Using Linked Data to improve the query performance of Patent Data

Xiaoting Yin

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-78
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-78.html

May 16, 2013

DO NOT CIRCULATE

Using Linked Data to improve the query performance of Patent Data

Master of Engineering Student in EECS

Xiaoting Yin

24100321

Outline

Abstract

Tens of thousands of related people, thousands of sensors and millions of transactions now work to create unimaginable amounts of information. People believe there are some values hidden behind these large amounts of data but find it unable to manage, analyze and manipulate these data to their advantage.

One of the main challenges in today's data-driven scenarios is not just the size and complexity of datasets, but the question of how to make sense of big data by combining it with other data and thus create valuable context for the data. Linked Data is one method for combining very rich and freely available public data and using it to be able to answer more advanced analytical questions. The next generation of analytics needs semantic reasoning where expert knowledge is combined with the analytics.

My project is to experiment with the state of the art, as well as current shortcomings, in Big data integration and analysis. We will build a search engine to integrate Public Linked Data into Patent Data to provide more sematic contexts and values to each patent. We build the interface and connection to retrieve the traditional data from the front-end website.

Introduction

One of the main challenges in today's data-driven scenarios is not just the size and complexity of datasets, but the question of how to make sense of big data by combining it with other data and thus create valuable context for the data. Linked Data is one method for combining very rich and freely available public data (such as Wikipedia, Freebase, data.gov, Geonames……), and using it to be able to answer more advanced analytical questions. The Linked Data cloud is growing constantly, and data integration and analysis skills are becoming increasingly important in many fields of science and engineering[1].

Data Integration and Analysis based on Linked Data are facing two major challenges: The first step is to integrate data by converting it to Resource Description Framework (RDF) and making sure that concepts in two previously disconnected datasets are identified and mapped (entity resolution and matching). The second step is to identify those analysis tasks that can be solved within the technical framework of Linked Data (using SPARQL)[2], or to figure out ways how to externalize these tasks into more efficient Big Data analysis frameworks (such as Hadoop or similar frameworks).

The objective of this project is to experiment with the state of the art, as well as current shortcomings, in Big Data integration and analysis, and to go through all the steps required to fuse datasets, properly connect them through entity resolution and matching, and then solve relevant analysis questions either within the framework of

Linked Data, or by using additional Big Data tools. After project completion, we will acquire skills in identifying and solving problems occurring when big datasets need to be connected with public data and then analyzed.

This paper explores the integration of Big Data with Linked Data in two aspects. First, converting Big Data into Linked Data format and apply Linked Data methodologies to solve Big Data problems. Second, see the Linked Data as a public Big Dataset and combine it with the raw/mediate results of Big Data and adding more values. We will go through all these details from cleaning data, which ensures all data could be linked with common property, to conversion different data formats, which defines the semantic relationship among these, and interlink or publish the analysis results using some Data Analysis toolkits.

Literature Review

With the development of many related technologies, data is generated and accumulated in an amazing speed, which exceeds the processing capacity of conventional database systems. People try to come up with corresponding storage architectures and representation approaches to tackle these and dig out the hidden values. In this article, we briefly introduces common techniques in data storage and representation and mainly talks about Linked data, which is our project approach to integrate and make analysis of Big data.

Tens of thousands of related people, thousands of sensors and millions of transactions now work to create unimaginable amounts of information. People believe there are some values hidden behind these large amounts of data but find it unable to manage, analyze and manipulate these data to their advantage. The coming of big data has spawned Data Science. Based on analytics, Data Science produces novel data representations from existing data through translation, integration, transformation, and other techniques.

Data mining is a very useful tool to implement analysis process of large scale amount of structured data in data warehouses. However, many experts wonder that traditional databases could still be effective in terms of handling large amounts of data, sometimes petabyte range. Besides, it is difficult to employ statistical analysis theories to analyze non-numerical data, such as large scales of video or audio files. Hence, people come up with new data storage mechanisms – MongoDB[3], RDF[4],

NoSQL[5][6], etc.

