MOOCLink: Building and Utilizing Linked Data from Massive Open Online Courses

Sebastian Kagemann

Arizona State University—Polytechnic Campus

CRA-W Distributed Research Experience

Mesa, Arizona 85212, USA

sakagema@indana.edu

Srividya Kona Bansal

Arizona State University—Polytechnic Campus

Department of Engineering

Mesa, Arizona 85212, USA

srividya.bansal@asu.edu

*Abstract*—Linked Data is an emerging trend on the web with top companies such as Google, Yahoo and Microsoft promoting their own means of marking up data semantically. Despite the increasing prevalence of Linked Data, there are a limited number of applications that implement and take advantage of its capabilities, particularly in the domain of education. We present a project, MOOCLink, which aggregates online courses as Linked Data and utilizes that data in a web application to discover and compare open courseware.

Keywords—linked data; education; ontology building

# Introduction

Linked Data involves using the Web to create typed links between data from different sources [1]. Source data may vary in location, size, subject-matter and how congruously it is structured. Typed links produced from this data create uniformity and define properties explicitly in an effort to make data easier to read for machines. This is opening up opportunities for applications that were previously impractical such as Berners-Lee’s intelligent agents [2].

Linked Data is relatively unexplored in the domain of education. Although there are several data models for structuring educational data as well as repositories adopting these models [3], Linked Data-driven educational applications are far and few between. As a result, initiatives such as the LinkedUp Challenge have surfaced to encourage innovative applications focused on open educational data [4].

Massive Open Online Courses or MOOCs are online courses accessible to anyone on the web. Hundreds of institutions have joined in an effort to make education more accessible by teaming up with MOOC providers such as Coursera and edX [5]. Delivering course content through lecture videos as well as traditional materials such as readings and problem sets, MOOCs encourage interactivity between professors and students around the world by way of discussion forums and graded assessments.

Coursera, a leading MOOC provider, offers a RESTful API [6] for most information associated with their course catalog. This includes properties such as a courses’s title, instructor, and syllabus details. Although Coursera’s course catalog data is easily accessible as JSON, there is no option to retrieve and use it in a Linked Data format such as the Resource Description Framework (RDF). Moreover, there is little to no Linked Data available for MOOCs or an ontology that denotes properties unique to MOOCs.

In order to incorporate MOOC data into the Linked Data cloud as well as demonstrate the potential of Linked Data when applied to education, we propose to (i) build or extend an RDF ontology that denotes MOOC properties and relationships (ii) use our ontology to generate Linked Data from multiple MOOC providers and (iii) implement this data in a practical web application that allows users to discover courses across different MOOC providers.

# Background

## Resource Description Framework

The most notable model for Linked Data is the Resource Description Framework (RDF), which encodes data as subject, predicate, object triples [7]. The subject and object of a triple are both Uniform Resource Identifiers (URIs), while the predicate specifies how the subject and object are related, also using a URI. For the purposes of this paper, Linked Data is presented as RDF/XML, an XML syntax for RDF, which is also used in the implementation of our application.

## Simple Protocol and RDF Query Language

Simple Protocol and RDF Query Language or SPARQL is an RDF query language that allows users to retrieve and manipulate data stored as RDF [8]. SPARQL is used as our application’s query language in order to retrieve data to populate web pages with course information. A SPARQL endpoint for our data can be accessed at http://sebk.me:3030/sparql.tpl.

## Linked Data Principles

Tim Berners-Lee published a set of rules for publishing data on the Web so that data becomes part of a global space in which every resource is connected. The rules are as follows:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When somone looks up a URI, provide useful information (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things

These principles provide a basis for contributing to a Linked Data cloud in which a variety of datasets from different fields of human knowledge are interconnected. Our project aims to abide by these principles.

## Linked Education Data

Many models have been devised for structuring educational data, among the most popular are the IEEE Learning Object Metadata (LOM) specification and Sharable Content Object Reference model (SCORM). LOM is encoded in XML and includes nine categories with sub-elements that hold data. An RDF binding for LOM exists [9], however development is halted at the time of this paper’s writing [10]. SCORM is an extensive technical standard, typically encoded in XML, that defines how educational content should be packaged, how it is delivered, and how learners navigate between different parts of an online course [11]. An RDF binding of SCORM has yet to be developed. Rather than using the defunct LOM binding or creating a new binding of SCORM to RDF, we chose to extend a vocabulary meant for Linked Data, Schema.org, for which an RDF mapping exists [12], to include properties unique to open courseware.

