UniVisit

A Semantic Web Application - SER 594

Author's Name: Arpit Jaiswal, Devanshi Panu, Senthamil Sindhu, Sweta Singhal
Software Engineering, CIDSE, Arizona State University.

arpit.jaiswal@asu.edu , devanshi.panu@asu.edu, sblash@asu.edu , sweta.singhal@asu.edu

***Abstract*—This report elucidates UniVisit - a semantic web application which facilitates looking up information on universities and provides guidelines for users while visiting these universities. It provides weather statistics, information on crime rate and restaurants located near these universities. This favors recruiters, players, parents, exchange students, researchers and scholars who are planning on a short trip to the university. A semantic data model is developed initially to establish the relationship between the gathered datasets, ontology is created and the linked data is queried using SPARQL query language.**

***Keywords*—UniVisit, Semantic Data Model, Linked Dat**

I. Introduction

This application is developed on the basis of semantic integration of a variety of datasets. Initially, the datasets on Yelp, crime rate, OpenWeather Map and universities were gathered. A semantic data model was developed establishing the meaning of the instances involved. The datasets are logically structured so that we can get meaningful information while querying them. The main goal of the project is to incorporate all the semantic web concepts learnt as a part of this coursework. It includes technologies like XML, XSLT, Protege, Apache Jena framework for building Semantic Web and Linked Data applications. Other tools, technologies and frameworks were also used in the course of development of this application. As there is a growing need for developing semantic web application, there a lot of tools which is available for this purpose. Tools such as Karma, Google Refine and other online validator tools were used as a part of the development process.

II. Relevant Work

There are similar applications like Stupidsid, which is popular among students. It focuses on college reviews including information on infrastructure, location, faculty etc. USnews website provides data about graduate schools, online programs, ranking and advice for finding the best schools. This also helps students to narrow down their college preferences based on factors like location, tuition fees, rankings etc. BigFuture provides information on career opportunities and financial costs incurred. It gives a step-by-step college plan right from exploring the career areas to paying for college. Though these applications provide good amount of information with respect to universities, our main focus was to provide data for a large set of user groups. Thus, our application was developed keeping in mind the diverse user groups and guiding them throughout the planning process.

III. How our application is unique?

Our application gets information on university surroundings by three search criterias namely university name, city name and state name. The user can enter any of these keywords which queries the database using SPARQL query language. An interesting thing is that the application covers a wide range of user groups including researchers, scholars, university guest, exchange students, recruiters and of course students.
Anyone developing a similar application in the future can reuse our data model and documentation to increase the functionality of their own application.
The web application has the following special features:

- Our application displays the twitter feeds on the results page based on the search parameter entered by the user. Thus, the user can get updates on trending news and events happening around the university.
- We have SPARQL endpoint for the semantic community to develop new queries and test the results against our dataset. The user can enter custom SPARQL queries and query the database. This is useful for semantic development mainly to get any set of desired data.

IV. Key functionality of UniVisit

The main goal of our project is to model a collection of datasets semantically and display meaningful information to the user. Linked data is necessary for this purpose. So, the main focus was on integrating the data. Using this linked data, anyone can get data corresponding to the given search query. The input can be either the university name or the city name. So, the data in the corresponding datasets will be pulled out based on the user query. The user can thus know about the vicinity of the university by getting clear picture on restaurants, the safe places about the neighbourhood of the university. One can also predict the weather and plan their trip

accordingly. Semantic web concepts and the technology specified as a part of W3C standards helped us in the course of development.

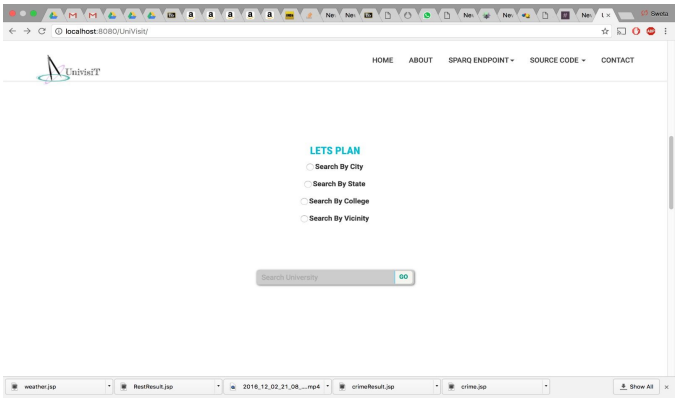

Fig. 1. Search keys for UniVisit application

The above screenshot shows the search criteria for displaying information about the vicinity.