The paper is organized as follows: Section II introduces Big Data, its sources and properties. Section III gives an overview of Big Data representations, tools and analytics. Section IV describes our capstone project and Section V draws the conclusion.

I Big Data

In information technology, big data is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, curation, storage, search, sharing, analysis, and visualization[7].

Big data is characterized by the three *V*s: volume, velocity, and variety. From terabytes to petabytes, the volume of data is encroaching on exabytes. Many experts predict that zetabytes (1021) are reachable within the next several years. Velocity is not only relative to the fast changing speed of the data we already have, but also is concerned with how fast we gather data. Some companies collect data at the rate of multiple petabytes per year. Some companies have stored data that changes at the speed of terabytes per year. The variety of data (structured data or unstructured data) is continually evolving. Twitter did not exist just a few years ago. But nowadays multiple petabytes of Twitter data have been generated[8].

II Related Work

This session will cover current approaches into new schemes for data representation in petabyte ranges. There are various big data storage and computation architectures, such as: HDFS[9], MapReduce[10], NoSQL databases (e.g. Hive, Pig, Zookeeper[11][12] etc), etc. These tools provide efficient access to large blocks of data.

Representation approaches are the following: XML, Resource Description Framework (RDF), micro-formats. Among these RDF is the most popular and widely accepted by some large search engine companies, like Google and Yahoo. RDF represents data as triples and describes the semantics of the RDF tags in the triple representation[13]. Triple stores can accumulate data in terabytes range but is hard to be scaled into petabyte range.

How to design the RDF graphs is the first problem in developing RDF triple stores, e.g., how to define the RDF tags so they are meaningful in a specific scenario. In the early 1970s, John Sowa from IBM, came up with conceptual graphs and used it to visualize a problem, which is changeable to an RDF description.

One of the main challenges in today's data-driven scenarios is not just the size and complexity of datasets, but the question of how to make sense of Big data by combining with other data and thus create valuable context for the data. Linked Data is one method for combining very rich and freely available public data(such as Wikipedia, Freebase, data.gov, etc) and using it to be able to answer more advanced analytical questions. Linked data is a graph and can also be represented by RDF[14]. In Linked Data, the subject and the object of the triple are nodes and the predicate of the

triple is an edge label. With mathematical graph theory, abundant metrics can be derived from the graph structure. These include metrics such as single shortest path, reachability and centrality and provided useful information for problem-solving. The Linked data cloud is growing constantly, and data integration and analysis skills are becoming increasingly important in many fields of science and engineering[15].

Graph analytics is a popular class of algorithms that takes RDF graphs as input and gives some useful hints to solve problems. Google and Yahoo are the best examples of graph analytics search engines.

Data integration and Analysis based on Linked data are facing two major challenges[16]: The first step is to integrate data by converting it to RDF (the Linked data model) and making sure that concepts in two previously disconnected datasets are identified and mapped (entity resolution and matching). The second step is to identify those analysis tasks that can be solved within the technical framework of Linked data (using SPARQL). or to figure out ways how to externalize these tasks into more efficient Big data analysis frameworks, such as Hadoop[17].

Usually, big data is processed on parallel systems. People often develop analytics based on MapReduce and MPI - two systems. Hadoop has grown into a basic infrastructure to support both MPI and MapReduce algorithms. Advanced analytics try to gain some useful hints from large scale data sets.

III Our Project

Based on this literature review, we can easily find that the next generation of analytics needs semantic reasoning where expert knowledge is combined with the analytics.

The objective of this project is to experiment with the state of the art, as well as current shortcomings, in Big data integration and analysis, and to go through all the steps required to fuse datasets, properly connect them through entity resolution and matching, and then solve relevant analysis questions either within the framework of Linked data, or by using additional Big data tools.

