ClassName>
      <SValue>0</SValue>
      <ID>0</ID>
      <Properties/>
   </OwlClass>
   <OwlClass>
      <Label>.txt</Label>
      <ClassName>dl:txt</ClassName>
      <SValue>0</SValue>
      <ID>0</ID>
      <Properties/>
   </OwlClass>
</ArrayOfOwlClass>
```

Figure 4.7 Schema of File Type Instances returned by getIndividuals Function

In Figure 4.8, all information of dataset published by the admin is stored in digital library domain ontology. If the admin update or delete the context of the document, information will be changed in ontology dataset.



Available services

File Upload:	/DigitalLibrary/upload
Graph Store Protocol:	/DigitalLibrary/data
Graph Store Protocol (Read):	/DigitalLibrary/get
HTTP Quads:	/DigitalLibrary/
SPARQL Query:	/DigitalLibrary/query
SPARQL Query:	/DigitalLibrary/sparql
SPARQL Update:	/DigitalLibrary/update

Statistics

Name	Overall	Overall good	Overall bad	Graph Store Protocol	HTTP Quads	File Upload	SPARQL Query	SPARQL Query	SPARQL Update	Graph Store Protocol (Read)
/DigitalLibrary	34	21	7	0	0	0	20 (1 bad)	0	0	14 (6 bad)

Dataset size

Note this may be slow and impose a significant load on large datasets:

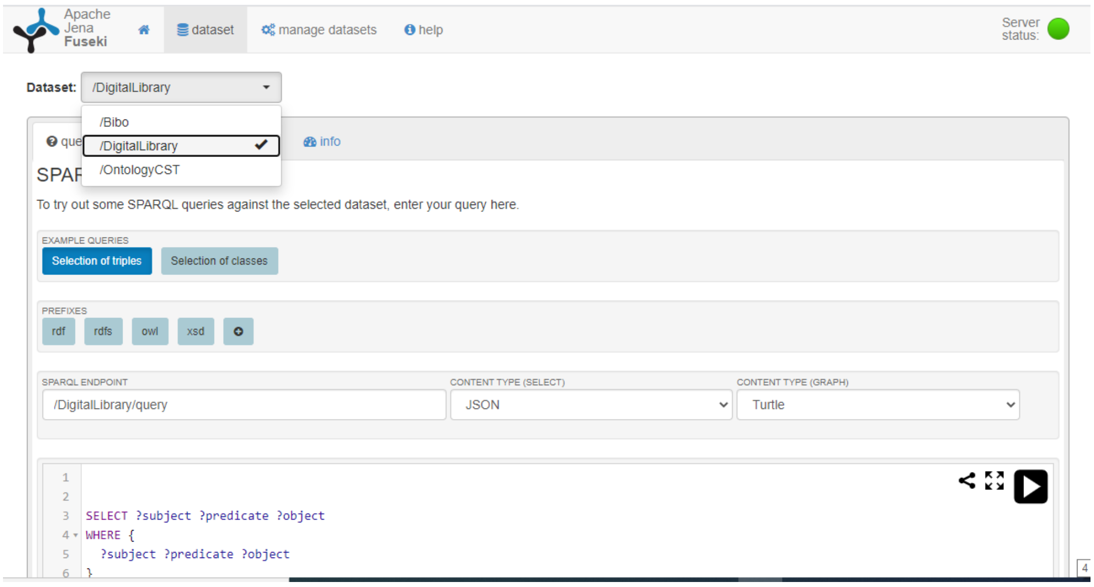
graph name:	triples:
default graph	8188

Ongoing operations

TBD. Will list any long-lasting operations that are ongoing or recently completed, e.g. backups.

Figure 4.8 Dataset uploaded to Digital Library Ontology on Apache Jena Fuskei Server

As this system ontology is implemented basing on, library ontology is created on Protégé editor v3.5 and ontology file is stored and executed on the Fuseki Server. And then its dataset or information are stored in library ontology when data is uploaded. The number of datasets and their information uploaded by admin can be seen in the graph in Figures 4.9, 4.10, 4.11, 4.12, 4.13, and 4.14.

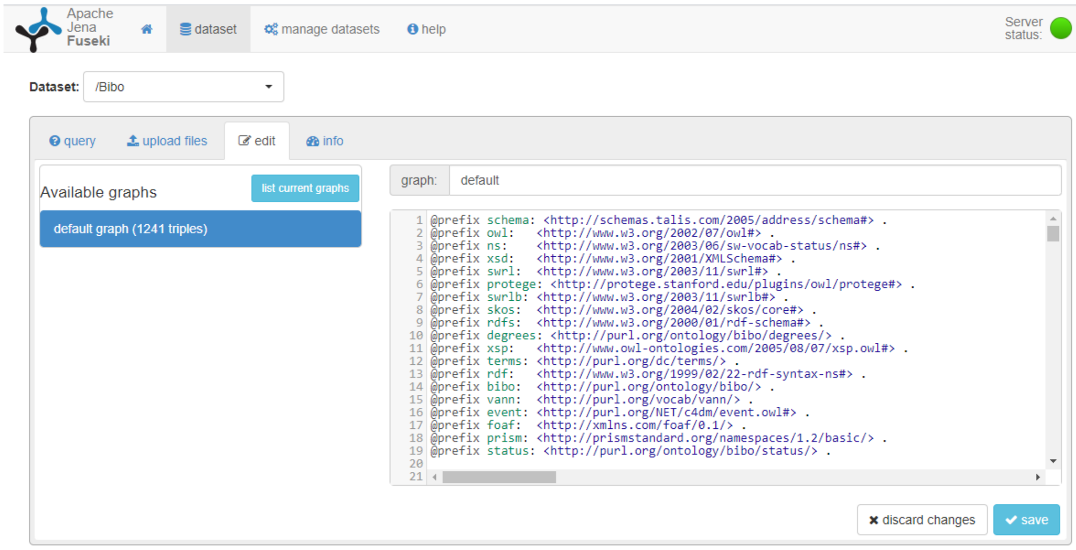


To try out some SPARQL queries against the selected dataset, enter your query here.

SPARQL ENDPOINT: /DigitalLibrary/query
CONTENT TYPE (SELECT): JSON
CONTENT TYPE (GRAPH): Turtle

```

SELECT ?subject ?predicate ?object
WHERE {
  ?subject ?predicate ?object
}
```

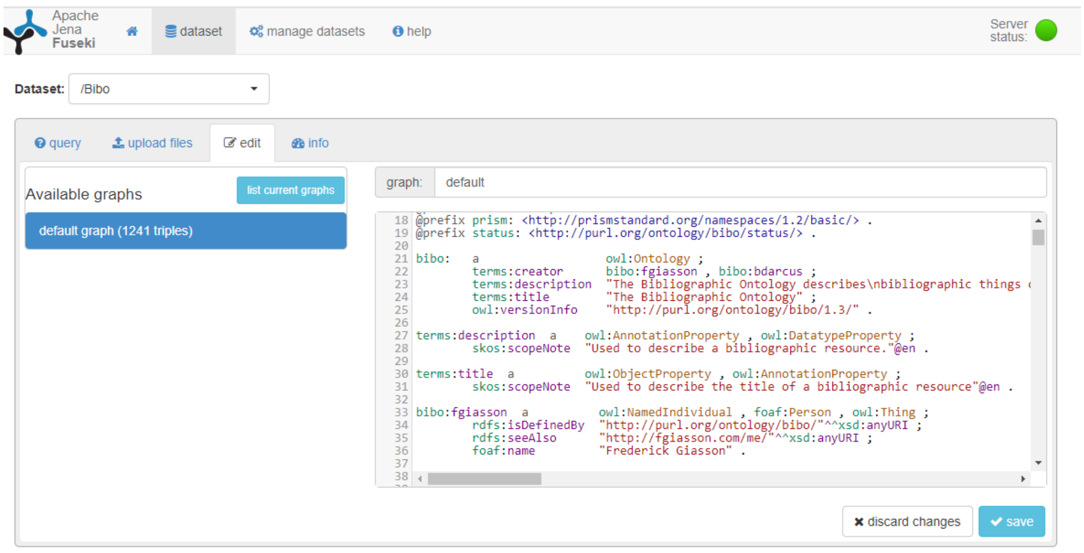
Figure 4.9 The Digital Library created in Fuseki Server



graph: default

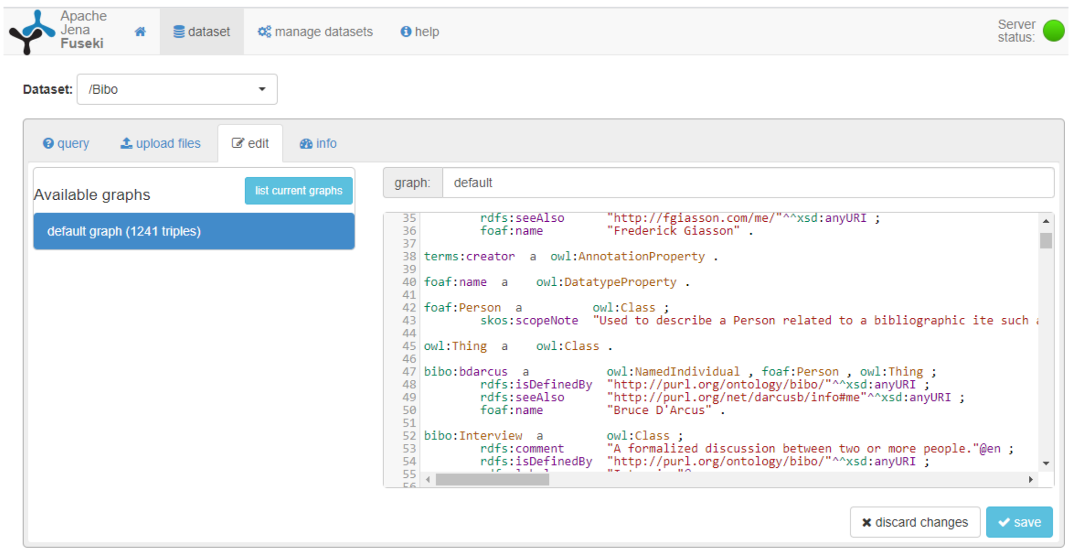
```
@prefix schema: <http://schemas.talis.com/2005/address/schema#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix ns:    <http://www.w3.org/2003/06/sw-vocab-status/ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix swrl:  <http://www.w3.org/2003/11/swrl#> .
@prefix protege: <http://protege.stanford.edu/plugins/owl/protege#> .
@prefix swrlb: <http://www.w3.org/2003/11/swrlb#> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix degrees: <http://purl.org/ontology/bibo/degrees/> .
@prefix xsp:   <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .
@prefix terms: <http://purl.org/dc/terms/> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix bibo:  <http://purl.org/ontology/bibo/> .
@prefix vann:  <http://purl.org/vocab/vann/> .
@prefix event: <http://purl.org/NET/c4dm/event.owl#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix prism: <http://prismstandard.org/namespaces/1.2/basic/> .
@prefix status: <http://purl.org/ontology/bibo/status/> .
```

Figure 4.10 The dataset on the Digital Library



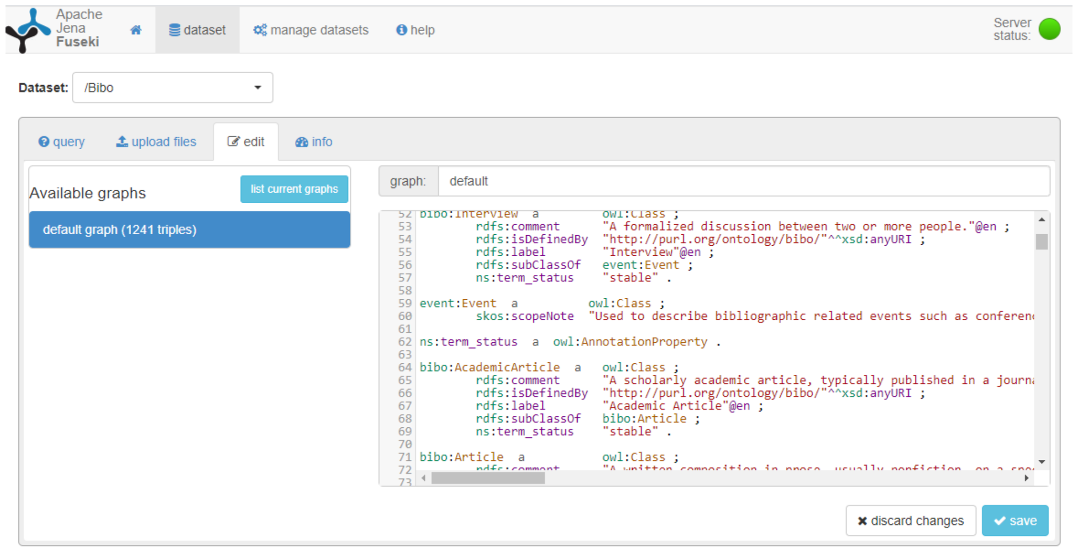
```
@prefix prism: <http://prismstandard.org/namespaces/1.2/basic/> .
@prefix status: <http://purl.org/ontology/bibo/status/> .

