Spring 2014

SEMANTIC SIMILARITY BASED INFORMATION RETRIEVAL AS APPLIED TO MOOCs

Krishna Nitin Tenali
San Jose State University

.

SEMANTIC SIMILARITY BASED INFORMATION RETRIEVAL AS APPLIED TO MOOCs

A Thesis

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 Science

By

Krishna Nitin Tenali

Dec 2013

The Designated Thesis Committee Approves the Thesis Titled

SEMANTIC SIMILARITY BASED INFORMATION RETRIEVAL AS APPLIED TO MOOCs

by

Krishna Nitin Tenali

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

Dec 2013

ABSTRACT

Over the last few years there has been a significant development in the e-learning industry that provides online courses to the public. Due to the drastic improvement in technology and the Internet, this form of education reaches many people across boundaries. There is vast set of courses currently provided by various sources, which range from the latest technologies in the field of computer science to any topic in history. Since the invention of e-learning, there has been constant improvement of user friendly tools to enhance the learning process. In the span of the last three years, many websites have come into existence that provide online courses. Some of the best universities in the United States and other universities throughout the world have also started to provide online courses that students can easily attend. It has become very difficult for a student to pick the right online course. Hence, an application that can integrate the courses provided by various e-learning websites like Coursera, Udacity and Edx would be very helpful. The student can compare the regular courses provided by his or her university with the courses offered by the e-learning websites and can enroll in similar online courses to get a better understanding of the subject.

The objective of this project is to use semantic similarity techniques to identify the MOOCs [massive open online courses] offered by the e-learning websites which are similar to the regular courses offered by the university.

ACKNOWLEDGEMENTS

I would like to thank to my advisor Dr. Chris Tseng for his continuous guidance and support throughout this project. Also I would like to thank the committee members Mr. Ronald Mak and Mr. Venkatasubramanian Soundararajan for their suggestions and valuable advice.

TABLE OF CONTENTS

List of Tables and Figures

1. Project Overview

1.1 Introduction

The Internet is a huge collection of data that is mostly semi-structured or unstructured. It is very important for websites to be light and responsive. They are designed in such a way that data retrieval is very quick and efficient. Hence the websites do not use the structured data model that is normally associated with databases but instead use XML or JSON to enable huge amounts of data to be stored and retrieved efficiently. The technologies used in the Internet are designed in such a way that each website can store and present the data differently. With the Internet expanding across the world and the data on the Internet is accessible by everyone, that makes Internet the most efficient and effective platform. Due to the flexibility and scope of the Internet, a lot of development is taking place to invent new technologies to constantly improve the visibility of its content. Many efficient methods have been employed for the storage and retrieval of data. The same content and its interface can be developed in multiple ways using various technologies due to the lack of a fixed structure or hierarchy of data across the Internet. Due to the constant changes involved, designing and developing a standard algorithm or technique to extract data from all the web pages has become very difficult.

In this project we have considered only San Jose State University's regular course data and the MOOCs [massive open online courses] data. The data from San Jose State University and MOOCs websites like Coursera, Udacity, and Edx are extracted using various data extraction techniques, and the data is stored in a relational database. The goal of the project is to build a web application that will take San Jose State's course

details as input and use a semantic similarity technique to compare items such as the course title and course description against MOOCs in order to identify similar courses. The purpose is to help the student pick the most relevant MOOC compared to the University's regular course. This would help the student to learn the subject in detail.

The project is divided into two stages. The first stage extracts the data from MOOC websites using web mining techniques. The second stage compares the courses and gives a score based on the level of similarity using semantic similarity techniques.

1.2 Introduction to Web Mining

Web mining is the process of identifying data patterns on the Internet by using various data mining methods. Internet content is generated from different data sources. The mining of content on the Internet can be divided into three sections.

1.2.1 Web Usage Mining

Many companies and educational institutions are investing time and effort in research to find new methodologies to improve the process of mining data from the web. Web usage mining is especially useful for e-commerce websites, where the company is concerned about the user's interest and would like to present the data in their website according to the user's interest. The data from server logs and mouse events of the user are captured and the data is used to identify the user's requirement. Some users might look for textual data and some might look for media. This information is later used to analyze and predict the user's interest. Then the user is provided with information they

are looking for, thereby making the website more user-friendly. Typically a web application would store the IP address of the user, the sections of the web page accessed by the user, the searches made by the user, and the user's mouse events.

1.2.2 Web Structure Mining

Web structure mining is a method to understand the relationship between various hyperlinks available in a website. This would help the website designers to understand the hierarchy of the website and to connect relevant information through links. These links will help the user to access the content of the website.

1.2.3 Web Content Mining

Web content mining is an efficient method to extract useful data from the web page contents. Since the data is semi-structured, the lack of a data model makes it very difficult to develop a standard technique to identify the information from the websites. Hence the extraction of this data is also challenging. Websites generally use XML or JSON to store the data because those formats are very easy to handle and do not have any data model like a relational database. Web content can be divided into two points of view, the IR view and the database view. The IR view or information retrieval view, deals with extracting information from semi-structured data source. The database view tries to determine if the data is stored in the database. In this project, we focus on web content mining and use various techniques to extract the data from the websites.

1.3 Introduction to Phrase Similarity

Phrase similarity is the second stage and the core portion of the project. Course data from San Jose State University and the MOOC websites are extracted using content mining techniques. Then a regular San Jose State University course is compared with all MOOC courses using the course description from the respective courses and the similarity score calculated using various semantic similarity techniques. Phrase similarity techniques can be broadly classified in two groups.

1.3.1 Lexical similarity

If the language is similar then lexical similarity would help us analyze two word sets and identify the measure of similarity between them. The similarity scores are in the range of zero to one, where one means that there is a complete overlap between the two word sets and the word sets are very similar to each other. If the score is zero then there are no common words between the two sets. This technique can also be used to find the genetic relationship between languages. If the scores are high it can also be assumed that the two languages can be dialects.

1.3.2 Semantic similarity

Semantic similarity is used to identify the relatedness of two word sets. This technique is used in various applications related to artificial intelligence, information retrieval, and natural language processing. The scores are usually in the scale of zero to one. If the two word sets are similar then the scores are closer to one and if the score is

zero then the word sets are not similar. Some of the common techniques use to implement semantic similarity are the hierarchical model or the statistical model like the vector space model. In both the techniques the word sets are represented as nodes and the similarity score is calculated based on distance between the two word sets. If the distance is more, then the word set are less similar and if the distance is less, then the word sets are similar. Semantic similarity measures can be classified into the following categories like topological similarity, edge-based, node-based, pairwise, groupwise, statistical similarity and semantics-based similarity.

2. Data Extraction

Data extraction is the first stage in this project. The course-related data from San Jose State University and MOOC websites are the data that we will be working on, so it is very important to extract the data from all these websites. Since we are concerned only about the content on the websites, we use web content mining to extract the data. Web content mining is very similar to text mining and data mining. Some of the methodologies used in data mining are used in the web content mining since both deal with information extraction. In web content mining the data is either semi-structured or unstructured whereas in data mining the data is more structured. Due to the exponential growth in web content and its usage, many applications are built to use this data and present the data in a user-friendly manner. There are various problems involved in extracting data from the Internet due to lack of a standard structure used by websites to store the data. Since the solution for these problems is very important in various real time applications, a lot of research is being done in this field. Using web scraping we can

convert the semi-structured data from a website to a more structured format that can be stored in a relational database. Since each website stores and presents the data differently, the data we require might not always be present in the same location, and the same method cannot be used to extract the data across various websites. Every web page should be analyzed to exactly determine the location of data. Some websites display information that is not required. It is critical to extract only the information that is required and scrap the rest of the data before storing the data in the database. Otherwise, it would involve a lot of overhead to process the data every time it is fetched from the database. Some of the techniques used to implement web scraping involve the use of a DOM/HTML parser.

