Exploiting Linked Data and Big Data for Semantic Patent Discovery

COEN 296
Introduction to the Cloud Computing – Instructor Professor Dr. Ming-Hwa Wang

Team Project by

Sathappan Subramanian, Sushil Dhilpe, and Udaya Yalamanchi

August 2013
Abstract

About this document

This document discusses how the Big Data can be easily exploited through various Linked Open Data databases, to present consolidated, meaningful and elaborate information of a Patent or an Inventor. The approach in this document uses similar concept of searching & interlinking documents on World Wide Web i.e. a webpage only human can understand, but now this information on the webpage is machine readable and can be treated a legitimate data. In other words, we want to use Linked Data to enrich the content. This concept is also implemented in various Linked Open Data databases thus making a powerful tool for a user and a computer that is looking for just about the right information.

Enter the Semantic Web technologies of RDF and SPARQL, which create a pathway for cost-effective, efficient, and powerful data analysis.

Audience

This document is intended for all the students, data scientist and analyst who want to explore the Value of Big Data
Document Revision Level

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>August 2013</td>
<td>Initial release of the Document for internal review</td>
</tr>
<tr>
<td>0.2</td>
<td>September 2013</td>
<td>Initial release of the Document for internal review &amp; submission</td>
</tr>
</tbody>
</table>

Contributors and Acknowledgement

The information included in this document represents the expertise, feedback, and suggestions of a number of professionals. The authors would like to recognize and sincerely thank Professor Dr. Ming-Hwa Wang of Santa Clara University.
Table of Contents

Chapter 1. Introduction .................................................................................................................. 5
Chapter 2. Theoretical Basis and Literature Review ................................................................. 7
Chapter 3. Hypothesis .................................................................................................................... 9
Chapter 4. Methodology ............................................................................................................... 10
Chapter 5. Implementation ......................................................................................................... 16
Chapter 6. Data Analysis and Discussion ................................................................................... 27
Chapter 7. Conclusions and Recommendations ...................................................................... 29
Chapter 8. Bibliography ............................................................................................................. 30
Chapter 9. Appendices ............................................................................................................... 31
Abbreviations and Acronyms ..................................................................................................... 32

Table of Figures

Figure 1. ........................................................................................................................................... 13
Figure 2. .......................................................................................................................................... 13
Figure 3. .......................................................................................................................................... 14
Figure 4. .......................................................................................................................................... 14
Figure 5. .......................................................................................................................................... 15
Figure 6. .......................................................................................................................................... 25
Figure 7. .......................................................................................................................................... 25
Figure 8. .......................................................................................................................................... 26
Chapter 1. Introduction

Objective

Big Data is commonly-used term to describe data that exceeds the processing capacity of conventional database systems and is about our ability to exploit that information. The Value of Big Data any organization is looking for is in analytical use, enabling new products and optimize actionable outcome.

According to report from Cisco, by 2016, global Internet traffic is expected to reach a staggering 1.3 Zetta-bytes annually. To put that into perspective, it's the equivalent of 38 million DVDs per hour. This data explosion shows no sign of abatement and is likely to accelerate with new data types (in both senses: record structure; and social as well as transactional) together with greater access to networked devices (such as smart meters and smartphones with geo-positioning data).

Optimal approach and solution to Big Data for an organization or enterprise is to use Linked Data as the reference for Big Data systems, to query against, providing analysis, search (patent in our case) and alerting. The current approaches for Big Data emphasize the ability to deal with Volume and Velocity. But our objective is to work from a different approach, interlinking Variety of Big Data and creating Valuable context using Linked Open Data (LOD) sources that already exists in the Linked Data Cloud. To name few Linked Open Data sources data sets, Semantic Web Dog Food, DBpedia, DBTune, and RKBExplorer. To keep our scope of investigation just to one or two, we have used DBpedia LOD to search against patent or an inventor query from Patent database and present Valuable information in a web user interface.

Due to enormous resources available on Linked Data, we created our own data in Resource Description Framework (RDF) format such as relational data or XML data (Structured data and has its own schema). Here in our case, we have attempted to convert existing patent data into Web-of-Data compliant (RDF) format, by assigning Uniform Resource Identifiers (URIs) to entities that creates Patent Semantic Web or Patent Graph.