bibo:   a                   owl:Ontology ;
        terms:creator       bibo:fgiasson , bibo:bdarcus ;
        terms:description   "The Bibliographic Ontology describes\nbibliographic things
        terms:title         "The Bibliographic Ontology" ;
        owl:versionInfo     "http://purl.org/ontology/bibo/1.3/" .

terms:description  a    owl:AnnotationProperty , owl:DatatypeProperty ;
        skos:scopeNote  "Used to describe a bibliographic resource."@en .

terms:title  a          owl:ObjectProperty , owl:AnnotationProperty ;
        skos:scopeNote  "Used to describe the title of a bibliographic resource"@en .

bibo:fgiasson  a          owl:NamedIndividual , foaf:Person , owl:Thing ;
        rdfs:isDefinedBy  "http://purl.org/ontology/bibo/"^^xsd:anyURI ;
        rdfs:seeAlso      "http://fgiasson.com/me/"^^xsd:anyURI ;
        foaf:name         "Frederick Giasson" .

```

Figure 4.11 List of Dataset on Digital Library Ontology



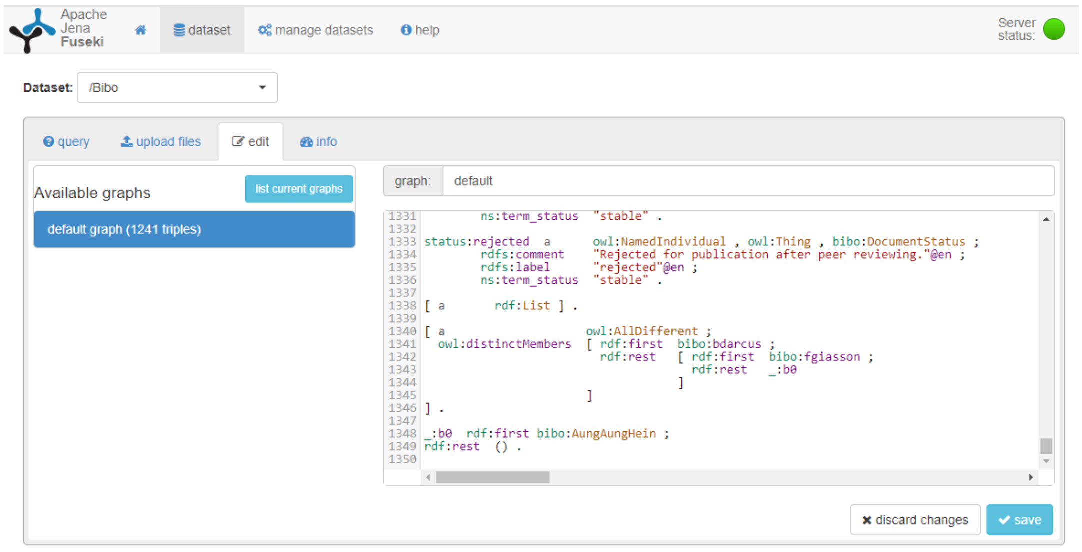
```
        rdfs:seeAlso      "http://fgiasson.com/me/"^^xsd:anyURI ;
        foaf:name         "Frederick Giasson" .

terms:creator  a  owl:AnnotationProperty .

foaf:name  a    owl:DatatypeProperty .

foaf:Person  a          owl:Class ;
        skos:scopeNote  "Used to describe a Person related to a bibliographic ite such

owl:Thing  a    owl:Class .

bibo:bdarcus  a           owl:NamedIndividual , foaf:Person , owl:Thing ;
        rdfs:isDefinedBy  "http://purl.org/ontology/bibo/"^^xsd:anyURI ;
        rdfs:seeAlso      "http://purl.org/net/darcusb/info#me"^^xsd:anyURI ;
        foaf:name         "Bruce D'Arcus" .

bibo:Interview  a         owl:Class ;
        rdfs:comment      "A formalized discussion between two or more people."@en ;
        rdfs:isDefinedBy  "http://purl.org/ontology/bibo/"^^xsd:anyURI ;
```

Figure 4.12 List of Dataset on Digital Library Ontology



```
bibo:Interview  a          owl:Class ;
        rdfs:comment      "A formalized discussion between two or more people."@en ;
        rdfs:isDefinedBy  "http://purl.org/ontology/bibo/"^^xsd:anyURI ;
        rdfs:label        "Interview"@en ;
        rdfs:subClassOf   event:Event ;
        ns:term_status    "stable" .

event:Event  a          owl:Class ;
        skos:scopeNote  "Used to describe bibliographic related events such as conferen

ns:term_status  a  owl:AnnotationProperty .

bibo:AcademicArticle  a   owl:Class ;
        rdfs:comment      "A scholarly academic article, typically published in a journa
        rdfs:isDefinedBy  "http://purl.org/ontology/bibo/"^^xsd:anyURI ;
        rdfs:label        "Academic Article"@en ;
        rdfs:subClassOf   bibo:Article ;
        ns:term_status    "stable" .

bibo:Article  a           owl:Class ;
        rdfs:comment      "A written composition in prose, usually nonfiction, on a spec
```

Figure 4.13 List of Dataset on Digital Library Ontology



```
        ns:term_status  "stable" .

status:rejected  a      owl:NamedIndividual , owl:Thing , bibo:DocumentStatus ;
        rdfs:comment    "Rejected for publication after peer reviewing."@en ;
        rdfs:label      "rejected"@en ;
        ns:term_status  "stable" .

[ a       rdf:List ] .

[ a                     owl:AllDifferent ;
  owl:distinctMembers  [ rdf:first  bibo:bdarcus ;
                         rdf:rest   [ rdf:first  bibo:fgiasson ;
                                      rdf:rest   _:b0
                                    ]
                       ]
] .