IV Conclusion

It is always difficult to make sense of Big data with respect to its size and complexity. Thus people came up with various storage and computation architectures and representation approaches. Among the storage and computation architectures, Distributed file systems stand out due to its parallel property and Linked data get most attention because of more values embedded in the graph structures. Our project is to integrate Big data with Linked data and organize Big data or meaningful information in Linked data format.

Design and implementation (Methodology)

In our project, we used Patent Data as our dataset. This dataset includes all inventor names from the U.S. utility patent database, from 1975 to the end of 2010, which is convenient to uniquely identify inventor careers. We first extracted some small pieces of Big Data and tried to go through all steps to convert Big Data into Linked Data format(RDF) to explore the pros and cons of the conversion procedure. Then we want to add more values by adding Public Linked Data to Big Data(Patent Data), we can combine Linked Data search with Big Data query to provide more semantic context and reduce the ambiguity for Patent Data query. In convenience for our evaluation and deliverable of this new search strategy, we also designed a website interface to display the search results. In this paper, I mainly introduce the steps to convert Big Data into Linked Data and how to build the connection with Patent Data database and retrieve the query results.

I Converting Big Data into Linked Data

1. Tidy all patent data to ensure they could be linked with the common property – location and delete other columns that unrelated to this scenario.
2. Then, we need to use Google refine to convert those data into RDF (choose RDF namespace prefix and define the semantic relationship between different data).
3. Selecting Vocabularies

There is no definitive directory that can be consulted to find suitable vocabularies and ontologies. We want to apply the following criteria to select the vocabulary:

(1) Usage and uptake – is the vocabulary in widespread usage? Will using this vocabulary make a data set more or less accessible to existing Linked Data applications?

(2) Coverage – does the vocabulary cover enough of the data set to justify adopting its terms and ontological commitments?

4. Making Links within a Data Set

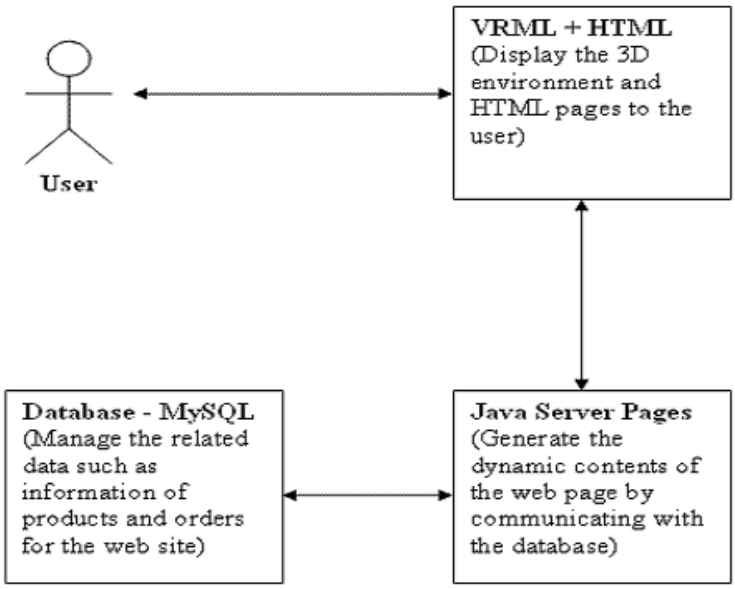
We should create links within and between data sets. Both aspects are essential in ensuring that a data set is integrated with the Web at large

5. Setting RDF Links Manually

Since our data set is not very large, we choose manual interlinking approach.

Once target data sets have been identified, these can be manually searched to find the URIs of target resources for linking. If a data source doesn't provide a search interface, such as a SPARQL endpoint or a HTML Web form, a Linked Data browser can be used to explore the data set and find the relevant URIs.

II New Search Strategy – Patent Data Database Query



The above picture describes the whole structure of Patent Data Query from the front-end website to the back-end database. In this paper, I mainly introduce the information query and retrieval regarding to the Patent Data.

1. Design the schema for the converted Patent Data in the back-end storage
2. Build the connection between the website interface with the backend Patent Data storage.