2.1 DOM Parser

The Document Object Model is a standard to interact with the objects stored in XML and HTML documents. All browsers use a model like DOM to render an HTML page. Typically, HTML code consists of nodes, these nodes are in a tree like structure known as the DOM tree with the document object forming the topmost node. When a browser renders an HTML document, it parses the document in order to display the contents. The HTML Parser accesses and traverses the HTML document. It is also important that the HTML Parser identifies various HTML tags. The HTML parser also provides various functions to extract specific portions of the HTML content. HTML parsers are written in various languages. For this project we have used Jsoup.

2.2 Using Jsoup

Jsoup is a Java library that provides various methods to extract and manipulate the data from an HTML document. Jsoup reads the HTML document and parses it similarly to the DOM parser to identify various nodes. These nodes are represented in a tree structure, and this tree can be traversed and the required data can be extracted using various methods provided by Jsoup. The following are some of the methods in Jsoup to extract data.

Document doc = Jsoup.connect("http://www.sjsu.edu/").get();

This above method can be used to render an HTML document and find the required data. The “connect” method make connections to the URL provided and “get” method parses the HTML document. An exception is thrown if there is any error in fetching the HTML page.

Elements getid = doc.select("div[id=top_subsite]");

The “select” method helps to traverse the HTML document and go to the particular location where the required data is found. In the above case we traverse to the “div” tag in the HTML page where the “id” attribute is “top_subsite”

String value = getid.get(1).attr("href")

The “get” method extracts data from the HTML page. The “attr” method helps us to selects the value for the attribute “href”.

String content = getid.get(1).text();

The text method can be used to extract the text portion between the HTML tags.

```
Elements list = getid.select("a[href]");
for (int i=0;i<list.size();i++)
{//Extract a particular data
}
```

The above snippet of the code is used to loop over all the nodes in the HTML document with *"href"* attribute and extract data. There are many other methods provided by Jsoup which would help us to efficiently extract data from an HTML document. Jsoup is a very efficient API and to use all its methods, we can import its jar file into our Java program. In this project we have focused on extracting course related data from San Jose State University, Edx, Udacity, and Coursera.

2.2.1 Extracting San Jose State University's data

The objective of the project is to compare similarity or relatedness of a regular course offered at San Jose State University with MOOC courses. The first step is to identify the hierarchy used in San Jose State University course website and understand the location of data. The data extraction process begins with the extraction of course related information. There are about seventy four departments in San Jose State University, and each department offers many courses.

We parse the HTML document which contains all the departments in San Jose State University http://info.sjsu.edu/web-dbgen/catalog/departments/all-departments.html.

Using Jsoup the list of department names and their corresponding URLs are extracted. This data is stored in hash map with department names as keys and the URLs as their values. The department URL is then parsed and the course-related URL is extracted. This HTML document contains the list of courses offered by a particular department. The value of the href attribute of each course is extracted using the "attr" method. This value would give us the URL to the particular course.

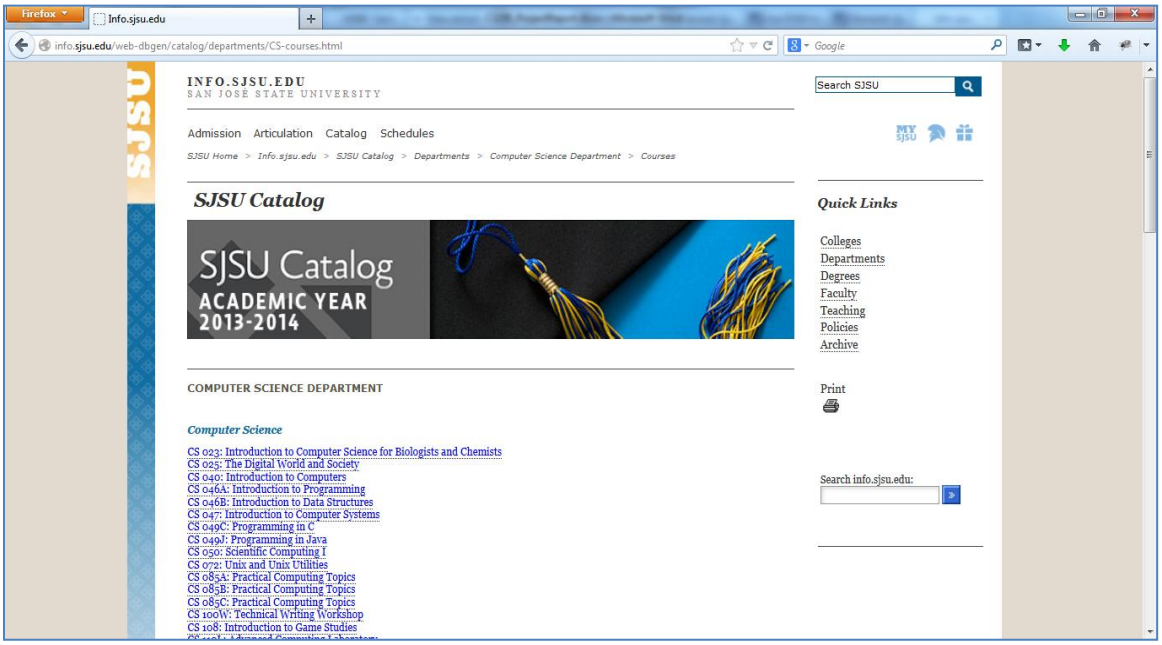


Fig 1: San Jose State University web page that contains course data from the Computer Science Department.

We parse this particular HTML document and identify the tag which contains data related to course name, course number, description, grading and units. This data is extracted and stored in the database.

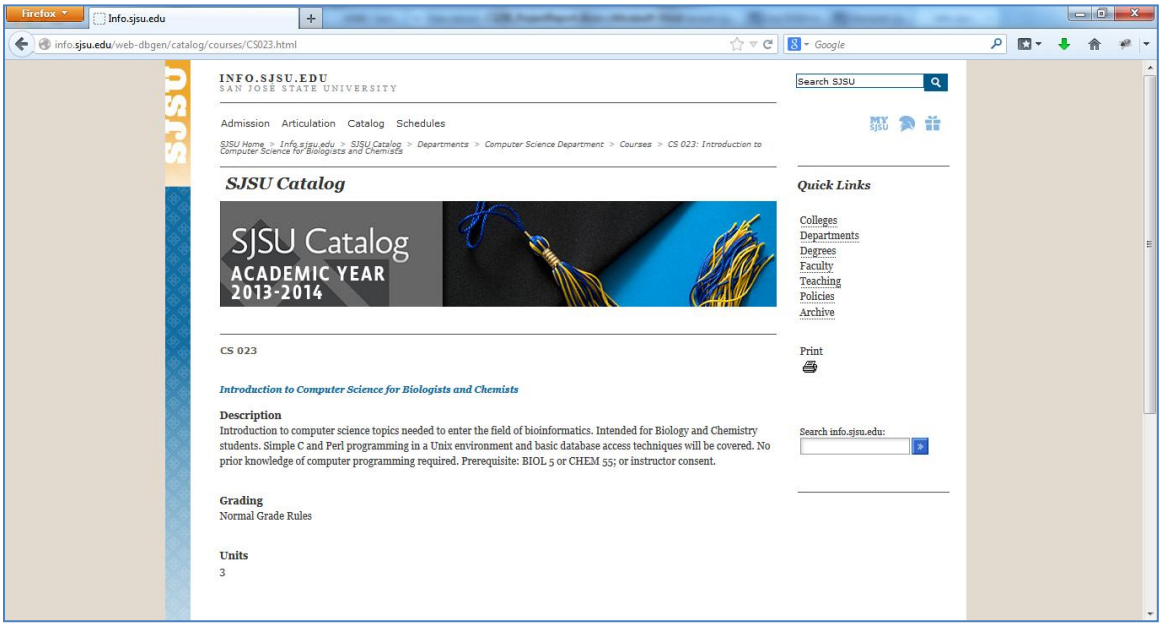


Fig 2: Web Page containing course-related information.

