CSCI 586 Project Report

Spring 2016

Group 6

Topic: Curriculum or Research Guide for the USC Viterbi School of Engineering[1a]

Song, Lingnan

Liu, Xiaobo

Lin, Tai-Yu

Wang, Mengze

I Project Description

The goal of this project type is to build an ontology of curriculum and research guide for USC Viterbi school of engineering. Five aspects of the domain have been considered in this project: course, faculty, PhD student, PostDoc student and publication. We go through the process of data collection and clean, ontology creation, data modeling, query setup and user system construction. In the final system, we build an interface which enables the users to do various kinds of search in these four areas.

II Scope of the Project

We limit our scope to Computer Science, Electronic Engineering and Informatics. We collect the data from the following five aspects:

- Course
- Faculty
- PhD Student
- PostDoc Student
- Publication

III Steps involved

We firstly collect data from website and then format the data on the five areas. Since there are more than 20K publications in total, all of us focus on data retrieval and formatting at the first period. We get publication data from a website called DBLP, and for the rest which can not be found on DBLP, we manually search the data from Google Scholar. After data collection, two of us focus on ontology creation and data modeling, and the other two focus on Sesame web service which helped us to build the final system.

Step 1: Data collection and clean

The first step is data collection and clean. For data collection, we use Chrome extension tool web scraper to get related information about course, faculty, PhD student and postdoc from Viterbi websites of three departments. When deal with publication information, we get related publications of each faculty from DBLP websites. When the faculty cannot be found on DBLP, we use Google study to search his/her publications and fill out the publication form manually. In addition, if we are able to find personal website of the professor, we use it as a supplement to make the publication related data complete.

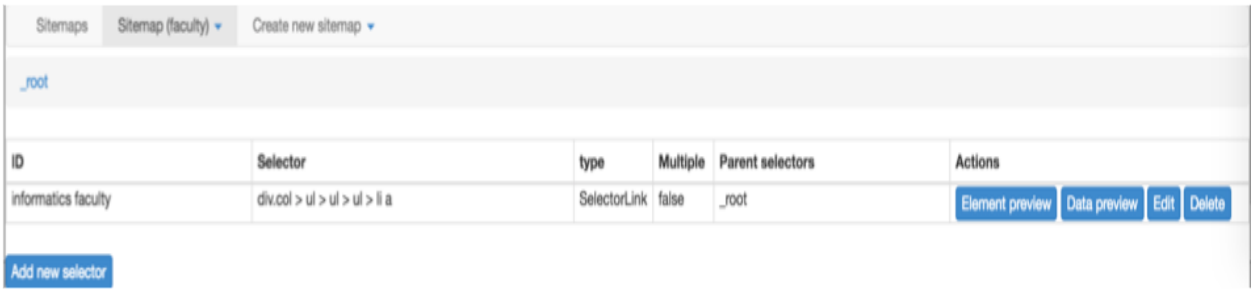


Figure 1. Chrome Web Scraper

It mainly involves four steps in data clean:

The first step is to make the data of the same area consistent with each other in different department forms. Below are the three tables with attributes we need in the final query processing. We discard all unnecessary attributes and let all forms of one specific area consistent with the attributes below.

CourseID	CourseName	Unit	Instructor	Prerequisite	Corequisite

Figure 2. Course attributes

Name	Alias	Category	Position	Cellphone	E-mail	Office	Research Area

Figure 3. Faculty attributes

Index	Category	Author	Title	Year	URL

Figure 4. Publication attributes

Next, we need to make all format of one specific attribute consistent to each other. For example,

1) To make all cellphone number in the format like xxx-xxx-xxxx
2) To make all faculty / PhD name in the format like first name (empty space) last name,
3) To display authors of one publication, each author takes one line in the 'Author' column,

This preprocessing optimizes the format of the final query result and make the user interface clean and tidy.

Then, we try to deal with when one professor has several names from different website. In the faculty form, we make the first column 'Name' stores the name of a faculty from school website, and we create column 'Alias'. If the professor

has different format of name, we add it to the column ‘Alias’. This helps us when we want to query publications according to faculty name.

At last, we add ‘Index’ column in publication form in order to relate it to faculty form. Usually one publication is written by several authors and we need to track the one from Viterbi school. It is like a foreign key which helps us to search publications of our faculty.

Step 2: Ontology Creation

After data collection and data cleaning, we separate our data into four categories to make the data more organized, they are course, department, person and publications. In each category, we also have sub-categories and their sub-categories under them to make the data being more specific. For example, in Person category, we separate all persons into 3 sub-categories, cs_person, ee_person and inf_person. In each sub-category, we also divide them if they are students or faculties. Next, we are trying to find the relationships among these organized data such as cs_faculty isInstructorOf cs_course to make the data more meaningful and useable. We then draw the ontology draft.

In our project, we use Protégé as our ontology creation tool which is developed by National Institute of General Medical Sciences at Stanford University. Protégé is an open-source platform and ontology editor that users are able to construct domain models and knowledge-based applications with ontologies. Based on the ontology draft we draw, we can easily create ontologies in Protégé, as shown in Figure 5.