We build an interface which is mainly responsible for communications with database, process SQL queries, update and return corresponding results[18]. For every request, we design a corresponding method to deal with it, e.g. getCon() is used to connect to the back-end database, doDelete() is used to delete some data from the database. For getConnection method, we apply DriverManager interface to obtain a local connection instance. To avoid SQLException here, we use try-catch solution to connect to the URI-identified database and identify usernames and passwords. For

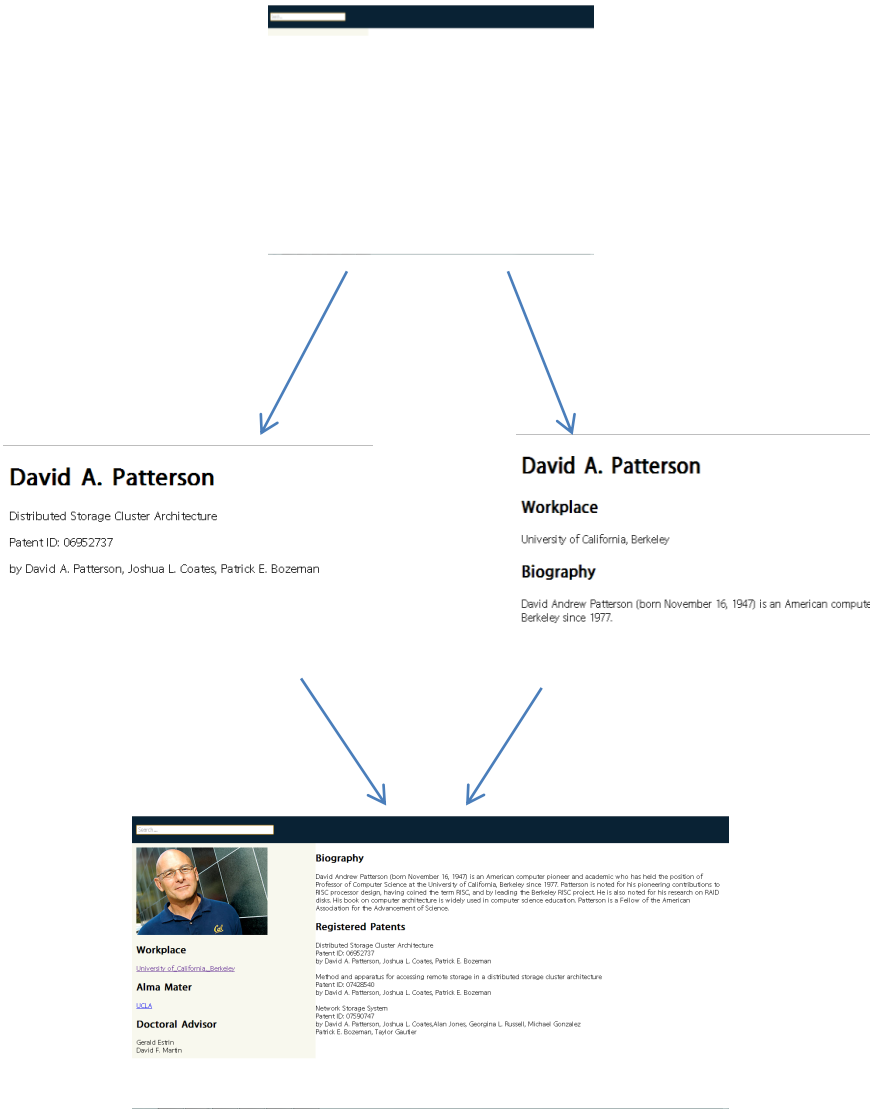
doDelete method, we use a PreparedStatement interface to process SQL queries. We designed several parameters, like –sql and params, in doDelete method to keep track of the main goal of each query. The parameter –sql represents the specified criterion usually extracted in the WHERE clause. We can use set methods in PreparedStatement interface to initialize the query criterion and execute method to get the query results.