The class curriculum around Big Data addressed the importance of mining big data and harnessing valuable information from the meaningful data. This project exploits the attribute, the Value of big data.

In reading various papers and web resources around this topic, we found some challenges such as extending or contributing back the patent related information because the integrated patent information lacked the conversion of patent database into RDF format and hence couldn’t interlink to Linked Data cloud for public consumption. In this paper, we will address the optimal solution, by converting the patent data into RDF format and making it available for public consumption. We explore the issues around ranking since our query in this approach is specific.
Statement of the problem

Provide patent database available from various data sources in Resource Description Framework format and interlinking this patent data to the Linked Open Data Cloud, delivering enriched content & relationship of a specific user query that is both human and machine readable.

Area or scope of investigation

1. Interlinking patent data in Linked Open Data (LOD) Cloud
2. Use similar methodology to connect to other LOD by using different data sources from the paper
3. Storing the RDF data back in local or linked data cloud
4. Although this paper addresses limited scope of interlinking patent data to Linked Open Data Cloud, Enterprise can use this methodology to relate their traditional databases to linked data
Chapter 2. Theoretical Basis and Literature Review

Definition of the problem

Provide patent database available from various data sources in Resource Description Framework format and interlinking this patent data to the Linked Open Data Cloud, delivering enriched content & relationship of a specific user query that is both human and machine readable.

Theoretical background of the problem

Since the patent database spans almost over three century, the size of this database is huge. There are many places to search for individual patents -- the US Patent and Trademark Office (USPTO) and Google Patent Search are two examples. But sometimes that’s not enough. If you’re trying to identify trends in innovation over time or analyze all the patents relevant to your invention, it helps to have all the patent related information handy and even more relate similar inventions and inventors through Linked Data. The trouble is this information is raw and in size of terabytes (TB) of data. The only way to deliver and circulate this information is on DVDs and other physical media. Despite USPTO and Google have attempted to make this data available, the conversion to RDF format and linking back in the Linked Data Cloud will be major undertaking but will present the information that is not just human readable but made available for machine consumption as well.

If time permits, we will attempt to link such information in the Linked Data and make it available to all the students, data scientists, research communities etc.

Related research to solve the problem

Following papers and web resources studies in this area made an attempt to resolve the problem –

2. Learning from Linked Open Data Usage: Patterns & Metrics – By Knud Moller, Michael Hausenblas, Richard Cyganiak, Siegfried Handschuh, Web Science Conference
3. Searching RDF Graphs with SPARQL and Keywords
4. Towards Big Linked Data: A Large-scale, Distributed Semantic Data Storage
5. Browsing Unbounded Social, Linked Data Instances with User Perspectives – by Hyun Namgoong, Sungkwon Yang. Mina Song, Hong-Gee Kim
6. LinkedLab: A Linked Data Platform for Research Communities – by Fariz Darai and Ruli Manurang
Advantage – The researched papers mentioned earlier provide readily available source for patent database that we could download as a local database in our development environment and query against it. Another advantage described in the papers clearly mention how by publishing patent data in Linked Open Data Cloud provides a good justification for a change from data warehousing/mining culture to extrapolating the Value from the Big Data that many organizations, data scientists, and new inventors are looking for.

Disadvantage – The research fell short of converting patent database into RDF format and could not exploit Linked Data and interlink multiple relationship to a specified query

How to mitigate this problem

We will describe in detail our approach to mitigate this issue in subsequent chapters. We are trying to come with an optimal solution, by converting the patent data into RDF format and making is available for public consumption. We explore the issues around ranking since our query in this approach is specific.

Our solution will attempt to convert patent database into RDF format and take advantage of Linked data resources/properties to display most valuable information of a specified query by a user or a machine. This is something other researcher didn’t attempt in their project.

Our solution if deployed in large scale, will be eventually made public to all the research communities to access various patents and its relationship with other inventions, inventors etc. This will eventually benefit organization in reducing the cost that is usually incurred in disputing litigations over patents and inventions. Organizations can redirect their Research and Development cost elsewhere.
Chapter 3. Hypothesis

The goal of our project was to convert patent data into Resource Description Framework format and interlink with Linked Data Cloud to generate meaningful enriched content that will be web-data compliant. Due time constraints, we planned to simulate the large data set size conversion of patent data into RDF format and interlink this data in look alike Linked Data Cloud.