_:b0  rdf:first bibo:AungAungHein ;
rdf:rest  () .
```

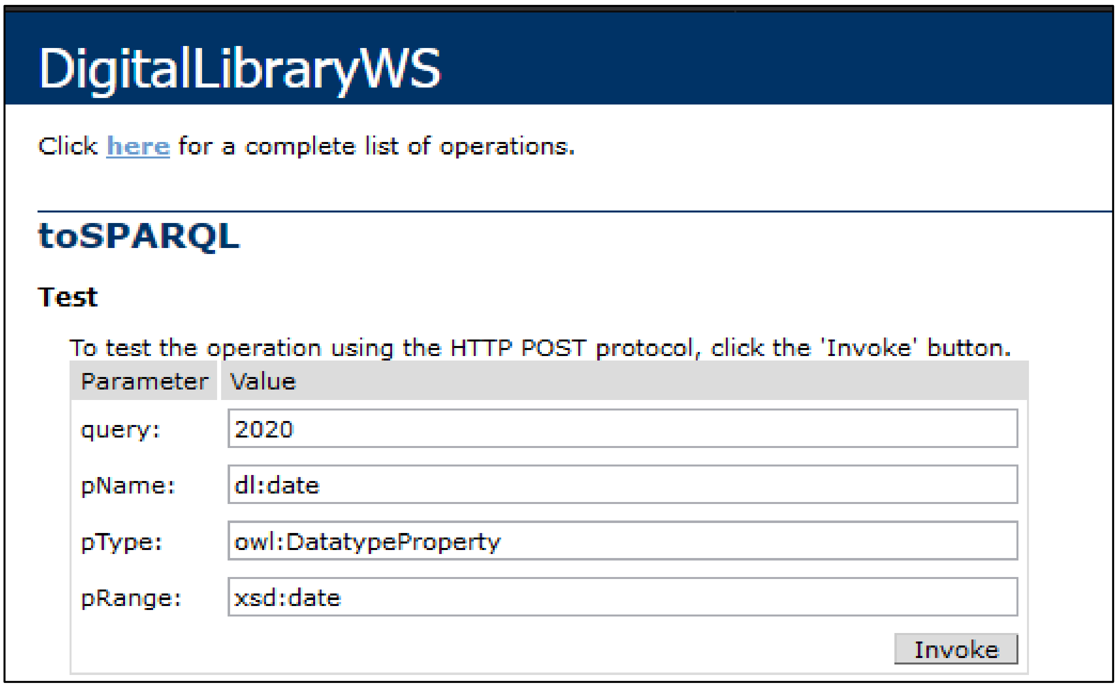
Figure 4.14 List of Dataset on Digital Library Ontology

Formatting SPARQL query, context matching, calculating similarity and evaluating IR results are the main functions for the retrieval module of Digital Library web service. The description of these functions and sample testing of implementation is shown in below.

- tokenizeQuery: tokenizing the words and preprocessing the text from user query, such as removing punctuation, special characters, numbers and so on

- toSPARQL: converting tokenized keywords to SPARQL query according to the selected property by user.
- contextMatch: matching the context of documents and retrieving them form Ontology dataset by formatted SPARQL query.
- calculateSimilarity: calculating the similarity between retrieved documents and user query.
- rank: ranking the retrieved documents by their similarity values.
- evaluateIR: evaluating the precision, recall, and f-measure scores of IR by the number of retrieved and relevant documents.
- saveIRResult: saving the precision, recall, and f-measure scores of IR in the database.
- getIRResult: getting the precision, recall, and f-measure scores of IR from the database.

Testing the toSPARQL function of retrieval module of web service with sample input parameters is shown in Figure 4.15. Query, name, type, and range of property are given in this example. The formatted SPARQL query in XML format returned by this function is shown in Figure 4.16.


DigitalLibraryWS

Click here for a complete list of operations.

toSPARQL

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
query:	2020
pName:	dl:date
pType:	owl:DatatypeProperty
pRange:	xsd:date

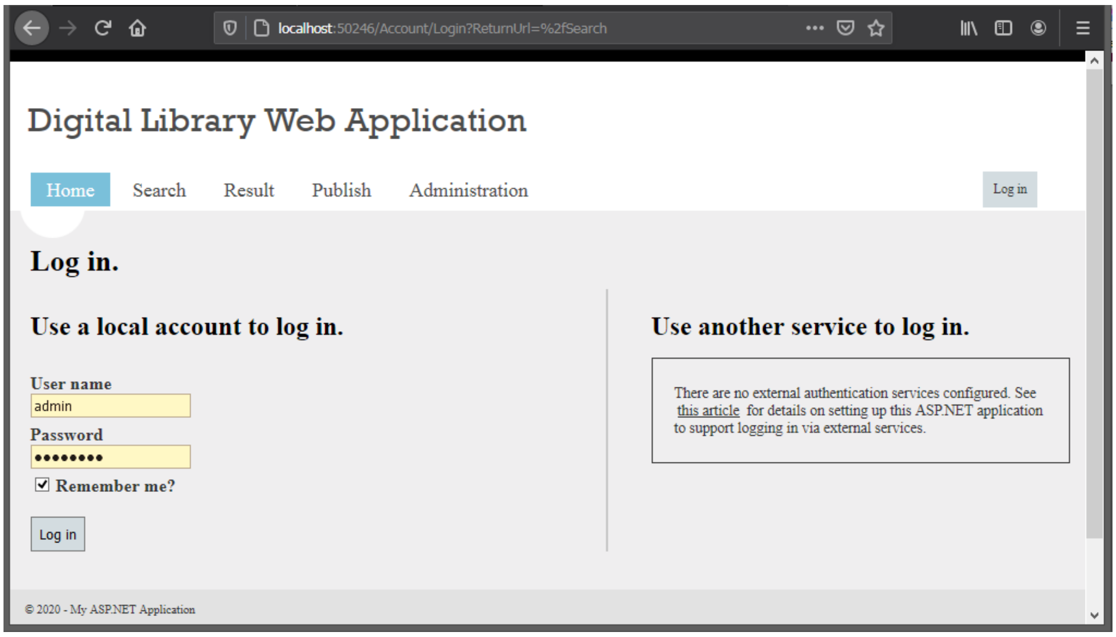
Figure 4.15 Testing the toSPARQL Function of Web Service