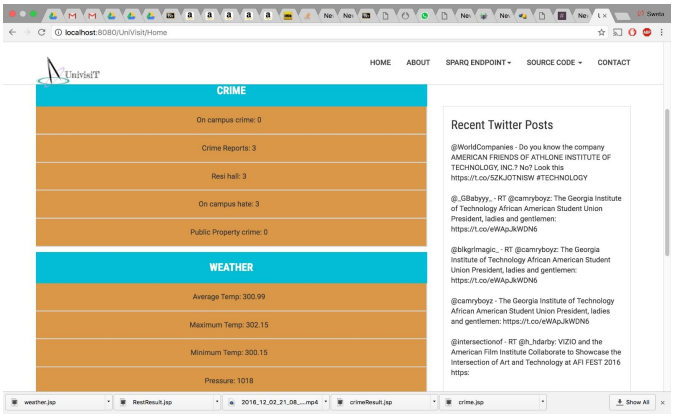

Fig. 2. Related twitter posts on the university

Our application also provides trending tweets on a particular university to know about the latest news and events happening currently. The above image demonstrates tweets with respect to the university that is searched for.

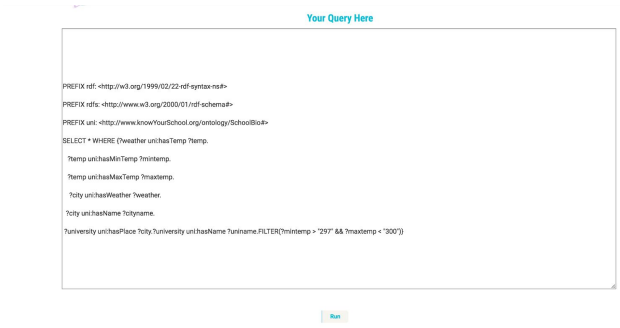

Fig. 3. Custom queries

The above custom query illustrates fetching all the university details which lie between a particular temperature range.

V. DATASETS

The data sets used in our applications are Weather, Restaurant, University and Crime data. Data was readily available for latter two and we had to write data scrapers for former two. We used university and city list from University data set to gather data from Openweathermap API and Yelp API. Since the data set for restaurants was huge we limited 5 restaurants per city. The application comprises of data collected from four different datasets. The following are the datasets falling under the scope of our project:

- University Dataset: This dataset contains fields such as contact information, address of the university, latitude, longitude and university identity information.
- Crime Dataset: This dataset contains fields such as on-campus crime rate, crime reports, public property crime, residential hall crime etc.
- OpenWeather Map API: This API provides all types of data associated with weather like minimum temperature, maximum temperature, average temperature, pressure, humidity, clouds, rains etc.
- Yelp dataset: It contains information pertaining to restaurants near a school, information such as timing, price range, reviews, ratings and facilities such as delivery, wheelchair accessible, alcohol, wifi, takeout, accepts credit card etc.

VI. SEMANTIC DATA MODEL

The ontology was designed using Protege by forming object and data properties to provide knowledge based solutions. Protege is a free and open source ontology editor used for semantic purposes. The ontology design can favor reusing this domain knowledge for future work or some relevant purpose. It also explicitly organizes the classes and structures them semantically.

Our semantic data model consists of all the relationships defined between various classes in our system. To explain the semantic relationship between the classes, let's consider a partial design of our system as below. The figure illustrates all the properties associated with the university class. Each property has a domain and a range. Properties like 'hasContact' contains the university class as their domain. The university class has subclasses like contact, InstAddress which in turn has subproperties like website address, fax, phone number, street address, statecode, zip etc. These properties have their range as Long, Literal etc.