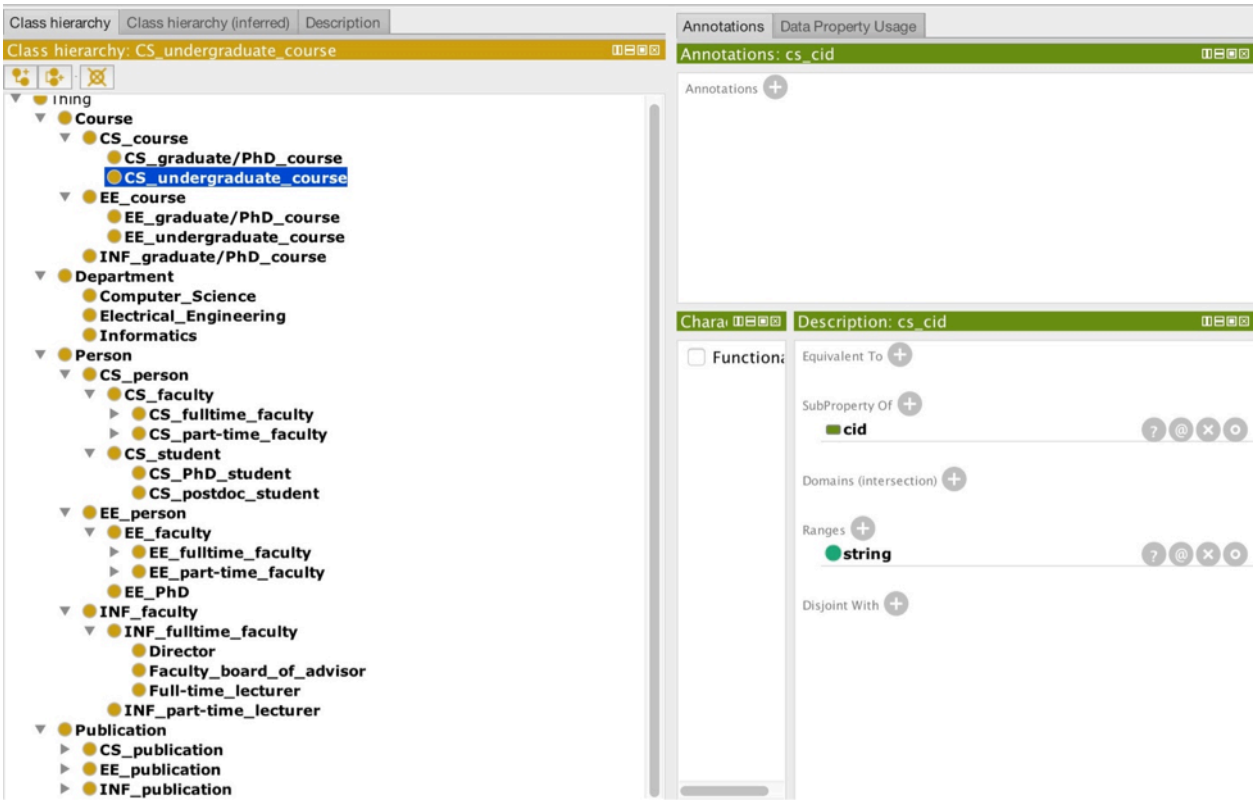


Figure 5. Created ontology using Protégé

In order to build the relationships among the data for the ontology creation, two concepts, data property and object property, are involved in this project. Data property explicitly describes the property of the data, which gives the definition of one object. Object property usually describes the relationship between two data properties. In the example mentioned above, cs_faculty and cs_course are data properties and csInstructorOf is the object property. Figure 6 provides lists of data properties and object properties. They play very important roles during data modeling. After creation of this ontology, we output the ontology in OWL (Web Ontology Language) format for further use.

