Spring 2013

Mobile Presentation of Unstructured Information

Shailesh Benake
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Computer Sciences Commons](#)

Recommended Citation

Benake, Shailesh, "Mobile Presentation of Unstructured Information" (2013). *Master's Projects*. 314.
DOI: <https://doi.org/10.31979/etd.zz4h-ke2z>
https://scholarworks.sjsu.edu/etd_projects/314

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Mobile Presentation of Unstructured Information

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment of the

Requirements for the

Degree Master of Computer Science

By

Shailesh Benake

Spring 2013

© 2013
Shailesh Benake
ALL RIGHTS RESERVED
SAN JOSÉ STATE UNIVERSITY

The Undersigned Writing Project Committee Approves the Writing Project Titled

Mobile Presentation of Unstructured Information

by

Shailesh Benake

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Chris Tseng, Department of Computer Science 05/24/2013

Dr. Soon Tee Teoh, Department of Computer Science 05/24/2013

Mr. Anand Sastry, VP, Qolsys 05/24/2013

ABSTRACT

Since the advent of online education in 1994 by CALCampus , many improvements have been made for effectiveness of e-learning. Video/audio conferencing, synchronous education system and many such advances in multimedia communication have made this system more popular among the masses. However with many online education websites, competing to make the same course, it's important for user to find course structure of his interest. What makes even more challenging for a learner is, to decide how good will be the learning from a course provided by a particular site. For example open online course sites like edx.org, canvas.net, coursera.org etc provide similar types of course but the course structure and outcome of the courses are different. Hence it is imperative to integrate such information, in order to get more relevance of similar courses. What gives more advantage is, if a student can find similar courses offered by his university on massive open online course websites. This will surely provide a student more details about the course apart from his college curriculum, making his learning experience more unique.

So the objective of this project is, to gather semi-structured and unstructured data from MOOC (massive open online course) websites. Perform similarity check among the courses from these websites and provide likeness between the courses provided by the university and by the MOOC websites on mobile.

ACKNOWLEDGEMENT

I would like to thank Dr. Tseng for his technical guidance and his constant support during the writing project. Dr. Tseng's timely advice and his expertise in web technologies have helped me all through this project. I would also like to thank Dr. Teoh and Mr. Anand Sastry for being my committee members and help me through this process.

Table of Contents

1. PROJECT OVERVIEW	8
1.1 INTRODUCTION	8
1.2 MOOC WEBSITES AND SCORE EVALUATIONS	9
2. TOOLS AND FRAMEWORKS	11
1.1 SIMPLEHTMLDOM PARSER	11
1.2 UMBC - SEMANTIC TEXTUAL SIMILARITY SYSTEM:	13
1.3 PHONEGAP MOBILE PLATFORM:	17
1.4 XAMPP:	20
3. PROJECT DESIGN.....	23
1.1 DESIGN OUTLINE	23
a. Data Identification Phase:	24
b. Data Extraction Principles:.....	24
c. Formulation of Structural Data (Data Staging):	25
d. Data Presentation:	27
1.2 SCORE EVALUATION PROCESS:	27
a. Two Pass Score Evaluation:	27
b. One Pass Score Evaluation:.....	28
4. PROJECT IMPLEMENTATION	29
1.1 DOM PARSING TECHNIQUE:	31
1.2 SCORE CALCULATION	36
1.3 MOBILE PRESENTATION OF DATA:	38
5. TECHNICAL CHALLENGE:	43
6. TESTING AND RESULTS:	45
1.1 DATA VALIDATION:	45
7. CONCLUSION AND FUTURE WORK:.....	46
8. REFERENCES:	48

List of Table and Figures

Figure 1: UMBC Phrase Similarity Service	15
Figure 2: UMBC Phrase Similarity API Service	16
Figure 3: Phonegap application development	17
Figure 4: Adobe Phonegap Build	19
Figure 5: XAMPP for windows	20
Figure 6: XAMPP for control panel for windows	21
Figure 7: PhpMyAdmin for windows	22
Figure 8: Project design Outline	23
Figure 9: Wrappers to perform secondary parsing	25
Figure 10: Database design for Course database	26
Figure 11: HTML data extract using file_get_html method	31
Figure 12: Block of Unstructured data, regarding instructor information	32
Figure 13: Main-Page data of edx.org for level 1 parsing	33
Figure 14: Sub-Page data of edx.org for level 2 parsing	34
Figure 15: Application communication with Server	39
Figure 16: Application page when launched	41
Figure 17: Course result page on mobile	42
Figure 18: MOOC Courses similar to “CS 046A”	43
Figure 19: Source of Coursera.org page with no HTML data.	44
Figure 20: Json data link for coursera.org.	45
Figure 21: Data validation check list.	46
Figure 22: Course output for Testing.	46

1. Project Overview

1.1 Introduction

The aim of this project is to access the semi-structured and unstructured data from the websites and present it on the mobile in a structured format. For this project we have focused on MOOC (Massive open online courses) websites and San Jose State university webpage's for data. The MOOC websites provide free online courses to the masses. The secondary goal of the project is to provide comparison among the courses posted by MOOC websites, and to find similarity between MOOC courses and the University offered courses. The similarity score provided here will benefit the students in finding courses of their interest and help them in making decision among the best MOOC sites for a particular course.