3. Retrieve the patent data at the front-end by querying the backend storage

After the previous step, we get the satisfied results. First we will encapsulate each result to a separately JavaBean class and then build a List Set to store all these JavaBeans. In this case, we can use Struts2.0 Iterator interface to go through all the results and display them according to their properties.

Discussions

I UnitTest

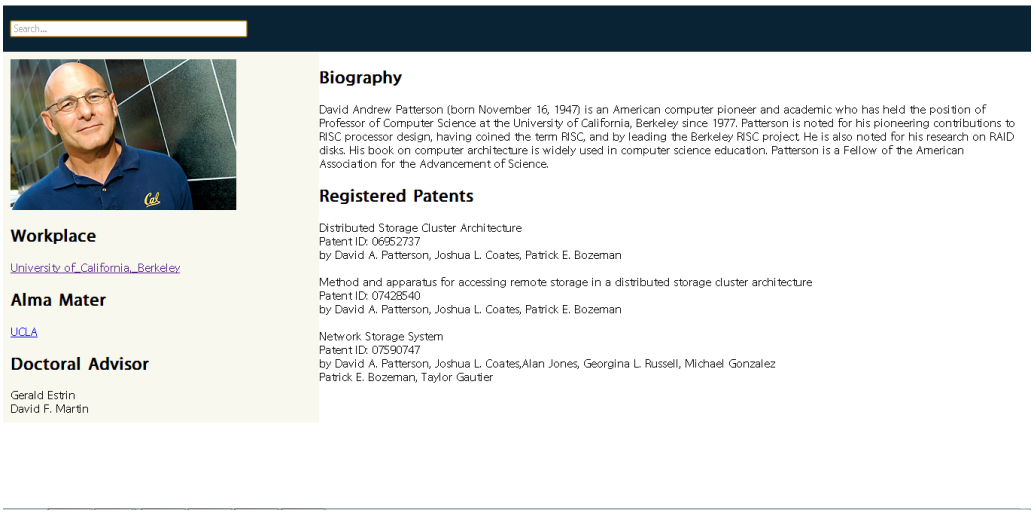


We apply MVC model to build the search engine to query DBpedia and Patent Data. So we apply Unit Test to check whether the connections between models with views and models with controllers work well. In this step, we only just build some simple web pages to testify given some specific query input, like faculty professor or some other persons who own a patent, the model can extract the input and open the

connection with local data base or send the HTTP query to the remote data cloud and display the result on a new web page.

II Explanation of Result

The contents of the return 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 and his advisor or students. The screenshot is the below:



The left side includes the professor's workplace, his advisor and students. Some of the results are web links which users may find potentially interesting to look through. This is a start point of the application of Linked Data. It can help you find more information centered around your original query. Starting from that point, you can change your following query criteria and search more information underlying relative. The middle side includes query results from Patent Database. We can retrieve the stored patent information, like patent number and name in the local database.

III Evaluation

We first want to talk about two approaches. The first approach is converting Patent Data into Linked Data and doing patent search just in remote linked data cloud. The second approach is querying both in remote linked cloud and local data base simultaneously. We tried the first approach: converting the patent data into RDF format, integrating these converted patent data into the linked data cloud and then querying just in the remote linked data cloud. In this approach, the data conversion is the key step and we used Google refine to implement this conversion. But we find it is not feasible to convert large amounts of data for the following reasons: Google refine requires manual labor to detect the dirty data and then clean data. It takes around one hour to clean 50 MB data. Moreover, it is not the case to deal with big data. Google refine can’t hold large amounts of data, like GigaByte scale and then convert these big data. Therefore, it seems unrealistic to convert all the big data into RDF and combine with other public Linked data to dig out more values.

Then we want to talk about the scalability problem of the second approach. Actually it is not very time-efficient to query both remotely and locally. In the remote data cloud, the single query requires information retrieval across different servers and this is very time expensive. However in the local data base query, the size of data is small compared with the amount of total data in the remote end. Sometimes the local query results are ready and just wait for the response from the remote query. In this case, we can get the query results in parallel but combine different data sources in serial.

