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 engineering

I. 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.

II. BACKGROUND

A. *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.

B. *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.

C. *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.

D. 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.

E. 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.

III. 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.

A. 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.

B. 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.

C. 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.

D. *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.

IV. Implementation

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

A. *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. Figure 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.

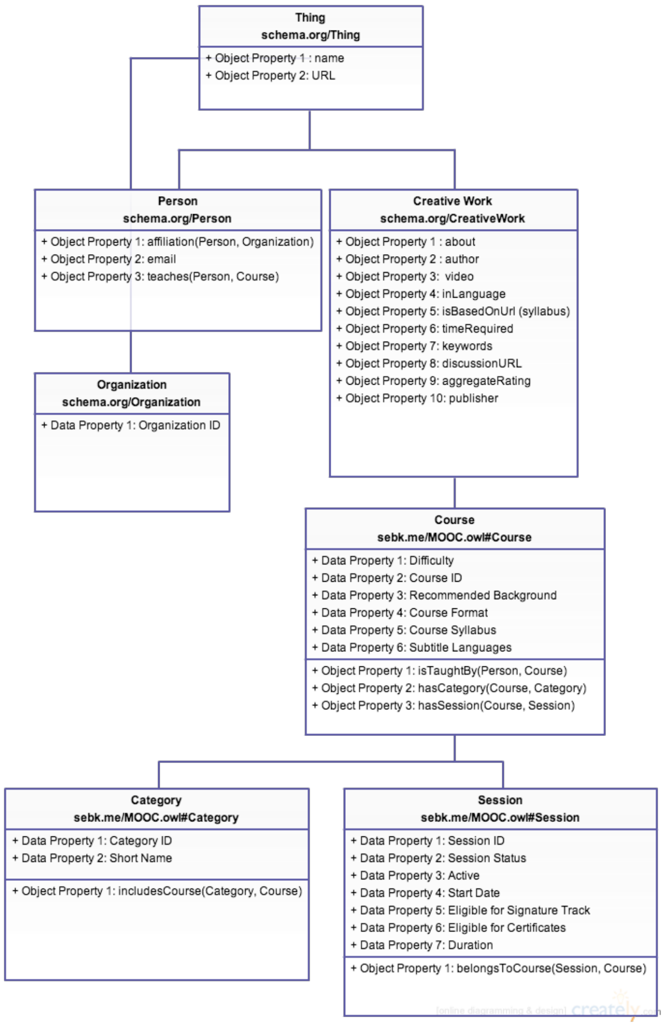

Thing
schema.org/Thing
+ Object Property 1 : name
+ Object Property 2: URL
Person
schema.org/Person
+ Object Property 1: affiliation(Person, Organization)
+ Object Property 2: email
+ Object Property 3: teaches(Person, Course)
Creative Work
schema.org/CreativeWork
+ Object Property 1 : about
+ Object Property 2 : author
+ Object Property 3: video
+ Object Property 4: inLanguage
+ Object Property 5: isBasedOnUrl (syllabus)
+ Object Property 6: timeRequired
+ Object Property 7: keywords
+ Object Property 8: discussionURL
+ Object Property 9: aggregateRating
+ Object Property 10: publisher
Organization
schema.org/Organization
+ Data Property 1: Organization ID
Course
sebk.me/MOOC.owl#Course
+ Data Property 1: Difficulty
+ Data Property 2: Course ID
+ Data Property 3: Recommended Background
+ Data Property 4: Course Format
+ Data Property 5: Course Syllabus
+ Data Property 6: Subtitle Languages
+ Object Property 1: isTaughtBy(Person, Course)
+ Object Property 2: hasCategory(Course, Category)
+ Object Property 3: hasSession(Course, Session)
Category
sebk.me/MOOC.owl#Category
+ Data Property 1: Category ID
+ Data Property 2: Short Name
+ Object Property 1: includesCourse(Category, Course)
Session
sebk.me/MOOC.owl#Session
+ Data Property 1: Session ID
+ Data Property 2: Session Status
+ Data Property 3: Active
+ Data Property 4: Start Date
+ Data Property 5: Eligible for Signature Track
+ Data Property 6: Eligible for Certificates
+ Data Property 7: Duration
+ Object Property 1: belongsToCourse(Session, Course)

Fig. 1. Diagram of Schema.org ontology extension.