So if one is interested in knowing which site is offering a course on "Introduction to Computer Science", but his interest is only in programming related course structure. The results from this project will help him identify courses in sync with his interest and also provide relativity among them. The person also has an option to mix and match the sections from 2-3 similar courses provided by different sites, making his learning experience more relevant. Another important scenario is, if the student needs to get a similar course structure between the course offered by his university and the sites. The scores from the result will be decisive in finding the match.

The data from which the scores are generated are fetched from unstructured and semi-structured data sources. As stated, the sources of data are the MOOC websites and the university

site from which data is scraped using parser. The final data used is the output of the staging process, the input to the staging process includes irregular patterns of data.

Making the results available on mobile was important as it is one of the ubiquitous modes of communication. The challenge was to integrate data wrt to a particular course and make it presentable for Mobile UI interaction. Also making an application for only one platform was a biased approach. So I used the Phonegap platform which allows the developer to develop the application using the HTML5, CSS and JQuery. This code can then be used to generate binary or executable files for 6 different platforms namely iOS, Android, WebOS, Symbian, Windows and Blackberry.

1.2 MOOC Websites and Score Evaluations

Massive open online courses provide online courses to the masses, which aims to project large scale participation for the courses it offers. Presumably the courses offered by these sites are for free and some of the sites offer course from reputed scholars across the globe. For eg “coursera.org” offers “Programming in Scala” for free, this is taught by the Martin Odersky designer of Scala [11]. These sites do provide courses from various departments, namely computer science, economics, health and medicine, social science, management etc where every site having different course structure. But they have a common approach of connecting tutors with participants on a distributed network for course discussions. They also provide auto-online feedback, online assessments and moreover some of them also provide certifications and placements. Such are the advantages one can get for free from MOOC websites.

The current applications just provide courses from various MOOC websites in an integrated format and few similarity searches based on keyword. But there is no measure to evaluate the courses based on semantics of the course details. The idea here is to facilitate the user with such finesse. So for this project we have considered few popular MOOC sites such as Coursera, Edx, Udacity and Canvas. Our aim is to use the data from these sites for course evaluations in the project. The descriptive metadata of these sites are semi-structured and in some cases they are unstructured too. The data collection is one of the primary objectives of this project. Using proper techniques will ensure in good quality of content, which will benefit for qualitative analysis in next stage of the project. So the method we have undertaken for extraction is a three step approach. Once we obtain the data the next step is to formulate and cache the data, for this step we are using MySQL database system. The cached data is then used for similarity calculations and to find relativity among the courses.

As mentioned we are comparing the university courses with the MOOC courses, for which we are scrapping the data from San Jose State University website. So for our evaluations we will be considering all the courses offered by SJSU, whose information will be fetched in the similar way as with the MOOC websites. For score evaluations we have used different approaches in order to achieve higher order precision and relevance. One such approach is to find phrase similarity among the title of the courses, this will fetch us a score. The score is valued against a threshold score, the threshold score is obtained from manual evaluations and the precision score. All the courses with score above threshold scores will be subjected to next evaluation of course description phrase similarity process. The score is again obtained from this phase; the weighted score is calculated from the scores of the two phases. This weighted

score helps in determining how relevant the two courses are. Similarly we have devised few approaches which we will discuss in the later part, to get maximum weighted score among two courses.

2. Tools and Frameworks

We have used open source tools and frameworks at different stages of the project, right from extraction phase to the presentation phase. Following are the tools:

1.1 Simplehtmldom parser

Simplehtmldom parser helps in parsing the html document object model and allows you to manipulate html in a convenient way [3]. As we know the document object model (DOM) is cross platform and language independent, providing access to interact with objects in HTML, XML documents. It requires php 5+ versions and one of the advantages it endows with is, it supports even invalid html. It has robust in-built functionalities, which facilitates fetching of html contents in single line of script. Following are the few functionalities it provides:

- To access the entire DOM of a html file we have:

```
$html = file_get_html('sample.htm');
```

- To access the entire DOM from a page:

```
$html = file_get_html('www.knollp.com');
```

- To access all the anchor tags of an html page we have:

```
$a_tags = $html->find('a');
```

- If we want to extract the div tag with id 'falcon' the utility it provides is:

```
$div_tag = $html->find('div[id=falcon]');
```

Similarly we can use the advanced way too do the above step is,

```
$div_tag = $html->find('#falcon');
```

- We can also extract multiple tags with similar attributes, in this case we have all the anchors and images with the "title" attribute.

```
$a_attr = $html->find('a[title], img[title]');
```

- Other than tags we also have option to access comments and text in the html,

```
$text_data = $html->find('text');
```

- If one needs to traverse tags using the nested approach , it is possible by using,

```
// Find all <li> in <ul>
foreach($html->find('ul') as $ul)
{
    foreach($ul->find('li') as $li)
    {
        // do something...
    }
}
```

- One of the most convenient approach it provides for traversing a DOM, wherein a parent child relationship exists between the tags in the html is,

```
// Example
echo $html->find("#div1", 0)->children(1)->children(1)->children(2)->id;
// or
echo $html->getElementById("div1")->childNodes(1)->childNodes(1)->childNodes(2)->getAttribute("id");
```

Simplehtmldom is easy to use , download the source file from a trusted site. Next set the file in the folder where you are going to set your scripts for parsing the webpage's. In your scripts just include the file name as *"include ('simple_html_dom.php');"*. Once the setup is done, develop the scripts using stated methods in order to fetch the data from web pages.

