MOOCLink: An aggregator for MOOC offerings from various providers

Chinmay Dhekne, Srividya K. Bansal

School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Mesa, AZ 85212 USA
srividya.bansal@asu.edu

Abstract: With so many online courses offered as Massive Open Online Courses (MOOC), it is becoming increasingly difficult for an enduser to gauge which course best fits their needs. It would be very helpful for them to make a choice if there is a single place where they can visually compare the offerings of various MOOC providers for the course they are interested in. Previous work has been done in this area through the MOOCLink project that involved integrating data from Coursera, EdX, and Udacity and generation of semantically linked data. The research objective of this paper is to devise algorithms to include data from new MOOC providers and maintain the quality of data through MOOCLink application, as there are lots of new courses being constantly added and old courses being removed by MOOC providers. We present the integration of data from various MOOC providers and algorithms for incrementally updating linked data to maintain their quality in order to provide an interactive MOOC aggregator environment for students to be engaged with up-to-date data.

Keywords: Linked Data; Semantic Web; Engineering Education; Learning Management Systems.

1. Introduction

Online education has been gaining widespread importance in the last 5 years. According to the New York Times, 2012 became "the year of the MOOC", as various providers such as EdX, Udacity, Coursera, etc. emerged. Multiple course providers offer courses with similar syllabi and content. The user is spoilt for choices in terms of which online course to choose, as there is such a fine line between courses by different providers. The user has to scan through multiple providers each with their own webpage and browse through all their course offerings. Also, in many cases, it so happens that the users want to learn a specific topic within a domain. In such a scenario, it becomes increasingly difficult for the user to look through the syllabi of all courses offered by different providers before arriving at a decision as to which provider to go ahead with. There exists initial work in this area in the form of semantic web (Bizer, 2009) application called MOOCLink (Kagemann, 2015; Dhekne, 2017). With MOOCLink, the user can make an informed decision by visiting the MOOCLink website and using the enhanced SPARQL enabled search engine, he/she can quickly compare courses offered by various MOOC providers, all in the one place. The aim of this interactive aggregator system is to help users solve the problem of looking through multiple course providers. However, even with this integrated solution in place, maintaining the highest level of data quality is still a concern. Especially in the field of online education, where course offerings are dynamic, it is all the more important that relevant data is being displayed to the end users at all times. The paper presents extraction and linking of data from various course-providers to drive the MOOCLink web application and a strategy to solve the data quality issue. Linked data is the method of connecting and publishing related structured data on the web (Bizer, 2009). Linked Open Data (LOD) is being used in a number of interesting and useful Web and mobile applications including MOOCLink. A general architecture of the MOOCLink application is shown in Figure 1.

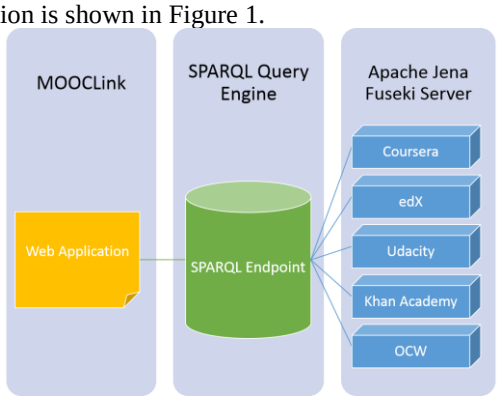

Figure 1: MOOCLink - High-level Architecture

Maintenance of data quality in this paper refers to the removal of outdated and/or irrelevant data and also the addition of new data being constantly added by various course providers. The focus of this paper is to "devise algorithms to maintain linked data quality (in the domain of online education) and evaluate the approach through the implementation of MOOCLink web application which serves as an aggregator of all courses that are available". This project focuses primarily on integrating data from 5 different course providers, publishing it as linked data and maintaining the quality of the linked data generated. Specifically, this paper presents the algorithms for maintaining quality of data and experiments for evalution of the algorithms.

The rest of the paper is organized as follows: Related work is presented in section 2. Section 3 presents data collection followed by Linked data generation in Section 4. Section 5 presents the algorithms for maintaining data quality followed by experimental results and conclusions.