B. *Web Crawling*

We retrieve Coursera's course properties via their course catalog API but use screen scrapers for edX and Udacity. This section details the process of writing a Scrapy crawler for edX in Python. After starting a new Scrapy project in the terminal, we define the Item or container that will be loaded with scraped data. This is done by creating a scrapy.Item class and defining its attributes as scrapy.Field objects as shown in Figure 2. These attributes, which are ultimately output as JSON, are named slightly different than our RDF properties. As a consequence, the naming scheme is mapped to the types defined in our ontology later in our Jena methods, which convert the collected JSON into RDF/XML.

```
class EdxItem(Item):
    name = Field()
    about = Field()
    instructor = Field()
    school = Field()
    courseCode = Field()
    startDate = Field()
    url = Field()
    length = Field()
    effort = Field()
    prereqs = Field()
    video = Field()
    category = Field()
```

Fig. 2. Defining attributes for the EdXItem container

In Figure 3 we subclass CrawlSpider (scrapy.contrib.spiders.CrawlSpider), and define its name, allowed domains, as well as a list of URLs for the crawler to visit first. The subsequent lines of code open a list of edX categories, iterate through those categories and append URLs to start_urls. These appended URLs lead to paginated lists of courses in each category. This method was chosen over using edX's "All Courses" listing as a start URL in order to gather category names from each edX course. Category names are currently not explicitly defined on each edX course page.

```
class EdXSpider(CrawlSpider):
    name = "edx"
    allowed_domains = ["edx.org"]
    start_urls = []
    file = open('data/category_map', 'r')
    for line in file:
        url="https://www.edx.org/course-list/allschools/"
+ line.strip() + "/allcourses"
        start_urls.append(url)
```

Fig. 3. Defining the spider and start_urls

Figure 4 shows our first parse method which takes a URL from our list of start URLs defined in Figure 3, selects each course URL listed on the page using the XPath selector "//strong/a/@href", then iterates through each site. For each site, a new EdxItem is declared, "url" and "category" are assigned, and a new scrapy.Request is made for the URL selected, calling the parse_details method detailed in Figure 5. The complete EdxItem is then assigned to the Request.meta attribute, and the request is appended to a list of every request made during the crawl.

```
def parse_sites(self, response):
    filename = response.url.split("/")[-2]
    open(filename, 'wb').write(response.body)
    sel = Selector(response)
    sites = sel.xpath('//strong/a/@href').extract()
    sites.pop(0) # remove home directory link
    requests = []
```

```
    for site in sites:
            item = EdxItem()
            item['url'] = site
            item['category']=response.request.url.split("/")[5]
            request=scrapy.Request(site,callback=self.parse_details)
            request.meta['item'] = item
            requests.append(request)
     return requests
```

Fig. 4. Parsing start_urls

The next method, parse_details, shown in Figure 5, collects attributes from individual course pages using XPath selectors. The majority of these selectors are tailored to the HTML classes provided by edX, which denote course properties provided on the course page.

```
def parse_details(self, response):
        filename = response.url.split("/")[-2]
        open(filename, 'wb').write(response.body)
        sel = Selector(response)
        item = response.meta['item']
        item['name'] = sel.xpath('//h2/span/text()').extract()
        item['about']=
sel.xpath('//*[@itemprop="description"]').extract()
        item['instructor']=sel.xpath('//*[@class="staff-
title"]/text()').extract()
        item['school']=sel.xpath('//*[@class="course-detail-school
item"]/a/text()').extract()
        item['courseCode']=sel.xpath('//*[@class="course-detail-
number item"]/text()').extract()
        item['startDate']=sel.xpath('//*[@class="course-detail-
start item"]/text()').extract()
        item['length']=sel.xpath('//*[@class="course-detail-length
item"]/text()').extract()
        item['effort']=sel.xpath('//*[@class="course-detail-effort
item"]/text()').extract()
        item['prereqs']=sel.xpath('//*[@class="course-section
course-detail-prerequisites-full"]/p/text()').extract()
        item['video']=
sel.xpath('/html/head/meta[@property="og:video"]/@content').extrac
t()
        return item
```