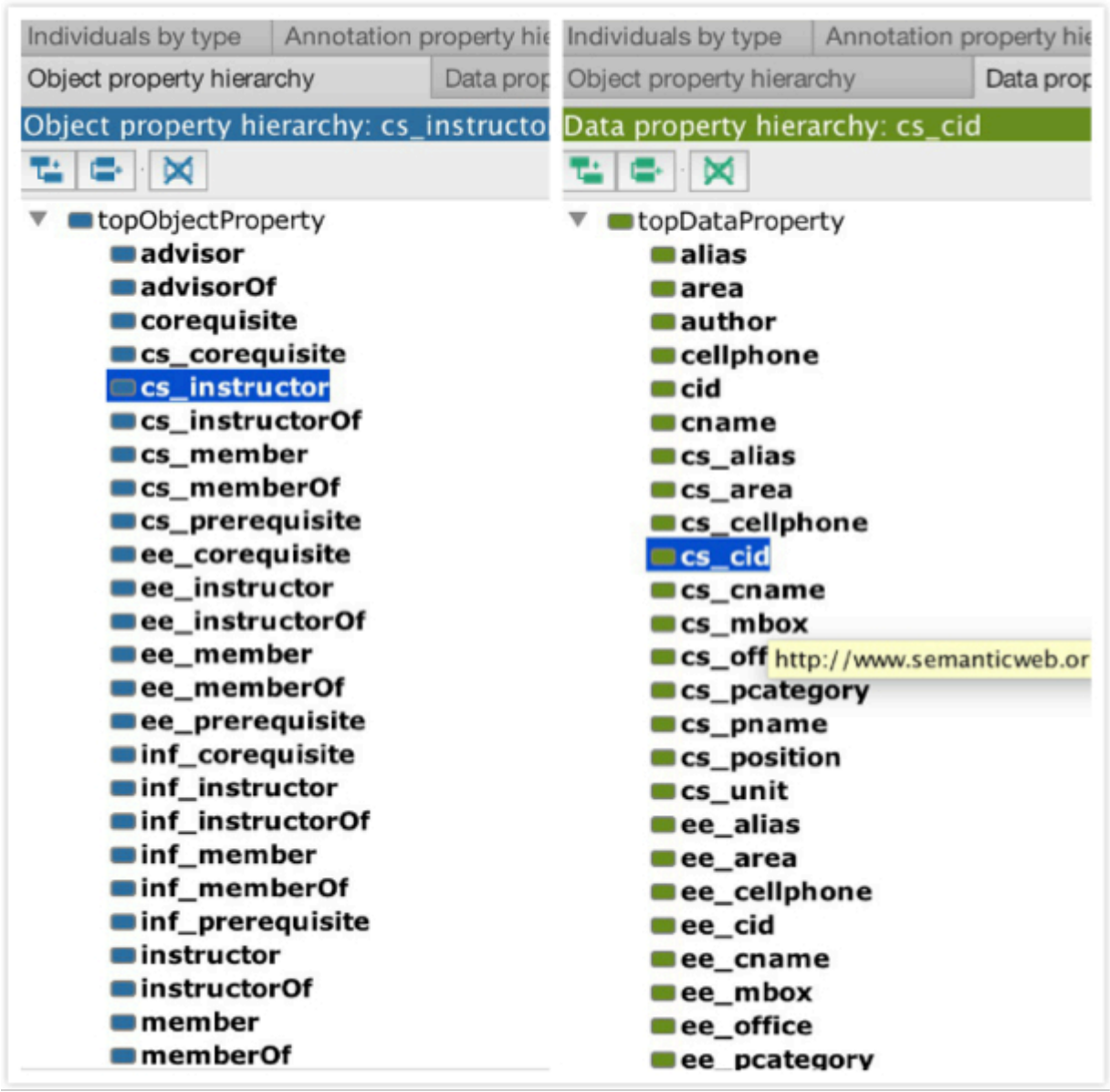


Figure 6. Object properties and data properties

Step 3: Data Modeling

In our project, we choose to use Karma for data modeling. Karma is an information integration tool that enables users to quickly and easily integrate data from a variety of data sources. We firstly import our source files(csv) and created ontology(owl) into karma and then mapped the data properties and the object properties for each instance. Figure 7 provides an example of the modeled data. The shallow grey bar shows the property between two nodes. It might be data property such as cs_cid, cs_cname in this example, or it can be object property such as cs_instructorOf. If the bottom node is not a property of itself, we need to add a new instance such as cs_faculty1 in this case. The arrow shows the mapping direction. After data modeling, we can generate the

Resource Description Framework(RDF) based on each source files and ontology for building query system.