We parse this particular HTML document and identify the HTML tags with data related to course name, course number, description, grading, and units. This data is extracted and stored in the database. There are some challenges while extracting prerequisite related information, since this information is part of the description tag. The description is parsed and then if prerequisite data is available then that portion of the data is stored separately. Some course don't have prerequisites and some have only prerequisite data and not course descriptions, Java exceptions are used to handle each case separately. The semi-structured data related to courses are extracted from San Jose State University's website and is stored in the database.

2.2.2 Extracting MOOCs course data

After extracting course-related data from San Jose State University's website,

now the second stage is to extract the course-related data from MOOC websites like Edx, Coursera, and Udacity.

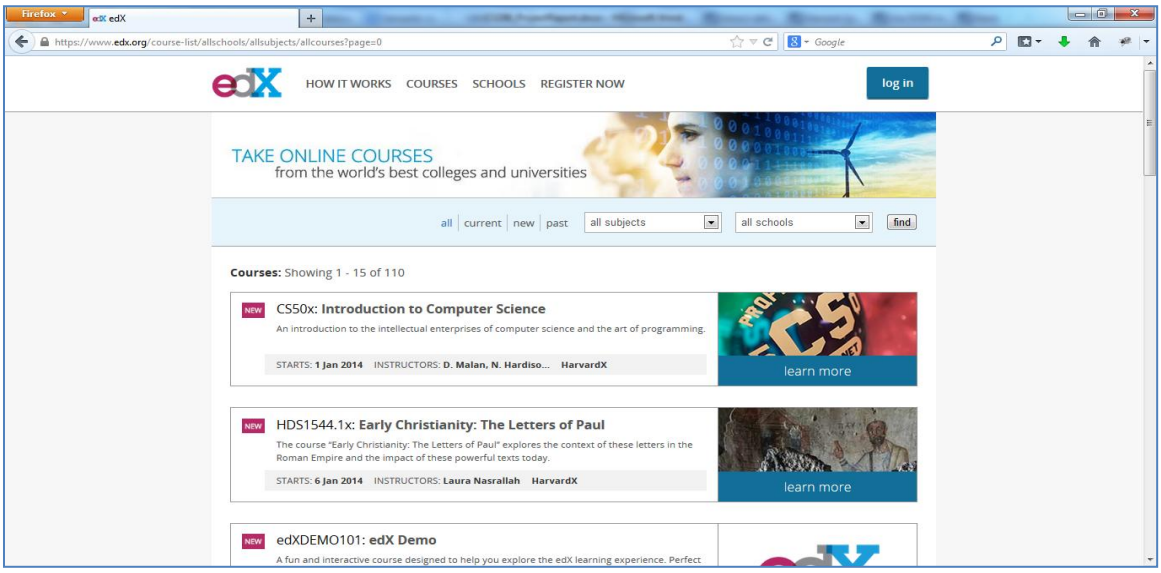


Fig 3: Web Page from Edx containing course information.

Each website uses a different hierarchy to store its data. In the case of Edx, we parse the HTML document at https://www.edx.org/course-list/allschools/allsubjects/allcourses?page=0 which contains the courses offered by Edx under all the categories. We identify all the courses, extract the course name and the URL related to each particular course. We parse the HTML document which contains information related to the particular course. Using the various methods available in Jsoup, we extract information such as course name, course description, course image, professor's name, start date, and video link. Some of the courses have exceptions in the description and start date information which are handled using Java exceptions.

Udacity is another MOOCs website whose structure is similar to Edx. By using the

"attr" and *"text"* methods from Jsoup, the course-related information is extracted from Udacity website. But in the case of Coursera, the course list is generated during runtime and do not have course-related content in the HTML document.

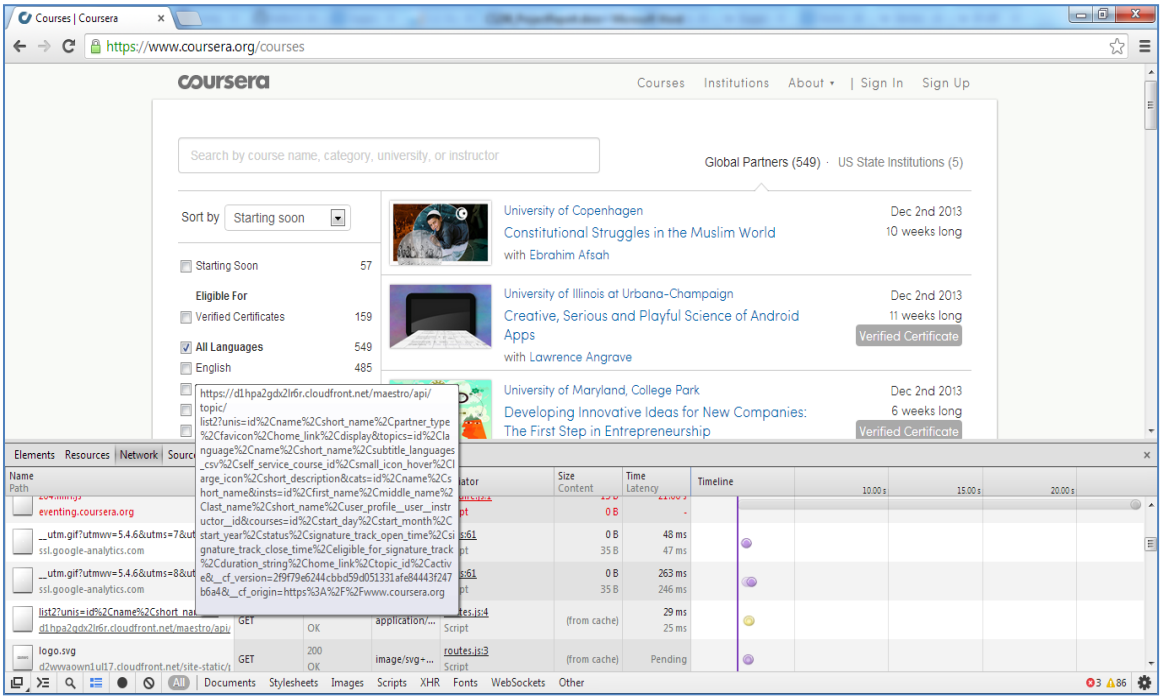


Fig 4: Web page from Coursera containing the course information.

Coursera stores the course-related data in a JSON file at the URL https://www.coursera.org/maestro/api/topic/list2. We parse the JSON and extract the topic id for each course and append the id to the URL https://www.coursera.org/maestro/api/topic/information?topic-id to get information related to a particular course. We parse this JSON file to extract the information such as course name, course description, course image, professor's name, start date, and video link.

3. Project Design

The project can be divided into three phases: the Data Extraction phase, the Similarity phase, and the Presentation phase.

3.1 Data Extraction Phase

In this phase we use some of the web content extraction techniques to extract semi-structured data from the websites and convert it into a structured data that will help us access the data effectively in the later stages. This extraction phase can be sub-divided into three stages: Data Identification, Data Extraction, and Data Processing.