Fig. 5. Parsing indiviudal course pages

Navigating to the Scrapy project folder in the terminal and issuing the command "scrapy crawl edx –o items.json" yields a JSON file containing all items scraped from edX. This works for initializing our data although an item pipeline component will need to be developed in the future to validate our scraped data and check for duplicates.

C. *RDF Data*

The first section of our RDF (Fig. 6) 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.

```
<rdf:RDF xmlns="http://sebk.me/MOOC.owl#"
    xml:base="http://sebk.me/MOOC.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:schema="http://schema.org"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#">
```

Fig. 6 RDF/XML namespace declarations

In Figure 7 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.

```
<rdf:Description rdf:about="http://sebk.me/MOOC.owl#coursera_course_83">
   <Course_Format>The class consists of lecture videos, 8 - 15 minutes in length. These
contain 2-3 integrated quiz questions per video. There are standalone quizzes each
week.  Total lecture time is ~ 14 hours.&lt;br&gt;</Course_Format>
   <Course_ID>83</Course_ID>
   <hasSession>http://sebk.me/MOOC.owl#coursera_session_410</hasSession>
   <rdf:type rdf:resource="http://sebk.me/MOOC.owl#Course"/>
   <hasCategory>http://sebk.me/MOOC.owl#category_10</hasCategory>
   <schema:name>Drugs and the Brain</schema:name>
   <isTaughtBy>http://sebk.me/MOOC.owl#coursera_instructor_640696</isTaughtBy>
   <schema:timeRequired>4-6 hours/week</schema:timeRequired>
   <schema:inLanguage>en</schema:inLanguage>
   <hasSession>http://sebk.me/MOOC.owl#coursera_session_971466</hasSession>
   <Recommended_Background>Neuroscience, the most interdisciplinary science of the
21st century, receives inputs from many other fields of science, medicine, clinical
practice, and technology. Previous exposure to one or more of the subjects listed in
"Suggested Readings" will provide a good vantage point, as we introduce material
from these subjects.</Recommended_Background>
   <hasCategory>http://sebk.me/MOOC.owl#category_3</hasCategory>
   <schema:about>What happens in the body when a person smokes a cigarette? After
several weeks of smoking? When a person takes antidepressant or antipsychotic
medication? A drug for pain, migraine, or epilepsy? A recreational drug? Neuroscientists
are beginning to understand these processes. You’ll learn how drugs enter the brain, how
they act on receptors and ion channels, and how “molecular relay races” lead to changes
in nerve cells and neural circuits that far outlast the drugs themselves. “Drugs and the
Brain” also describes how scientists are gathering the knowledge required for the next
steps in preventing or alleviating Parkinson’s, Alzheimer’s, schizophrenia, and drug
abuse.</schema:about>
</rdf:Description>
```