In 2013, the Learning Resource Metadata Initiative (LRMI) specification was incorporated into Schema.org’s vocabulary for tagging educational content [13]. The properties added in this adoption introduced fields for online course details including the type of learning resource, time required, and so on. While there is significant overlap between LRMI’s additions to Schema.org, Schema.org’s Creative Work properties and MOOC course details like those provided in Coursera’s API, several crucial missing data fields such as syllabus details, course difficulty, and predicates linking courses to other objects, make it necessary to extend the vocabulary for MOOC data.

## Building Ontologies

After determining that there was no educational resource ontology that denoted every property needed to create linked MOOC data, we chose to extend Schema.org’s ontology. In order to support uniformity in our data, we use RDF/XML from ontology creation to final data generation. Schema.rdfs.org hosts an RDF/XML version of Schema.org’s ontology, which we imported into Stanford Protégé to extend with additional types and properties.

2014 CRA-W Distributed Research Experience for Undergraduates

# MOOCLink

MOOCLink is a web application which aggregates online courses as Linked Data and utilizes that data to discover and compare online courseware. This section of the paper outlines our approach including choosing our providers, modeling the data, data generation, and the development of our web application.

## MOOC Providers

Coursera is the largest MOOC provider in the world with 7.1 million users in 641 courses from 108 institutions as of April 2014 [14]. These courses span 25 categories including 4 subcategories of computer science. All course details are retrievable by HTTP GET method using Coursera’s RESTful course catalog API and returned in JSON.

edX is another premier MOOC provider with more than 2.5 million users and over 200 courses as of June 2014 [15]. edX courses are distributed among 29 categories, many of which overlap with Coursera’s. edX does not provide an API for accessing their course catalog, however, as of June 2013, edX’s entire platform is open-source.

Udacity, founded by Google VP, Sebastian Thrun, is a vocational course-centric MOOC provider with 1.6 million users in 12 full courses and 26 free courseware as of April 2014 [16]. The majority of Udacity courses are within the field of computer science. Udacity does not provide an API for accessing their course catalog data.

## Data Model

Schema.org is organized as a hierarchy of types, each associated with a set of properties. CreativeWork is Schema.org’s type for generic creative work including books, movies, and now educational resources. The LRMI specification adds properties to CreativeWork including the time it takes to work through a learning resource (timeRequired), the typical age range of the content’s intended audience (typicalAgeRange), as well as specifying properties previously available in Schema.org’s CreativeWork type like the subject of the content (about) and the publisher of the resource (publisher) [17].

CreativeWork provides a base type for our ontology extension, which adds types Course, Session, Category and their associated properties drawn from MOOC data. The extension is made in Stanford Protégé [18], which we use to import the Schema.org vocabulary mapped to RDF at Schema.RDFS.org. In the GUI of Protégé, types and properties are added as well as sample individuals to be used as a model for data generation. The final product is an ontology in which classes are defined using OWL, the Web Ontology Language [19], and in which data and object properties are defined using RDFSchema, which provides basic elements for the description of ontologies [20]. The hierarchy of the ontology extension is outlined in the Implementation section of this paper.

## Data Generation

Coursera’s list of courses is accessible using their RESTful API and JSON as the data exchange format. Using Requests, a Python HTTP library [21], we retrieved a full list of courses, universities, categories, instructors and course sessions in JSON using the GET method.

edX and Udacity do not have an API therefore it became necessary to use a scraper to obtain course data. We used Scrapy, an open-source screen scraping and web crawling framework for Python [22] to retrieve properties from edX and Udacity course pages with XPath selectors. Scrapy supports multiple formats for storing and serializing scraped items; we export our data as JSON to maintain uniformity with the data we retrieve from Coursera.

After collecting the data, we create Linked Data from JSON using Apache Jena, an open source Semantic Web framework for Java [23]. First, we import the ontology model we created in Protégé to retrieve the types and properties to be assigned. We use three methods, one for each course provider, to read JSON specific to each provider (using Google Gson [24]), map property names from JSON to our ontology’s properties, and write each property to the RDF graph. The RDF is output in a flat file, serialized as RDF/XML.