1.2 UMBC - Semantic Textual Similarity System:

To evaluate the text similarities semantically in our project, we are using the system developed by the UMBC ebiquity research group. The API provided to access this system, is used as medium to supply our text inputs to the system. The score is generated based on the semantic relation among the text rather than the lexical relation within the words. For example, the verb "run" must be semantically related to the noun "Leg" or "Body". Thus the system provides range of scores from highest to lowest based on similar words and non-similar words correspondingly.

The system requires two phrase inputs which are to be matched, with variable lengths. There are two criteria to be mentioned when calculating the score; one is to provide whether the user wants to calculate score using concept similarity or relation

similarity. In Concept similarity, the phrase is divided into terms and then the word co-occurrences are counted against the entire corpus. This mode offer's a more accurate context but it only works for comparing words within the same parts of speech.

On the other hand Relation similarity works similar to concept similarity, but it facilitates us to calculate semantic similarity between the words with different Parts of Speech. The second criterion is selection of corpus against which the phrase similarity score is evaluated [1]. That is in order to generate a dependable word co-occurrence statistics; a stable text corpus of a very large volume is required. So this system provides “Refined Stanford WebBase corpus” and “LDC English Gigawords Corpus”. For our evaluations we are using the Relation similarity and Stanford WebBase corpus combination. As tests have shown that, words in a phrase from different POS have a tendency to fetch a lower score, when using the concept similarity option. Also the paper presents the results that stable word co-occurrence statistics was obtained when Stanford WebBase corpus was used.

The screenshot shows a web browser window with the address bar displaying 'swoogle.umbc.edu/SimService/GetSimilarity'. The page title is 'UMBC Phrase Similarity Service'. Below the title, there is a 'Go back' link. The main content area contains two input fields for phrases: 'Phrase 1: Introduction to Computer Programming' and 'Phrase 2: Introduction to Programming in Java'. Below these fields, there are radio buttons for 'Type' (Concept Similarity (noun phrase only) and Relation Similarity) and 'Corpus' (Refined Stanford WebBase corpus and LDC English Gigawords Corpus (American newswire services only)). A 'Get Similarity' button is located below the corpus selection. At the bottom, the text 'The similarity score is 0.6666667' is displayed.

Figure 1: UMBC Phrase Similarity Service

The above image shows how we can manually check the scores of two phrases from the UMBC Phrase Similarity Service website. But for our project we are using the API service provided by this system. The score generated by this API can be extracted using SimplehtmlDOM parser methods.

Figure 2: UMBC Phrase Similarity API Service

In the API link we are providing the phrases and the criteria against which the scores are to be calculated. The score fetched is available for our analysis on the page or can be accessed programmatically too.

1.3 Phonegap Mobile Platform:

Phonegap is a free open source framework for developing mobile applications using web API's that can execute on multi platforms [9]. So one can develop an application using HTML5, JavaScript and CSS once and execute it on mobile platforms like iOS, Android, Symbian, Blackberry, Windows and WebOS.

Figure 3: Phonegap application development

The framework provides access to hardware functionalities of the phone platform like accelerometer, camera, compass, geolocation, media etc. This enables the developer to use the signals from the interface managers of the phone. Also if you are using Phonegap build provided by Adobe to compile the code, then you can write your code using any editor. There are no such restrictions on using a particular IDE or environment to develop applications using this framework. The only disadvantage this platform has is execution speed. When the application to be developed requires more resources and more hardware

interactions, its performance drops down as JavaScript is slower when compared to native development languages of a mobile platform.

For our project we are using Phonegap build tool to generate binary and executable files for respective mobile platforms. The process involved is as follows,

- The application files such as Html5, JavaScript and CSS should be placed in a directory.
- The directory should be zipped using WinZip only as gzip, tar, rar or 7zip do not work.
- You should have a developer account on Adobe Phonegap Build to upload your code.
- In your account upload the zip file, you require developer key in case you need .dmg format of file for the application to run on iOS platform.
- After loading, the Phonegap build will generate the .exe, .bin platform dependent files for every platform mentioned.
- Once the files are generated then using the QR code scanner of your phone or barcode scanner, scan the QR code generated by the build.
- Every phone platform requires some settings to be performed in order to install applications directly into the phone. So one has to take the necessary steps for application to run on his phone.
- The scanning of QR code will download and install the application into the phone. The application will have a Phonegap build logo on your phone. Launch the application for use.