```
    <string>
       PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
       PREFIX owl: <http://www.w3.org/2002/07/owl#>
       PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
       PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
       PREFIX dl: <http://www.owl-ontologies.com/library.owl#>
       SELECT ?document ?i
       WHERE {
          ?document dl:date ?i.
          FILTER (?i='2020'^^xsd:date)
       }
    </string>
```

Figure 4.16 The Formatted SPARQL query returned by toSPARQL Function

4.3.2 Implementation of Digital Library Web Application

The user interface is designed and implemented as a web application in ASP.NET platform for testing the operations of web services. Tow types of roles for the user: Admin Role and User Role are defined in web applications. Admin can edit all the resources of IR system for Digital Library, such as management of user information, the publication of documents to Ontology dataset and manipulation of their information. The users can search for digital documents by keywords and property of documents. This application consists of five menus: Home, Search, Result, Publish, and Administration. All of these menus are available only for authenticated users. The admin and users must be login to our Digital Library web application by the "Login" page as shown in Figure 4.17.


localhost:50246/Account/Login?ReturnUrl=%2fSearch

Digital Library Web Application

Home Search Result Publish Administration Log in

Log in.

Use a local account to log in.

User name: admin

Password: ••••••••

☑ Remember me?

Log in

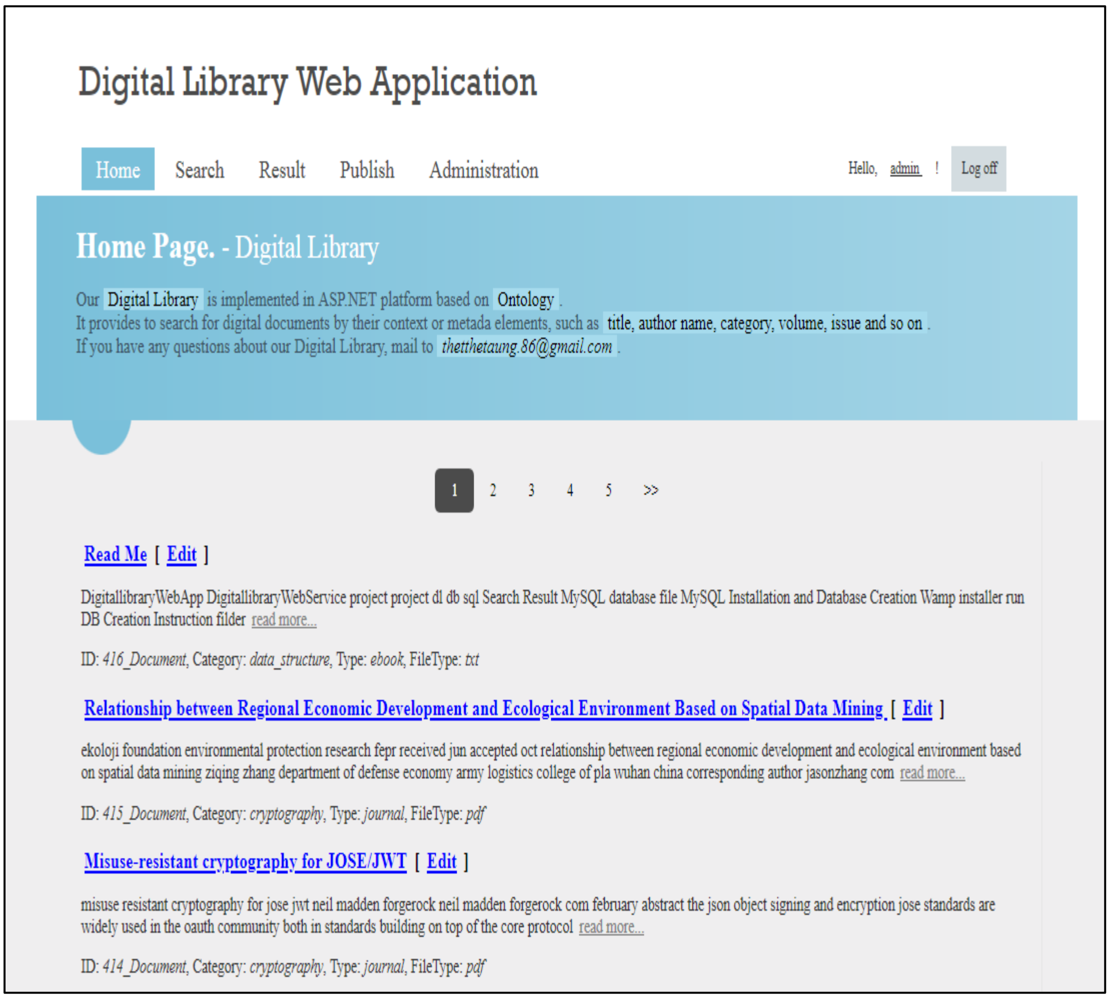
Use another service to log in.

There are no external authentication services configured. See this article for details on setting up this ASP.NET application to support logging in via external services.

© 2020 - My ASP.NET Application

Figure 4.17 Login Page of Digital Library Web Application

There is no registration process in our system. The registration for user accounts is done only by the admin of our system. After the login process is performed, the webpage of the "Home" menu can be seen as shown in Figure 4.18. The "Administration" and "Publish" menus, and "Edit" option for documents are available only for the admin account.



Home Page. - Digital Library

Our Digital Library is implemented in ASP.NET platform based on Ontology .
It provides to search for digital documents by their context or metada elements, such as title, author name, category, volume, issue and so on .
If you have any questions about our Digital Library, mail to *thetthetaung.86@gmail.com* .

Read Me [Edit]

DigitallibraryWebApp DigitallibraryWebService project project dl db sql Search Result MySQL database file MySQL Installation and Database Creation Wamp installer run DB Creation Instruction filder read more...

ID: *416_Document*, Category: *data_structure*, Type: *ebook*, FileType: *txt*

Relationship between Regional Economic Development and Ecological Environment Based on Spatial Data Mining [Edit]

ekoloji foundation environmental protection research fepr received jun accepted oct relationship between regional economic development and ecological environment based on spatial data mining ziqing zhang department of defense economy army logistics college of pla wuhan china corresponding author jasonzhang com read more...

ID: *415_Document*, Category: *cryptography*, Type: *journal*, FileType: *pdf*

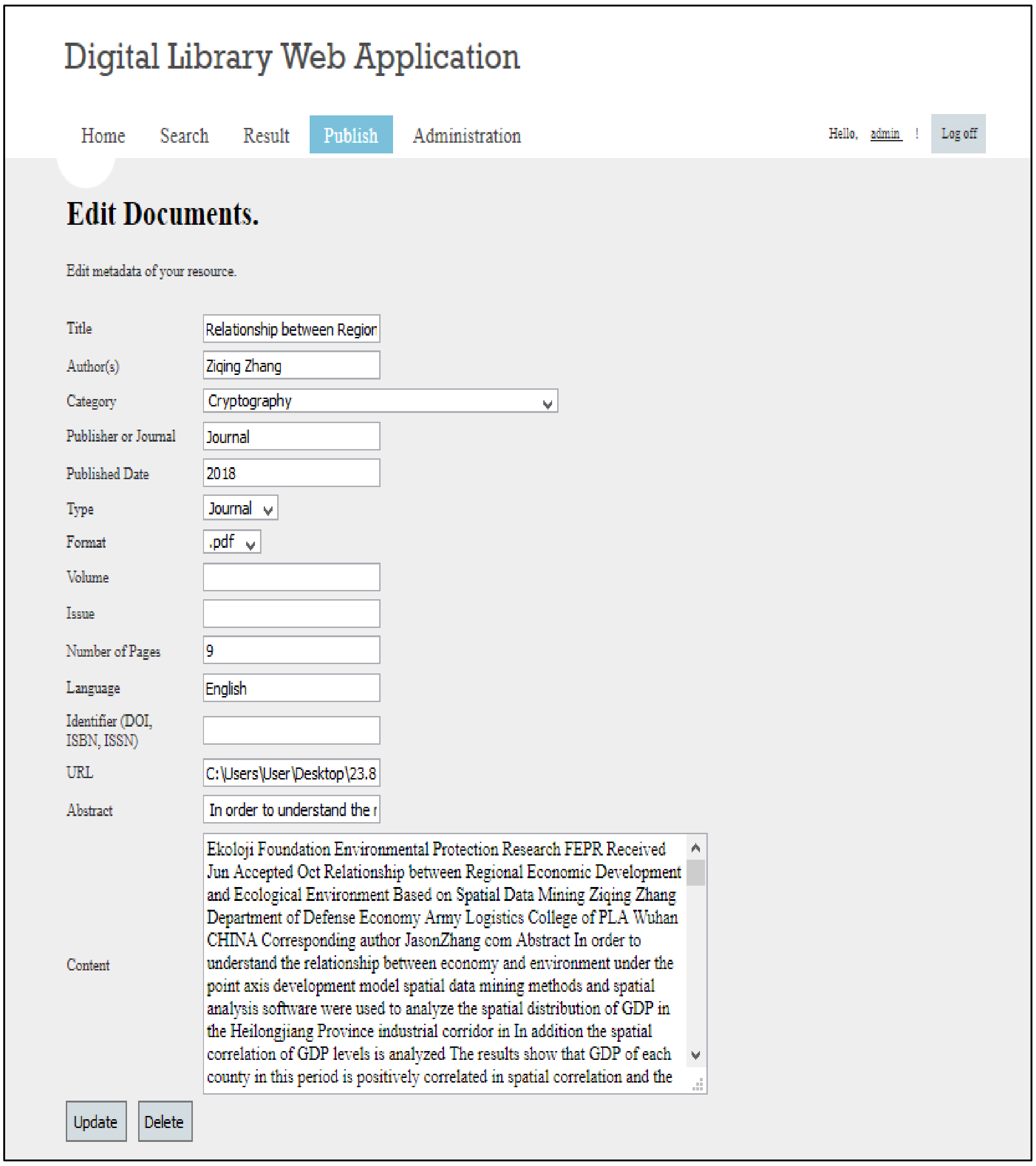
Misuse-resistant cryptography for JOSE/JWT [Edit]

misuse resistant cryptography for jose jwt neil madden forgerock neil madden forgerock com february abstract the json object signing and encryption jose standards are widely used in the oauth community both in standards building on top of the core protocol read more...

ID: *414_Document*, Category: *cryptography*, Type: *journal*, FileType: *pdf*

Figure 4.18 Home Page of Digital Library Web Application

On this "Home" page, the library user and admin can see all the list of documents from our Digital Library by clicking the pagination buttons. The titles of the documents are the links for downloading them. When the "Edit" link of a specific document is clicked by the admin, the "Edit" page will be displayed as shown in Figure 4.19. On this page, all the metadata of a specific document can be updated or the whole document can be deleted by the admin.



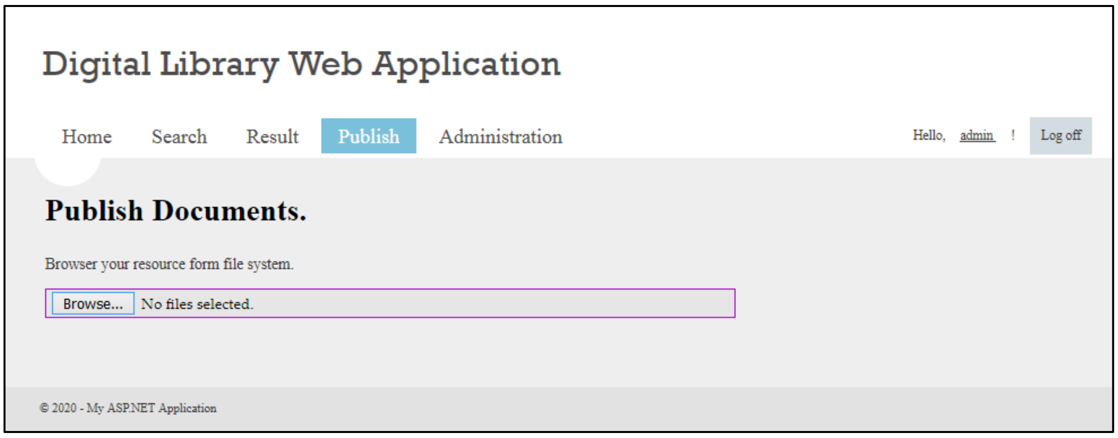
Edit Documents.

Edit metadata of your resource.

Title	Relationship between Regior
Author(s)	Ziqing Zhang
Category	Cryptography
Publisher or Journal	Journal
Published Date	2018
Type	Journal
Format	.pdf
Volume	
Issue	
Number of Pages	9
Language	English
Identifier (DOI, ISBN, ISSN)	
URL	C:\Users\User\Desktop\23.8
Abstract	In order to understand the r
Content	Ekoloji Foundation Environmental Protection Research FEPR Received Jun Accepted Oct Relationship between Regional Economic Development and Ecological Environment Based on Spatial Data Mining Ziqing Zhang Department of Defense Economy Army Logistics College of PLA Wuhan CHINA Corresponding author JasonZhang com Abstract In order to understand the relationship between economy and environment under the point axis development model spatial data mining methods and spatial analysis software were used to analyze the spatial distribution of GDP in the Heilongjiang Province industrial corridor in In addition the spatial correlation of GDP levels is analyzed The results show that GDP of each county in this period is positively correlated in spatial correlation and the

Figure 4.19 Edit Page of Digital Library Web Application

Admin not only can edit the metadata of documents but also can publish them to Ontology dataset. The "Publish" page which is shown in Figure 4.20 is used to browse a file from the computer and import it to our dataset and file storage. Once a file is browsed, its' content, URL, and format are automatically extracted by our application. The values for all the rest of the properties of the document should be filled by the admin manually in the publication process. The extracted and filled metadata of a document entitled "Introduction to SPARQL" is shown in Figure 4.21.

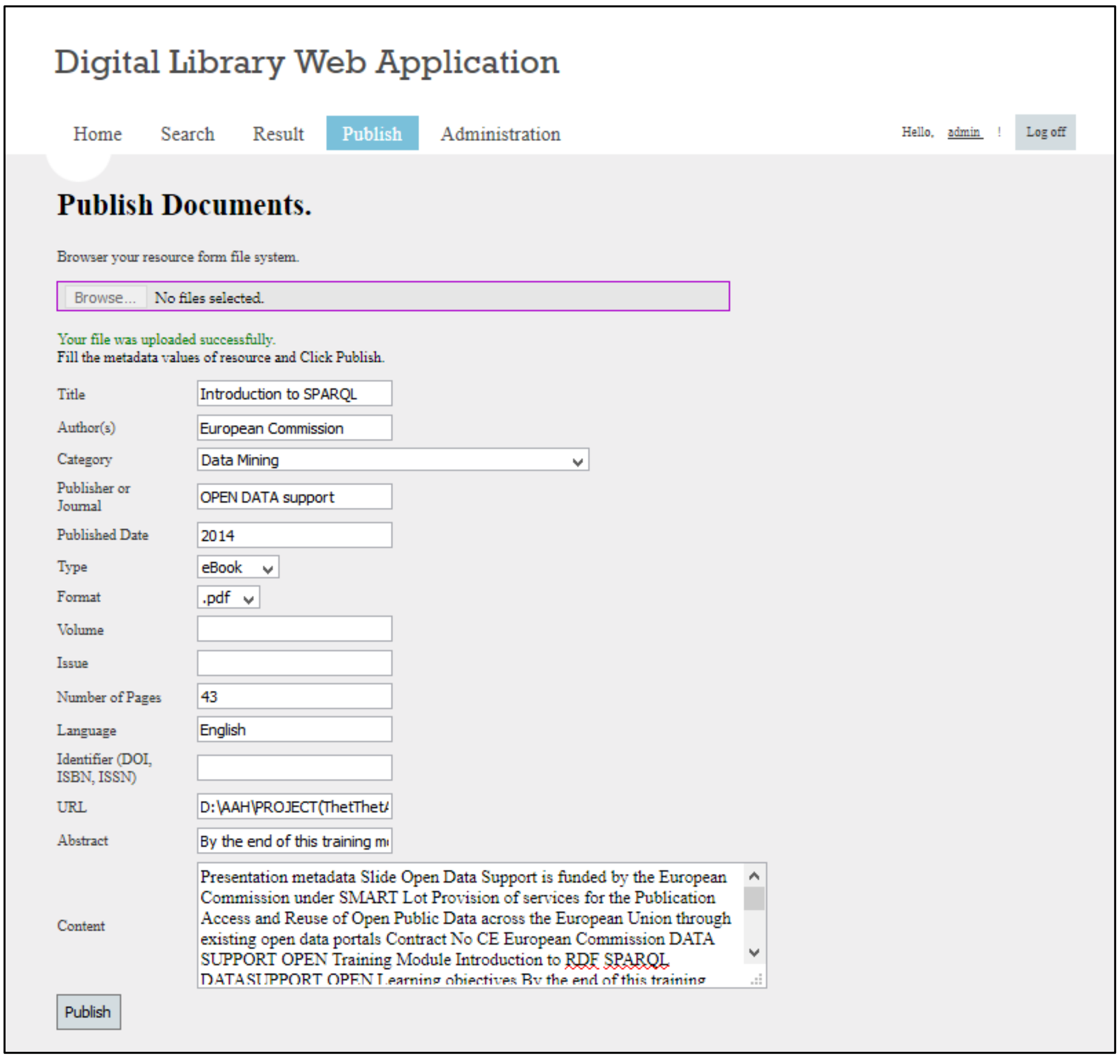


Publish Documents.

Browser your resource form file system.

No files selected.

Figure 4.20 Publish Page for Browsing the Document



Publish Documents.

Browser your resource form file system.

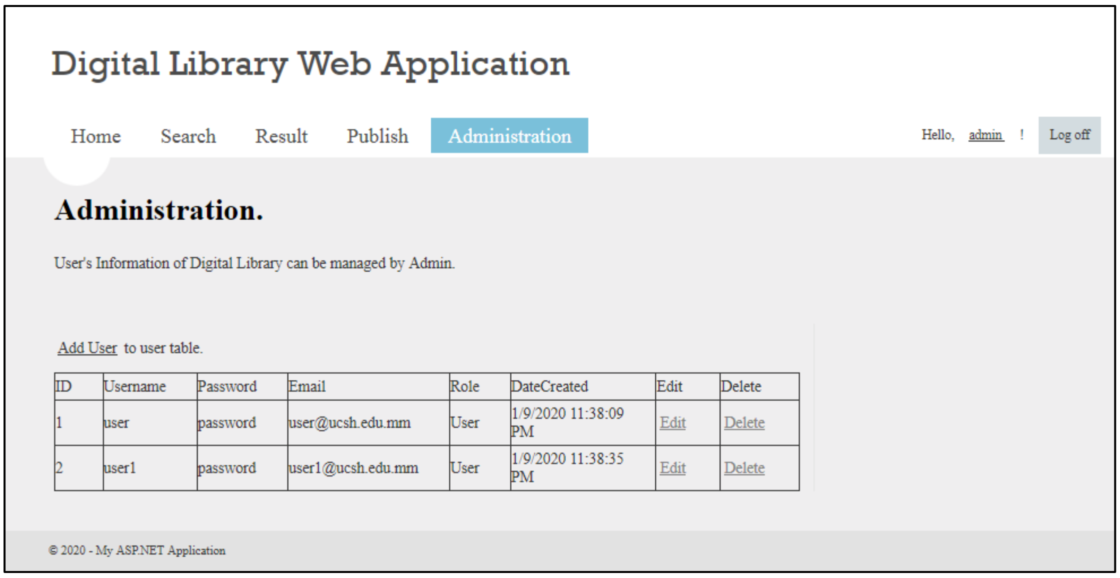