As recommended by Berners-Lee in his Linked Data Principles, each property is assigned a URI. HTTP URIs are used so these resources can be looked up. Because the majority of our data points to URIs within the same RDF, hash URIs are used to identify the local resources. More details on the naming schemes of these URIs are available in the Implementation section of this paper, where our RDF data for Coursera, edX, and Udacity is discussed in detail.

## Web Application

To create an application that is appealing to the eye, we used Bootstrap [25], a responsive HTML, CSS, and JavaScript framework to create a UI consistent on both mobile and desktop devices. Rather than designing each component of the website, we repurposed a business template for displaying online courses. The template incorporates many current web design trends including parallax scrolling and image carousels, which give it an up-to-date look and feel.

Fuseki is a sub-project of Jena that provides an HTTP interface to RDF data. For the web component of our project, we upload our data into a stand-alone Fuseki server from which we can query and update our RDF by <what>. Fuseki includes a built-in version of TDB (Tuple Data Base), which indexes the contents of the file as opposed to storing RDF as a flat file [26].

Google App Engine was initially explored as the web development framework for its automatic scaling and Java support. Unfortunately an incompatibility between App Engine and Apache Jena was found as App Engine supports URL Fetch to access web resources [27] while Jena’s SPARQL processor, ARQ, implements their HTTP requests using their own HTTP operation framework [28]. Additionally, App Engine’s limits on request and response size (10 megabytes and 32 megabytes respectively) as well as their maximum deadline on request handlers (60 seconds) could potentially cause problems if our queries executed slower than anticipated.

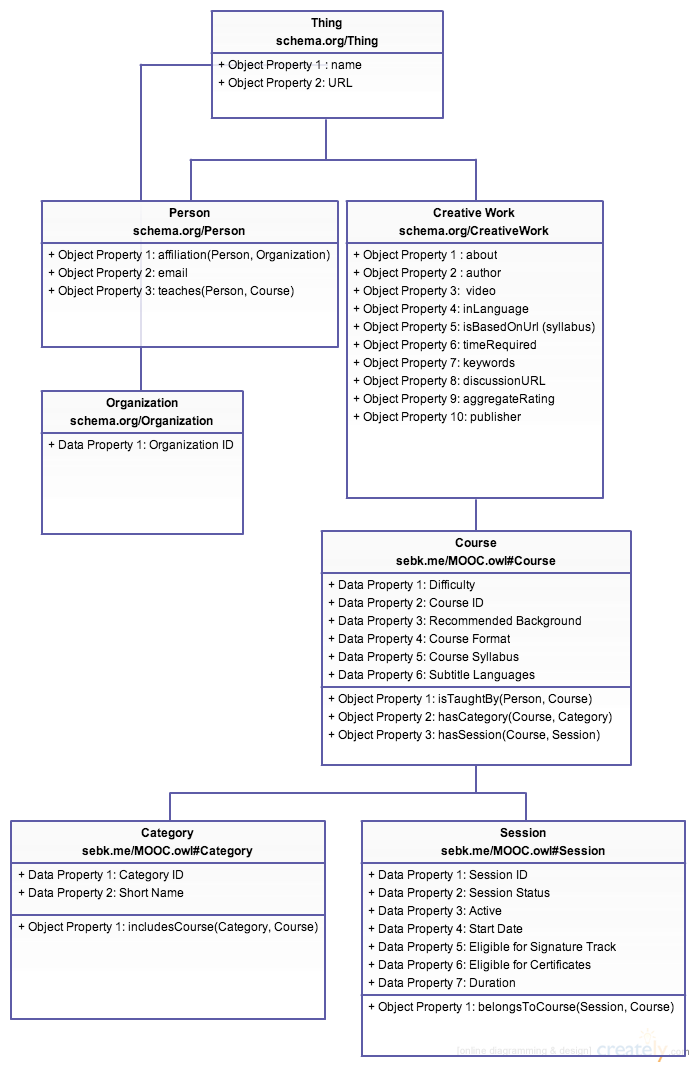
Because of App Engine’s constraints, we opted for a combination of Apache Tomcat as our web server / servlet container and JavaServer Pages (JSPs) and Java Servlets for dynamic web page generation over App Engine’s similar Java solution. Each JSP uses HTML, CSS and JS from the Bootstrap template for the UI. Jena ARQ then queries our RDF on the Fuseki server and writes the course details onto respective pages. Screenshots of the application demo can be found in the Implementation section of this paper.