2. Related Work

There are two aspects of Linked Data quality that need to be studied and strategies devised. The first aspect is to maintain data quality in terms of adding or replacing missing information, coupling and de-coupling of wrong relationships amongst data entities. The second aspect of maintaining data quality is by making sure that the data published on the Web, is up to-date. Approaches are developed to ensure that there's no outdated copy of data (Zaveri, 2012). Lots of research and development has been made for the first aspect of data quality. The paper "Crowdsourcing Linked Data Quality Assessment" explains the first aspect in detail (Acosta, 2013). This research focuses on using crowdsourcing as a means to maintain data quality. Analysis of the most commonly occurring problems was done and a model was designed which aimed to eliminate these issues. This model consisted of two different approaches. First, a contest which targets an expert crowd of Linked Data researchers; Second, by means of paid microtasks published on Amazon Mechanical Turk. The dataset for this research was DBpedia and these two novel approaches were evaluated against the data published on DBpedia. The results show that crowdsourcing is an effective mechanism to eliminate data quality issues, and this approach could potentially be included in the Linked Data curation process.

Another research named "Test-driven Evaluation of Linked Data Quality" aims to solve data quality issue using test-driven software development (Kontokostas, 2014). This research proposes that vocabularies, ontologies, datasets should be accompanied by test-cases, which help in maintaining a basic level of data quality. The need was felt especially because the amount of Linked Data being generated is huge and not all of the data is of top quality. Some of it might have a strong structured knowledge base at the back end but some datasets are not curated at all. This leads to an inconsistent environment, thereby requiring a cleansing of data using test-cases. One major advantage of using this approach was that domain-specific semantics can be encoded in the test-cases, in turn making it possible for data quality to surpass normal quality heuristics.

A. Comparison of Tools (MOOC Aggregators)

In this section, we list existing MOOC aggregators and comparing their functionality with MOOCLink.

- *Class Central:* is a MOOC aggregator that aims to list course offerings from multiple MOOC providers to help the user make an informed decision. The user can search by subject, or by simply searching a keyword and the underlying system of Class Central will pull all courses relevant to search query. MOOCLink offers various other parameters for searching courses, i.e. by Course format (Static/Dynamic), Course name, Course Category, etc.
- *MOOC-List:* provides a course list based upon providers, universities, instructors and tags. The search by tags mechanism is nothing but tagging each course into domains that encompass that course. For example, a course such as "Java Programming Fundamentals" will have tags such as "Programming", "Object-oriented", etc. Majority of search is based on the tags.
- *CourseTalk:* The course search on the CourseTalk website is based on either subject name, provider or on the basis of free/paid courses. CourseTalk has introduced an innovative concept of leaderboards. Based on user reviews, instructor reviews, content ratings and provider reviews, the leaderboards are developed. This gives way to healthy competition and it proves to be a motivational factor to learn for the students.
- *Degreed*: is one of the most popular MOOC aggregator. This aggregator keeps track of all educational data of the user. This educational data is not limited to what the user learns at Degreed, but it extends to the books read, MOOC courses completed, college degrees earned and so on. Moreover, weekly analysis of the user's learning of each week is also provided which helps the user to keep track of their work throughout the week.

B. MOOCLink

MOOCLink aggregator system differentiates itself from other MOOC aggregators by using semantic web technologies that allows for publishing of the integrated MOOC datasets as Linked Open Data on the cloud. This opens up the data to the community as a whole and can be tweaked and modified as required by the community for other research as well. Another aspect which sets MOOCLink apart from the above listed MOOC aggregators is the power of MOOCLink to be able to search on a number of search parameters. MOOCLink supports usage of parameters such as "Search by Name", "Search by Category", "Search by Course Type (Free/Paid)", "Search by Course Provider", "Search by Start/End Date", "Search by Course Description". The semantic data model and MOOCLink ontology for the integrated data from different course providers is presented in our previous work (Kagemann, 2015; Dhekne, 2017).

3. Data Collection

For this project, data was collected from various MOOC providers. Figure 2 provides an overview of the data sources and their format.