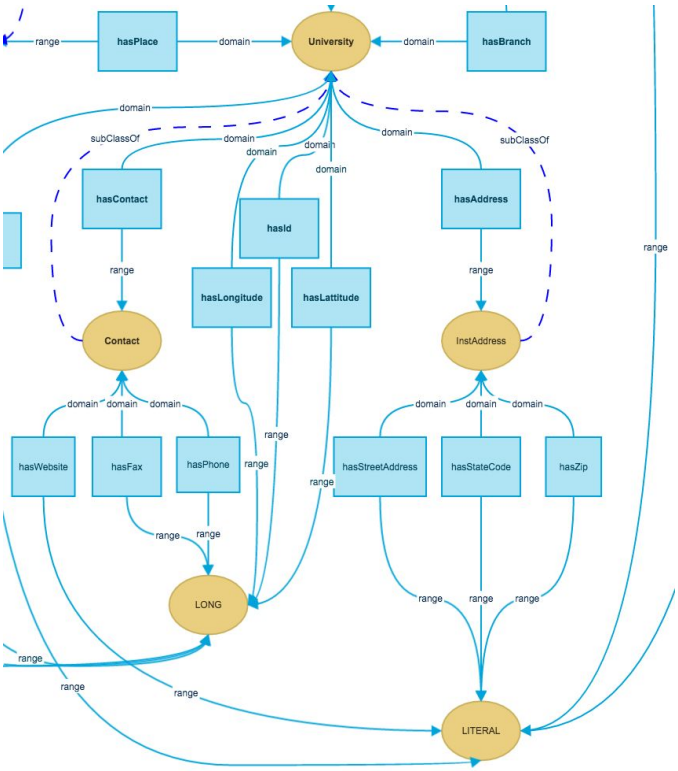

Fig. 4: Data model representing university class

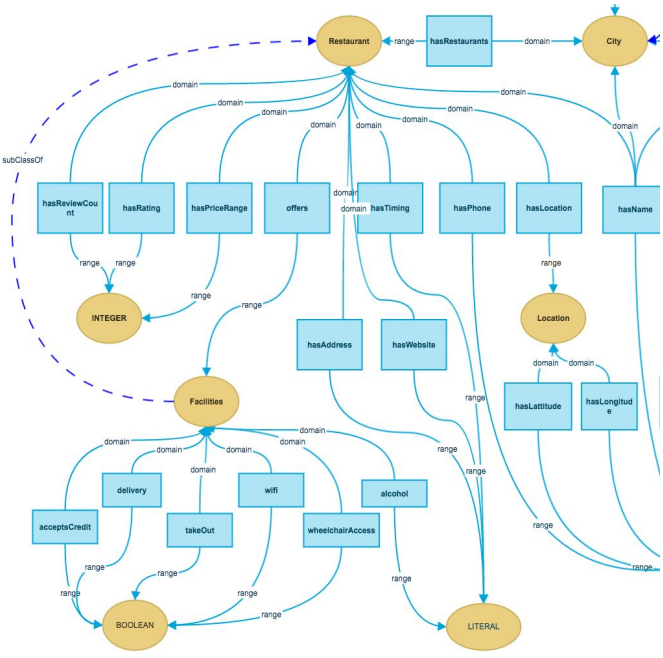

Fig. 5: Data model representing Restaurants class

There are classes like restaurants, city, location, facilities, weather, temperature and branch in our ontology which has meaningful properties and links to other classes. The below image shows the restaurant class and the associated properties. It has subclasses like facilities that comprises of properties like acceptsCredit, delivery, takeOut, wifi, wheelchairAccess, alcohol etc.

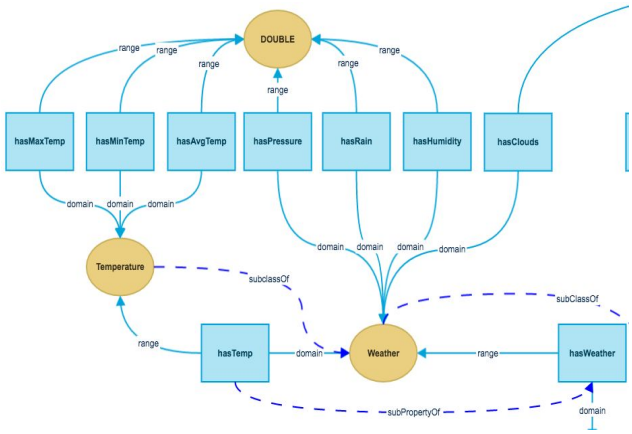

Fig. 6: Data model representing Weather class

Fig. 6 depicts that the Weather class has a subclass called temperature which has properties like hasMaxTemp, hasMinTemp, hasAvgTemp, hasPressure, hasRain, hasHumidity, hasClouds etc.