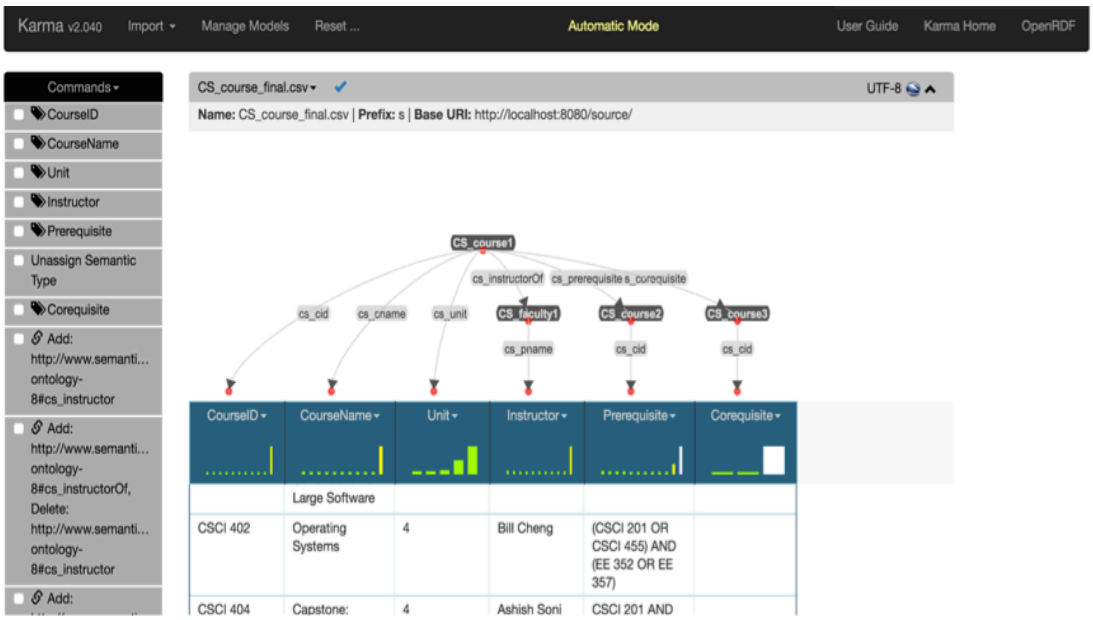


Figure 7. Data Modeling with Karma

Step 4: Search Query

- Sesame Workbench & Sesame Server

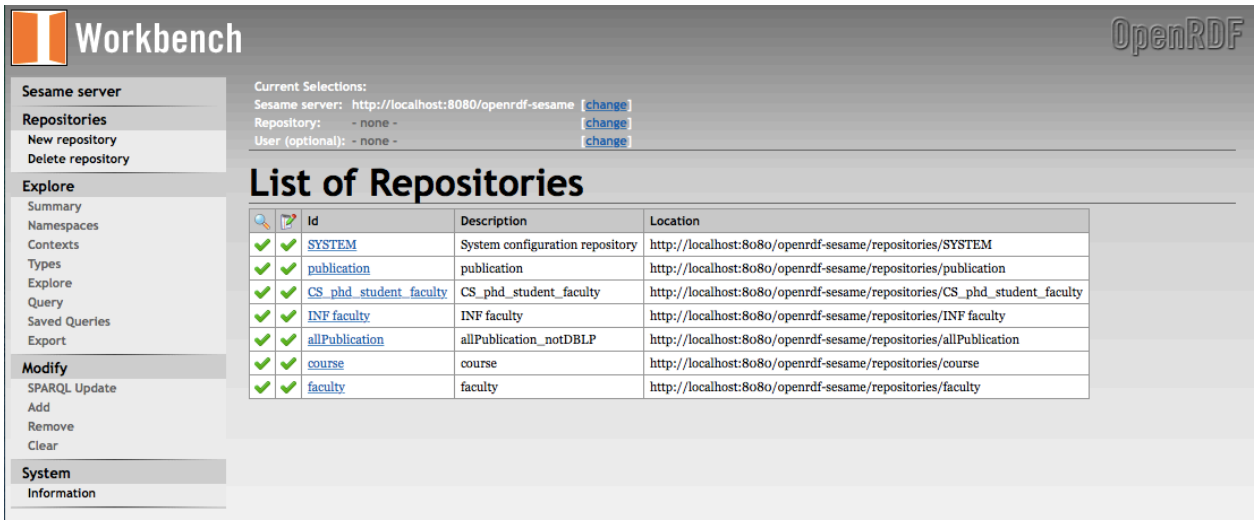


Figure 8. Sesame Workbench

This web is the main tool of our query over RDF data. It is actually an OpenRDF Workbench, which is a web application for interacting with Sesame. We use Workbench to create and manage repositories in Sesame Server's SYSTEM repository. In terms of Sesame, it is a powerful java framework for processing and operating RDF data, also it supports SPARQL to query over

RDF data. So we put and store our RDF data which is produced in Karma on this local server and then query over these data via SPARQL. One of its merits is that its easy-to-use API can be connected to all leading RDF storage solutions.

- Query

In our system, we created four types of search: faculty search, PhD search, course search, and publication search. In total, there are nine queries provided in our system. Followings are details of each queries, including examples, SPARQL codes, and results that are displayed on our user interface for each query.

1. Faculty Search

1) Retrieving a particular faculty information (Name, Category, Position, Cellphone, Email, Office, and Research Area) via his name or alias. Ex: Retrieving a faculty data whose name or alias contains “Dennis”

* SPARQL Code:

```
PREFIX about: <http://example.org/data/CS_faculty_final.csv#>
PREFIX about1: <http://example.org/data/ee_faculty_final.csv#>
PREFIX about2: <http://example.org/data/INF_faculty_final.csv#>

SELECT
DISTINCT ?Name ?Category ?Position ?Cellphone ?E_mail ?Office ?Research_Are
a
WHERE
{{?x about:Category ?Category .
  ?x about:Name ?Name .
  ?x about:Position ?Position .
  Optional{?x about:Cellphone ?Cellphone}
  Optional{?x about:E_mail ?E_mail}
  Optional{?x about:Office ?Office}
  Optional{?x about:Research_Area ?Research_Area}
  Optional{?x about:Alias ?Alias}
  FILTER(regex(?Name,'^Dennis','i')||regex(?Name,'Dennis','i')
          ||regex(?Alias,'^Dennis','i')||regex(?Alias,' Dennis','i'))}
 UNION{
  ?x about1:Category ?Category .
  ?x about1:Name ?Name .
  ?x about1:Position ?Position .
  Optional{?x about1:Cellphone ?Cellphone}
  Optional{?x about1:E_mail ?E_mail}
  Optional{?x about1:Office ?Office}
```

```
  Optional{?x about1:Research_Area ?Research_Area}
  Optional{?x about1:Alias ?Alias}
  FILTER(regex(?Name,'^Dennis','i')||regex(?Name,'Dennis','i')
          ||regex(?Alias,'^Dennis','i')||regex(?Alias,' Dennis','i'))}
 UNION{
  ?x about2:Category ?Category .
  ?x about2:Name ?Name .
  ?x about2:Position ?Position .
  Optional{?x about2:Cellphone ?Cellphone}
  Optional{?x about2:E_mail ?E_mail}
  Optional{?x about2:Office ?Office}
  Optional{?x about2:Research_Area ?Research_Area}
  Optional{?x about2:Alias ?Alias}
  FILTER(regex(?Name,'^Dennis','i')||regex(?Name,'Dennis','i')
          ||regex(?Alias,'^Dennis','i')||regex(?Alias,' Dennis','i'))}
}ORDER BY(?Name)
```

*Result:

Faculty	Category	Position	Cellphone	Email	Office	Research Area
Dennis McLeod	tenured and tenure track faculty	Professor	213-740-4504	mcleod@usc.edu	PHE 406	Structured domain ontologies; semantic web; database semantic heterogeneity resolution and inter-database correlation; service-ba information access and delivery frameworks

2) Searching faculty information (Name, Category, Position, Cellphone, Email, Office, and Research Area) by specifying his department and category.
Ex: Retrieving a faculty data who belongs to “Tenured and Tenure Track Faculty” in the Computer Science department.