3.1.1 Data Identification Stage

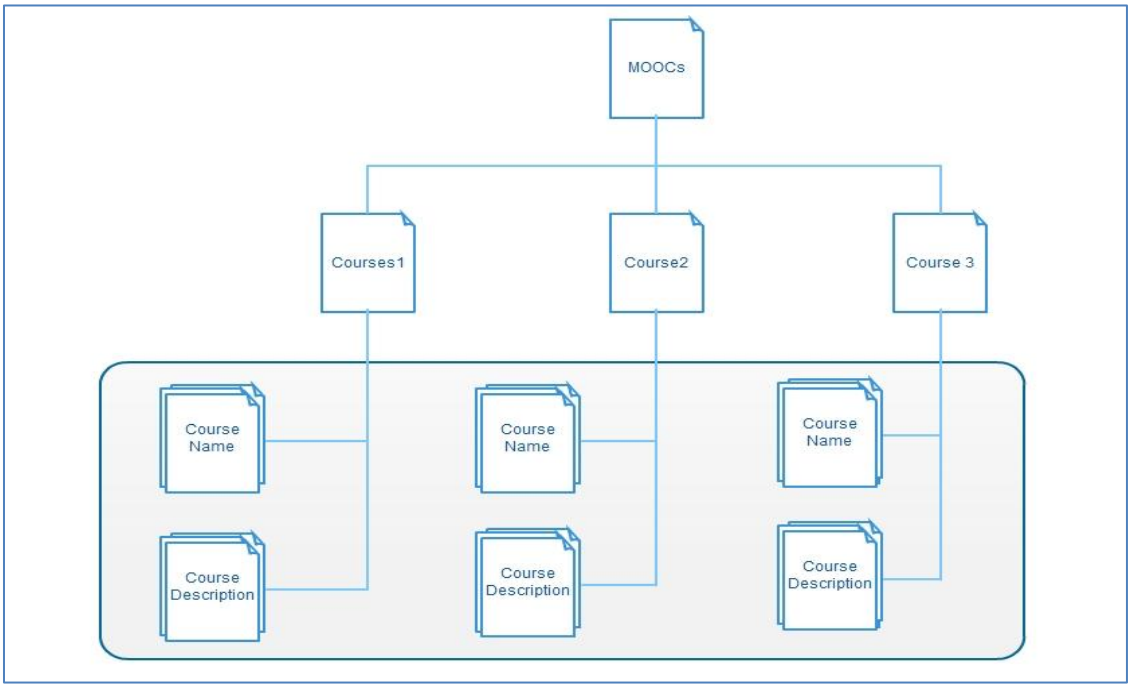


Fig 5: Data Identification stage.

In the data identification stage, we finalize the list of MOOC web pages that we will consider for this project. The data available in these websites are semi-structured data or unstructured. We have considered San Jose State University, Edx, Coursera, and Udacity. Each website should be analyzed in depth to understand the hierarchy of the website and the exact location of the data in these websites.

3.1.2 Data Extraction Stage

In the data extraction stage, we construct algorithms to extract information from the identified sources. We convert the static HTML document into a DOM tree in which the HTML tags form the nodes of the tree. We can access the child nodes and scrape the metadata of the web page to extract the relevant data. Using various methods provided by Jsoup we can access these sources, traverse through its web pages and extract data from them.

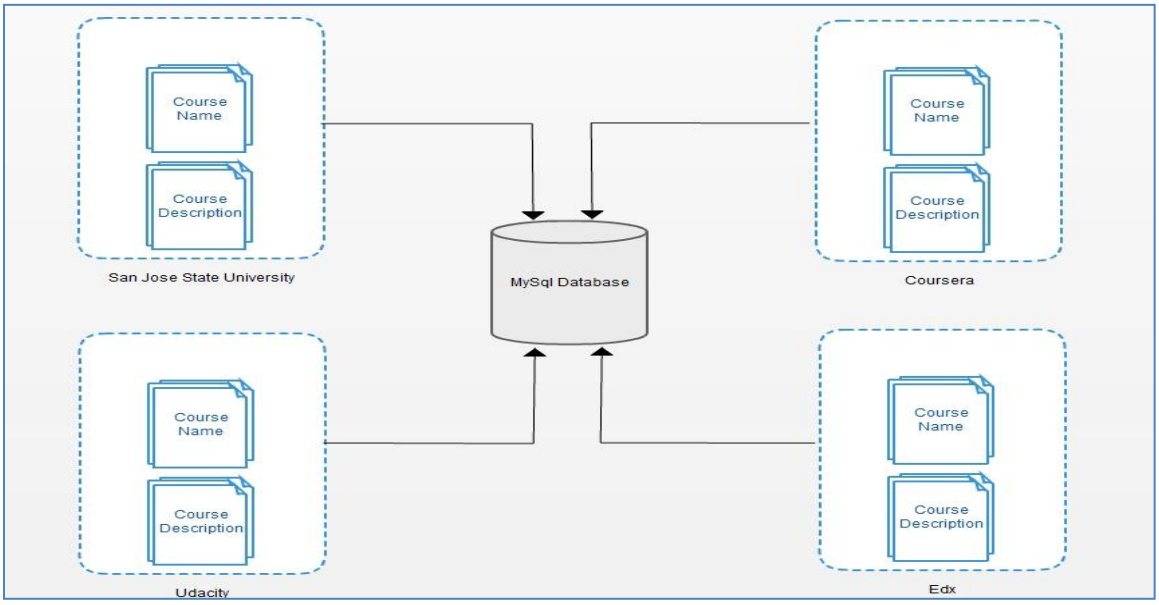


Fig 6: Data Extraction stage.

3.1.3 Data Processing Stage

We use four tables to store data. We store San Jose State University's data in courses table, and we store the course data scraped from MOOCs in course_data and professor-related data in coursedetails. We have a scorecompare table that is populated after the Similarity phase which contains the similarity scores of MOOC courses when compared with San Jose State University's courses. We have a foreign key and primary key relationship between the fields in course_data and course details tables.

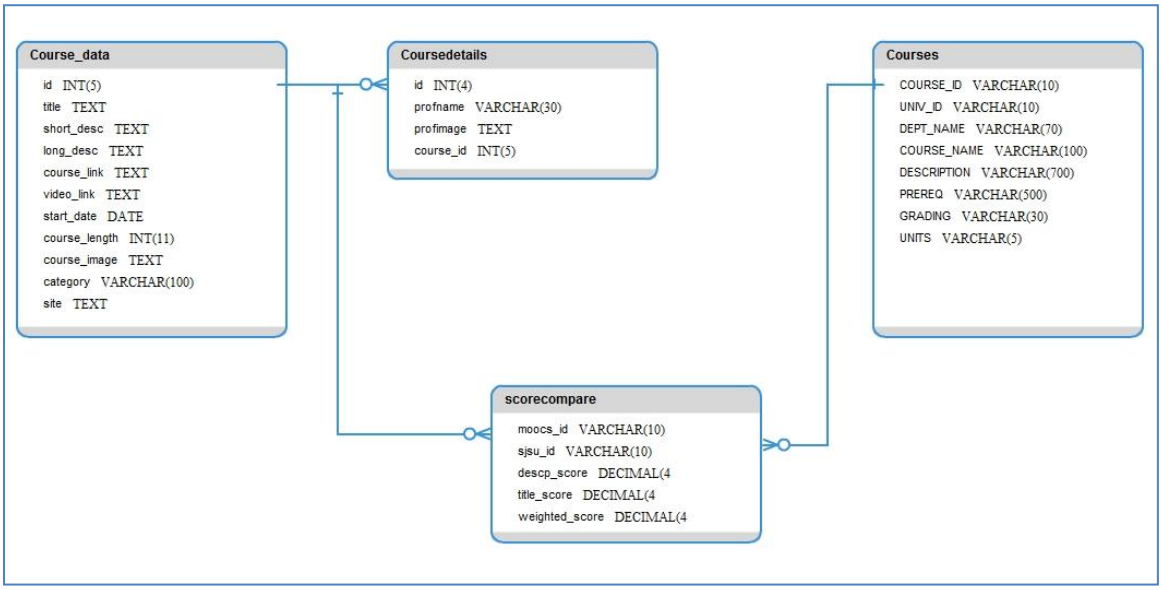


Fig 7: Data Processing stage

3.2 Similarity Phase

Similarity phase is the core section of this project that deals with semantic text similarity. Two courses are compared and a similarity score is generated. We compare all the MOOC courses with San Jose State University courses, and the corresponding scores

for each set of courses are then stored into scorecompare database table. To calculate the similarity score we use the system developed by TakeLab as a part of SemEval 2012. This system is developed in Python using WordNet and Support Vector Machines. WordNet is a lexical database that groups words into sets based on their semantic relation. This is a commonly used methodology in various text analysis applications. Support Vector Machines is a machine learning algorithm that consists of various learning models to analyze data and identify patterns. Both Support Vector Machines and WordNet will be explained in detail in next few sections.