There are multiple hypothesis listed below

1. Convert the unstructured patent big data into structured RDF format that is readable to both human and machine
2. Linking the patent data to Linked Data within next refresh cycle time of DBpedia
3. Making the patent data for public consumption, although this may come with certain restrictions on how the data is written back in Linked Open Data cloud
Chapter 4. Methodology

We have categorized our methodology and our approach in following five steps

1. Search for objects in our system via online collection DBpedia
2. Reference our objects via sqlite in patent database
3. Combine the results with our web form using SPARQL and Jena API queries
5. Connect the application to our SPARQL endpoint

Pre-requisite

- Collect input data from [https://github.com/funginstitute/downloads](https://github.com/funginstitute/downloads)
The above source is from Berkeley University and can download related patent data sets in sqlite3 form. Due to limited scope of our project, we are using three databases from the nine databases listed below

  - patent.sqlite3
  - class.sqlite3
  - inventor.sqlite3
  - patdesc.sqlite3
  - invpat.sqlite3
  - assignee.sqlite3
  - citation.sqlite3
  - lawyer.sqlite3

Alternatively, you can also downloads patent data from USPTO Patent data source link - [http://www.google.com/googlebooks/uspto-patents.html](http://www.google.com/googlebooks/uspto-patents.html)

- Access to Linked Open Data i.e. DBpedia, from the development environment
DBpedia.org is a community effort to extract structured information from Wikipedia and to make this information available on the Web. DBpedia allows you to ask sophisticated queries against Wikipedia and to link other datasets on the Web to Wikipedia data
Here we assume that the WikiPedia page for requested first name and last name exists

Search for objects in the system via online collection DBpedia

Access the linked data from dbpedia.org e.g [http://live.dbpedia.org/page/Larry_Page](http://live.dbpedia.org/page/Larry_Page)

Reference the objects via sqlite in patent database

Collect the data from Fung Institute from SQLite/pgadmin III to RDBMS format. Next, define Schema for patent data and convert into RDF so we can integrate both sources of the data together.
Combine the results with our web form using SPARQL and Jena API queries


Browse examples of our data using Wikipedia

Validate the data using Wikipedia from the fields our web interface such birth date, alma mater etc.

Connect the application to our SPARQL endpoint

Finally, reference the objects via URIs using D2RQ that convert the final results in RDF format.
Algorithm design

Steps –
1. Collect and Validate the end user input
2. Use different error conditions if not within specified requirements
3. Connect to DBpedia.org
4. Connect to Patent database
5. Use SPARQL and Jena API to combine the results
6. Combine/link the results
7. Present the linked data to the end user

Language and Tools

Java, SQL, SPARQL

**Java** is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture.

**SQL** (Structured Query Language is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS). Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language and a data manipulation language.

**SPARQL** is similar to SQL but this is used to query Linked data which is stored as RDF.

Linux Server, PostgresSQL, Apache Tomcat, Apache Jena, Eclipse, JSP, Google refine, SQLite, SQL/JDBC, Hibernate, Enterprise Architect etc.
High Level Use Case Diagram

Block diagram in Figure 1 describes complete structure of Patent Data Query

![Diagram](image)

Figure 1.

The contents of the web page mainly include two query results from DBpedia and Patent Database. The structure of the user interface is that patent data is in the middle and surrounded by basic information of the patent inventor, like biography, picture if available and his advisor or students.

Validate the hypothesis

After the results are generated, query independently from DBpedia, Patent Data, WikiPedia and D2RQ server page to match the results.

- The results from our local Patent database to a specific query is shown in Figure 2.

![Figure 2](image)
• The results from the linked data DBpedia database to a specific query is shown in Figure 3

http://live.dbpedia.org/page/Larry_Page

Figure 3.

• The results from the combined results in RDF format to a specific query is shown in Figure 4

Figure 4.
The results as seen from our web interface to a specific query is shown in Figure 5.