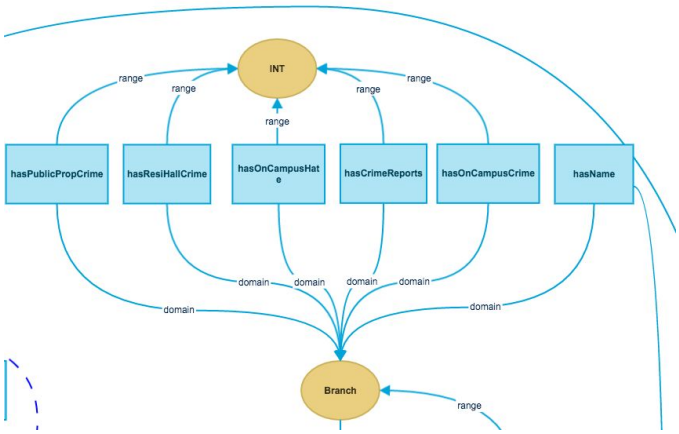

Fig. 7: Data model representing University branch class

VII. Challenges Faced During Data Integration

Since, the data is miscellaneous (i.e.) from heterogenous sources, there were many issues faced during the integration process. The following were the challenges faced while integrating the data:

- The data was not available in the desired format.
- The data had many unwanted characters like colon which Karma was not able to recognize. We had to modify all the columns to standard naming style.
- Karma does not allow us to load an existing mapping on data and make changes to it. So, if the window is lost and you have to make any changes, we need to redo everything from scratch.

- We created a column for city name in all the datasets which in turn resulted in creation of redundant tuples. So, we had to correct everything from scratch.
- The size of data was unmanageable. The tuples were in hundred thousands. So, we trimmed the dataset by considering only universities belonging to a particular number of states.

VIII. Instance Generation

Web-Karma was used for RDF instance generation for UniVisit web application. Semantic types were added for each column and links were established for informative data. Subclasses, data and object properties were modeled to gather useful information for the user. The ontology design suggested the relationships which eased the process of instance generation. The turtle file generated by Karma was converted into RDF file by using EasyRDF. As a result, we created and published the RDF data consisting of all the relationships that were defined in the model.

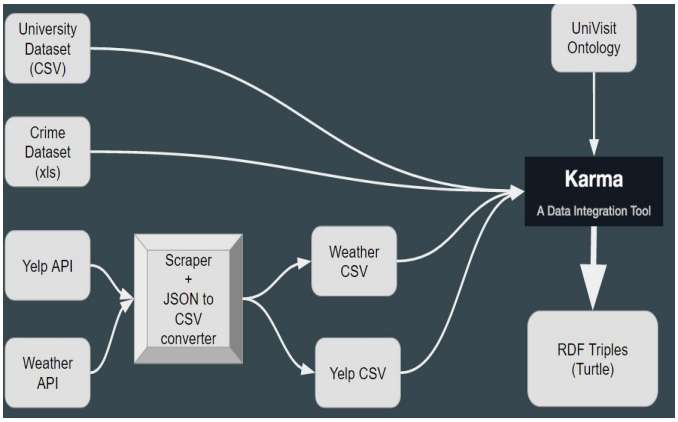

Fig. 8: Workflow for data integration

The above image demonstrates how data in different formats are fed to Karma along with the Ontology file to extract the RDF triples.