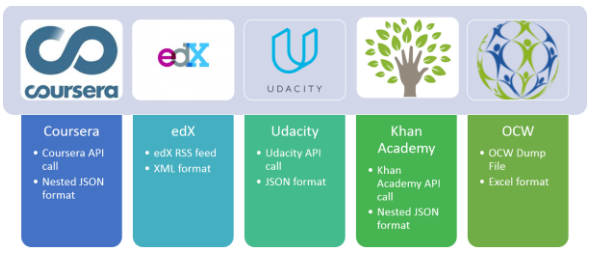

Figure 2: Data Sources for MOOCLink

A. Coursera

Coursera is the first amongst the five MOOC providers

from which data is being pulled for MOOCLink. Coursera is an educational technology company that offers various Massive online courses. These courses belong to various domains - Physics, Medicine, Computer Science, Biology, Business, etc. Coursera offers all courses free of cost. There is an option to sign up for a signature track to obtain a completion certificate, but for an additional fee. It offers an API for developers who wish to obtain data about their courses. This API is a catalog API that exposes all of Coursera's courses, instructors and partnering universities. It is available publicly and does not need authentication (Coursera API). Data is retrieved in JSON format. Coursera also provides various optional parameters that can be included in the API query, such as the "includes" parameter. This parameter enables the developer to include other entities such as instructor list, category list and university list in one single call to the API endpoint.

B. edX

edX is MOOC provider founded by Massachusetts Institute of Technology and Harvard University in May 2012. edX offers various online courses with a very broad range of disciplines, with some courses being offered without any cost. edX focuses mainly on the weekly learning pattern. Also, in addition to providing online courses, edX plays an important role in the research for open and distant learning. The concept of learning via edX is novel in its own way. Each course consists of a series of videos that the student goes through and a learning exercise at the end of every few videos that aim to test the knowledge imparted to the student until that point. In the earlier phase of development done in the MOOCLink project (Kagemann, 2015), data was obtained by using screen-scraping techniques. However, edX now provides an API endpoint for developer use (EdX API).

C. Udacity

Udacity is a for-profit organization founded at Stanford University with a number of free computer science classes. The courses comprise of videos with closed captions and integrated quizzes and follow-up homework that helps students evaluate what they have learned in a progressive manner. Being a for-profit organization, as was the case with edX during the initial phase of the MOOCLink project (Kagemann, 2015), there was no API available to obtain data from Udacity. Hence, a web scraper was developed for Udacity using Scrapy. The scraper was developed in Python programming language. This was a challenge because it was difficult to keep the development stable with the ever-changing website design of Udacity. This pain was alleviated when they released their own API endpoint to obtain course catalog information (Udacity API).

D. Khan Academy

Khan academy is a non-profit organization providing free online education. The courses are in the format of short video lectures. Khan Academy introduced the concept of "badges" in order to make the learning process more competitive. There are six levels of badges and the learners in a particular course can compete amongst themselves to create a healthy competitive atmosphere. Khan Academy is the first amongst new data sources added to the MOOCLink ecosystem. Makers of Khan Academy have a developer API in place to pull data from (Khan Academy).

E. OpenCourseWare (OCW)

OpenCourseWare (OCW) is a free and open digital publication of courses that are created at universities and are published on the internet. OCW has gained worldwide recognition with over 29,000 courses available in over 70 languages. OCW is the second new data source which is added to the MOOCLink ecosystem. OCW also provide a developer API and their API endpoint (OCW API). As the API is under development, it was decided that the excel dump file would be a more stable.

4. Linked Data Generation

The process of Linked Data generation starts right from capturing raw data from MOOC providers, building a semantic web ontology for the system, conversion of the raw data captured from MOOC providers into RDF data (in TTL format) compliant with the ontology, creating and maintaining a SPARQL endpoint (using Apache Jena Fuseki Server), and finally executing SPARQL queries on the generated RDF data. The entire process is depicted in Figure 3. Data collected from the MOOC providers is in the form of raw data files (json format, spreadsheets, or traditional database tables) that is mapped to the semantic data model of the system.

Karma Web is an "information integration tool that enables users to quickly and easily integrate data from a variety of data sources" (Szekely, 2013). Karma web has made it possible to seamlessly convert the raw data into appropriate RDF data in TTL format while using ontologies as a basis for integrating information of the system. It provides a visual representation of the raw data and enables the user to create relationships and hierarchies between entities. Karma Web is a very user-friendly, easy to use and robust tool that provides integration support with hierarchical data models such as XML, JSON, and KML. Raw JSON data from Coursera was converted into RDF using this tool.