3.3 Presentation Phase

XAMPP is an open source web server stack which helps developers publish their applications on their local system [10]. XAMPP is an acronym where X denoted cross platform, which can be deployed across different operating systems. A denotes apache HTTP server, M is for MySql database and PP denotes PHP and Perl script interpreters.

We designed and developed two web pages which we deployed on a local server using XAMPP. The web pages use PHP and the MySql database. The data scraped from San Jose State University and the MOOCs are also stored. We developed a web page to help the user identify the courses offered by MOOCs that are similar to courses offered at San Jose State University.

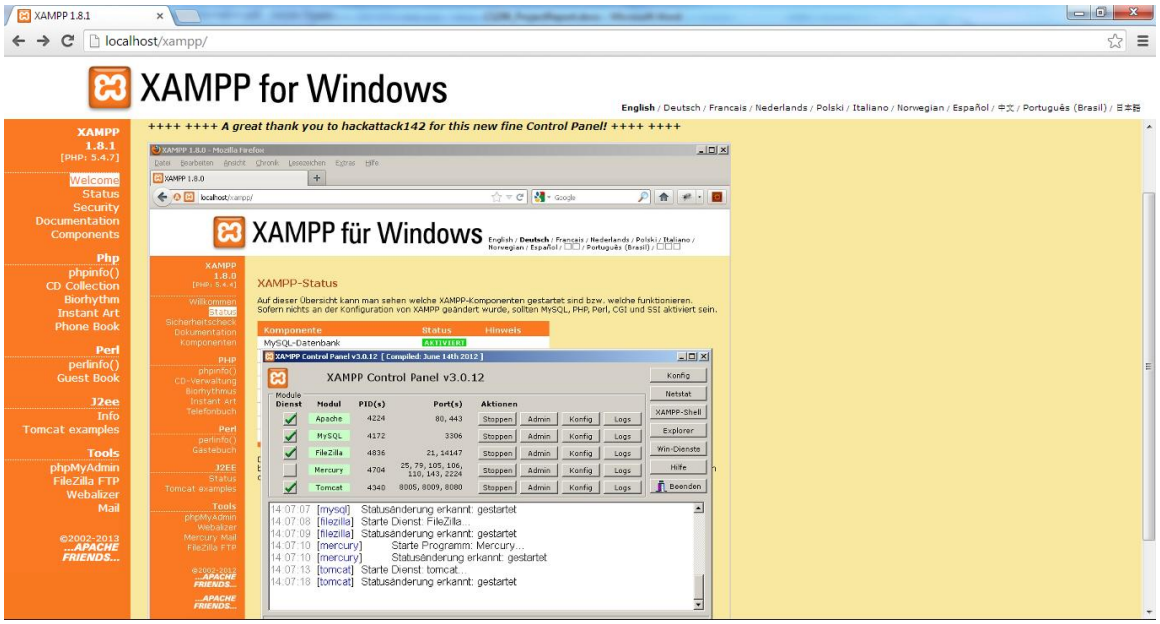


Fig 8: XAMPP

This webpage was developed using PHP, Jquery and MySql database. The following is Jquery snippet that implements AJAX functionalities like retrieving records from the database without refreshing the whole page.

```
$(document).ready(function() {
    $('#dropdwn1').load("getDept.php",function(){
    $('#dropdwn1').change(function(){
        var checkA = $("#dropdwn1").val();
        $('#dropdwn2').load("getCrs.php",{'dept': checkA},function(){
                $('#dropdwn2').change(function(){
                    var value = $("#dropdwn2").val();
                    $('#table1').load("getSJSUcrs.php",{'sjsucrs': value},function(){
                                });
```

```
                    });
                });
            });
        });
});
function test(id)
{
window.open("http://localhost/SJSU_MOOCS/Moocsdetails.php?val="+id ,"_blank");
}
```

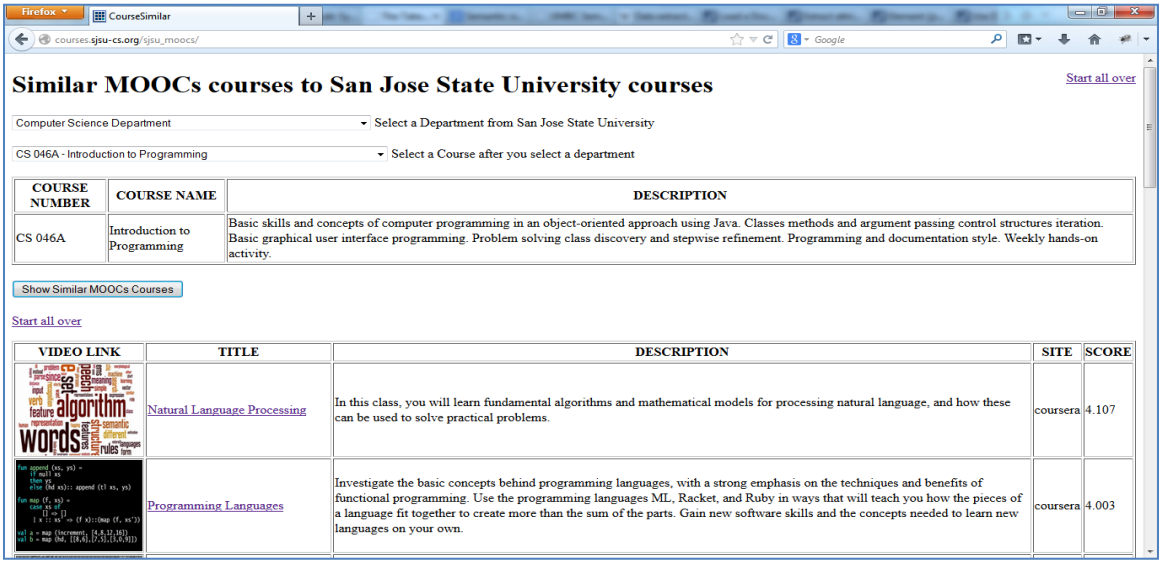


Fig 9: Web page to compare the courses.

The web page lets the user select a department from the first dropdown menu which is populated from the courses table in the database. This dropdown contains the list of departments at San Jose State University and allows the user to select a department. Using Jquery, we dynamically populate the next dropdown menu with the courses offered

by the selected department. We query the table course with the course name and retrieve the data and display a table with the course number, course name, and the course description. If the user is interested in finding the similar course offered by MOOCs that are related to the selected San Jose course, the user would click on the button.

We now query the scorecompare table with the San Jose State University course number selected by the user and retrieve the related MOOCs data from the course_data table and scores from scorecompare table. A table with MOOCs course name, course description, site name and scores are displayed. Since the video link data and course link data were also extracted from all the MOOCs websites, the link for the course and the video are hyperlinked to the course name and the course image. After analyzing the similarly scores of various courses, we have come to a conclusion that courses with similarity score of 3.8 and above are similar and courses less than that are not similar.

4. Semantic Similarity

Semantic similarity is a measure of identifying the level of relatedness between a set of texts. This has been researched in the field of natural language processing and machine learning. There are three approaches to implement semantic similarity.