No files selected.

Your file was uploaded successfully.
Fill the metadata values of resource and Click Publish.

Title	Introduction to SPARQL
Author(s)	European Commission
Category	Data Mining
Publisher or Journal	OPEN DATA support
Published Date	2014
Type	eBook
Format	.pdf
Volume	
Issue	
Number of Pages	43
Language	English
Identifier (DOI, ISBN, ISSN)	
URL	D:\AAH\PROJECT(ThetThet/
Abstract	By the end of this training m
Content	Presentation metadata Slide Open Data Support is funded by the European Commission under SMART Lot Provision of services for the Publication Access and Reuse of Open Public Data across the European Union through existing open data portals Contract No CE European Commission DATA SUPPORT OPEN Training Module Introduction to RDF SPARQL DATASUPPORT OPEN Learning objectives By the end of this training

Figure 4.21 Publish Page with Metadata Values

The "Administration" menu of the web application is used to manage the accounts of library users. An account consists of information about username, password, email, role, and date created. Admin can define the new user by clicking the link "Add User" in the "Administration" page which is shown in Figure 4.22.



Administration.

User's Information of Digital Library can be managed by Admin.

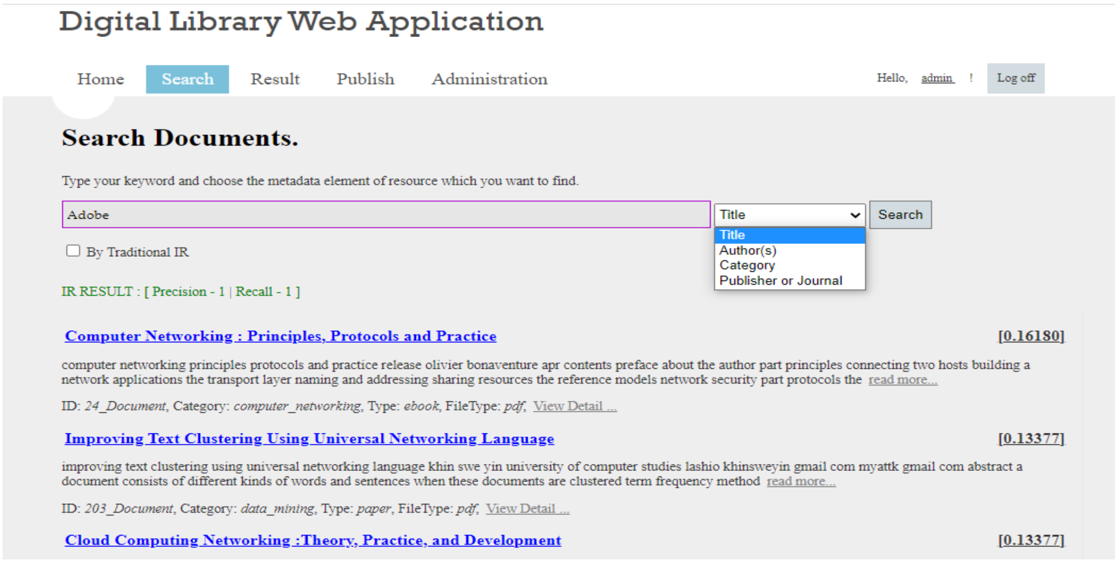
Add User to user table.

ID	Username	Password	Email	Role	DateCreated	Edit	Delete
1	user	password	user@ucsh.edu.mm	User	1/9/2020 11:38:09 PM	Edit	Delete
2	user1	password	user1@ucsh.edu.mm	User	1/9/2020 11:38:35 PM	Edit	Delete

© 2020 - My ASP.NET Application

Figure 4.22 Administration Page with User Information

The main functional page in our Digital Library web application is the "Search" page. On this page, both users and admin can search for different types of documents by given query with the different property of document: Title, Author(s), Category, Publisher or Journal. In Figure 4.23, the retrieved document is shown as a result of searching for document in Author(s) property by given keyword. In the retrieving process, the evaluation of precision and recall is performed for the current search.



Search Documents.

Type your keyword and choose the metadata element of resource which you want to find.

Adobe

By Traditional IR

IR RESULT : [Precision - 1 | Recall - 1]

Computer Networking : Principles, Protocols and Practice [0.16180]

computer networking principles protocols and practice release olivier bonaventure apr contents preface about the author part principles connecting two hosts building a network applications the transport layer naming and addressing sharing resources the reference models network security part protocols the read more...

ID: *24_Document*, Category: *computer_networking*, Type: *ebook*, FileType: *pdf*, View Detail ...

Improving Text Clustering Using Universal Networking Language [0.13377]

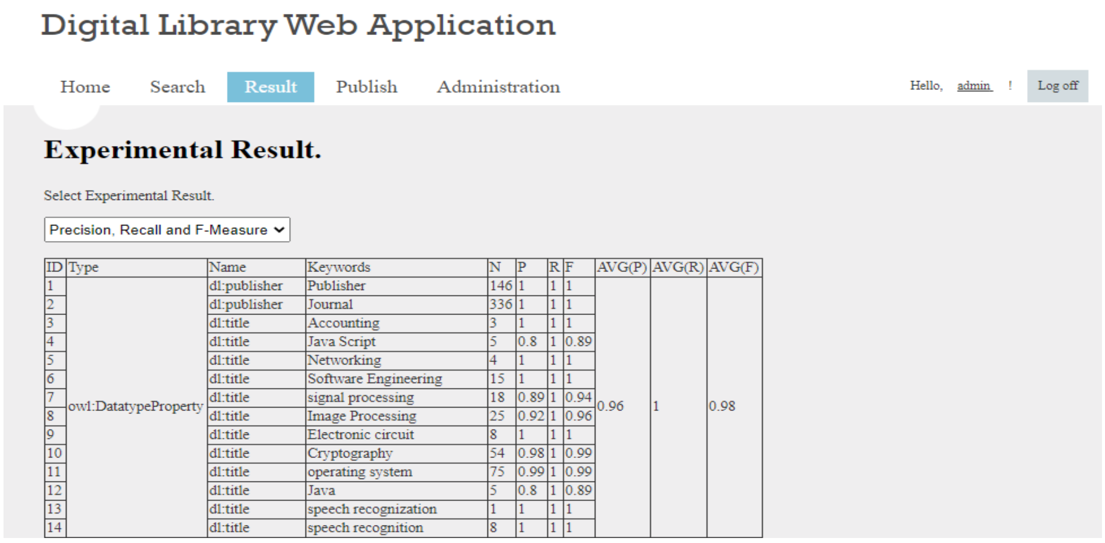
improving text clustering using universal networking language khin swe yin university of computer studies lashio khinsweyin gmail com myattk gmail com abstract a document consists of different kinds of words and sentences when these documents are clustered term frequency method read more...

ID: *203_Document*, Category: *data_mining*, Type: *paper*, FileType: *pdf*, View Detail ...

Cloud Computing Networking :Theory, Practice, and Development [0.13377]

Figure 4.23 Search Page of Digital Library Web Application

The "Result" menu is designed and implemented for displaying the results of IR in detail. These results consist of precision, recall, f-measure. The results for all tested queries are shown on this page. As the result, tested queries are grouped by type of properties: DatatypeProperty and ObjectProperty. The “Result” pages with DataTypeProperty and with ObjectProperty of the web application is shown in Figure 4.24 and Figure 2.25. The table of experimental results and its values will be explained in the performance analysis section.



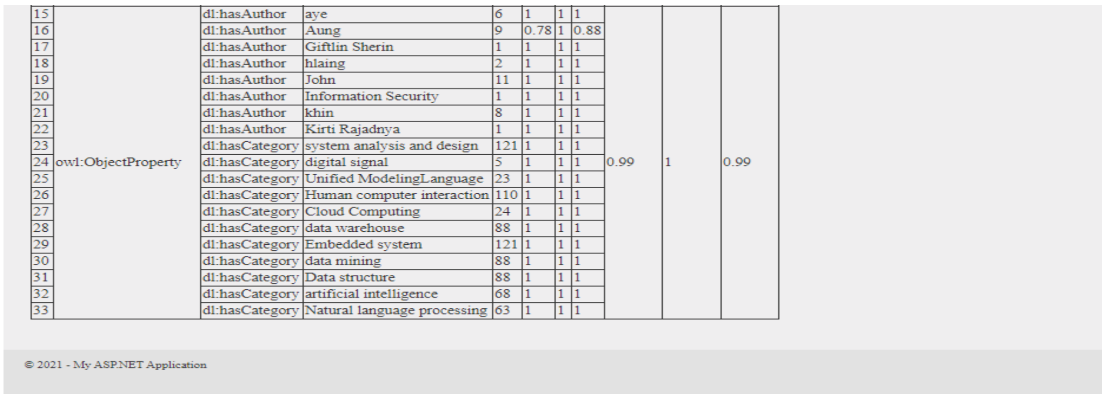
Experimental Result.

Select Experimental Result.

Precision, Recall and F-Measure

ID	Type	Name	Keywords	N	P	R	F	AVG(P)	AVG(R)	AVG(F)
1		dl:publisher	Publisher	146	1	1	1			
2		dl:publisher	Journal	336	1	1	1			
3		dl:title	Accounting	3	1	1	1			
4		dl:title	Java Script	5	0.8	1	0.89			
5		dl:title	Networking	4	1	1	1			
6		dl:title	Software Engineering	15	1	1	1			
7	owl:DatatypeProperty	dl:title	signal processing	18	0.89	1	0.94	0.96	1	0.98
8		dl:title	Image Processing	25	0.92	1	0.96			
9		dl:title	Electronic circuit	8	1	1	1			
10		dl:title	Cryptography	54	0.98	1	0.99			
11		dl:title	operating system	75	0.99	1	0.99			
12		dl:title	Java	5	0.8	1	0.89			
13		dl:title	speech recognization	1	1	1	1			
14		dl:title	speech recognition	8	1	1	1			

Fig 4.24 Result Page with DataTypeProperty of Digital Library Web Application



15		dl:hasAuthor	aye	6	1	1	1			
16		dl:hasAuthor	Aung	9	0.78	1	0.88			
17		dl:hasAuthor	Giftlin Sherin	1	1	1	1			
18		dl:hasAuthor	hlaing	2	1	1	1			
19		dl:hasAuthor	John	11	1	1	1			
20		dl:hasAuthor	Information Security	1	1	1	1			
21		dl:hasAuthor	khin	8	1	1	1			
22		dl:hasAuthor	Kirti Rajadnya	1	1	1	1			
23		dl:hasCategory	system analysis and design	121	1	1	1			
24	owl:ObjectProperty	dl:hasCategory	digital signal	5	1	1	1	0.99	1	0.99
25		dl:hasCategory	Unified ModelingLanguage	23	1	1	1			
26		dl:hasCategory	Human computer interaction	110	1	1	1			
27		dl:hasCategory	Cloud Computing	24	1	1	1			
28		dl:hasCategory	data warehouse	88	1	1	1			
29		dl:hasCategory	Embedded system	121	1	1	1			
30		dl:hasCategory	data mining	88	1	1	1			
31		dl:hasCategory	Data structure	88	1	1	1			
32		dl:hasCategory	artificial intelligence	68	1	1	1			
33		dl:hasCategory	Natural language processing	63	1	1	1			

© 2021 - My ASP.NET Application

Figure 4.25 Result Page with ObjectProperty of Digital Library Web Application

4.4 Performance Analysis

To show the performance of the system, 33 queries for different properties of documents were tested by using 415 training documents that include various file types (.doc, .pdf, .txt). These testing queries are related to Object and Datatype Properties. The training documents are collected from the Google search engine.

To evaluate the performance of Ontology-based IR system for Digital Library, precision, recall, and F-measure methods are used as shown in Equations 4.1, 4.2, and 4.3.

Precision (P)

$$P = TP / (TP+FP) \tag{4.1}$$

Recall (R)

$$R = TP / (TP+FN) \tag{4.2}$$

F-Measure (F)