*SPARQL Code:

```
PREFIX about: <http://example.org/data/CS_faculty_final.csv#>
SELECT
DISTINCT ?Category ?Name ?Position ?Cellphone ?E_mail ?Office ?Research_Are
a
WHERE
{?x about:Category ?Category .
 ?x about:Name ?Name .
 Optional{?x about:Position ?Position}
 Optional{?x about:Cellphone ?Cellphone}
 Optional{?x about:E_mail ?E_mail}
 Optional{?x about:Office ?Office}
 Optional{?x about:Research_Area ?Research_Area}
 FILTER (regex(?Category,'^tenured and tenure track faculty$','i'))
```

```
}ORDER BY ASC(?Category) ASC(?Name)
```

*Result:

Faculty	Category	Position	Cellphone	Email	Office	Research Area
Aiichiro Nakano	tenured and tenure track faculty	Professor	213-821-2657	anakano@usc.edu	VHE 610	High performance computing and simulation
Aleksandra Korolova	tenured and tenure track faculty	WiSE Gabilan Assistant Professor				Data privacy, privacy-preserving algorithms and technologies; big d applications; and large-scale data mining of social and information networks
Ari Requicha	tenured and tenure track faculty	Gordon Marshall Professor	213-740-4502	requicha@usc.edu	SAL 202	Molecular robotics, programmable automation
Barry Boehm	tenured and tenure track faculty	Director, CSSE	213-740-4927	boehm@sunset.usc.edu	SAL 326	Systems engineering, software engineering

2. PhD Search

1) Retrieving PhD students' information (Name, Email, Faculty Advisor) via his name. Ex: Retrieving a PhD student data whose first name or last name contains "Yinuo"

*SPARQL Code:

```
PREFIX about: <http://example.org/data/CS_phd_student_faculty.csv#>
SELECT Distinct ?Name ?Email ?FacultyAdvisor
WHERE
{?x about:Name ?Name.
 ?x about:Email ?Email.
 ?x about:FacultyAdvisor ?FacultyAdvisor.
 FILTER (regex(?Name,'^Yinuo','i')||regex(?Name,' Yinuo','i'))
}ORDER BY ASC(?Name)
```

*Result:

PhD student	E-mail	Faculty Advisor
Yinuo Zhang	yinuozha@usc.edu	Viktor Prasanna

2) To count how many PhD students that a particular faculty advisor has and to list all his PhD students Ex: ① Listing all Viktor Prasanna's PhD students

*SPARQL Code:

```
PREFIX about: <http://example.org/data/CS_phd_student_faculty.csv#>
SELECT Distinct ?FacultyAdvisor ?Name
WHERE
{?x about:FacultyAdvisor ?FacultyAdvisor.
?x about:Name ?Name.
FILTER (regex(?FacultyAdvisor,'^Viktor Prasanna$','i'))
}ORDER BY ASC(?FacultyAdvisor) ASC(?Name)
```

②To get Viktor Prasanna's total number of PhD students

*SPARQL Code:

```
PREFIX about: <http://example.org/data/CS_phd_student_faculty.csv#>
SELECT Distinct (count(distinct ?Name)as ?Total)
WHERE
{?x about:FacultyAdvisor ?FacultyAdvisor.
 ?x about:Name ?Name.
 FILTER (regex(?FacultyAdvisor,'^Viktor Prasanna$','i'))
}
```

*Result: Our user interface finally combines both querying results

Faculty Advisor	PhD Name
Viktor Prasanna	Ajitesh Srivastava
Viktor Prasanna	Alok Gautam Kumbhare
Viktor Prasanna	Charith Wickramaarachchi
Viktor Prasanna	Chung Ming Cheung
Viktor Prasanna	Gregory Harris
Viktor Prasanna	Michail Misyrlis
Viktor Prasanna	Om Prasad Patri
Viktor Prasanna	Palash Goyal
Viktor Prasanna	Vasileios Zois
Viktor Prasanna	Yinuo Zhang
Viktor Prasanna, Yogesh Simmhan	Saima Aman
Total PhD Students:	11

3. Course Search

1) To obtain all course information (Course Name, Unit, Course ID, Instructor, Prerequisite, Corequisite) via Course ID Ex: To obtain CSCI 586's course information

*SPARQL Code:

```
PREFIX about: <http://example.org/data/CS_course_final.csv#>
SELECT
DISTINCT ?CourseName ?Unit ?CourseID ?Instructor ?Prerequisite ?Corequisite
WHERE
{?x about:CourseID ?CourseID .
?x about:CourseName ?CourseName .
?x about:Instructor ?Instructor .
?x about:Unit ?Unit .
Optional{?x about:Prerequisite ?Prerequisite}
Optional{?x about:Corequisite ?Corequisite}
Filter regex(?CourseID, '^csci 586$', 'i')
}
```

*Result:

Course ID	Course Name	Unit	Instructor	Prerequisite	Corequisite
CSCI 586	Database Systems Interoperability	4	Dennis McLeod	CSCI 585	

2) To search course information (Course Name, Unit, Course ID, Instructor, Prerequisite, Corequisite) by specifying the instructor Ex: To search all course information of which instructor's first name or last name contains "Saty"