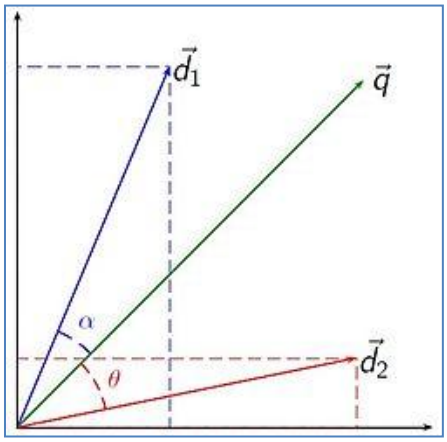
4.1 Vector Space Mode

Vector space model is an algebraic model in which the texts are represented as a vector.

$d = (d_1, d_2, d_3 \ldots d_z)$

$q = (q_1, q_2, q_3 \ldots qz)$

Each value in the vector is a non-zero value which weight of the term in the document. This weight can be calculated by tf-idf, i.e., term frequency-inverse document frequency. The similarity between texts can be identified using cosine similarity. This can be calculated by comparing the deviation of angles between the query vector and the document vector.



$$\text{Cos } \theta = d_2 * q \,/\, | d_2 | | q |$$

Fig 10: Cosine similarity.

In the above figure d_1 and d_2 are two document vectors and q is the query vector. The value of α and θ will give the relevance measure between the query and the two documents d_1 and d_2, respectively.

4.2 Corpus-based and Knowledge-based Measures

This technique is based on an assumption that if two word sets are semantically similar then we should be able to align their words. This alignment quantity is the similarity measure.

4.2.1 Corpus-based Measure

The corpus-based method depends on the degree of similarity between the two word sets using the information from a large corpus. The co-occurrence of a word with the count is collected in a large corpus where the degree of dependency between the two word sets is calculated. The occurrence of a term in the corpus can also be determined using singular value decomposition in a term by document matrix. When singular value decomposition is applied to the matrix, it decomposes into three matrices. Similarity is calculated using cosine similarity on a lower dimensional space and hence this method is more efficient than the vector space model [9].

4.2.2 Knowledge-based Measure

Knowledge-based similarity is dependent on WordNet. WordNet is a semantic network of words. The similarity between two words can be determined using their relative positions in the knowledge base hierarchy. The two words can have high similarity score if the words are in the same WordNet synset or if one word is a hypernym of another word [9].

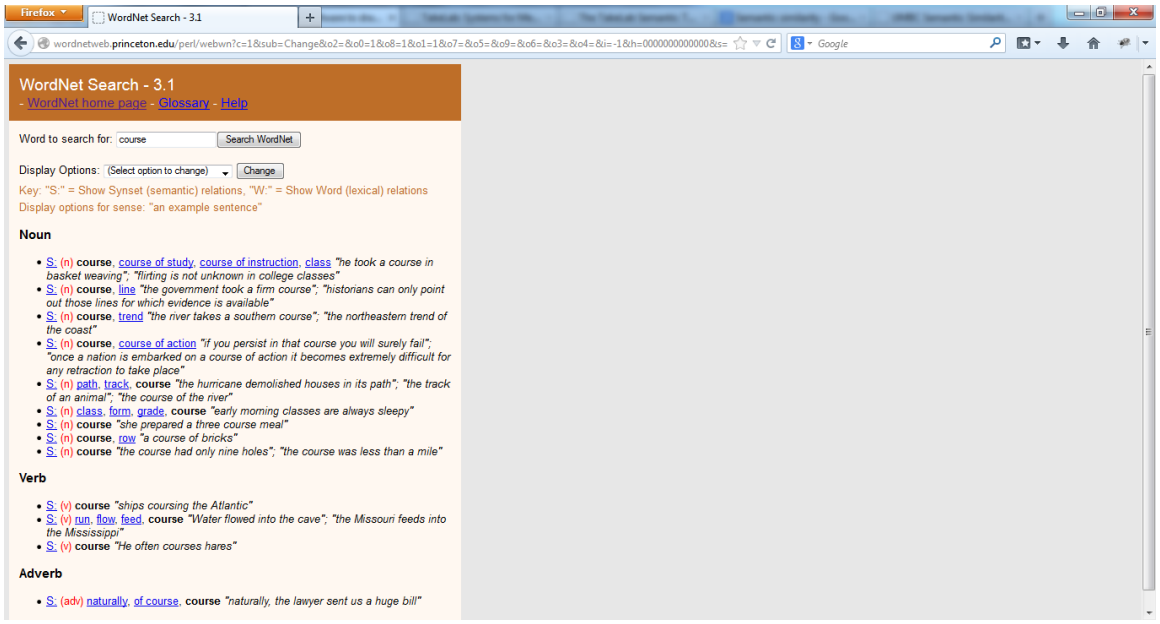


Fig 11: WordNet

WordNet is a hierarchically organized lexical database that groups words into synonyms called synsets. It provides semantic relations between synonym sets. Based on the grammatical rules, WordNet categorizes the words as nouns, verbs, adjectives and adverbs. Morphologic functions are used in order to deduce the root form of the word and only the root is stored in the database.

A sentence is partitioned into a list of tokens and these words are stemmed to find the root of the word. Using the WordNet database, an appropriate word is substituted and the similarity is computed based on the pairs of words.

4.3 Support vector machines

In this approach, different measures of machine learning models are used to analyze data. The support vector machine takes a set of input and predicts the scores for

a set of input. An SVM model is a representation of the data as points in space, and the data belonging to different categories are divided by a gap. The new data are then mapped into the same space and based on which category is predicted.

Some of the advantages of SVM are its effectiveness in high dimensional spaces. It uses only a subset of the training points in the decision function to make it very efficient. In this project we use a support vector regression and the model is developed from support vector classification in which only a subset of the training data is used. The training points that lie beyond the margin do not affect the cost function for the model. Analogously, the Support Vector Regression model depends only on a subset of the training data.