Figure 4: Adobe Phonegap Build

The figure 4 shows the adobe Phonegap build interface. You can access this page by using the link <https://build.phonegap.com/apps> .

1.4 XAMPP:

XAMPP is an open source cross platform web server stack for developers who want to test their application on their local system [10]. X stands for cross platform so it can be installed on any operating system, A for apache http server, M for MySQL database, PP for PHP and Perl script interpreters respectively. In this project we have developed the server using PHP and for caching of data we have used the MySQL db. So XAMPP has provided the environment for implementing the backbone of the project.

Figure 5: XAMPP for windows

The server can be controlled by using the XAMP control panel. The control panel also provides logs of every module, provides configuration access and command line control.

Figure 6: XAMPP for control panel for windows

As mentioned XAMPP also provides MySQL support, so the PhpMyAdmin tool is used in our project to setup the caching module. The database is designed to cache the

data from scrapping the sites, used in staging process for data cleansing and finally to cache the scores obtained from phrase similarity of course titles and course description.

Figure 7: PhpMyAdmin for windows

All the program and resource files which need to be tested using XAMPP should be placed in the “htdocs” folder. Then to execute we can use any browser, in the browser we have to provide the path to the file as http://localhost/folder_name/file_name .

3. Project Design

1.1 Design outline

The project can be divided into 4 steps, namely identification of data, extraction of data, the staging process and finally reporting phase. This can be represented as,

Figure 8: Project design Outline

a. Data Identification Phase:

In this phase we identify the MOOC and SJSU webpage's we require for our evaluations. The data that we are looking out for is mainly semi structured data or in some cases it can be even unstructured. The information we are interested in, is details about a course, like course title, course dates, duration, instructor information etc. We identified coursera.org, udacity.com, canvas.net and edx.org, for university data we considered the sjsu.edu for comparison.

b. Data Extraction Principles:

This stage defines the principles on how to extract the information from the identified sources. We are using methods such as Dom parsing, http programming in cases where we can access link to the source databases, devised a wrapper for detecting pages that consists information of our interest and for a particular case we have also scraped the metadata in order to relate the actual data.

In Dom parsing, the static html data is transformed into a relational tree, so that we can access the child nodes using parent-child relation among the tags. This method we are using mainly for all the sites that we have identified. In http programming technique we are using the link to fetch json data from the source database. This process is used for extracting data from coursera.org, wherein every html page was generated dynamically. To perform second level of parsing wherein we are extracting the subpages, we have devised wrappers in order to identify the information

we are looking for. Once we get the particular sub page then we are using the Dom parsing technique to fetch the contents.

Figure 9: Wrappers to perform secondary parsing

c. Formulation of Structural Data (Data Staging):

In this process, we are performing cleansing of data; this will ensure the correct format of data is obtained. The data usually is not formatted or sometimes we can also get garbage data usually when dealing with unstructured data. The unstructured data that we are dealing here does not have any format or we do not have metadata in order predict the existence of expected data. So for this we will extract the whole data and

then perform lexical analysis on the corpus. The outcome of this process will give us likely data in string format, which can be further formatted for caching it.

Once we get a clean data, we the cache it into the MySQL database. We are using a relational database for caching the data. The database for our project consists of 4 tables namely “Courses” to store the data from sjsu.edu, “Course_data” and “Course_details” for storing MOOC data and “match” table for storing the similarity scores among the courses.

Figure 10: Database design for Course database

Once the staging data is loaded into the database we truncate the temporary data. This process is true for every cycle of data scraping process we perform.

d. Data Presentation:

This phase serves two purposes one is to display the data and secondly to test if valid data has been retrieved. We are developing user interface for web as well as phone. So when the user will search for a particular course the result will consist of the course title, introduction video's, course homepage link, professor details, course date and length and the weighted score.

1.2 Score Evaluation Process:

As discussed, score evaluation is one of the important processes of this project. This process helps in finding how similar are two courses with respect to each other. The course data extracted from the MOOC and SJSU websites are used as the input corpus. We will use the following methods:

a. Two Pass Score Evaluation:

In this method, we are calculating the scores using step by step approach. The course data consists of course title, course description, course date etc. So the first step is to score the similarity between the titles. For example, title 1: "Introduction to Computer Programming" and title 2: "Introduction to Computer Science and Programming" will fetch a score of 0.875. In the next step we will only score those two courses whose step 1 result is above threshold value. In this step we will score the course descriptions and the category of the two courses. So we have three scores in total, the weighted score can then be calculated as,

$$\text{Weighted Score} = 5 * \text{title score} + 4 * \text{course description score} + 1 * \text{category score} / 10$$

The threshold value here is calculated by trial and error validation process. We manually sampled courses from every category of courses to see whether they match. Once it was confirmed we also checked the score and on averaging the scores we zeroed down to 0.74 as the threshold value.

b. One Pass Score Evaluation:

In this scheme, we are comparing every course with every other course and scoring it against course title, description and category. Once the three scores are obtained we calculate the weighted score for every combination using above equation. Then pick the top 5 weighted scores for every course and cache it.