$$F = 2 * [(P*R) / (P+R)] \tag{4.3}$$

Where TP denotes the number of relevant documents in retrieved documents. FP is the number of non-relevant documents in retrieved documents. FN denotes the number of relevant documents in non-retrieved documents.

Precision is the ability to retrieve top-ranked documents that are most relevant. The recall is the ability of the search to find all of the relevant items in the corpus. This means that the precision is the exactness and the recall is the completeness of the IR system. The f-measure is just a combination of the exactness and completeness of the system.

The precision, recall, and f-measure values of experimental results for the ObjectProperty are shown in Table 4.1.

Table 4.1 Precision, Recall and F-measure Results for ObjectProperty

PropertyName	Kyewords	NofRetrieved	P	R	F
dl:hasAuthor	Information Security	1	1	1	1
dl:hasAuthor	Khin	8	1	1	1
dl:hasAuthor	Kirti Rajadnya	1	1	1	1
dl:hasAuthor	John	11	1	1	1
dl:hasAuthor	aye	6	1	1	1
dl:hasAuthor	Aung	9	0.78	1	0.88
dl:hasAuthor	Giftlin Sherin	1	1	1	1
dl:hasAuthor	hlaing	2	1	1	1
dl:hasAuthor	myo	5	0.8	1	0.89
dl:hasCategory	system analysis and design	121	1	1	1
dl:hasCategory	data mining	88	1	1	1
dl:hasCategory	Unified ModelingLanguage	23	1	1	1
dl:hasCategory	artificial intelligence	68	1	1	1
dl:hasCategory	Human computer Interaction	110	1	1	1
dl:hasCategory	Natural language processing	63	1	1	1
dl:hasCategory	digital signal	5	1	1	1
dl:hasCategory	Embedded system	121	1	1	1
dl:hasCategory	Data structure	88	1	1	1
dl:hasCategory	Cloud Computing	24	1	1	1
dl:hasCategory	Data warehouse	88	1	1	1
AVERAGE			0.98	1	0.99

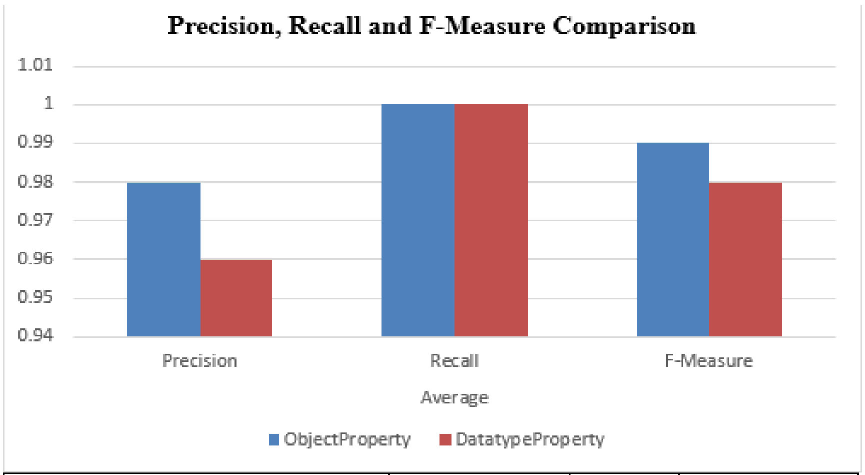
In the above table, the precision (P), recall (R), and f-measure (F) values for four ObjectProperty of documents are shown. The recall for all properties is 1 and the average precision for all properties is 0.98. The average F-measure value is 0.99. According to these results, the exactness and completeness of Ontology-based IR systems in ObjectProperty is over 98%. The precision, recall, and f-measure values of experimental results for the DatatypeProperty are shown in Table 4.2.

Table 4.2 Precision, Recall and F-measure Results for DatatypeProperty

PropertyName	Kyewords	NofRetrieved	P	R	F
dl:publisher	Publisher	146	1	1	1
dl:publisher	Journal	336	1	1	1
dl:title	Accounting	3	1	1	1
dl:title	Java Script	5	0.8	1	0.9
dl:title	Networking	4	1	1	1
dl:title	Software Engineering	15	1	1	1
dl:title	signal processing	18	0.89	1	0.9
dl:title	Image Processing	25	0.92	1	1
dl:title	Electronic circuit	8	1	1	1
dl:title	Cryptography	54	0.98	1	1
dl:title	operating system	75	0.99	1	1
dl:title	Java	5	0.8	1	0.9
dl:title	speech recognization	1	1	1	1
dl:title	speech recognition	8	1	1	1
AVERAGE			0.96	1	0.98

In the above table, the precision (P), recall (R), and f-measure (F) values for four DatatypeProperty of documents are shown. The average precision for all properties is 0.96 and the recall for all properties is 1. The average F-measure value is 0.98. According to these results, the exactness and completeness of Ontology-based IR systems in DatatypeProperty is 96%.

The average results of Ontology-based IR system for ObjectProperty and DatatypeProperty are compared and described with the bar chart in Figure 4.26. According to the comparison results of precision, recall, and f-measure, the Ontology-based IR system is more accurate in ObjectProperty type because the values for this property are all instances of an OWL class.



	Precision	Recall	F-Measure
Object Property	0.98	1	0.99
Datatype Property	0.96	1	0.98

Figure 4.26 Comparison Results of Precision, Recall and F-Measure

To evaluate the performance of proposed system, the processing time of IR is compared with traditional IR system. The processing time of both proposed IR and traditional IR system is recorded in database for each tested query. And then the average value of processing time for both IR systems is calculated and grouped by type of query property. The unit of processing time in our experiment is in milliseconds. The average processing time results for the ObjectProperty are shown in Table 4.3.

Table 4.3 Average Processing Time Results for ObjectProperty

PropertyName	Keywords	Processing Time(ms)	
		Proposed-IR	Traditional-IR
dl:has Author	Information Security	526	1378
dl:has Author	Khin	528	1196
dl:has Author	Kirti Rajadnya	515	1829
dl:has Author	John	568	1492
dl:has Author	aye	501	1340
dl:has Author	Aung	532	1395
dl:has Author	Giftlin Sherin	239	959
dl:has Author	hlaing	465	856
dl:has Author	myo	623	1630
dl:hasCategory	system analysis and design	528	1262
dl:hasCategory	data mining	500	1300
dl:hasCategory	Unified ModelingLanguage	457	1383
dl:hasCategory	artificial intelligence	499	1335
dl:hasCategory	Human computer Interaction	646	6639
dl:hasCategory	Natural language processing	744	6449
dl:hasCategory	digital signal	2500	2795
dl:hasCategory	Embedded system	2660	2925
dl:hasCategory	Data structure	399	1484
dl:hasCategory	Cloud Computing	634	1377
dl:hasCategory	Data warehouse	326	1730
AVERAGE		720	2038

As a result, the minimum processing time of proposed IR system for ObjectProperty queries is 239 milliseconds and the maximum is 2500 milliseconds. The maximum processing time of traditional IR system for ObjectProperty queries is 720 milliseconds. According to the comparison result of average processing time which shown in Table 4.3, the proposed IR system is more than two times faster than the traditional IR system in searching for ObjectProperty type queries.

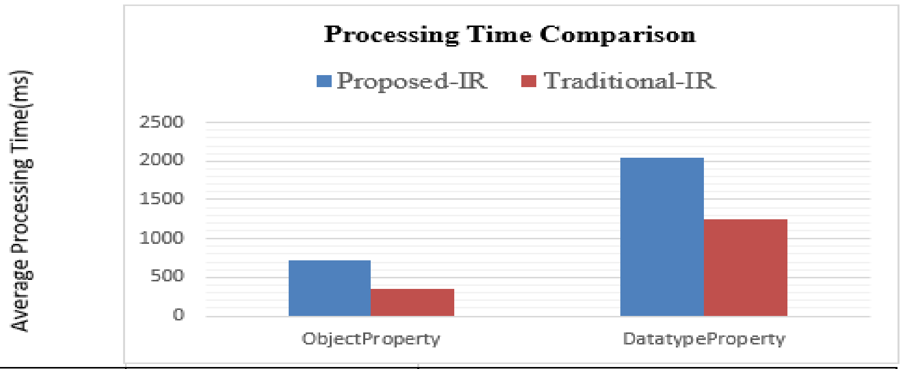
The average processing time results for the DatatypeProperty are shown in Table 4.4. As a result, the minimum processing time of proposed IR system for DatatypeProperty queries is 233 milliseconds and the maximum is 2660 milliseconds. The maximum processing time of traditional IR system for DatatypeProperty queries is 6639 milliseconds. The average value of processing time for our proposed system is 720 milliseconds and traditional IR system is 2038 milliseconds. According to the comparison result of average processing time which shown in Table 4.4, the proposed IR system is more than three times faster than the traditional IR system in searching for DatatypeProperty type queries.