Fig. 7. Sample of Coursera course data in RDF/XML

edX courses (Fig. 8) 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.

```
<rdf:Description rdf:about="http://sebk.me/MOOC.owl#edx_course_OEE101x">
        <schema:name>Our Energetic Earth</schema:name>
        <schema:video>http://www.youtube.com/v/lQc13b-
g2io?version=3&amp;amp;autohide=1</schema:video>
        <schema:inLanguage>English</schema:inLanguage>
        <hasSession>http://sebk.me/MOOC.owl#edx_session_OEE101x</hasSession>
        <isTaughtBy>http://sebk.me/MOOC.owl#edx_instructor_OEE101x</isTaughtBy>
        <schema:timeRequired>2-3 hours per week (6 weeks)</schema:timeRequired>
        <hasCategory>http://sebk.me/MOOC.owl#category_308</hasCategory>
        <Course_ID>OEE101x</Course_ID>
        <Recommended_Background>None.</Recommended_Background>
        <schema:about>&lt;span itemprop="description"&gt;&lt;h4&gt;
        *Note - This is an Archived course*&lt;/h4&gt;
     &lt;p&gt;&lt;span&gt;This is a past/archived course. At this time, you can only
explore this course in a self-paced fashion. Certain features of this course may not be
active...</schema:about>
        <rdf:type rdf:resource="http://sebk.me/MOOC.owl#Course"/>
</rdf:Description>
```

Fig. 8. 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.

```
<rdf:Description rdf:about="http://sebk.me/MOOC.owl#udacity_course_cs259">
   <Recommended_Background>&lt;p&gt;Basic knowledge of programming and Python
at the level of Udacity CS101 or better is required. Basic understanding of Object-
oriented programming is helpful.&lt;/p&gt;</Recommended_Background>
   <hasCategory>http://sebk.me/MOOC.owl#category_12</hasCategory>
   <hasSession>http://sebk.me/MOOC.owl#udacity_session_cs259</hasSession>
   <schema:name>Software_Debugging</schema:name>
   <Course_ID>cs259</Course_ID>
   <rdf:type rdf:resource="http://sebk.me/MOOC.owl#Course"/>
   <schema:inLanguage>English</schema:inLanguage>
   <schema:timeRequired>Assumes 6hr/wk</schema:timeRequired>
   <schema:about>&lt;div&gt;&#xD;
&lt;p&gt;In this class you will learn how to debug programs systematically, how to
automate the debugging process and build several automated debugging tools in
Python.&lt;/p&gt;&#xD;
&lt;/div&gt;&lt;strong&gt;Why Take This Course?&lt;/strong&gt;&lt;div&gt;&#xD;
&lt;p&gt;At the end of this course you will have a solid understanding about systematic
debugging, will know how to automate debugging and will have built several functional
debugging tools in Python.&lt;/p&gt;&#xD;
&lt;/div&gt;&lt;strong&gt;Prerequisites                                              and
Requirements&lt;/strong&gt;&lt;div&gt;&#xD;
&lt;p&gt;Basic knowledge of programming and Python at the level of Udacity CS101 or
better is required. Basic understanding of Object-oriented programming is
helpful.&lt;/p&gt;&#xD;
&lt;/div&gt;&lt;p&gt;See the &lt;a href="https://www.udacity.com/tech-requirements"
target="_blank"&gt;Technology            Requirements&lt;/a&gt; for            using
Udacity&lt;/p&gt;</schema:about>
   <schema:video>http://www.youtube.com/channel/UC0VOktSaVdm </schema:video>
   <isTaughtBy>http://sebk.me/MOOC.owl#udacity_instructor_cs259</isTaughtBy>
</rdf:Description>
```