Apache Jena Fuseki Server is used for hosting RDF/TTL files. It is basically a SPARQL server. There are multiple modes in which Apache Fuseki server can work as an operating system service, as a Java Web app, or even as a standalone server. Apache Jena Fuseki Server provides a intuitive graphical user interface where the user can upload multiple RDF/TTL files upon which SPARQL queries are performed. Another important feature of Apache Jena Fuseki server is the "manage dataset" feature. Using this feature, the user can create multiple datasets and choose from them to execute SPARQL queries. Also, for a particular dataset, the user is provided with the ability to add and/or remove more files to the particular dataset.

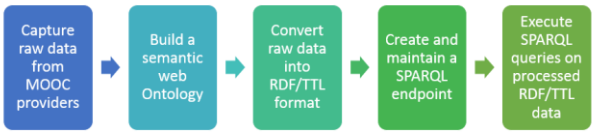

Figure 3: Linked Data Generation Process

An alternative to Karma Web Integration technique to convert raw data from data sources to RDF or TTL format, using scripts for conversion. While the Karma Web is a beautiful and easy technique to convert raw data into RDF/TTL format, it has to be done manually and requires human intervention. The end goal of this project is to maintain the data quality on the MOOCLink website. Using our own scripts, allows automation of the process of linked data generation. As a general rule, different scripts have been developed for different data sources. This is because the data format of each data source is different though the underlying semantic data model for the integrated data is common. The same OWL file is used in all scripts. The conversion algorithm from raw data format to RDF/TTL format follows the following steps:

- Loading all ontology properties and classes from the OWL file into an Apache Jena object in Java.
- Parsing raw data file and dividing each data object into two HashMaps, i.e. a property map and a class map. Property map contains all properties of a particular course and class map contains all classes that are present in that course.
- Jena object, Property map and the class map together are loaded into the Apache Jena library for Java and the appropriate RDF/TTL file is generated.

5. Maintaining Quality of Linked Data

This section will focus on maintenance of Linked Data quality. The web of today is extremely dynamic and information presented and viewed by the user at a point of time may become irrelevant within a day or even earlier than that. To be able to cope up with the dynamic nature of data, we need strategies so that the users are interacting with the latest copy of data. MOOCLink application is a MOOC aggregator that must provide users latest data on course offerings. We propose two algorithms for maintaining Linked Data Quality in this project:

- Naïve Batch Update
- Incremental Update

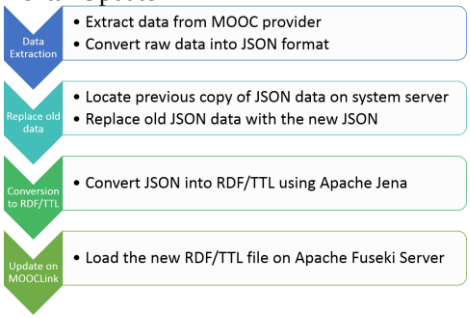

Figure 4: Naive Batch Update Approach

A. Naïve Batch Update Algorithm

This approach is a simplistic and traditional way of maintaining data quality shown in figure 4. The first step is to pull new data from the MOOC provider. This process of data extraction from MOOC provider is explained in detail in section 5. Once new data is extracted from a MOOC provider, it is converted into an appropriate JSON format with proper indentation. The indentation makes sure that the JSON document is human readable if there's a need for a manual override during the working of the system. The next step is to locate the previous copy of JSON data on the server. MOOCLink app is hosted in Apache Tomcat server. This file resides under the "data" folder on the Tomcat server. Once the file is located, it is replaced by the new JSON file. The next step is to convert this "new" but raw JSON data into RDF/TTL file format. This is a crucial step because all SPARQL queries will be executed on the RDF/TTL data, and not the raw JSON data. The conversion from JSON to RDF/TTL is performed as explained in previous section. Once the file is converted to RDF/TTL file format, it completely replaces the old RDF/TTL file residing on the Apache Jena Fuseki Server. Once this change is done, the same process is followed for other MOOC providers.

B. Incremental Update Approach

The difference between the Naïve Batch update and the Incremental update is in the factors taken into consideration before obtaining updated Linked Data. In case of the former algorithm, simple replacement of the old JSON data with the newly obtained JSON data is done. Figure 5 depicts steps in this approach.