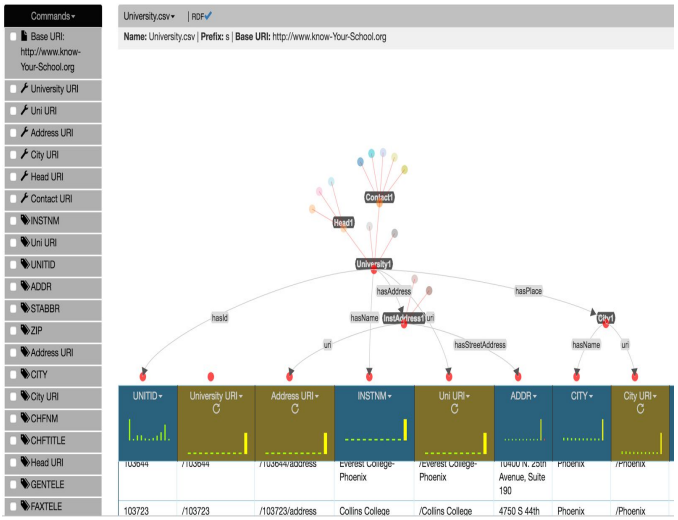

Fig. 9: Karma tool showing semantic types and linkages

The Karma tool allows us to map all the properties as per the ontology design which is loaded onto the application. The following is an example of Yelp RDF instance data which is generated as a result of converting the turtle file to RDF.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:yelp="http://www.knowYourSchool.org/ontology/SchoolBio#">
<rdf:Description
rdf:about="http://www.yelp.org/ontology/YelpAcademicDataset/FoodiesBrick&amp;Mortar/facility">
    <yelp:Delivery>false</yelp:Delivery>
    <yelp:Alochol>none</yelp:Alochol>
    <rdf:type rdf:resource="http://www.knowYourSchool.org/ontology/SchoolBio#Facilities"/>
    <yelp:WiFi>true</yelp:WiFi>
    <yelp:WheelChairAccessible>false</yelp:WheelChairAccessible>
    <yelp:AcceptsCreditCard>true</yelp:AcceptsCreditCard>
    <yelp:TakeOut>true</yelp:TakeOut>
  </rdf:Description>

  <yelp:City rdf:about="http://www.yelp.org/ontology/YelpAcademicDataset/Kennewick">
    <yelp:hasRestaurants>
      <yelp:Restaurant rdf:about="http://www.yelp.org/ontology/YelpAcademicDataset/FoodiesBrick
        <yelp:hasName>Foodies Brick &amp; Mortar</yelp:hasName>
        <yelp:hasReviewCount>117</yelp:hasReviewCount>
        <yelp:offers rdf:resource="http://www.yelp.org/ontology/YelpAcademicDataset/FoodiesBric
        <yelp:hasRating>4</yelp:hasRating>
        <yelp:hasAddress>[308 W Kennewick Ave, Kennewick, WA 99336]</yelp:hasAddress>
        <yelp:hasLocation>
          <yelp:Location rdf:about="http://www.yelp.org/ontology/YelpAcademicDataset/FoodiesBri
            <yelp:hasLongitude>-119.1218689</yelp:hasLongitude>
            <yelp:hasLatitude>46.2088269</yelp:hasLatitude>
          </yelp:Location>
        </yelp:hasLocation>
        <yelp:hasWebsite>https://www.yelp.com/biz/foodies-brick-and-mortar-kennewick?adjust_cre
        <yelp:hasPhone>5095910424</yelp:hasPhone>
        <yelp:hasTiming>Mon 11:00 am -  11:00 pm Tue 11:00 am -  11:00 pm Wed 11:00 am -  11:00
        <yelp:hasPriceRange>1</yelp:hasPriceRange>
      </yelp:Restaurant>
    </yelp:hasRestaurants>
```

Fig. 10: RDF instance data for Yelp dataset

The following is an example of University RDF instance data which is generated with the help of Karma. It is the result of converting the produced turtle file to RDF.

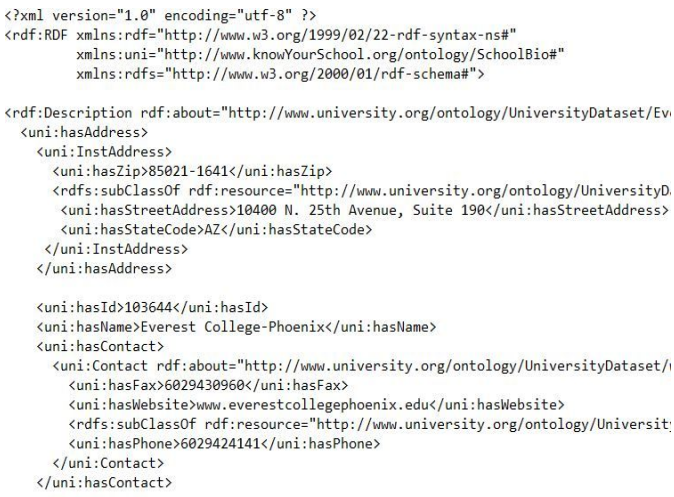

Fig. 11: RDF instance data for University dataset

The below image represents the RDF data produced for the weather dataset.

Fig. 12: RDF instance data for Weather dataset

IX. QUERYING THE LINKED DATA

We have employed SPARQL query language for getting relevant data from our database. This language is used to query hundreds of thousands of triples in our datasets. The queries were written for the following requirements stated:

- Given university, display all the details of that university and related crime report, weather information and nearby restaurants.
- Given city, display all the universities in it.
- Based on location of university, show universities nearby.
- Given university, show all restaurants nearby.
- Given city, show all restaurants.
- Given university, show all restaurants based on price.
- Given university, show all restaurants with rating greater than a particular value.
- Given university, show all restaurants with presence of specific facilities.
- Hottest/Coolest universities based on average temperature.
- Universities having minimum and maximum temperature within a range.
- All universities with humidity less than a given value.
- Universities with least/most campus crime.
- Universities with least/most campus hate.
- Universities with least/most public property crime.
- Universities with least/most campus crime reports.
- Universities with least/most residence hall crime ost campus crime.
- Universities with least/most campus hate.
- Universities with least/most public property crime.
- Universities with least/most campus crime reports.
- Universities with least/most residence hall crime.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uni: <http://www.knowYourSchool.org/ontology/SchoolBio#>
SELECT ?uniName ?cityName ?resName ?rating ?reviewCount
?priceRange ?timing ?phone ?wifi ?delivery ?longitutde ?latitude
WHERE {?university uni:hasName ?uniName.
 ?university uni:hasPlace ?city.
 ?city uni:hasName ?cityName.
 ?city uni:hasRestaurants ?restaurant.
 OPTIONAL {?restaurant uni:hasName ?resName}.
 OPTIONAL {?restaurant uni:hasPhone ?phone}.
 OPTIONAL {?restaurant uni:hasRating ?rating}.
 OPTIONAL {?restaurant uni:hasReviewCount ?reviewCount}.
 OPTIONAL {?restaurant uni:hasLocation ?location}.
 OPTIONAL {?location uni:hasLatitude ?latitude}.
 OPTIONAL {?location uni:hasLongitude ?longitutde}.
 OPTIONAL {?restaurant uni:hasTiming ?timing}.
 OPTIONAL {?restaurant uni:hasPriceRange ?priceRange}.
 OPTIONAL {?restaurant uni:offers ?facility}.
 OPTIONAL {?facility uni:WiFi ?wifi}.
 OPTIONAL {?facility uni:Delivery ?delivery}.
 FILTER(?uniName = "American Institute of Technology")
}
```