Fig. 9. Sample of Udacity course data in RDF/XML

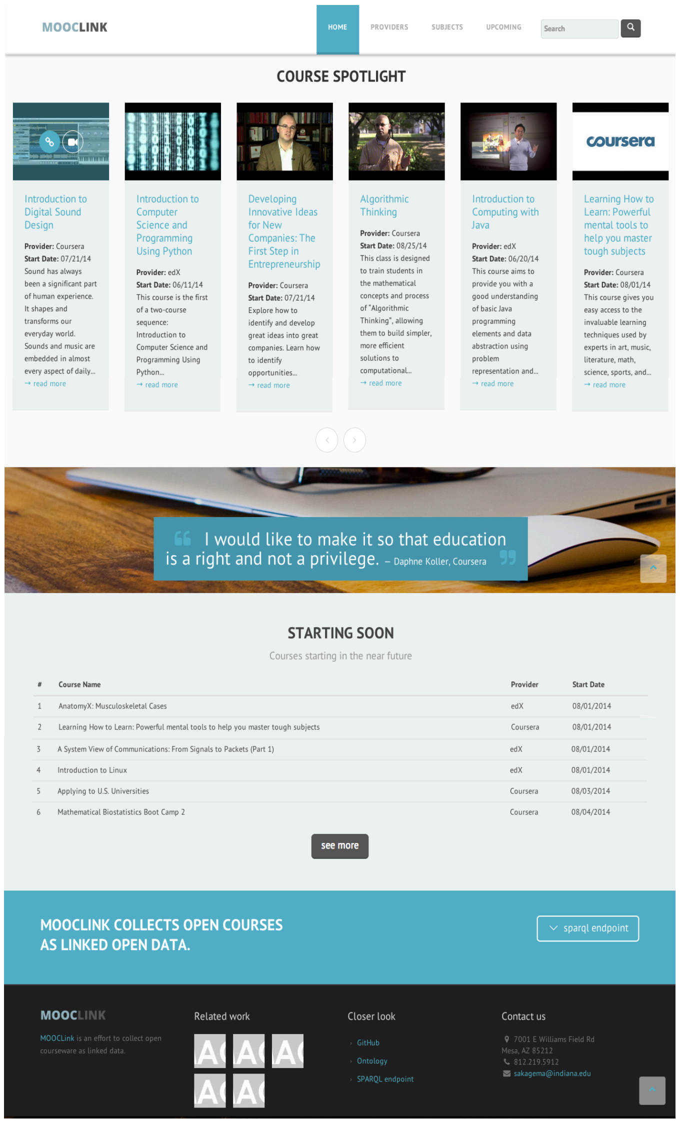
D. *UI Screenshots*

MOOCLink’s home page, shown in Figure 10, is headed by a navbar which contains links to “Providers”, “Subjects”, and “Upcoming” pages. “Providers” drops down into a list of Coursera, edX, and Udacity links when hovered over. Each of these pages is a paginated list of courses from their respective provider. “Subjects” leads to a table of links to our 36 categories. Clicking on one of these retrieves a paginated list of courses in that category. “Upcoming” is an extension of the starting soon table featured at the bottom of the home page, listing courses starting within the next month. The last item on the navbar is a search bar which retrieves courses by SPARQL query, an example of which is shown in Figure 14.

The “Course Spotlight” is an image carousel which features courses that might be popular with average users. Currently it is a mix of introductory courses we have hand-picked. In the future we hope to base these courses on statistics incorporated into our RDF such as how many users have enrolled or completed a specific course. This will depend on the future availability of MOOC statistics via provider API or otherwise.

The course images in this carousel are retrieved from YouTube. The IDs for these videos are taken from our RDF which collects links to introductory videos. Where an ID is not provided, a placeholder logo for the provider, such as the Coursera logo in the rightmost course, is shown. Hovering over a course prompts links to the MOOCLink course details page and the introductory video. The course title, provider, start date, as well as the beginning of the course summary are provided below the course images.

Below the course spotlight, an image with a quote from Daphne Koller is shown. On further scrolling (which creates a parallax effect with the image) the "Starting Soon" section follows with 7 courses starting in the next month. This is followed by a link to our SPARQL endpoint and a footer featuring a brief project description, links to related work as well as our contact information.



COURSE SPOTLIGHT

Introduction to Digital Sound Design
Provider: Coursera
Start Date: 07/21/14
Sound has always been a significant part of human experience. It shapes and transforms our everyday world. Sounds and music are embedded in almost every aspect of daily...
→ read more

Introduction to Computer Science and Programming Using Python
Provider: edX
Start Date: 06/11/14
This course is the first of a two-course sequence: Introduction to Computer Science and Programming Using Python...
→ read more

Developing Innovative Ideas for New Companies: The First Step in Entrepreneurship
Provider: Coursera
Start Date: 07/21/14
Explore how to identify and develop great ideas into great companies. Learn how to identify opportunities...
→ read more