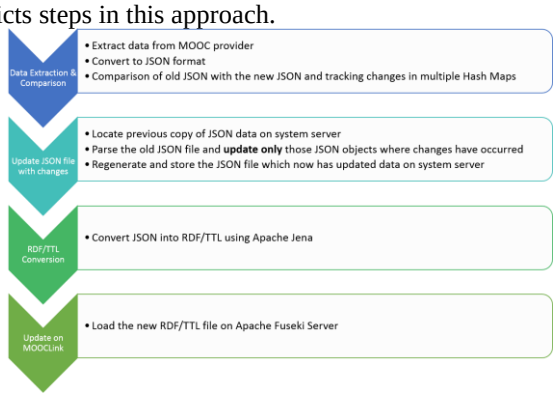

Figure 5: Incremental Update approach

In the first step of the incremental update approach, the original JSON file residing on the server is compared with the new JSON file. This is done using Hash Maps, as this data structure fits in perfectly to satisfy the requirement and also has a faster retrieval time (O(1)). Two levels of Hash Maps are created, i.e. one on the class level, and the other on the property level of a class. As seen in figure 5, an outer Hash Map is created for each class, i.e. "Course", "Category", and so on. Also, for each of these Hash Maps, there is an inner Hash Map created with key being the property of that class and value being the data associated with that property. For example, in case of the "Course" class, an outer Hash Map is created with key-value pair as:

<course_ID>, <Hash Map<String, String>>

The inner Hash Map contains the key-value pairs such as (<Course_ID>, "123"), (<Course_desc>, "description") and so on. This process ensures that all the new JSON data is captured in these Hash Maps. Once this data is captured, the old JSON file residing on the system server is parsed, object-by-object, and each class and each property of that class is compared with the new JSON file stored in Hash Map format. A "Course Change Counter" and "Course Add/Delete Counter" is maintained to be incremented whenever a course change or course addition/deletion is encountered between the two sets of data respectively. These counter variables are also used in the "provider score" module that is described later in this section. The next step of this process is to update the changes to the original JSON file. The old JSON file is parsed and the changes, which are again captured in the Hash Map data structure in the previous step, are updated in the original JSON file. The last two steps of the Incremental update approach is same as the Naïve batch update, that is conversion to RDF/TTL and reflecting the changes on the web application.

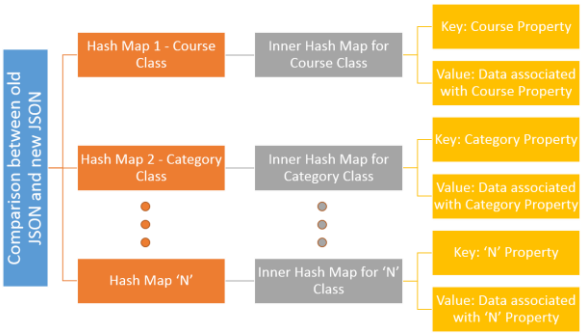

Figure 6: JSON comparison in Incremental Update approach

C. Decision Maker Module

The Decision maker module is responsible to identify one of the approaches described based on a "Threshold" value that is calculated using various factors. Figure 7 presents the steps in the "decision maker" module. The first step in this process is to extract new data from each MOOC provider as presented in section 5. When a fresh copy of raw data is obtained from the MOOC provider, the next step is to take this new copy of data and compare it with the previous raw data of the same MOOC provider residing on the project server. While comparing the new data and the old data, the system looks at two parameters: Percentage change and Provider Score.

The "percentage change" parameter tells how much percentage of data has been changed in the new data file as compared to the older version of data. For example, if 100 courses are retrieved in the first run, and in the next run, 100 courses are retrieved but it is found that out of the new 100 courses, 36 courses have changes in them. These changes may be changes to "course name", "course description", "category", etc. This shows that the percentage change is 36%. It also signifies that the remaining 64 courses are unchanged. The "provider score" parameter helps to rank various MOOC providers in descending order of their score. This parameter is helpful in the real-world where unlimited amount of bandwidth is not a reality. There are bandwidth limitations in terms of how much amount of data can reside on the system server and/or how much amount of data can be downloaded using the services of your Internet Service Provider (ISP). While the latter is not much of a concern in an industry setting, the former is a real-world issue. Coming up with a provider score for each MOOC provider will ensure that if there arises a bandwidth constraint, then the system is in a position to target only those MOOC providers that have a greater provider score than the others. This will, in turn, make sure that maximum number of data updates is done in each run of the check for Data Quality. The provider score is calculated based on three factors: (i) size of data (number of courses retrieved from the data source); (ii) number of Courses added or deleted; (iii) number of course details modified. Based on these factors, each data source is ranked. The next parameter is "threshold". This parameter will decide when to use the Naïve batch update vs Incremental update. The threshold value is defined as the "percentage change at which one approach becomes better than the other in terms of efficiency".