The processing time comparison result of both IR systems for ObjectProperty and DatatypeProperty queries are described with the bar chart in Figure 4.19. The average processing time of proposed IR system for ObjectProperty queries is 499 milliseconds and DatatypeProperty queries is 610 milliseconds. According to this comparison results, the proposed Ontology-based IR system is faster in ObjectProperty query than the DatatypeProperty query because the values for this property are all instances of an OWL class.

Table 4.4 Average Processing Time Results for DatatypeProperty

PropertyName	Keywords	Processing Time(ms)	
		Proposed-IR	Traditional-IR
dl:publisher	Publisher	233	1664
dl:publisher	Journal	239	1648
dl:title	Accounting	257	1619
dl:title	Java Script	510	1668
dl:title	Networking	280	1104
dl:title	Software Engineering	241	891
dl:title	signal processing	440	973
dl:title	Image Processing	337	943
dl:title	Electronic circuit	351	1109
dl:title	Cryptography	365	965
dl:title	operating system	276	1150
dl:title	Java	250	1491
dl:title	speech recognization	856	1090
dl:title	speech recognition	266	1058
AVERAGE		350	1241

The average processing time of Ontology-based IR system for ObjectProperty and DatatypeProperty are compared and described with the bar chart in Figure 4.27. According to the comparison results of Proposed-IR and Traditional-IR, the Ontology-based IR system with objectProperty is faster than in Datatype Property type.



	ObjectProperty	DatatypeProperty
Proposed-IR	720	2038
Traditional-IR	350	1241

Figure 4.27 Processing Time Comparison for Proposed and Traditional IR

4.5 Summary

The overview design diagram of the Ontology-based information retrieval system is presented in this chapter. The proposed system is implemented as the IR system by using Domain Ontology. The main point of proposed IR system is the formatting of SPARQL query and context matching process by using SPARQL query. The Ontology-based IR system for Digital Library is implemented based on Service-Oriented Architecture (SOA) by using the XML based web service technology and ASP.NET. The architecture of the proposed system consists of file storage for documents, one ontology dataset, and two programming components: web service and web application. To show the performance of the system, 33 queries for different properties of documents were tested by using 415 training documents. To evaluate the performance of Ontology-based IR system for Digital Library, precision, recall, and F-measure methods are used. According to the comparison results of precision, recall, and f-measure, the Ontology-based IR system is more accurate in ObjectProperty type and also ObjectProperty is faster than DatatypeProperty in processing time with miliseconds.

CHAPTER 5
CONCLUSION AND FURTHER EXTENSIONS

This thesis intends to develop to retrieve documents from the Ontology dataset not only by the keywords but also by the metadata of documents. The various components of this system are investigated and their contributions to the overall performance of the system are analyzed. In this chapter, the main contents of the thesis are concluded, advantages and limitations of the system, and future work are suggested.

5.1 Conclusion

The proposed system presents the implementation of Ontology-based information retrieval for Digital Library. This system introduces a system that users can use to retrieve digital documents from the Ontology dataset. The ontology method is used to represent the context model based on digital library resources. Ontology plays a key role in the evolution of digital libraries. In interoperability at the semantic level, context-sensitive query processing over heterogeneous information resources requires the matching of concepts. The system presents the available heterogeneous information sources and improves the accuracy of information retrieval using semantic web technology. In addition, the system can help users to reduce the consuming time for surfing the information they wanted.

5.2 Advantages and Limitations of the System

The proposed system serves user-friendly, high-performance, and scalable semantic search for information from the digital library. As a result, the Ontology-based IR system is more accurate in searching for ObjectProperty type.

Information retrieval by SPARQL query produces exact results; in the case of keyword search, it produces all results containing keywords including non-relevant documents. The exactness and completeness of the IR system are proved by the average value of F-measure which obtains over 95%.

Moreover, the use of Ontology for Digital Library is more flexible and interchangeable than the use of Relational Databases. It provides a chance to extend and define metadata for other resources easily without modifying the implementation.

However, our proposed IR model doesn’t support to transform the user query in natural language into SPARQL format. And also, it provides to search for only digital documents.

5.3 Further Extensions

The proposed system is tested by using only the dataset with document resources. The dataset can be extended with multimedia resources, such as video, audio, and others, by modifying the Digital Library Ontology. Obtaining a better result in the formatting of SPARQL query is a motivation for further research work such as the development of an algorithm to transform the natural language query to SPARQL.

AUTHOR’S PUBLICATIONS

[1] Thet Thet Aung, Khin Lay Myint, Hlaing Htake Khaung Tin,”Ontology based Information Retrieval System for Digital Library”, to be published in the National Journal of Parallel and Soft Computing(NJPSC 2021), Yangon, Myanmar, 2021.

[2] Thet Thet Aung, Myat Mon Khaing, Khin Shin Thant, Hlaing Htake Khaung Tin, “Senitment Analysis On Who Southeast Asia Region Organization (Who Sero) User Comment Review And Opinion Mining”, International E-Conference In Association with International Engineering Journal For Research& Development, ICIPPS June 2020, E-ISSN NO:- 2349-0721, IMPACT FACTOR : SJIF - 6.549, pp. 314-321.

[3] Thet Thet Aung, Khin Shin Thant, Myat Mon Khaing, “Finding Out the Possible Benefits Using Social Media for Higher Education During the Epidemic Covid-19”, University Journal of ICT in Multidisciplinary Issues on Arts & Science, Engineering, Economics and Education, June, UJIM 2020, Hinthada, Myanmar, ISBN 978-99971-0-872-2, Volume 1, Issue 1, pp. 59-63.

[4] Khin Shin Thant, Thet Thet Aung, Hlaing Htake Khaung Tin, “Exploring Better Teaching Methods By Surveying from Computer Science Students (Case Study for Software Engineering)”, International Journal of Research & Innobation in Applied Science, IJRIAS May 2020, EISSN 2454-6194, Volume V, Issue V, pp. 93-96.