So the above two methods are implemented to calculate similarity scores among MOOC courses and SJSU, MOOC courses respectively.

4. Project Implementation

The process that we are following to implement this project is as follows:

- a. Cache the data from every MOOC website identified and university website.
- b. Perform data formatting and cleaning if necessary.
- c. Calculate scores using the score evaluation methods and cache it.
- d. Display of the data.

The first step as discussed in the design phase is to identify the data from the MOOC sites and the SJSU website. So using the simplehtmldom parser method, we fetch the entire webpage content to manually analyze the html data.

```
include('simple_html_dom.php');

// Retrieve the DOM from a given URL
$folder = file_get_html("https://www.edx.org/courses");

echo $folder;
```

The output is the entire page content, but we are interested in very specific tags which have course information. The below output shows a part of the html content we are interested in. We observe the course title given in “span” tag having class= “course-number”, course image given in “img” tag , course start-date given in “span” tag with class “start-date”. So in this way we identify data for every site,

```
<article id="UTAustinX/UT.2.01x/2013_Sept" class="course">
  <span class="status">New</span>
  <a href="/courses/UTAustinX/UT.2.01x/2013_Sept/about">
    <div class="inner-wrapper">
      <header class="course-preview">
        <hgroup>
          <h2><span class="course-number">UT.2.01x</span> Ideas of the 20th Century</h2>
        </hgroup>
        <div class="info-link">&#x2794;</div>
      </header>
      <section class="info">
        <div class="cover-image">
          
        </div>
        <div class="desc">
          <p>Learn how philosophy, art, literature, and history shaped the last century and the world today.</p>
        </div>
        <div class="bottom">
          <a href="/university_profile/UTAustinX" class="university">UTAustinX</a>
          <span class="start-date">Sep 15, 2013</span>
        </div>
      </section>
    </div>
    <div class="meta-info">
      <p class="university">UTAustinX</p>
    </div>
  </a>
</article>
```

Figure 11: HTML data extract using file_get_html method

Once we identify the dataset, we set up the extraction rules for every page of our interest. This method is true for only data which are semi-structured and can be identified with the help of metadata or tags. While evaluating subpages for Coursera.org, we came across unstructured data, wherein the data was just a flat file with no tags. Using the parsing method was not yielding the correct results. Therefore here we used the string split technique (explode in php) wherein chunk of the data was segregated to form sentences and then parsing it to fetch the right data.

Director of the Stanford Artificial Intelligence Lab the main AI research organization at Stanford with 20 professors and about 150 students post docs. At Stanford he teaches Machine Learning which with a typical enrollment of 350 Stanford students is among the most popular classes on campus. His research is primarily on machine learning artificial intelligence and robotics and most universities doing robotics research now do so using a software platform (ROS) from his group. In 2008 together with SCPD he started SEE (Stanford Engineering Everywhere) which was Stanford's first attempt at free online distributed education. Since then over 200000 people have viewed his machine learning lectures on YouTube and over 1000000 people have viewed his and other SEE classes' videos. Ng is the author or co-author of over 100 published papers in machine learning and his work in learning robotics and computer vision has been featured in a series of press releases and reviews. In 2008 Ng was featured in Technology Review's TR35 a list of 35 remarkable innovators under the age of 35. In 2009 Ng also received the IJCAI Computers and Thought award one of the highest honors in AI. [university_logocourse-ids\[16152970311\]](#) [displaytrue recommended_background](#)

Figure 12: Block of Unstructured data, regarding instructor information

After categorizing of the data to be extracted for each website, we then develop the extraction principles. The process we are going to discuss here is for Edx.Org website. So once we get the complete html data for a page using `file_get_html` method. We then use the Dom parsing method for parsing the data and extract it.

1.1 Dom Parsing Technique:

In this technique we traverse the html page from one parent node to its child nodes. We use `simplehtmldom` parser to check for the existence of nodes and then extract it.

```
include('simple_html_dom.php');

$folder = file_get_html("https://www.edx.org/courses");

foreach ($folder->find('.courses-listing-item') as $e){
    if(strpos($e->find('#.course',0)->childNodes(0)->text(),'New') !== false){
        $elink= $e->find('#.course',0)->childNodes(1)->getAttribute('href');
    }
    else{
        $elink= $e->find('#.course',0)->childNodes(0)->getAttribute('href');
    }
}
```


Here we are getting the entire html page content, then we are using the ‘find’ method to check whether “.courses-listing-item” is present. So if it’s present then we fetch the child nodes. The goal is to fetch the “href” attribute of an anchor tag as shown below.

```
<li class="courses-listing-item">

<article id="UTAustinX/UT.2.01x/2013_Sept" class="course">
  <span class="status">New</span>
  <a href="/courses/UTAustinX/UT.2.01x/2013_Sept/about">
```

After successful extraction of href link tag, we then access the subpages by using this link. This page again undergoes the same process as the main page, to get the html page content.

```
//program link
$elink = "https://www.edx.org$elink";
$subfolder = file_get_html("$elink");
```

This leads to second level parsing of data, wherein individual course data can be extracted.