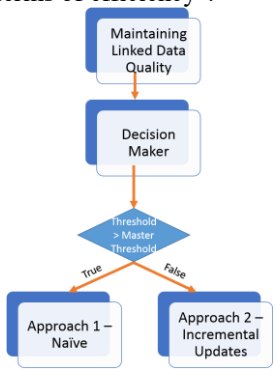

Figure 7: Decision Maker module

As seen in figure 7, the threshold value is responsible for deciding whether to update data using Approach 1 or Approach 2. A master threshold value was evaluated after extensive experimentation on the data sets. The process of evaluation of the master threshold value is presented in the next section. Once the master value is obtained, it is compared with the change percentage of the current run. If the change percentage is less than the master threshold value, then the "Incremental Update approach" is used. On the other hand, if the change percentage is greater than the master threshold value, then the "Naïve approach" is used.

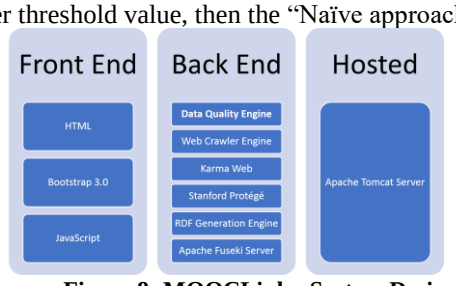

Figure 8: MOOCLink - System Design

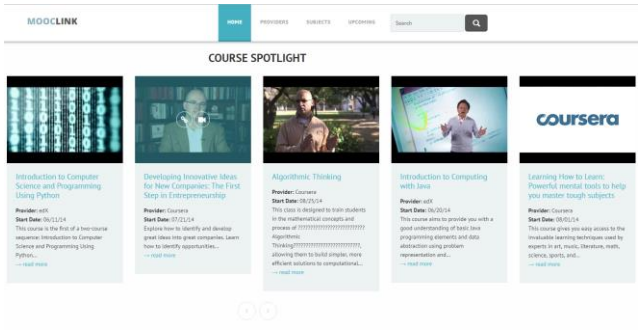

Figure 9: MOOCLink Webapp screenshot

D. Web Application Development

MOOCLink is hosted on Apache Tomcat 8 Server. The GUI is developed in Javascript and HTML with Bootstrap 3.0 library. The most important feature of the MOOCLink application is the "Search" bar. The user can search for any course or even keywords and the underlying SPARQL query engine will search throughout the RDF triples and return a list of courses with an exact or at least a partial match with the keyword. Figure 8 and 9 shows the MOOCLink system design and web application screenshot.

6. Experimental results

In order to evaluate search results obtained through MOOCLink, "precision" and "recall" values of the results were used. By definition, precision is defined as "the fraction of documents retrieved that are relevant to the user's information need" (Zhu, 2004). On the other hand, Recall is defined as "the fraction of documents that are relevant to the query and are successfully retrieved" (Zhu, 2004). In reference to this project, precision would give us the percentage of courses that are retrieved by the MOOCLink system and are relevant to the information queried by the user. Recall would give us the percentage of courses that are relevant to the user query and are retrieved using MOOCLink. Table 1 presents a summary of the data loaded into MOOCLink in terms of the number of semantic RDF triple, size of data and number of courses. Table 2 presents the precision and recall values for each provider.

Table 1: Linked data in MOOCLink

Provider	# of Triples	Size of Data (KB)	# of Courses
Coursera	3480	2522	1220
edX	1680	990	993
Udacity	564	1092	131
Khan Academy	1338	84409	142
OCW	55756	37019	29696
TOTAL	**62818**	**126032 KB**	**32182**

Table 2: Precision and Recall of search queries

Provider	Precision (%)	Recall (%)
Coursera	100	94.89
edX	100	100
Udacity	100	100
Khan Academy	100	100
OCW	100	91.23