Algorithmic Thinking
Provider: Coursera
Start Date: 08/25/14
This class is designed to train students in the mathematical concepts and process of "Algorithmic Thinking", allowing them to build simpler, more efficient solutions to computational...
→ read more

Introduction to Computing with Java
Provider: edX
Start Date: 06/20/14
This course aims to provide you with a good understanding of basic Java programming elements and data abstraction using problem representation and...
→ read more

Learning How to Learn: Powerful mental tools to help you master tough subjects
Provider: Coursera
Start Date: 08/01/14
This course gives you easy access to the invaluable learning techniques used by experts in art, music, literature, math, science, sports, and...
→ read more

I would like to make it so that education is a right and not a privilege. – Daphne Koller, Coursera

STARTING SOON

Courses starting in the near future

#	Course Name	Provider	Start Date
1	AnatomyX: Musculoskeletal Cases	edX	08/01/2014
2	Learning How to Learn: Powerful mental tools to help you master tough subjects	Coursera	08/01/2014
3	A System View of Communications: From Signals to Packets (Part 1)	edX	08/01/2014
4	Introduction to Linux	edX	08/01/2014
5	Applying to U.S. Universities	Coursera	08/03/2014
6	Mathematical Biostatistics Boot Camp 2	Coursera	08/04/2014

MOOCLINK COLLECTS OPEN COURSES AS LINKED OPEN DATA.

MOOCLINK
MOOCLink is an effort to collect open courseware as linked data.

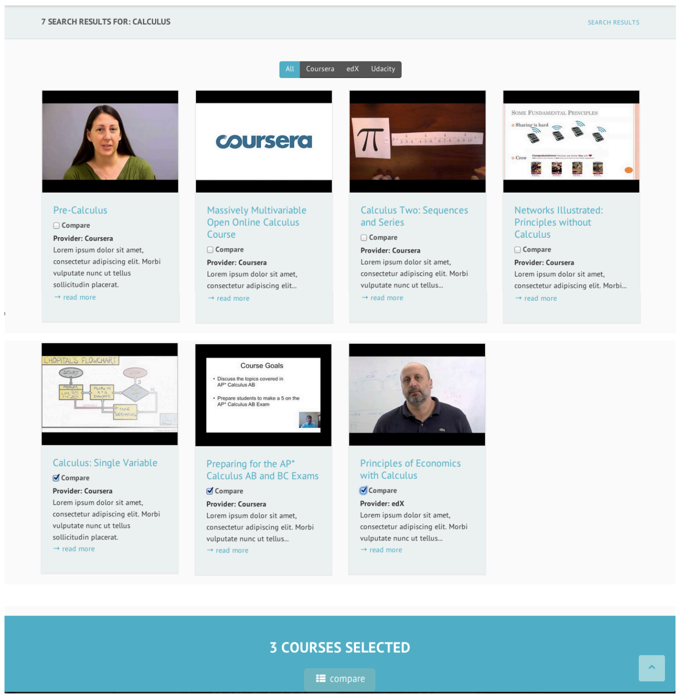
Related work

Closer look
› GitHub
› Ontology
› SPARQL endpoint

Contact us
7001 E Williams Field Rd
Mesa, AZ 85212
812.219.5912
sakagema@indiana.edu

Fig. 10. Screenshot of MOOCLink home page

Search results are shown in a 4-column image grid as shown in Figure 11. The images of these courses are retrieved by the same method as the home page carousel. Courses on the search results page are filterable by course provider. Checkboxes are provided below course titles to flag which courses one would like to compare in more detail. In this example, the bottom three courses, "Calculus: Single Variable", "Preparing for the AP Calculus AB and BC Exams" and "Principles of Economics with Calculus" are flagged for comparison.