Figure 5.
Chapter 5. Implementation

Sample Code (refer detailed programming details in the appendix)

- **Web form and User Interface sample code**

```jsp
<%@ page import="patent.*" %>
<%@ page import="rdf.*" %>
<%@ page import="java.util.*" %>

<%

LinkedPatentData lpData = new LinkedPatentData();
LinkedData linkedData = null;
List <PatentData> patentList = null;

String aliasFirst = "";
String firstName = "";
String lastName = "";

String cmd = request.getParameter("action");
try {
    if ("Search".equals(cmd)) {
        aliasFirst = request.getParameter("aliasfirst");
        firstName = request.getParameter("firstname");
        lastName = request.getParameter("lastname");
        linkedData = lpData.getLinkedData(aliasFirst, lastName);
        patentList = lpData.getPatentData(firstName, lastName);
    }
} catch(Exception e) {
    e.printStackTrace();
}
%

<head>
<title>Patent & Linked Data.</title>
</head>

<body bgcolor="white">
<br>
<form method=POST action=linkedpatent.jsp>
    <font size=4> First Name :<input type=text name="firstname" value="<%=firstName%>" size=20>
</font> <font size=4> Alias First Name :<input type=text name="aliasfirst" value="<%=aliasFirst%>"size=20>
</font> <font size=4> Last Name :<input type=text name="lastname" value="<%=lastName%>"size=20>
</font> <input type=submit name=action value="Search">
</form>
```
<hr>

<table border="1" style="width: 700px;">
  <tr>
    <td height="200px" valign="top" width="200px"><img src="<%=linkedData==null? "" : linkedData.getDepiction()%>" height="200px" width="200px"></td>
    <td valign="top">
      <table border="0">
        <tr>
          <td>Birth Name:</td>
          <td><%=linkedData==null ? "" : linkedData.getBirthName()%></td>
        </tr>
        <tr>
          <td>Birth Date:</td>
          <td><%=linkedData==null ? "" : linkedData.getBirthDate()%></td>
        </tr>
        <tr>
          <td>Alma Mater:</td>
          <td><%=linkedData==null ? "" : linkedData.getAlmaMater()%></td>
        </tr>
        <tr>
          <td>Known for:</td>
          <td><%=linkedData==null ? "" : linkedData.getKnownFor()%></td>
        </tr>
        <tr>
          <td>Total Patents:</td>
          <td><%=patentList==null ? "" : patentList.size()%></td>
        </tr>
      </table>
    </td>
  </tr>
</table>

<br>

<table border="1" style="width: 1000px;">
  <tr>
    <th valign="top" width="10%">Patent Id</th>
    <th valign="top" width="30%">Title</th>
    <th valign="top" width="40%">Abstract</th>
    <th valign="top" width="10%">Application Date</th>
    <th valign="top" width="10%">Grant Date</th>
  </tr>
  <% if (patentList != null) {
    for (PatentData patentData : patentList) { %>
    <tr>
      <td><%=patentData.getPatentId()%></td>
      <td><%=patentData.getTitle()%></td>
      <td><%=patentData.getAbstract()%></td>
      <td><%=patentData.getApplicationDate()%></td>
      <td><%=patentData.getGrantDate()%></td>
    </tr>
  <% } %>
</table>
• Patent linked database collection

package patent;

// Generated Aug 24, 2013 11:22:26 AM by Hibernate Tools 4.0.0

import java.util.List;
import javax.naming.InitialContext;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.hibernate.LockMode;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

import static org.hibernate.criterion.Example.create;

/**
 * Home object for domain model class Inventor.
 * @see patent.Inventor
 * @author Hibernate Tools
 */

public class InventorHome {

    private static final Log log = LogFactory.getLog(InventorHome.class);

    private final SessionFactory sessionFactory = getSessionFactory();

    protected SessionFactory getSessionFactory() {
        try {
            //return (SessionFactory) new InitialContext().lookup("SessionFactory");
            return new Configuration().configure().buildSessionFactory();
        } catch (Exception e) {
            log.error("Could not locate SessionFactory in JNDI", e);
            e.printStackTrace();
            throw new IllegalStateException("Could not locate SessionFactory in JNDI");
        }
    }
}
public void persist(Inventor transientInstance) {
    log.debug("persisting Inventor instance");
    try {
        sessionFactory.getCurrentSession().persist(transientInstance);
        log.debug("persist successful");
    } catch (RuntimeException re) {
        log.error("persist failed", re);
        throw re;
    }
}

public void attachDirty(Inventor instance) {
    log.debug("attaching dirty Inventor instance");
    try {
        sessionFactory.getCurrentSession().saveOrUpdate(instance);
        log.debug("attach successful");
    } catch (RuntimeException re) {
        log.error("attach failed", re);
        throw re;
    }
}