Fig. 13: SPARQL query for selecting restaurants near a university

The above query illustrates selecting all the restaurants near a university. The university name, city name, restaurant name, ratings, reviews, price range for the restaurant, timing, phone number, wifi, delivery facilities, longitude, latitude are selected with the university name as the search key. Also the details of the restaurant are displayed only if it is available in the dataset.

```
PREFIX rdf: <http://w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uni: <http://www.knowYourSchool.org/ontology/SchoolBio#>
SELECT * WHERE {?weather uni:hasTemp ?temp.
?temp uni:hasMinTemp ?mintemp.
?temp uni:hasMaxTemp ?maxtemp.
?city uni:hasWeather ?weather.
?city uni:hasName ?cityname.
?university uni:hasPlace ?city.
?university uni:hasName ?uniname.
FILTER(?mintemp > "297" && ?maxtemp < "300")
}|
```

Fig. 14: SPARQL query for selecting universities with a particular temperature range

The above query chooses the university name, contact information, address etc. with the filter condition specifying the minimum and maximum temperature.

```
"PREFIX rdf: <http://w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uni: <http://www.knowYourSchool.org/ontology/SchoolBio#>
SELECT *
 WHERE {
?university uni:hasPlace ?City.  ?university uni:hasName ?Name.
 OPTIONAL {    ?university          uni:hasId ?id }.
OPTIONAL {     ?university          uni:hasAddress ?instAddress}.
OPTIONAL {     ?university          uni:hasHead ?head }.
OPTIONAL {     ?university          uni:hasLongitude ?longitude }.
OPTIONAL {     ?university          uni:hasLatitude ?latitude }.
OPTIONAL {     ?university          uni:hasContact ?contact}.
OPTIONAL { ?contact uni:hasApplicationURL ?applicationURL}.
OPTIONAL { ?contact      uni:hasFax ?fax}.
OPTIONAL { ?contact      uni:hasWebsite ?website}.
OPTIONAL { ?contact      uni:hasPhone ?phone}.
OPTIONAL { ?head uni:hasPerson ?person}.
OPTIONAL { ?head uni:hasTitle ?title}.
OPTIONAL {  ?instAddress uni:hasStateCode ?statecode}.
OPTIONAL {  ?instAddress uni:hasZip ?zip}.
OPTIONAL {  ?instAddress uni:hasStreetAddress ?streetaddress}.
FILTER(?Name = '" + s + "') }"
```