7 SEARCH RESULTS FOR: CALCULUS

SEARCH RESULTS

Pre-Calculus
☐ Compare
Provider: Coursera
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi vulputate nunc ut tellus sollicitudin placerat.
→ read more

Massively Multivariable Open Online Calculus Course
☐ Compare
Provider: Coursera
Lorem ipsum dolor sit amet, consectetur adipiscing elit...
→ read more

Calculus Two: Sequences and Series
☐ Compare
Provider: Coursera
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi vulputate nunc ut tellus...
→ read more

Networks Illustrated: Principles without Calculus
☐ Compare
Provider: Coursera
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi...
→ read more

Calculus: Single Variable
☑ Compare
Provider: Coursera
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi vulputate nunc ut tellus sollicitudin placerat.
→ read more

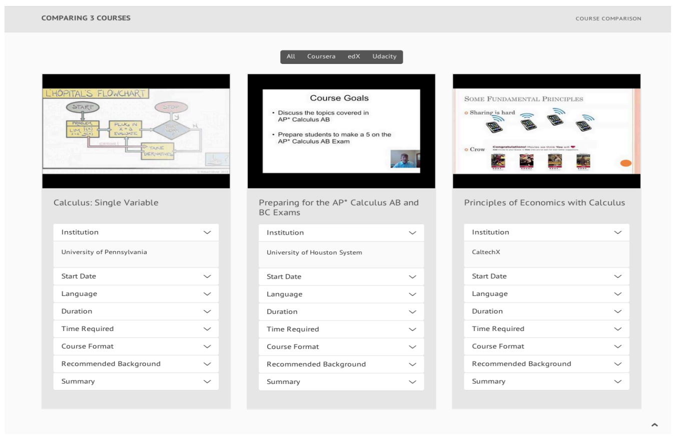
Preparing for the AP* Calculus AB and BC Exams
☑ Compare
Provider: Coursera
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi vulputate nunc ut tellus...
→ read more

Principles of Economics with Calculus
☑ Compare
Provider: edX
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi vulputate nunc ut tellus...
→ read more

3 COURSES SELECTED

Fig. 11. Screenshot of search results

In Figure 12, the table for the comparison of courses is shown. Information pertaining to each course is displayed as an "accordion" in which clicking on a property opens the field for each course being compared allowing for simple detail-by-detail comparison. Courses are once again filterable by course provider and hovering over the course image brings up a link to its corresponding MOOCLink "course details" page as well as a link to enroll in the course.


All
Coursera
edX
Udacity
L'HÔPITAL'S FLOWCHART
Course Goals
Discuss the topics covered in AP* Calculus AB
Prepare students to make a 5 on the AP* Calculus AB Exam
SOME FUNDAMENTAL PRINCIPLES
Sharing is hard
Crow

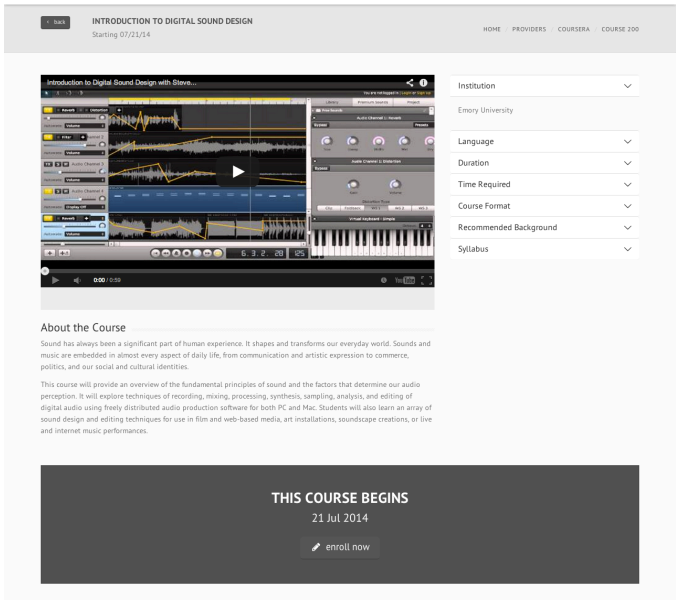
COMPARING 3 COURSES

COURSE COMPARISON

Calculus: Single Variable	Preparing for the AP* Calculus AB and BC Exams	Principles of Economics with Calculus
Institution	Institution	Institution
University of Pennsylvania	University of Houston System	CaltechX
Start Date	Start Date	Start Date
Language	Language	Language
Duration	Duration	Duration
Time Required	Time Required	Time Required
Course Format	Course Format	Course Format
Recommended Background	Recommended Background	Recommended Background
Summary	Summary	Summary