Figure 13: Main-Page data of edx.org for level 1 parsing

Figure 14: Sub-Page data of edx.org for level 2 parsing

Getting access to both the pages, we then extract the following data;

- Course description of the course:

```
//program short description
$edesc = $e->find('.desc',0)->text();
$edesc = trim($edesc);
```

- Category of the course which it belongs to:

```
//program category
$category = courseCat($ename[0],$ecname,$edesc);
```

- Course Start date:

```
//program start date
$esdate = $subfolder->find('.start-date',0)->text();
$estDate = calcDate($esdate);
```

- Course Title :

```
//program name
$subname = $subfolder->find('.intro'.0)->childNodes(0)->childNodes(0)->text();
$ename = explode(":", $subname);
    if(count($ename) == 2){
        $ecname = trim($ename[1]);
        $ecname = $ename[0].": ".$ecname;
    }
```

- Course image and video link :

```
//program image
$imglink = $subfolder->find('.hero',0)->childNodes(0)->getAttribute('src');
$image = "https://www.edx.org$imglink";

//Program Video link
if ($subfolder->find('#video-modal',0)->childNodes(0)->childNodes(0)){
    $sevidlink = $subfolder->find('#video-modal',0)->childNodes(0)->childNodes(0)->getAttribute('src');
}
else{
    $sevidlink = "http://www.youtube.com/watch?feature=player_embedded&v=XNaiOGxWeto";
}
```

In this way we access every relevant data using the Dom Parsing technique. This same process is used for other sites too.

The next step is to format the data; generally the data that we are getting is a string. We need some of the data to be of date format, numeric, or decimal so that we can use it for computations. So we are using functions to perform such formatting,

```

$norm = trim($date);
$norms = str_replace(" ", "", $norm);
$normdate = explode(" ", $norms);
$day = str_replace(",", "", $normdate[1]);
$normdate[0] = trim($normdate[0]);
if (count($normdate) > 2){
    $normdate[2] = trim($normdate[2]);
}
else{
    $normdate[2] = 0;
}

if(strcasecmp($normdate[0], "Jan") == 0 || strcasecmp($normdate[0], "January") == 0){
    $month = 1;
}
else if(strcasecmp($normdate[0], "Feb") == 0 || strcasecmp($normdate[0], "February") == 0){
    $month = 2;
}
else if(strcasecmp($normdate[0], "Mar") == 0 || strcasecmp($normdate[0], "March") == 0){
    $month = 3;
}

```

Here “\$date” is consisting of date in string format as “Sep 15, 2013”, so we are using string parsing techniques to split the \$date. Once we get the desired string formatting then we do check for every month. If the month is “JAN” we assign the new value as “\$normdate = 1”. The final date will be of the format “09-15-2013”, once we get this type of value we push it into the table column of type date. Similarly for numeric or decimal data we use the following formatting method

```

// $num_val is an object having a numeric value

$value = (string)$num_val;
$val = floatval($value);

```

Here we have \$num_val as object value from simplehtmldom, we convert it to string by using string casting. Then using floatval casting we convert the value to a decimal value.

1.2 Score Calculation

Following the staging and caching process, the score calculation is performed.

Here we will be using the UMBC phrase similarity API for our semantic phase evaluations.

The API link is :

```
$score= "http://swoogle.umbc.edu/SimService/GetSimilarity?operation=api&phrase1=$my  
&phrase2=$in&corpus=Refined%20Stanford%20WebBase%20corpus&type=Relation%20  
Similarity";
```

Here phrase1 and phrase2 are the two input parameters that should be provided programmatically in the program. The API fetches us the score on successful semantic comparison of the two phrases. The score can be fetched using file_get_contents method in php, here \$val will give us the formatted score value in decimal format.

```
$val = (string) file_get_contents ($score);  
  
$val = floatval($val);
```

This process is used to help in implementing the step score evaluation method and combined score calculation method. Both these methods are implemented in different way. The prior one requires two passes for score evaluation where in the first pass we calculate the title scores check against the threshold and save ids above the threshold. In the second pass we compute the scores from course description and category for saved ids from first pass.

Two Pass Score Evaluation → First Pass:

Phrase1: Introduction to Programming → from MOOC site

Phrase2: Introduction to Programming in Java → from SJSU site

The score is 0.834 which is above threshold of 0.74. The course id of the phrase 2 will be stored in the cache block.

Second Pass:

Phrase1: Basic skills and concepts of computer programming in an object-oriented approach using Java. Classes methods and argument passing control structures iteration. Basic graphical user interface programming. Problem solving class discovery and stepwise refinement. Programming and documentation style. Weekly hands-on activity. → from SJSU site

Phrase 2: In this class, you will learn basic skills and concepts of computer programming in an object-oriented approach using Java. → Course description from MOOC website

The score is 0.774 for this comparison. Also the comparison of category will generate a score of 1.0 as both of them have same category of computer science.