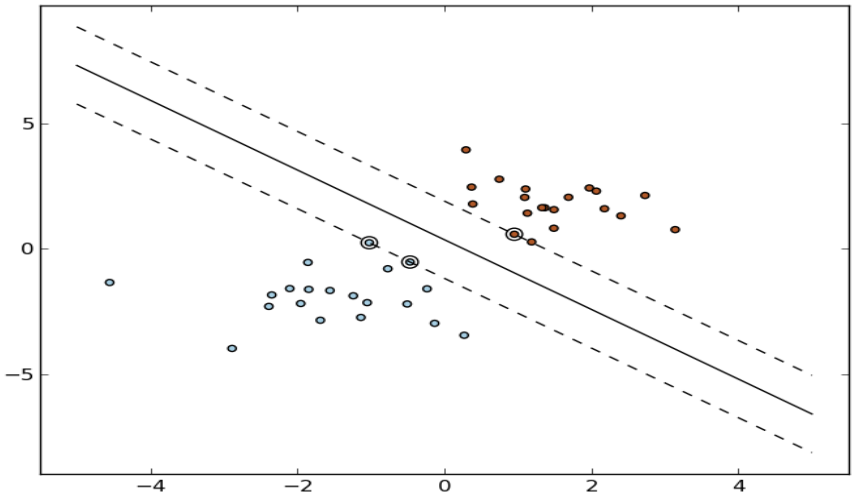


Fig 12: Support vector machine

5. Implementation

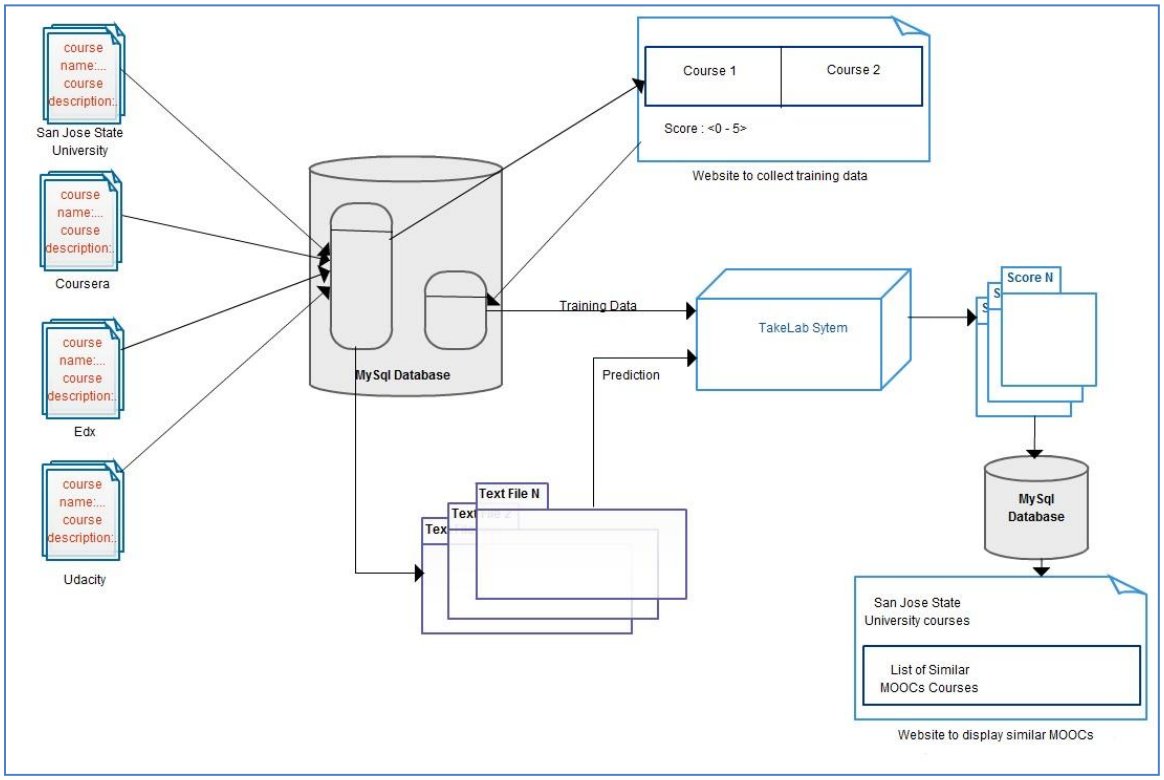


Fig 13: Project flow.

The initial step is to extract the data from San Jose State University's web site which can be done using "/SJSUCourses/HtmlParser/HtmlParser.java" and extract Edx data using "/SJSUCourses/MOOCS/src/moocs/Edx.java" and "/SJSUCourses/MOOCS/src/moocs/staff.java". The data will be stored in the database in courses and course_data tables respectively. In this project we have used the system built by TakeLab during SemEval. This system uses knowledge based measure from WordNet and support vector machines. WordNet is used to find the relationship between two words using a lexical database. Every word has a specific position in the knowledge-based hierachy tree depending on its context. The similarity score between two words is identified based on the relative positions on a hierachy tree. Each word has many

concepts based on the context. The NLTK library provides various methods to calculate the similarity score based on WordNet.While computing the similarity score we will take the maximum score over all the possible pairs of concept[10]

The text received as input during the training is preprocessed to remove all the stop words such as *the*, *is*, *at*, *which*, and *on*, special characters and also remove words like *not* and *am*. This preprocessing step will help the system to be more robust and focus on only the relevent words.

Using LIBSVM, the Support Vector Regression Model is trained with the training set and the SVM parameters C, g and p are genrated using grid search. This model is later used to compute the similarity scores of the sentences. This system is developed in Python and TakeLab also provides the training data that consists of pairs of sentences and their respective similarity scores.

5.1 Training Module

The training data provided by TakeLab consits of pairs of sentences and their respective similarity scores. These sentences are not specific to any context. The similarity score assigned to each pair of sentence range from zero to five, where zero is less similar and five is almost similar and these scores are based on human judgement. Since the existing training data is not related to courses, using this data will not help the system to predit the similarity correctly. For this project it is very important to use a training data related to courses. Hence we used the training data based on courses from all MOOCs websites and rated these courses based on our judgment. To effiectlvely

collect this traning data, we developed the web page to compare the course description of any two courses and let the user rate a pair of courses based on similarity.

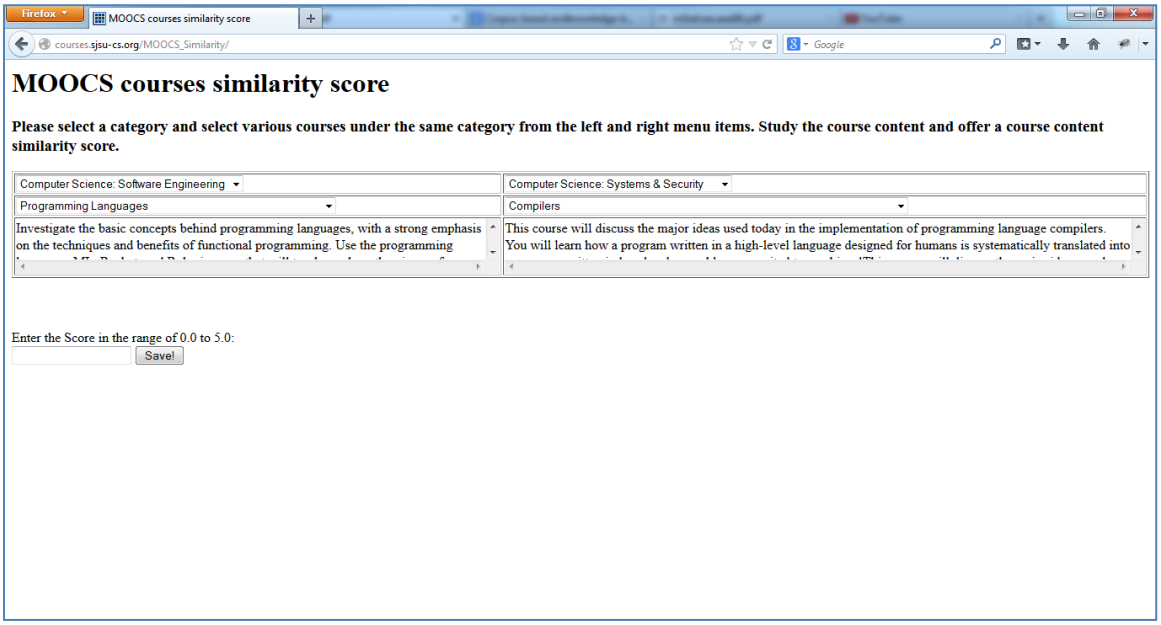


Fig 14: Web page to collect training data.

The web page "/MOOCS_Similarity/index.php" was developed using PHP and Jquery. It consists of two dropdown menus adjecent to each other. One menu contains the catogory data from the course_data table that contains the catogories of all the courses from MOOCs. Once the user selects a catogory using Jquery we populate the other dropdown menu with the list of courses offered in the particular catogory. When the user selects a course then we query the course_data table for the corresponding course. The course description of the particular course is displayed in the text box below the dropdown menus. Now the users can compare the course description between both the courses and rate the level of similarity. The program will check if the combination of

courses has already been selected. If the record is already available then we display the existing score for the combination. The user can retain the same score or give a new score to the pair of courses. The average of the new score and the old score is taken and updated in the table against the course. The training data used in this project is based on the course description. But if the training data needs to be changed to a different item like course name, the query used in "/MOOCS_Similarity/index.php" can be changed to retreive the respective column data and populate the dropdown menu. Now the user can compare the data that is based on a different field and can rate the new data based on similarity.