Fig. 15: SPARQL query for selecting university details based on the search query

The above figure selects all the university details corresponding to the given search key.

```
"PREFIX rdf: http://w3.org/1999/02/22-rdf-syntax-ns#
 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
 PREFIX uni: <http://www.knowYourSchool.org/ontology/SchoolBio#>
 SELECT * WHERE { " + " <" + s + "> "+ " uni:hasWeather ?weather.
?weather uni:hasPressure ?pressure.
?weather uni:hasRain ?rain.
?weather uni:hasClouds ?clouds.
?weather uni:hasHumidity ?humidity.
?weather uni:hasTemp ?temp.
?temp uni:hasMinTemp ?mintemp.
?temp uni:hasAvgTemp ?avgtemp.
?temp uni:hasMaxTemp ?maxtemp. }";
```

Fig. 16: SPARQL query for selecting weather details of a given university

```
"PREFIX rdf: <http://w3.org/1999/02/22-rdf-syntax-ns#>
 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uni: <http://www.knowYourSchool.org/ontology/SchoolBio#>
 SELECT * { " + " <" + s + "> "+ " uni:hasBranch ?branch.
?branch  uni:hasName  ?branchname.
?branch  uni:hasOnCampusCrime  ?oncampuscrime.
?branch  uni:hasCrimeReports  ?crimereports.
?branch  uni:hasResiHallCrime  ?resihall.
?branch  uni:hasOnCampusHate  ?oncampushate.
?branch  uni:hasPublic_PropCrime  ?publicpropertycrime. }";
```

Fig. 17: SPARQL query for selecting crime details for a particular university

```
"PREFIX rdf: <http://w3.org/1999/02/22-rdf-syntax-ns#>
 PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
 PREFIX uni: <http://www.knowYourSchool.org/ontology/SchoolBio#>
SELECT ?resName ?rating ?rcount ?phone
 WHERE { " + " <" + s + "> "+ " uni:hasRestaurants ?Res.
 ?Res uni:hasName ?resName.
?Res uni:hasRating ?rating.
?Res uni:hasReviewCount ?rcount.
 ?Res uni:hasPhone ?phone.  }";
```

Fig. 18: SPARQL query for selecting restaurants near a particular university

Acknowledgment

The authors would like to thank Dr. Srividya Bansal and Jaydeep Chakroborty for their support and encouragement in pursuing this project throughout the coursework. The professor's encouragement and reviews helped us in making the application unique and extending its functionality. This work was supported as a part of Academic project for SER 594 – Web Semantics in Arizona State University.

Video demo link: UniVisit

References

[1] Yelp API - Documentation: https://www.yelp.com/developers/documentation/v2/overview
[2] Yelp API Code Samples : https://www.yelp.com/developers/documentation/v2/examples
[3] OpenWeatherMap API Documentation : https://openweathermap.org/current
[4] Open WeatherMap Code samples : https://openweathermap.org/examples
[5] Karma Instalation: https://github.com/usc-isi-i2/Web-Karma/wiki/Installation
[6] Karma User Guide : https://github.com/usc-isi-i2/Web-Karma/wiki
[7] Apache Fuseki: Installation : https://jena.apache.org/download/#apache-jena-fuseki
[8] Apache Fuseki Documentation: https://jena.apache.org/documentation/
[9] Twitter API Unofficial Library : http://twitter4j.org/en/index.html
[10] Twitter API Documentation: https://dev.twitter.com/docs

[11] UniVisit Website template: https://themefisher.com/free-bootstrap-templates-for-responsive-html5-website-like-corporate-tech-wedding-real-estate-magazine-and-more-2016/

[12] Big Future : https://bigfuture.collegeboard.org/
[13] Stupidsid : http://www.stupidsid.com/
[14] USNews : http://www.usnews.com