public void attachClean(Inventor instance) {
    log.debug("attaching clean Inventor instance");
    try {
        sessionFactory.getCurrentSession().lock(instance, LockMode.NONE);
        log.debug("attach successful");
    } catch (RuntimeException re) {
        log.error("attach failed", re);
        throw re;
    }
}

public void delete(Inventor persistentInstance) {
    log.debug("deleting Inventor instance");
    try {
        sessionFactory.getCurrentSession().delete(persistentInstance);
        log.debug("delete successful");
    } catch (RuntimeException re) {
        log.error("delete failed", re);
        throw re;
    }
}

public Inventor merge(Inventor detachedInstance) {
    log.debug("merging Inventor instance");
    try {
        Inventor result = (Inventor) sessionFactory.getCurrentSession().merge(detachedInstance);
        log.debug("merge successful");
        return result;
    } catch (RuntimeException re) {
        log.error("merge failed", re);
    }
}
public Inventor findById(patent.InventorId id) {
    log.debug("getting Inventor instance with id: " + id);
    try {
        Inventor instance = (Inventor) sessionFactory.getCurrentSession().get("patent.Inventor", id);
        if (instance == null) {
            log.debug("get successful, no instance found");
        } else {
            log.debug("get successful, instance found");
        }
        return instance;
    } catch (RuntimeException re) {
        log.error("get failed", re);
        throw re;
    }
}

public List<Inventor> findByPatent(String patentId) {
    log.debug("getting Inventor instance with patent id: " + patentId);
    try {
        List<Inventor> results = (List<Inventor>) sessionFactory.openSession().createQuery("from Inventor where patent = ?").setString(0, patentId).list();
        log.debug("find by example successful, result size: " + results.size());
        return results;
    } catch (RuntimeException re) {
        log.error("get failed", re);
        throw re;
    }
}

public List<Inventor> findByFristNameLastName(String firstName, String lastName) {
    log.debug("getting Inventors with " + firstName + ":" + lastName);
    try {
        List<Inventor> results = (List<Inventor>) sessionFactory.openSession().createQuery("from Inventor where firstname like ? and lastname like ?").setString(0, firstName).setString(1, lastName).list();
        log.debug("find by example successful, result size: " + results.size());
        return results;
    } catch (RuntimeException re) {
        log.error("get failed", re);
    }
}

public List<Inventor> findByExample(Inventor instance) {
    log.debug("finding Inventor instance by example");
    try {
        List<Inventor> results = (List<Inventor>) sessionFactory
            .getCurrentSession().createCriteria("patent.Inventor")
            .add(create(instance)).list();
        log.debug("find by example successful, result size: "
            + results.size());
        return results;
    } catch (RuntimeException re) {
        log.error("find by example failed", re);
        throw re;
    }
}

• Façade process of patent database

package patent;
import java.util.ArrayList;
import java.util.List;
import rdf.LinkedData;
import rdf.RDFDataCollector;

public class LinkedPatentData {
    private InventorHome inventorHome;
    private PatentHome patentHome;
    private PatdescHome patdescHome;

    public LinkedPatentData() {
        inventorHome = new InventorHome();
        patentHome = new PatentHome();
        patdescHome = new PatdescHome();
    }

    public List<PatentData> getPatentData(String firstName, String lastName) {
        List<PatentData> result = new ArrayList<PatentData>();
        List<Inventor> inventors = inventorHome.findByFristNameLastName(firstName + "%",
            lastName);
        String[] patents = new String[inventors.size()];
        int i = 0;
        for (Inventor inventor : inventors) {
            InventorId id = inventor.getId();
patents[i++] = id.getPatent().trim();