# Implementation

This section outlines and provides relevant illustrations of (i) our extension of Schema.org’s ontology to include MOOC classes and properties, (ii) RDF data from three different MOOC providers and (iii) our web application.

## Ontology Schema

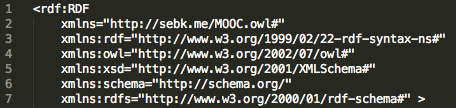
As mentioned in the previous section on our data model, Schema.org is organized as a hierarchy of types. Much like object-oriented programming, types inherit properties. Fig. 1 displays relevant properties from CreativeWork and other Schema.org types although our new types, Course, Session and Category, inherit more properties that are not shown. Sebastian’s personal domain, sebk.me, is used as a temporary namespace for the ontology.



1. Diagram of Schema.org ontology extension.

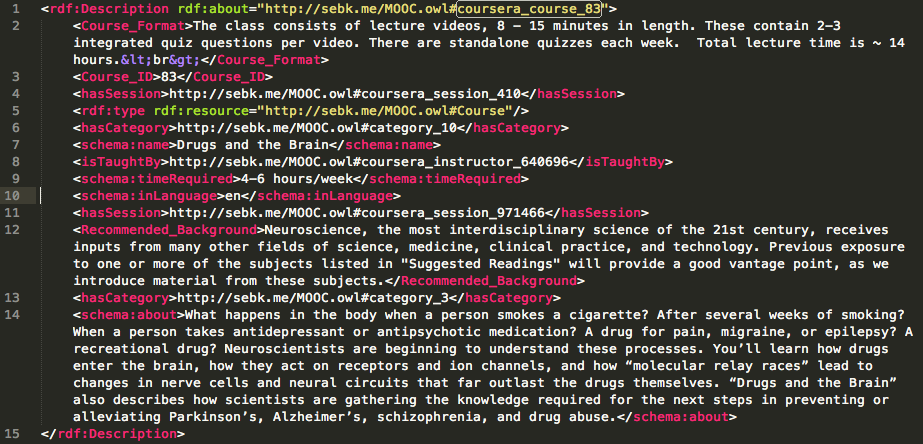
## RDF Data

The first section of our RDF (Fig. 2) invokes the namespaces associated with (line 2) our ontology, (line 3) RDF, (line 4) OWL, (line 5) XMLSchema, (line 6) Schema.org and (line 7) RDFSchema. These namespaces are used as prefixes to abbreviate URIs throughout the data.



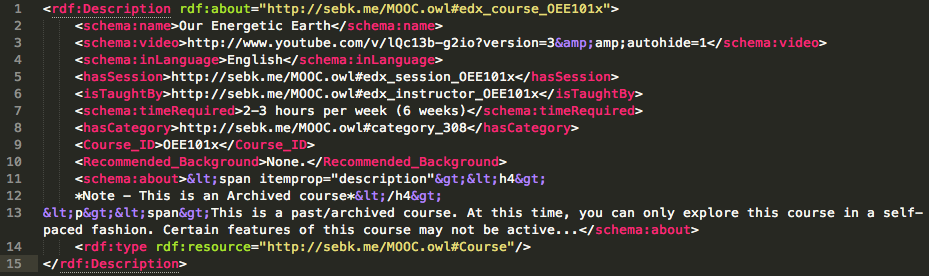
1. RDF/XML namespace declarations

In Figure 3 line 1, Coursera subject URIs (rdf:about) are set to “coursera\_course\_” concatenated with the course ID. They are preceded by the ontology’s namespace explicitly. Line 5 sets the type of resource to Course, a class provided by our MOOC ontology. Data and object properties mapped from Coursera data to the ontology such as Recommended\_Background are assigned values one by one. These properties using a “schema” prefix where the property is inherited from Schema.org and no prefix where the property is borrowed from our ontology. Lines 4 6, and 8 link the course to Coursera sessions, categories and instructors respectively. These linked classes follow a similar subject URI naming scheme to Coursera courses. They are also instantiated in RDF/XML in the same way, but with their own properties.