*SPARQL Code:

```
PREFIX about: <http://example.org/data/CS_course_final.csv#>
PREFIX about1: <http://example.org/data/EE_course_final.csv#>
PREFIX about2: <http://example.org/data/INF_course_final.csv#>
SELECT
DISTINCT ?Instructor ?CourseName ?Unit ?CourseID ?Prerequisite ?Corequisit
e
WHERE
{{?x about:CourseID ?CourseID .
  ?x about:CourseName ?CourseName .
  ?x about:Instructor ?Instructor .
  ?x about:Unit ?Unit .
  Optional{?x about:Prerequisite ?Prerequisite}
  Optional{?x about:Corequisite ?Corequisite}
  FILTER (regex(?Instructor,'^Saty','i')||regex(?Instructor,' Saty','i'))}
 UNION{
  ?x about1:CourseID ?CourseID .
  ?x about1:CourseName ?CourseName .
```

```
  ?x about1:Instructor ?Instructor .
  ?x about1:Unit ?Unit .
  Optional{?x about1:Prerequisite ?Prerequisite}
  Optional{?x about1:Corequisite ?Corequisite}
  FILTER (regex(?Instructor,'^Saty','i')||regex(?Instructor,' Saty','i'))}
 UNION{
  ?x about2:CourseID ?CourseID .
  ?x about2:CourseName ?CourseName .
  ?x about2:Instructor ?Instructor .
  ?x about2:Unit ?Unit .
  Optional{?x about2:Prerequisite ?Prerequisite}
  Optional{?x about2:Corequisite ?Corequisite}
  FILTER (regex(?Instructor,'^Saty','i')||regex(?Instructor,' Saty','i'))}
}ORDER BY ASC(?CourseID) ASC(?Instructor)
```

*Result:

Instructor	Course ID	Course Name	Unit	Prerequisite	Corequisi
Ulrich Neumann, Saty Raghavachary	CSCI 580	3-D Graphics and Rendering	4		
Dennis McLeod, Saty Raghavachary, Massoud Ghyam	CSCI 585	Database Systems	4		
Claire Bono, Saty Raghavachary	EE 455x	Introduction to Programming Systems Design	4		

3) To list all course IDs by specifying course units and department Ex: To get a list of Computer Science course ID with 4 units

*SPARQL Code:

```
PREFIX about: <http://example.org/data/CS_course_final.csv#>
SELECT DISTINCT ?CourseID ?CourseName
WHERE
{?x about:CourseName ?CourseName .
 ?x about:Unit ?Unit .
 ?x about:CourseID ?CourseID.
 ?x about:Unit '4' .
 filter regex(?CourseID, 'CSCI')
}ORDER BY ASC(?CourseID)
```

*Result:

Course ID	Course Name
CSCI 104L	Data Structures and Object Oriented Design
CSCI 170	Discrete Methods in Computer Science
CSCI 201L	Principles of Software Development
CSCI 270	Introduction to Algorithms and Theory of Computing
CSCI 310	Software Engeineering
CSCI 350	Introduction to Operating Systems
CSCI 353	Introduction to Internetworking
CSCI 360	Introduction to Artificial Intelligence
CSCI 401	Capstone:Design and Construction of Large Software Systems
CSCI 402	Operating Systems

4. Publication Search

1) To retrieve all faculty's publications by specifying his name Ex: To retrieve all faculty's publications whose first name or last name contains "Stephen"

*SPARQL Code:

```
PREFIX about: <http://example.org/data/11.csv#>
SELECT Distinct ?index ?title ?category ?author ?year ?url
WHERE
{?x about:index ?index .
 ?x about:title ?title .
 ?x about:category ?category .
 ?x about:author ?author .
 ?x about:year ?year .
 ?x about:url ?url .
 ?x about:index ?index.
 FILTER(regex(?index,'Stephen','i'))
}ORDER BY ASC(?index) DESC(?year)
```

*Result:

Result | Summary

USC faculty	Year	Title	Author	Category	URL
Stephen Crago	2015	Supporting High Performance Molecular Dynamics in Virtualized Clusters using IOMMU, SR-IOV, and GPUDirect.	Andrew J. Younge, John Paul Walters, Stephen P. Crago, Geoffrey Charles Fox	Conference and Workshop Papers	db/conf/vee/vee2015.html#YoungeWCF15
Stephen Crago	2015	Welcome to the IEEE Transactions on Big Data.	Stephen P. Crago	Journal Articles	db/journals/tbd/tbd1.html#Crago15
Stephen Crago	2015	Heterogeneous Cloud Computing: The Way Forward.	Stephen P. Crago, John Paul Walters	Journal Articles	db/journals/computer/computer48.html#CragoW1
Stephen Crago	2014	Evaluating GPU Passthrough in Xen for High Performance Cloud Computing.	Andrew J. Younge, John Paul Walters, Stephen P. Crago, Geoffrey Charles Fox	Conference and Workshop Papers	db/conf/ipps/ipdps2014w.html#YoungeWCF14
Stephen Crago	2014	Dynamic runtime optimizations for systems of heterogeneous architectures.	Geoffrey Phi C. Tran, Dong-In Kang, Stephen P. Crago	Conference and Workshop Papers	db/conf/hpec/hpec2014.html#TranKC14

2) Summary: It includes total number of all publications, number of publications for each publication category and number of publications in recent year, which are all by specifying a faculty's name Ex: ①To get number of publications for each publication category; all publications must belong to a faculty whose first name or last name contains "Stephen"