List<Patent> patentList = patentHome.findByPatentIds(patents);
for (Patent patent : patentList) {
  if (patent == null)
    continue;
  PatdescId pdescId = patdescHome.findByPatentId(patent.getId().getPatent()).getId();
  PatentData pd = new PatentData();
  pd.setPatentId(patent.getId().getPatent());
  pd.setApplicationDate(patent.getId().getAppdate());
  pd.setGrantDate(patent.getId().getGdate());
  pd.setAbstract(pdescId.getAbstract_());
  pd.setTitle(pdescId.getTitle());
  result.add(pd);
}
return result;
}

public LinkedData getLinkedData(String firstName, String lastName) {
  RDFDataCollector rdfCollector = new RDFDataCollector(firstName, lastName);
  rdfCollector.collectFromDBPedia();
  return rdfCollector.getLinkedData();
}

public static void main(String[] args) {
  // get RDF data
  LinkedData linkedData = linkedPatentData.getLinkedData("Larry", "Page");
  System.out.println(linkedData.toString());

  for (PatentData patentData : patentDataList) {
    System.out.println(patentData);
    System.out.println("\n");
  }
}
}
RDF Data collection and parser from DBpedia

```java
package rdf;

import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.util.FileManager;

public class RDFDataCollector {

    private String firstName;
    private String lastName;
    private LinkedData linkedData;

    public RDFDataCollector(String _firstName, String _lastName) {
        firstName = _firstName;
        lastName = _lastName;
        linkedData = new LinkedData(_firstName, _lastName);
    }

    public void collectFromDBPedia(){
        FileManager fManager = FileManager.get();
        fManager.addLocatorURL();
        String url = "http://dbpedia.org/data/" + firstName + "_" + lastName + ".rdf";
        Model model = fManager.loadModel(url);
        linkedData.setBirthDate(RDFDataParser.parseDBPedia(model, "http://dbpedia.org/ontology/birthDate"));
        linkedData.setBirthName(RDFDataParser.parseDBPedia(model, "http://dbpedia.org/property/birthName"));
        linkedData.setAlmaMater(RDFDataParser.parseDBPedia(model, "http://dbpedia.org/property/almaMater"));
        linkedData.setKnownFor(RDFDataParser.parseDBPedia(model, "http://dbpedia.org/property/knownFor"));
        linkedData.setDepiction(RDFDataParser.parseFOAFDepiction(model));
    }

    public LinkedData getLinkedData() {
        return linkedData;
    }
}
```

```java
package rdf;

import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.SimpleSelector;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.StmtIterator;
import com.hp.hpl.jena.sparql.vocabulary.FOAF;
import com.hp.hpl.jena.rdf.model.NodeIterator;
```
public class RDFDataParser {
    // to get dbpedia properties
    public static String parseDBPedia(Model model, String property) {
        String propertyValue = "";
        Property p = model.createProperty(property, "");
        StmtIterator iter =
            model.listStatements(new SimpleSelector(null, p, (RDFNode) null));
        while (iter.hasNext()) {
            Statement stmt = iter.nextStatement();
            if (stmt.getObject().isLiteral()) {
                Literal obj = (Literal) stmt.getObject();
                if (propertyValue.length() > 0)
                    propertyValue += ", ";
                propertyValue += obj.getString();
            }
        }
        return propertyValue;
    }

    // image
    public static String parseFOAFDepiction(Model model) {
        String foafValue = "";
        NodeIterator iter = model.listObjectsOfProperty(FOAF.depiction);
        while (iter.hasNext()) {
            RDFNode r = iter.next();
            if (foafValue.length() > 0) break;
            foafValue += r.toString();
        }
        return foafValue;
    }
}

- **Translate the combined results into RDF format using D2RQ**

ExtraUsing RDF conversion tools such as D2RQ, we present the combined result data into Resource Description Framework format, which can be eventually used for interlinking in the Linked Data Cloud.

D2R Server is a tool for publishing the content of relational databases on the Semantic Web, a global information space consisting of Linked Data. Data on the Semantic Web is modeled and represented in RDF. D2R Server uses a customizable D2RQ mapping to map database content into this format, and allows the RDF data to be browsed and searched – the two main access paradigms to the Semantic Web.

Requests from the Web are rewritten into SQL queries via the mapping. This on-the-fly translation allows publishing of RDF from large live databases and eliminates the need for replicating the data into a dedicated RDF triple store.
Design document and flowchart

- Following figure 6 is the use case diagram of patent linked data

![Use Case Diagram](image)

**Figure 6.**

- Following diagram is a high level sequence diagram

![Sequence Diagram](image)

**Figure 7.**
• Diagram 8 explains the relations of all the classes and objects we used

Figure 8.
Chapter 6. Data Analysis and Discussion

Output generation & analysis

- The output is generated as previously displayed in Figure 5, shows full query of the inventor and the related patents
- To analyze our output, we manually tested the data from multiple sources.
- For personal information, we verified the data from Wikipedia & DBpedia websites
- For Patent information, we executed SQL queries against our local patent database and Google patent search

Compare the output against hypothesis

The goal of our project was to convert patent data into Resource Description Framework format and interlink with Linked Data Cloud to generate meaningful enriched content that will be web-data compliant. Due time constraints, we planned to simulate the large data set size conversion of patent data into RDF format and interlink this data in look alike Linked Data Cloud.