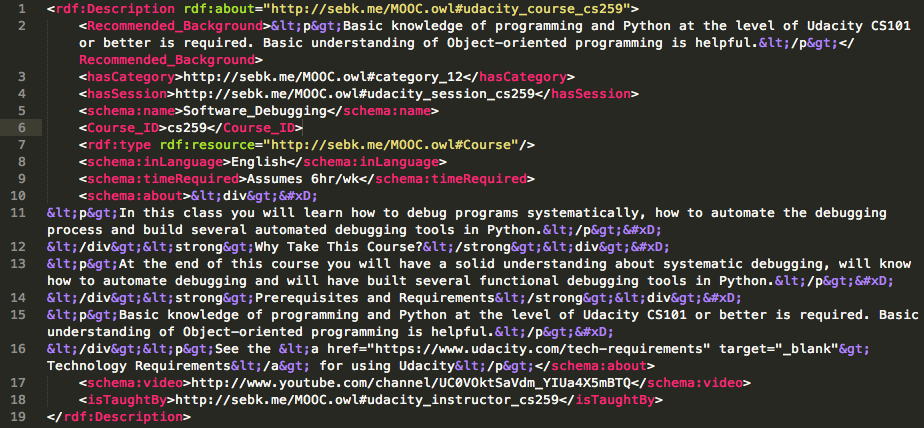
1. Sample of Coursera course data in RDF/XML

edX courses (Fig. 4) follow the same subject URI format as Coursera’s with “edx\_course\_” appended to edX’s mixed character and integer IDs. Note that on line 8 that the category is in the 300s rather than the single or double-digit category URIs seen in the previous RDF snippet. Although most of edX’s courses are mapped to categories drawn from Coursera JSON, some edX categories did not have an equivalent category. As a result, additional edX categories were added to our RDF with IDs starting at 300. Support to map relevant courses from other providers to these new categories is currently not implemented. An additional difference in this data is that the session and instructor (lines 5 and 6) do not have unique IDs. To accommodate this change, we append course IDs to the subject URIs of the edX session and instructor.



1. Sample of edX course data in RDF/XML

## Udacity follows the same subject URI scheme as Coursera and edX but with the course ID provided by Udacity. Similar to edX, instructors and sessions do not have their own IDs. We once again use the course IDs in place of instructor and session IDs for their subject URIs. We do not assign these course IDs to Session\_ID data properties however; this is to avoid any conflicts in future SPARQL queries.



1. Sample of Udacity course data in RDF/XML

## UI Screenshots

Discussion of home page layout. Items in nav bar. Carousel: how courses are picked, how images are retrieved, etc.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Screenshot of MOOCLink home page

Discussion of search results screen. Checkboxes to select courses to compare. More robust semantics and tagging should be incorporated to distinguish between things like Pre-Calculus and Calculus. Current search matches course names based on regular expressions but can be expanded to perform more precise search.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Search results

Discussion of course comparison table.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Course comparison table

Discussion of course pages.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Course details page

# Conclusion and Future Work

We have presented an extension of Schema.org’s Linked Data vocabulary, which incorporates types and properties found in online courses. The extended ontology allows for assignment of data properties relevant to MOOCs as well as object properties which link MOOC sessions, course details, categories and instructors together.

Also presented is our approach to collecting and generating Linked Data from three MOOC providers: Coursera, edX, and Udacity. Using Coursera’s API and two Scrapy crawlers for edX and Udacity, we collect MOOC data in JSON and convert it to RDF with Apache Jena.

We describe a prototype implementation of MOOCLink, a web application which utilizes the Linked MOOC Data to allow users to discover and compare similar online courses. A semantic web stack of Apache Tomcat as the web server and servlet container, Fuseki as the SPARQL server, TDB as the RDF store, and JavaServer Pages and Java Servlets to dynamically create webpages is proposed to achieve this. The functionality as well as the look and feel of the application are highlighted with UI screenshots.

Our future work will focus on: <incorporating demographic data, review section, developing crawlers to retrieve more consistent data, automating website updates, user profiles, course tracks, NLP of syllabi and summaries for more robust data & tagging>.

##### Acknowledgments

I’d like to thank Subway for $5 footlongs and remembering the honey mustard on my oven-roasted chicken.