*SPARQL Code:

```
PREFIX about: <http://example.org/data/11.csv#>
SELECT DISTINCT ?sumIndex1 ?sumCategory (count (Distinct ?title) as ?sumCategoryTotal)
WHERE
{?x about:index ?sumIndex1 .
 ?x about:title ?title .
 ?x about:category ?sumCategory.
 FILTER(regex(?sumIndex1,'^Stephen','i')||regex(?sumIndex1,' Stephen','i'))
}group by ?sumIndex1 ?sumCategory
order by ASC(?sumIndex1) ASC(?sumCategory)
```

②To count number of publications in latest five years; all publications must belong to a faculty whose first name or last name contains "Stephen"

*SPARQL Code

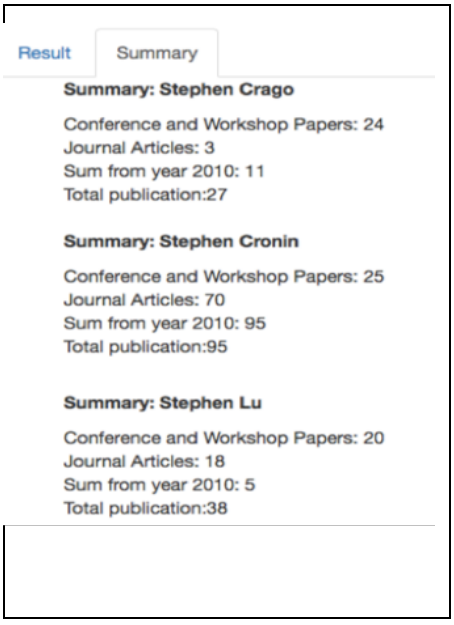
```
PREFIX about: <http://example.org/data/11.csv#>
SELECT DISTINCT ?sumIndex2 (count(distinct ?title)as ?sum2010totalPub)
WHERE
{?x about:index ?sumIndex2 .
 ?x about:title ?title .
 ?x about:year ?year .
 FILTER(xsd:integer(?year) >= 2010)
 FILTER(regex(?sumIndex2,'^Stephen','i')||regex(?sumIndex2,' Stephen','i'))
}group by ?sumIndex2
order by ASC(?sumIndex2)
```

③To count total number of publications which belong to a faculty whose first name or last name contains “Stephen”

*SPARQL Code

```
PREFIX about: <http://example.org/data/11.csv#>
SELECT DISTINCT ?sumIndex3 (count(distinct ?title)as ?sumTotalPub)
WHERE
{?x about:index ?sumIndex3 .
 ?x about:title ?title .
 ?x about:year ?year .
 FILTER(regex(?sumIndex3,'^Stephen','i')||regex(?sumIndex3,' Stephen','i'))
}group by ?sumIndex3
order by ASC(?sumIndex3)
```

*Result: Our user interface finally combines above 3 querying results



3) To list all publication which title contains a particular keyword Ex: To obtain all publications which title contains “big data”

*SPARQL Code

```
PREFIX about: <http://example.org/data/11.csv#>
SELECT DISTINCT ?title ?category ?author ?year ?url
WHERE
{?x about:title ?title .
 filter regex(?title, ‘big data', 'i')
 ?x about:category ?category.
 ?x about:author ?author .
 ?x about:year ?year .
 ?x about:url ?url .
}ORDER BY ASC(?index) DESC(?year)
```

*Result:

Category	Author	Title	Year	URL
Conference and Workshop Papers	Charalampos Chelmis, Jahanvi Kolte, Viktor K. Prasanna	Big data analytics for demand response: Clustering over space and time.	2015	db/conf/bigdataconf/bigdataconf2015.html#ChelmisKP15
Journal Articles	Craig A. Knoblock, Pedro A. Szekely	Exploiting Semantics for Big Data Integration.	2015	db/journals/aim/aim36.html#KnoblockS15
Conference and Workshop Papers	Daniel Kappler, Jeannette Bohg, Stefan Schaal	Leveraging big data for grasp planning.	2015	db/conf/icra/icra2015.html#KapplerBS15
Journal Articles	Jie Xu, Dingxiong Deng, Ugur Demiryurek, Cyrus Shahabi, Mihaela van der Schaar	Mining the Situation: Spatiotemporal Traffic Prediction With Big Data.	2015	db/journals/jstsp/jstsp9.html#XuDDSS15
Journal Articles	Stephen P. Crago	Welcome to the IEEE Transactions on Big Data.	2015	db/journals/tbd/tbd1.html#Crago15

Step 5: User Interface

In order to have a friendly user experience of our query system, we add the front-end design which is completed with HTML, CSS and JavaScript.