There are multiple hypothesis listed below

1. Convert the unstructured patent big data into structured RDF format that is readable to both human and machine
2. Linking the patent data to Linked Data within next refresh cycle time of DBpedia
3. Making the patent data for public consumption, although this may come with certain restrictions on how the data is written back in Linked Open Data cloud

Abnormal case explanation (the most important task)

There were few instances we couldn’t achieve the expected results from the linked data DBpedia database to a specific query due to many factors involved while entering a query in the web form. We have listed these few abnormal cases with brief explanation

- Some of the inventor names do not exists in DBpedia database, so querying for those inventors results doesn't provide the linked data
- Some of the inventors has commonly known first name which is different from their original first name. DBpedia and Wikipedia uses commonly known names while the patent database has the original first name, the causes our results not able to link data across multiple sources for such inventors. E.g. Bill Gates original name is William Henry (Bill) Gates III
- We couldn’t display correct size picture of the inventor in the Firefox browser. This caused webpage to skew and hide all our queried results. We fixed the issue immediately in our code
- We were unable to convert our combined results in URI form and ultimately convert in RDF format due to our patent database table required primary key field to query the RDF triples. Entire database was re-indexed and vacuumed before we could convert our results in URI/RDF format
Statistic regression

- This section doesn’t apply to our project

Discussion

Our approach doesn’t have any mathematical model however we verify the output data as previously mentioned through DBpedia and Patent data queries executed separately. The information that we have displayed is just hosted on our development environment but we feel that this information should be made available to the public consumption by hosting on a linked data server with SPARQL end point query. Since we worked on subset (1970 – 2010) of patent data available from USPTO, considering the huge amount of patent data available, this data should have been hosted on Hadoop Cluster.
Chapter 7. Conclusions and Recommendations

Summary and conclusions

The main objective of this project was to query data from two data sources and combining them in Resource Description Framework (structured or machine readable) format. This was achieved on our small development environment since the patent database was huge in size. We made a successful attempt by simulating how the information would look in Linked data cloud and established a framework to deploy such information for public consumption in linked data cloud.

Recommendations for future studies

As mentioned earlier in the area/scope of investigation, the ultimate goal is to establish such framework by interlinking patent data to the Linked Open Data Cloud and propose this approach to the Enterprise or Organizations that rely on big data information. This methodology if proven to be a successful approach, then enterprise can relate their traditional databases to the linked data.

We have tremendous opportunity in exploring other important information available in the linked data from other countries patent database, for example we have patent office from one of the countries in Asia approach our professor for such project deployment. If we are successful, we will attempt to take this as our future study and also explore other Linked Open Databases.
Chapter 8. Bibliography

**Books, White papers, Tech journals:**


2. Learning from Linked Open Data Usage: Patterns & Metrics – By Knud Moller, Michael Hausenblas, Richard Cyganiak, Siegfried Handschuh, Web Science Conference April, 2010, Raleigh, NC, USA


5. Browsing Unbounded Social, Linked Data Instances with User Perspectives – by Hyun Namgoong, Sungkwon Yang, Mina Song, Hong-Gee Kim, 2009


**Web resources:**

- [http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData](http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData)
- [http://jena.apache.org/index.html](http://jena.apache.org/index.html)
- [http://www.w3.org/TR/sparql11-query/](http://www.w3.org/TR/sparql11-query/)
- [http://d2rq.org/](http://d2rq.org/)
Chapter 9. Appendices

Program flowchart

Program source code with documentation

Complete program source and related input/output data is submitted to the SCU Design Center Linux Repository
### Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARQL</td>
<td>SPARQL Protocol And RDF Query Language</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>LOD</td>
<td>Linked Open Data</td>
</tr>
<tr>
<td>USPTO</td>
<td>United States Patent &amp; Trademark Office</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
</tbody>
</table>