IV Limitation

In this part, we want to talk about the accuracy of the approach combining Linked Data query with local database query. Since the DBpedia linked data cloud is extracted from Wikipedia database, the names of inventors include the full name (first name, middle name and last name). If the user doesn't input the full name, it will bring some context ambiguity to the query criteria. In this case, our patent data search engine is not as intelligent as Google Patent search because Google also integrates some page rank algorithms to give a prediction of what's the specific meaning of an ambiguous query input. We tried around 50 query inputs and the statistic accuracy and time complexity is the following:

Accuracy	73%
Time complexity/each query	0.3 ms

V Future work

The Patent Data search is used to explore whether the approach of combining query in Linked Data cloud and local database is feasible and bring more values than just individually query. Since we have validated this feasibility, we can make use of different kinds of public data to provide more contexts and information relevant to our local database, like Freebase, Geonames, etc.

To make our patent data search engine more user friendly, I think we can explore more page rank algorithms and integrate them to filter the query results after local

database search. With page rank algorithms, we will be less restricted to user input and provide some search result options based on the statistical information of the search input of all users.

Conclusions

In this project, we build a search engine to integrate Public Linked Data into Patent Data to provide more sematic contexts and values to each patent. We build the interface and connection to retrieve the traditional data from the front-end website. For the next step, we want to try other possible methods to improve the combination of Linked Data and Big Data. We need some Page Rank algorithms to improve the search engine query performance.

Reference

[1] Christian Bizer, Tom Heath and Tim Berners-Lee. 2009. Linked Data – The Story So Far.

[2] Tom Heath and Christian Bizer. 2011. Linked Data: Evolving the Web into a Global Data Space.

[3] Eelco Plugge, Peter Membrey and Tim Hawkins. 2010. The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing.

[4] Michel Klein 2001. XML, RDF, and relatives.

[5] Giuseppe Paterno. 1999. NoSQL Tutorial: A comprehensive look at the NoSQL database.

[6] Robin Hecht and Stefan Jablonski. 2011. NoSQL evaluation: A use case oriented survey.

[7] http://en.wikipedia.org/wiki/Big_data

[8] http://www.businesscomputingworld.co.uk/data-is-exploding-the-3vs-of-big-data

[9] Sanjay Ghemawat , Howard Gobioff and Shun-Tak Leung. 2003. The Google File System.

[10] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters.

[11] Fay Chang, Jeffrey Dean and Sanjay Ghemawat. 2006. Bigtable: A Distributed Storage System for Structured Data.

[12] Giuseppe DeCandia, Deniz Hastorun and Madan Jampani. 2007. Dynamo: Amazon's Highly Available Key-value Store.

[13] Ora Lassila and Ralph R. Swick. 1998. Resource Description Framework (RDF) Model and Syntax Specification.

[14] Christian Bizer, Tom Heath and Tim Berners-Lee. 2009. Linked Data – The Story So Far.

[15] Tom Heath and Christian Bizer. 2011. Linked Data: Evolving the Web into a Global Data Space.

[16] Erik Wilde. 2012. Integrating and Analyzing Big Data with Public Data.

[17] Bastian Quilitz and Ulf Leser. 2008. Querying Distributed RDF Data Sources with SPARQL.

[18] A servlet and JSP Tutorial. http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial

Appendix

Here I will list some code snippets described in methodology.

```
<?php
$q=$_GET["q"];
$con = mysql_connect('localhost', 'root', '');
if (!$con)
   {
   die('Could not connect: ' . mysql_error());
   }
mysql_select_db("PatentData", $con);
$sql="SELECT * FROM Data WHERE Name = '".$q."'";
$result = mysql_query($sql);
// only deal with the first result we found
// id, PatentName, Name
if($row = mysql_fetch_array($result)){
      echo json_encode($row);
}
mysql_close($con);
?>
```