So we calculate the weighted score as,

$$\text{Weighted Score} = (5*0.834 + 4*0.774 + 1*1)/10 = 0.8267$$

So the probability that these two courses are equal is 82.67%. In similar way we carryout te course comparisons for this method in two passes.

In One pass score evaluation process we have only one pass wherein we calculate all the three courses. Then we compute the weighted scores.

1.3 Mobile Presentation of Data:

For displaying the data on mobile we are using the Phonegap platform, the application that we are building is using the jqueryMobile framework. The framework provides all the Mobile UI components, event handlers and listeners along with the Phonegap API's.

Figure 15: Application communication with Server

The mobile application provides user interface for user to check the courses of his interest. The search box allows the user to enter complete course title or he can provide keywords into the search box. The request to server will be an Ajax call with input parameters to the server.


```
$.ajax({
  type: 'POST',
  dataType: 'json',
  data: {course_name: course_name},
  url: 'http://localhost/Course/GetCourse.php',
  success: function(data, status, xhr){
    console.log(jQuery.type(data));
    json_new = data;//JSON.stringify(data);
    //json_new = JSON.parse(data);
    console.log(jQuery.type(json_new));
    $.mobile.changePage("#dataPage");
    $(document).trigger("dataPageFired");
    console.log(json_new);
  }
}
```

In the above snippet we see the ajax call made to the server, using course-name as input parameter. We are expecting json as the return type data from the server. The server will receive the course details for which user has requested. It will send a query to MySQL DB for fetching the relevant courses.

```
$course = $_POST['course_name'];
if($course !== null){
  $crse = explode(" ", $course);
  foreach($crse as $course){
    $sql = "SELECT distinct course_data.id,course_data.title, course_data.short_desc,course_data.course_link,
    course_data.video_link,course_data.start_date,course_data.course_length,course_data.course_image,
    course_data.category,course_data.site,coursedetails.profname,coursedetails.profname
    FROM course_data, coursedetails
    where course_data.id = coursedetails.id and course_data.title LIKE '%$course%'
    Group By course_data.id,course_data.title";
    $result = mysql_query($sql) or die(mysql_error());
  }
}
```

The course details will be then formatted according to the user needs and converted into the json file using json_encode() method which converts array to json format. This json file will then be sent back to the application to display the result.

Figure 16: Application page when launched

The above figure shows the UI screen when the application is launched, the user can also use the top categories option to check the courses. So when the user selects one of the top four categories of courses, an Ajax call is made to the server and json file is retrieved. The json file is then used in dynamically generating the html page. The result that is sent from the server consists of scores w.r.t entered query, which are having scores in decreasing order of weighted scores.

Figure 17: Course result page on mobile

If the user wants to compare courses from his university with the MOOC courses then he can select the university option. Using this option will allow the user to enter the course id of the course. For example, if the user is interested to know if the MOOC courses are having similar course as “Introduction to Programming” having course id “CS 046A”. He will enter the “CS 046A” in the input and request the details through the application. The process is similar as mentioned above to get data from server. The output will be the best possible matches from MOOC with the university course.

Figure 18: MOOC Courses similar to “CS 046A”

The above result shows that the “CS 046A” course from computer science department catalog has best match with MOOC course “Introduction to Programming” provided by Udacity.com. The user can then click on the course image to look at the YouTube video, which provides an introduction to the course. He can also click on the course title which will direct the user to the course homepage.

5. Technical Challenge:

One of the main problems in this project was to fetch data from dynamically generated html pages and also its sub pages. If we consider the coursera.org website we find that the courses are getting fetched at runtime when we scroll down the page. Hence whenever we try to use the `file_get_html()` method to get the source of the page we do not get any html content.

Figure 19: Source of Coursera.org page with no HTML data.

For this purpose we had to find an alternative, either we had to perform screen scraping of the webpage which results in generating irregular data. This data will be of no use as we

cannot effectively differentiate the data for each course. This process also requires manual intervention and the results are sometimes unpredictable. Another way out was to ping the source server of the Coursera to fetch the data directly and then extract it accordingly. After a considerable amount of research we could find the link, from which we could fetch json data from the source server. The developer tool provided by Google chrome came handy, the network module helped in getting the link to the source server. Once we got the json data, we could convert it into a multi-array data structure in php. This data also provided link to every sub-page for 2nd level parsing of data. For 2nd level parsing the individual course page was again a json file. We had to follow the same process to get it, as with the main page.

Figure 20: Json data link for coursera.org.

6. Testing and Results:

1.1 Data Validation:

In this testing process we are checking whether data extracted from the websites are extracted properly and the format is structured. We are comparing the results between the data stored in the DB on extraction and with the output from the data presentation phase. We checked for random three course in this test scenario.