The recall values for Coursera and OCW are comparatively low due to the non-availability of access to the entire course catalog via the API's provided by the respective MOOC providers. For Coursera, a "pagination" parameter has been added to the API call that did not return all courses at the time of these experiments. OCW provides an Excel dump file complete Course Catalog information. Unfortunately, they don't have a mechanism in place to automate the updating of this Excel dump file as and when new courses are added on OCW. As a result, recall value was low as result comparison was done against an updated OCW course dataset with course information that was not present in Excel file used for computation in MOOCLink. For evaluation of maintenance of data quality, data was manually generated for 4 sets of experiments with different percentage of change in data from original data: 10%, 50%, 60%, and 90%. For each dataset both algorithms were executed. To ensure that all data files are being downloaded at the same download speed, a third party application called "Net Limiter" was used to keep the download speed uniform for each run of the experiment for both approaches. Results of these experiments show that the "master threshold value" is at 55%. It is observed that when the "percentage change" of the data of any MOOC provider is greater than 55%, then the Naïve batch update is more efficient. On the contrary, if the "percentage change" is less than 55%, then the Incremental update approach fares better as shown in figure 10.

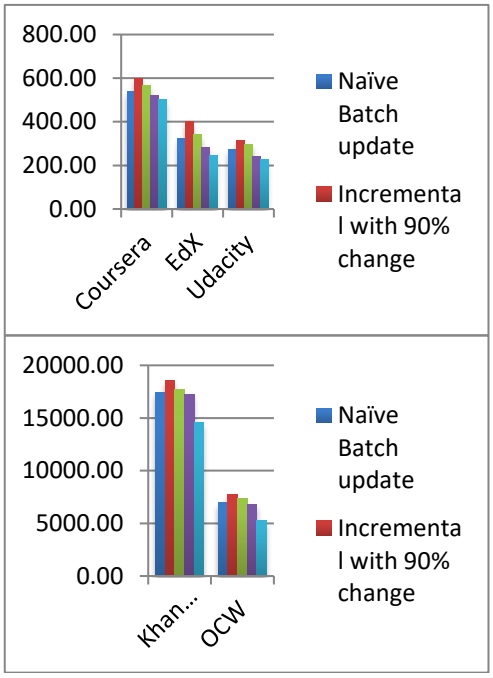

Figure 10: Results of Incremental update approach

7. Conclusions & Future Work

This paper presents an MOOC aggregator system for online education through integration of data from various MOOC providers and publishing it as Linked Open data thereby contributing to the community effort of LOD for data interchange on the web. Algorithms are presented to ensure that the integrated data is updated to maintain its quality. Future work includes incorporation of more MOOC providers and presenting a user-friendly interface for visual "Course Comparison" to better illustrate similarities and differences. The course comparison feature would help

users to understand differences in various MOOC offerings and help them choose the most appropriate course.

References

Zaveri, A., Rula, A., Maurino, A. (2012). Quality Assessment Methodologies for Linked Open Data: A Systematic Literature Review and Conceptual Framework. Journal of Semantic Web, pp. 1 - 33.

Kagemann, S., & Bansal, S. (2015). MOOCLink: Building and utilizing linked data from Massive Open Online Courses. Intl. Conference on Semantic Computing (ICSC), pp. 373–380.

Acosta, M., Zaveri, A., et al. (2013). Crowdsourcing Linked Data Quality Assessment. Intl. Semantic Web Conference (ISWC), pp. 260–276.

Kontokostas, D., Westphal, P. (2014). Test-driven evaluation of linked data quality. World Wide Web Conf. (WWW), pp. 747–757.

Coursera: https://api.coursera.org/api/

EdX API: https://www.edx.org/api/v2/report/course-feed/rss

Udacity API: https://www.udacity.com/public-api/v1/courses

Khan Academy: http://www.khanacademy.org/api/v1/topictree

OCW API: https://github.com/ocwc/ocwc-data

Szekely, P., Knoblock, C. (2013). Connecting the Smithsonian American Art Museum to the Linked Data Cloud. The Semantic Web: Semantics and Big Data: 10th International Conference, Montpellier, pp. 593–607.

Dhekne, C., & Bansal, S. (2017). Linking and Maintaining quality of data about MOOCs using Semantic Computing. Intl. Conference on Semantic Computing (ICSC), pp. 81–84.

Bizer, C., Heath, T., Berners-Lee, T. (2009) “Linked data-the story so far,” Intl. journal on semantic web & information systems, vol. 5, #3, pp. 1–22.

Zhu, Mu. "Recall, precision and average precision." Dept. of Statistics and Actuarial Science, University of Waterloo, Waterloo 2 (2004): 30.