Fig 12. Screenshot of course comparison table

A course page on MOOCLink, shown in Figure 13, is headed by the course title and start date as well as a breadcrumb for navigation. Next to the introductory video is the same accordion found on the course comparison table, listing relevant course properties. Below the video is the full course summary and a link to enroll in the course on its provider's webpage.



back

INTRODUCTION TO DIGITAL SOUND DESIGN

Starting 07/21/14

HOME / PROVIDERS / COURSERA / COURSE 200

Institution

Emory University

Language

Duration

Time Required

Course Format

Recommended Background

Syllabus

About the Course

Sound has always been a significant part of human experience. It shapes and transforms our everyday world. Sounds and music are embedded in almost every aspect of daily life, from communication and artistic expression to commerce, politics, and our social and cultural identities.

This course will provide an overview of the fundamental principles of sound and the factors that determine our audio perception. It will explore techniques of recording, mixing, processing, synthesis, sampling, analysis, and editing of digital audio using freely distributed audio production software for both PC and Mac. Students will also learn an array of sound design and editing techniques for use in film and web-based media, art installations, soundscape creations, or live and internet music performances.

THIS COURSE BEGINS

21 Jul 2014

enroll now

Fig 13. Screenshot of course details page

In order to generate search results and retrieve course properties from our RDF, SPARQL queries are performed. These are called on our application server using Jena's ARQ API which queries the RDF from a separate Fuseki Server. Currently we search for names in the RDF that contain the words searched using regular expressions. Figure 14 illustrates a search for "calculus". All items in the RDF are selected where they are of type mooc:Course and every course with a schema:name or course title containing "calculus" (ignoring case as defined by the regular expression filter) is returned.

```
PREFIX mooc: <http://sebk.me/MOOC.owl#>
PREFIX schema: <http://schema.org/>
        SELECT * WHERE {
            ?course rdf:type mooc:Course.
             ?course schema:name ?iname.
        FILTER (regex(?iname, "calculus", "i")).
        }
```

Fig. 14. Sample of a SPARQL query behind MOOCLink search

Although keyword search of course titles may yield relevant results, there is more work to be done in order to take advantage of the more extensive searches we can perform with SPARQL. For example, we might want to know what courses cover writing business plans. There are few courses which concentrate around business plan writing or have "business plan" in the title, although many business courses might have lectures which cover it. Because our MOOC RDF contains syllabi details, we can perform a SPARQL query which returns any course that mentions writing business plans allowing the user to compare relevant courses side-by-side to see that they enroll in the course that covers this topic most extensively. We aim to incorporate more robust search such as this example in future iterations of MOOCLink.

V. 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, reviews, developing an item pipeline for our crawlers, automating website updates, enabling user profiles, course tracks, natural language processing of syllabi and summaries for more robust data and search.

Acknowledgments

I would like to thank my mentor, Professor Srividya Bansal of Arizona State University for her continual support. This work is supported in part by the Distributed Research Experience for Undergraduates (DREU) program, a joint project of the CRA Committee on the Status of Women in Computing Research (CRA-W) and the Coalition to Diversify Computing (CDC).

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. (n.d.). Retrieved July 22, 2014, from http://coursera.org/

[15] edX. (n.d.). Retrieved July 22, 2014, from http://edx.org/.

[16] Udacity. (n.d.). Retrieved July 22, 2014, from http://udacity.com/

[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