5.2 Prepare data for training

The traning data should be a text file containing pairs of sentences separated by a tab space between them and another text file with similarity scores for those pairs of sentences.

```
A woman is slicing an onion.    A woman is cutting an onion.
A person is peeling shrimp. A person is preparing shrimp.
A woman is cutting broccoli.    A woman is slicing broccoli.
A man is playing a guitar.  A woman is playing the guitar.
```

Fig 15: Sample training data.

In our project the training data will contain the pairs of course descriptions collected from the web page "MOOCS_Similarity/index.php". Using the Java program "/SJSUCourses/FileRead/src/fileread.java" we query the contents of the compare table and write it to a text file using file operations. We used the two text files to train the

model.

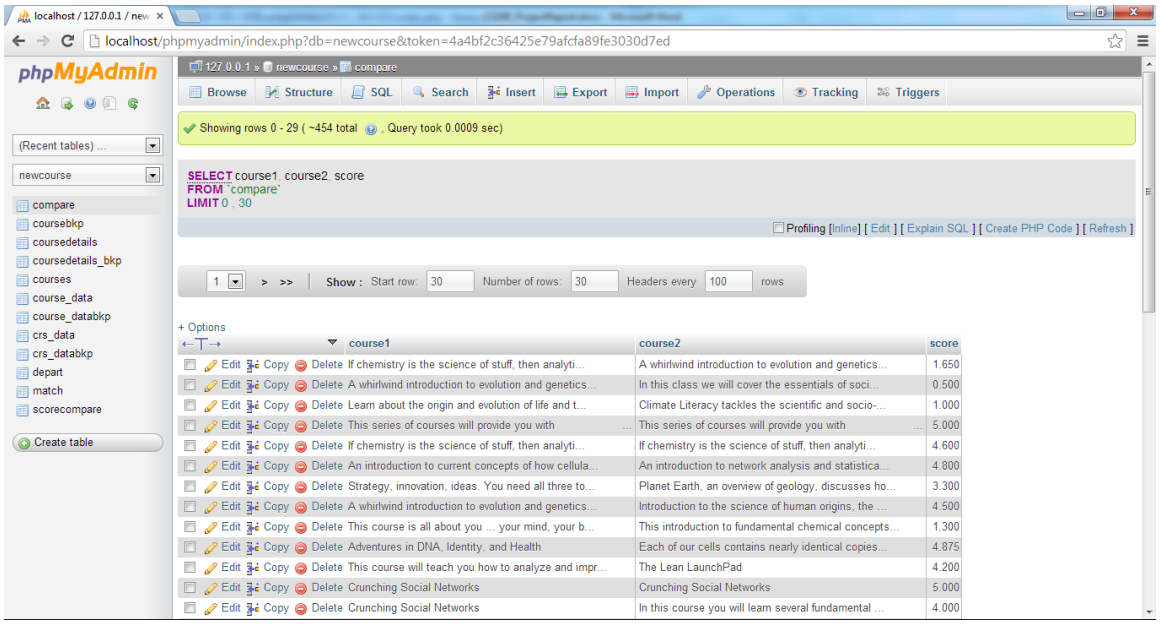


Fig 16: The database table that contains the training data.

5.3 How to Train

The text files that contain pairs of sentences and scores are taken as inputs for training. Training the model involves two steps. The first step is the preprocessing step where the stop words and special characters are removed, and WordNet is used to get a vector based on the similarity using the lexical database and its relative position in the WordNet tree. The optimal parameters are generated using grid search.

python takelab_simple_features.py <Text file with pairs of sentences> <Text file with scores> > <Output of this step is a set of vectors from WordNet>

svm-train -s 3 -t 2 -c 200 -g .02 -p .5 < WordNet output from the previous step >

model.txt

In the second step, we use the library LIBSVM to train the model and support vectors are constructed based on the optimal margin. The support vectors are then used to predict the score of new sentences.

5.4 Predict the scores

The input file containing the pair of sentences is preprocessed by removing the stop words and WordNet's lexical database is used to construct vectors based on the relative distance in WordNet tree.

python takelab_simple_features.py <Text file with pairs of sentences> > <Output of this step is a set of vectors from WordNet>

The system uses the support vectors generated for each pair of course descriptions for the courses corresponding to training data collected from the website to predict the similarity scores for new sentences. We use LIBSVM to process the input check with training model and generate similarity scores.

svm-predict < WordNet output from the previous step > model.txt <Output file to capture the scores>

Since we need to calculate the scores by comparing all the MOOC courses with the course offered at San Jose State University, the input files are generated with course description of the MOOCs courses and San Jose State University courses. There are around 600 MOOCs courses and 4024 San Jose State Universities course, so about 600

text files are generated with each MOOC course compared to all of San Jose State University's courses.

We automated the process of calculating the similarity scores for all the input files generated using the Python file similarityscore1.py. The filenames are stored in a list and using reading the input text file and using subprocess command we can execute Unix commands.

p = subprocess.Popen("python takelab_simple_features.py newResults/Input/" +filename +" > newResults/Input/" +predictfile,shell=True)

ph_ret = p.wait()

subprocess.Popen("svm-predict newResults/Input/" +predictfile +" model.txt newResults/Score/" +scorefile,shell=True)

Using the model generated after training and the Python script, the similarity scores of all the courses are generated. The scores are generated in text files and these files are used to populate the table scorecompare with all the similarity scores.

6. Conclusion and Future Work

The objective of the project to access semi-structured and unstructured data from the Internet, Parsing through the data was effectively accomplished by using Jsoup. The structure of the data was analyzed and location of the course-related data was identified. Using various methods provided by Jsoup we were able to extract the URLs and traverse through those web pages and extract the course-related data from San Jose State University's websites and the MOOC websites like Coursera, Edx, and

Udacity. This data was stored in the MySql database, thereby converting the semi-structured data is converted to structured data stored in a relation database.

Using the system built by TakeLab, using WordNet and Support vector machines. The data from the database was formatted to remove unwanted space and the training data was collected through a website. The users compare course descriptions of two courses and give a rating on how similar the courses are in the scale of zero to five with five being very similar and zero is not similar. Based on this training data, collected the model was trained and this trained model was later used to calculate the similarity scores between all the San Jose State University courses and MOOC courses. Using PHP and Jquery, a web page was designed to help to user select a course offered at San Jose State University and to identify the similar MOOC course.

As an enhancement to this project, a better training set can be collected and using which our model can be trained to provide better result. We can also include the course title in our comparison to get a more relevant result. We could also try to use a search engines like Nutch to crawl through various university websites which would help us to get a huge amount of data.

7. References

1. Web Mining: Information and Pattern Discovery on the World Wide Web
http://maya.cs.depaul.edu/~classes/ect584/papers/cms-tai.pdf

2. Web Content Mining and Structured Data Extraction and Integration

https://wiki.engr.illinois.edu/download/attachments/200017637/ResearchReport.pdf?version=1&modificationDate=1336540363000

3. WebDB: A System for Querying Semi structured Data on the Web

http://www.public.asu.edu/~candan/papers/jvlc01.pdf

4. Lu, H. (2005). Semantic Web Services Discovery and Ranking. The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2005), 157-160.

5. UMBC EBIQUITY-CORE: Semantic Textual Similarity Systems paper presented by Lushan Han, Abhay L. Kashyap, Tim Finin

6. D3 JavaScript data driven visualizations

http://d3js.org/

http://bl.ocks.org/mbostock

7. Jsoup API – HTML parser for Java.

http://jsoup.org/

8. Json parser for Java

http://json.org/java/

9. Corpus-based and Knowledge-based Measures of Text Semantic Similarity
http://www.cse.unt.edu/~rada/papers/mihalcea.aaai06.pdf

10. TakeLab: Systems for Measuring Semantic Text Similarity

http://aclweb.org/anthology//S/S12/S12-1060.pdf

11. Semantic Text Similarity system built by TakeLab

http://takelab.fer.hr/sts/