Data Validation Check		
Id = 5	Age of GlobalizationU	PASS
Id = 152	Nutrition and Physical Activity for Health	PASS
Id = 242	Intro to Sustainability	PASS

Resulting JSON data

```
[{"id":"5","title":"UT.3.01x : Age of GlobalizationU","category":"Engineering","website":"Edx","prof_name":"John Hoberman"}, {"id":"152","title":"Nutrition and Physical Activity for Health","category":"Health & Medicine","website":"Coursera","prof_name":"John M. Jakicic and Amy D. Ric"}, {"id":"242","title":"Intro to Sustainability ","category":"Science","website":"Canvas","prof_name":"Kyle Crider"}]
```

Figure 21: Data validation check list.

Figure 18 shows, we can see the json output which shows the data extracted from the database.

VideoLink	Course Name	Category	Start Date	Duration	Instructor	Image	Initiative
UT.3.01x : Age of GlobalizationU	Engineering	2013-09-01	16	John Hoberman		Edx
Nutrition and Physical Activity for Health	Health & Medicine	2013-09-09	6	John M. Jakicic and Amy D. Ric		Coursera
Intro to Sustainability	Science	2013-05-06	10	Kyle Crider		Canvas

Figure 22: Course output for Testing.

7. Conclusion and Future work:

The goal of accessing semi-structured and unstructured data was possible by using Dom parsing, http programming and also by devising wrappers for detecting pages, that have information of our interest. For unstructured data we used string functions to segregate the data into sentences. Then analyzing every sentence, by comparing them with the actual data from the site will give us accurate data. Furthermore to fetch the course data from coursera.org we had to identify the access link to the source server. With the help of Google chrome developer tool this was possible. Once the required quality of data was achieved we used the information to calculate the similarity among them.

The semantic phrase similarity is more valid way of checking relativity among the courses than lexical comparisons among the phrases. The results show that semantic phrase similarity also checks for parts-of-speech of every word in the phrase. The advantage being every word will be checked and scored against its semantically related words. The UMBC research on phrase similarity has provided us the advantage on carrying out our analysis. But an enhancement to this project would be developing a semantic phrase similarity server, using the corpus of only course information across the web. This corpus consisting of course information will help us in evaluating every course efficiently and the scoring between two courses will be of higher accuracy.

We started initially with the assumption that every data on web is straightforwardly accessible, which can be retrieved using simple parsing techniques. With growing advancements in front-end frameworks like backbone.js, require.js has provided more flexibility in handling data. Moreover popularity of NoSQL DB's has given access to store semi-structured and unstructured data. Retrieving data from such sources will be challenging. With ever changing the data, the scripts to access it will also need regular modifications to maintain high accuracy of retrievals.

We have used Phonegap platform in our project to display the results of our project on mobile. With multiple OS for mobile in the market, developing an application for only one platform will be not useful. Phonegap platform makes it possible to launch a single application on multiple platforms. Another advantage of using Phonegap, the developer need not know native programming language for every mobile platform. Application development in this platform requires one to have a good understanding in web technologies like HTML5, jQuery and CSS. To compile the code into binary and exec files we have the Adobe Phonegap build which makes the job easy for developers.

8. References:

- [1] UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems, by Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield and Jonathan Weese, http://ebiquity.umbc.edu/_file_directory_/papers/674.pdf
- [2] "The W3C Document Object Model (DOM)", Le Hégaré, Philippe (2002). World Wide Web Consortium. Retrieved January 10, 2009 <http://www.w3.org/2002/07/26-dom-article.html>
- [3] Php SimplehtmlDom Parser, Retrieved on May 6th, 2013 from website: <http://simplehtmldom.sourceforge.net/>
- [4] "Document Object Model (DOM)". W3C. Retrieved on 2012-01-12 <http://www.w3.org/DOM/#what>
- [5] Phonegap for developers, Retrieved on May 5th, 2013 from website: <http://phonegap.com/>
- [6] Semantic Text Similarity Using Corpus-Based Word Similarity and String Similarity, by Aminul Islam and Diana Inkpen, <http://140.134.131.64/wiki/images/1/1f/SemanticTextSimilarityUsingCorpus-BasedWordSimilarityandStringSimilarity.pdf>
- [7] Graph of Relations, by Anupam Joshi and Lushan Han, retrieved May 2nd, 2013 from website http://swoogle.umbc.edu/SimService/phrase_similarity.html
- [8] The Evaluation of Sentence Similarity Measures, by Palakorn Achananuparp, Xiaohua Hu and Shen Xiajiong <http://www.cis.drexel.edu/faculty/thu/research-papers/dawak-547.pdf>

- [9] The Phonegap/Cordova Misconception, May 2012,
<http://mobile.dzone.com/articles/phonegapcordova-misconception>
- [10] XAMPP for developers, Retrieved on May 6th, 2013 from website:
<http://www.apachefriends.org/en/xampp.html>
- [11] Coursera, <https://www.coursera.org/>
- [12] Edx, <https://www.edx.org/>
- [13] Canvas, <https://www.canvas.net/>
- [14] Udacity, <https://www.udacity.com/>
- [15] San Jose State University, <http://info.sjsu.edu/web-dbgen/catalog/degrees/all-degrees.html>