[5] Khaing Thazin New, Lwin Sandar Soe, Thet Thet Aung, “Effective Concept and Components of 4ps Marketing Information System”, International Journal of Research and Scientific Innovation (IJRSI), June 2020, ISSN No. 2321-2705, Volume VII, Issue VI, pp. 30-33.

[6] Khin Shin Thant, Thet Thet Aung, Myat Mon Khaing, Hlaing Htake Khaung Tin, “Comparison Of Loan Finance System Of Fullerton Myanmar And Woori Finance Myanmar”, International E-Conference In Association with International Engineering Journal For Research& Development, ICIPPS June 2020, E-ISSN NO:- 2349-0721, IMPACT FACTOR : SJIF - 6.549, pp. 339-344.

REFERENCES

[7] Apache Software Foundation (2020). SPARQL Tutorial. Access in https://jena.apache.org/tutorials/sparql.html

[8] B. Liu, “Web Data Mining”, 2nd Edition, Springer-Verlag Berlin Heidelberg, New York, (2011).

[9] Berners-Lee, T. (1997). Axioms of Web Architecture: Metadata. Access in http://www.w3.org/DesignIssues/Metadata.html 05/032014

[10] Bethesda, MD, Metadata Types and Functions, NISO. (2004) Understanding Metadata. From https://marciazeng.slis.kent.edu/metadatabasics/types.htm Access in 04/08/2020

[11] C. Xia, X. Wang, “Graph-Based Web Query Classification”, Proceedings of the 12th Web Information System and Application Conference IEEE, (2016) 241-244.

[12] Casarosa, V. (2010). A Conceptual Model for Digital Libraries. Elag 2007 Conference, Barcelona, From http://elag2007.upf.edu/papers/casarosa.pdf Access in 23/05/2014

[13] D. Hiemstra, “Information Retrieval Models”, Information Retrieval: Searching in the 21st Century, (2009) 1-18.

[14] Dan, Munteanu. (2007). Vector space model for document representation in information retrieval. The Annals of “Dunarea de Jos”, University of Galati Fascicle III Electrotechnics Electronics Automatic Control and Informatics, 43-46, December 2007.

[15] Deborah L. McGuinness, Frank. (2004). OWL Web Ontology Language-Overview, W3C Recommendation, 10 February 2004. From https://www.w3.org/TR/owl-features/ access in 23/01/2020

[16] DLF (1998). A working definition of digital library. From http://old.diglib.org/about/dldefinition.htm

[17] Dr. Glöckner, “Fuzzy Information Retrieval”, Seminar Presentation, University Hagen, Department of Computer Science, (2005).

[18] Dr. Varshil Bhagaji Dashrath, Role of Metadata in Digital Resource Management, International Journal of Digital Library Services (IJODLS), 4(3): 209-217, 2014.

[19] Ferran, N. and Minguillón, E. (2005). Towards personalization in digital libraries through ontologies. Library Management Vol. 26 No. 4/5, 2005 pp. 206-217. From http://eprints.rclis.org/9179/1/nferranf_LMJ.pdf Access in 26/05/2014

[20] Isah, A.; Mutshewa, A.; Serema, B. & Kenosi, L. (2013). Digital Libraries: Analysis of Delos Reference Model and 5S Theory. Journal of Information Science Theory and Practice 1(4): 38-47, 2013. From https://www.researchgate.net/publication/263624200_Digital_Libraries_Analysis_of_Delos_Reference_Model_and_5S_Theory Access in 04/08/2020

[21] K. M. Risvik, "Scaling Internet Search Engines: Methods and Analysis", Ph.D. Thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, (2004) 1-172.

[22] Matthew Horridge (2011). A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools, Edition 1.3, The University of Manchester, 24 March.

[23] Matthew Horridge1, Holger Knublauch (2004). A Practical Guide to Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools, Edition 1.0, The University of Manchester, 27 August.

[24] Metadata. (2014).Wikipedia, the free encyclopedia. Access in http://en.wikipedia.org/wiki/Metadata

[25] Nilesh Shewale , Dr. J. Shivarama (2018). Ontology based digital library search system for enhanced information retrieval in engineering domain, International Journal of Advanced Research, Ideas And Innovations In Technology (IJARIIT), Vol. 4, No. 5, ISSN: 2454-132X

[26] Ontology for Digital Library, Master Thesis, International Master in Digital Library Learning, (2015). From https://oda.hioa.no/nb/ontologies-for-digital-libraries/asset/dspace:10524/Tavares.pdf , Access in 04/08/2020

[27] Protégé, A free, open-source ontology editor and framework for building intelligent systems. From https://protege.stanford.edu/ Access in 04/08/2020.

[28] RDF Basics – Capsicum, What is RDF, 02 July 2018, From https://rebeccabilbro.github.io/rdf-basics/ Access in 04/08/2020

[29] Ruban S, Kedar Tendolkar, Austin Peter Rodrigues, Niriksha Shetty (2014). An Ontology-Based Information Retrieval Model for Domesticated Plants. International Journal of Innovative Research in Computer and Communication

Engineering (IJIRCCE), Vol. 2, No. 5, ISSN(Online): 2320-9801, ISSN(Print): 2320-9798

[30] S. J. Cunningham et al., “Applying connectionist models to information retrieval”, Brain-Like Computing and Intelligent Information Systems, (1999) 435-457.

[31] S. Prakash, HR Shashidhara, G.T.Raju, “The Role of an Information Retrieval in the Current Era of Vast Computer Science Stream”, International Journal of Soft Computing and Engineering (IJSCE), 3(3) (2013) 139-143.

[32] Salton, Gerard. Introduction to Modern Information Retrieval. McGraw-Hill

[33] Sawsaa, A. and Lu, J. (2014). Building an Advance Domain Ontology Model of Information Science (OIS), International Journal of Digital Information and Wireless Communications (IJDIWC) 4(2): 90-98.The Society of Digital Information and Wireless Communications, 2014.

[34] Shaimaa Salama, Mahmoud Adb Ellatif, Marwa Hosny Hassan (2017). Ontology Based Information Retrieval Model for Semantic Research Digital Library (SEMRDL), International Journal Of Computer Science and Information Security(IJCSIS), Vol. 15, No. 5, ISSN: 1947-5500

[35] SPARQL Query Language for RDF, W3C Semantic Web, SPARQL Working Group, 21 March 2013. From https://www.w3.org/2001/sw/wiki/SPARQL Access in 04/08/2020

[36] SPARQL Tutorial – Filters, Apache Software Foundation. From https://jena.apache.org/tutorials/sparql_filters.html Access in 04/08/2020.

[37] Steve Harris, Garlik, Andy Seaborne (2013). SPARQL 1.1 Query Language, W3C Recommendation 21 March 2013. From https://www.w3.org/TR/sparql11-query/ Access in 07/07/2020

[38] Vangie Beal, OWL - Ontology Web Language, Webopedia. From https://www.webopedia.com/TERM/O/Ontology_Web_Language.html Access in 04/08/2020.

[39] What is Digital Library, IGI Global. From https://www.igi-global.com/dictionary , Access in 04/08/2020