##### References

1. Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data-the story so far. International journal on semantic web and information systems, 5(3), 1-22. Chicago
2. Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific american*, *284*(5), 28-37.
3. Dietze, S., Yu, H. Q., Giordano, D., Kaldoudi, E., Dovrolis, N., & Taibi, D. (2012, March). Linked Education: interlinking educational Resources and the Web of Data. In Proceedings of the 27th Annual ACM Symposium on Applied Computing (pp. 366-371). ACM.
4. Linked Up Challenge. (n.d.). *LinkedUp Challenge*. Retrieved July 22, 2014, from http://linkedup-challenge.org/
5. Pappano, L. (2012). The Year of the MOOC. The New York Times, 2(12), 2012. Chicago
6. Coursera Catalog API. (n.d.). *- Coursera Technology*. Retrieved July 22, 2014, from https://tech.coursera.org/app-platform/catalog/
7. Klyne, G., & Carroll, J. J. (2006). Resource description framework (RDF): Concepts and abstract syntax.
8. Pérez, J., Arenas, M., & Gutierrez, C. (2006). Semantics and Complexity of SPARQL. In *The Semantic Web-ISWC 2006* (pp. 30-43). Springer Berlin Heidelberg.
9. Nilsson, Mikael, Matthias Palmér, and Jan Brase. "The LOM RDF binding: principles and implementation." Proceedings of the Third Annual ARIADNE conference, Leuven Belgium, 2003. 2003.
10. IEEE Learning Object Metadata RDF binding. (n.d.). *IEEE Learning Object Metadata RDF Binding*. Retrieved July 22, 2014, from http://kmr.nada.kth.se/static/ims/md-lomrdf.html
11. Bohl, O., Scheuhase, J., Sengler, R., & Winand, U. (2002, December). The sharable content object reference model (SCORM)-a critical review. In Computers in education, 2002. proceedings. international conference on (pp. 950-951). IEEE. Chicago
12. Schema.rdfs.org: a mapping of Schema.org to RDF. (n.d.). *- Home*. Retrieved July 22, 2014, from http://schema.rdfs.org/
13. Learning Resource Metadata Initiative. (2013, April 9). *:: World’s Leading Search Engines Recognize LRMI as Education Metadata Standard*. Retrieved July 22, 2014, from http://www.lrmi.net/worlds-leading-search-engines-recognize-lrmi-as-education-metadata-standard/
14. **Coursera stats <currently drawn from Wikipedia>**
15. **edX stats**
16. **Udacity stats**
17. Learning Resource Metadata Initiative. (n.d.). *:: The Specification*. Retrieved July 22, 2014, from http://www.lrmi.net/the-specification/
18. A free, open-source ontology editor and framework for building intelligent systems. (n.d.). *protégé*. Retrieved July 22, 2014, from http://protege.stanford.edu/
19. McGuinness, D. L., & Van Harmelen, F. (2004). OWL web ontology language overview. *W3C recommendation*, *10*(10), 2004.
20. RDF Schema 1.1. (n.d.). *RDF Schema 1.1*. Retrieved July 23, 2014, from http://www.w3.org/TR/rdf-schema/
21. Requests: HTTP for Humans. (n.d.). *Requests: HTTP for Humans — Requests 2.3.0 documentation*. Retrieved July 22, 2014, from http://docs.python-requests.org/en/latest/
22. Scrapy | An open source web scraping framework for Python. (n.d.). *Scrapy | An open source web scraping framework for Python*. Retrieved July 22, 2014, from http://scrapy.org/
23. Apache Jena. (n.d.). *Apache Jena*. Retrieved July 22, 2014, from https://jena.apache.org/
24. google-gson - A Java library to convert JSON to Java objects and vice-versa - Google Project Hosting. (n.d.). *google-gson - A Java library to convert JSON to Java objects and vice-versa - Google Project Hosting*. Retrieved July 22, 2014, from http://code.google.com/p/google-gson/
25. Bootstrap. (n.d.). *Bootstrap*. Retrieved July 22, 2014, from http://getbootstrap.com/
26. TDB Architecture. (n.d.). *Apache Jena*. Retrieved July 22, 2014, from https://jena.apache.org/documentation/tdb/architecture.html
27. Google App Engine. (n.d.). *URL Fetch API Overview*. Retrieved July 22 2014, from https://developers.google.com/appengine/docs/java/urlfetch
28. HTTP Authentication in ARQ. (n.d.). *Apache Jena -*. Retrieved July 22, 2014, from http://jena.apache.org/documentation/query/http-auth.html