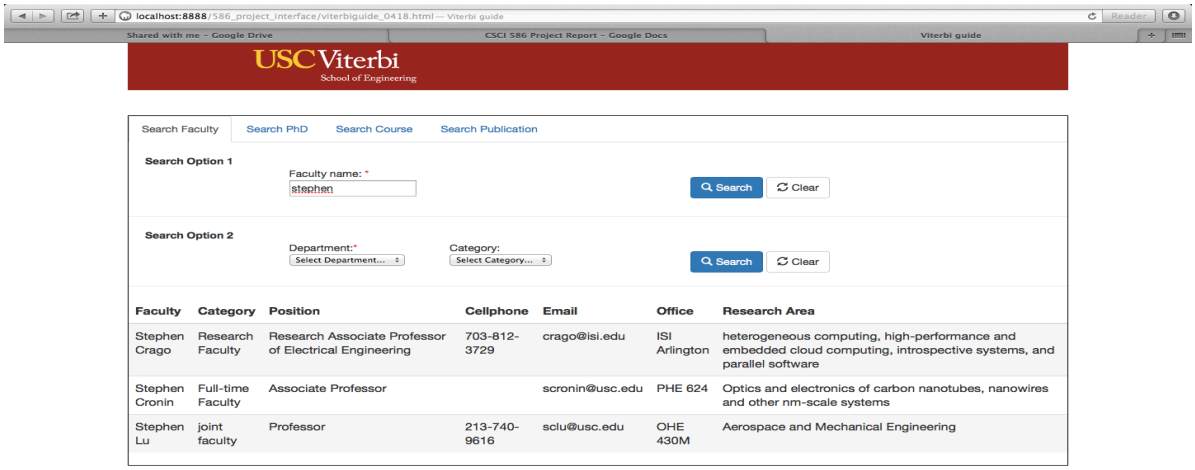


Figure 9. Search Faculty

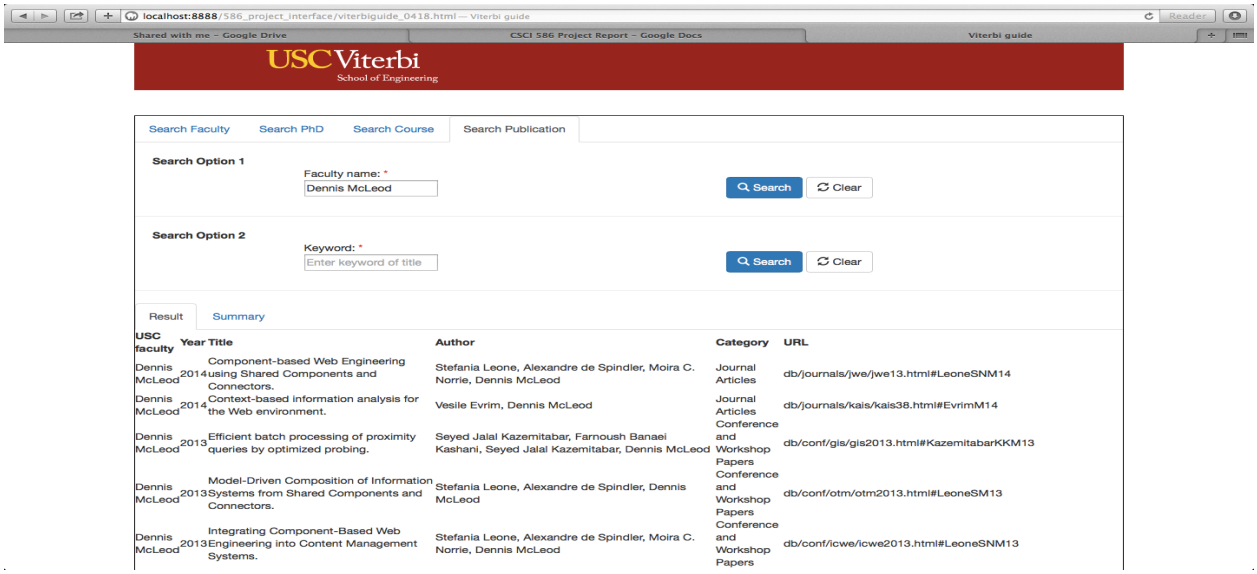


Figure 10. Search Publication

And we use PHP, which is server-side scripting language, to transmit the request from webpage to our local server and return the query result table back. The process is described as follows:

Figure 11. System Architecture

IV Technical Challenge

1. Standardize Faculty Name

As we mentioned in data cleaning section, we standardize all faculty names and record their standard names in “Name” attribute. But there are many faculties who can be called via different ways. For example, professor “Michael Zyda” can also be called “Mike Zyda.” In this situation, the querying system may get some troubles when searching a professor “Mike’s” publication in our database which only contains professor “Michael Zyda’s” data.

To solve this problem, we add an extra column “Alias” in our faculty tables which record all possible names for each faculty. Other tables involve attributes (FacultyAdvisor, Instructor, and Index) that are formed via faculty’s standard name. Thus, when the user queries some information via a faculty’s alias, PHP firstly requests Sesame to do faculty search and retrieve the faculty’s standard name. Then, PHP can obtain PhD, course, or publication data based on the standard name that is returned from previous faculty search.

2. PHP Queries Publication Data via Faculty Name

At the beginning, when the system queries publication data via faculty name, it has to search three faculty tables and one publication table at the same time. In other words, Sesame has to go through 20 thousand of data at the same time. This situation causes PHP stopping querying requests since the processing time is too long (over 2 minutes).

To narrow down the range of data that Sesame has to go through, we finally divide the whole work into two sub-works. Firstly, we let our system do faculty search and get all possible faculty’s standard names. Then, PHP sends querying requests with a parameter which wraps these faculty’s standard names to Sesame. Finally, our system can retrieve the publication data in 10 seconds.

V Contribution and Future work

In this system, we solve the alias problem through standardizing faculty name during data cleaning. We also use the benefits of PHP and ‘regex’ filtering to make queries more flexible and smarter, which means users can input the faculty’s alias name or only his last(first) name to get what they want. Moreover, other merits of this system is its friendly user interface and data statistics of the query results. However, there are still some aspects which need to be improved in the future work:

- Data Scalability, Maintenance
- Publication Rank: Top 3 most frequently cooperated USC faculty Data with one faculty. (It can be a fancy query which should be added in the

future but may takes a lot of time to deal with the inconsistent author name in each publication).

- Deployment on public web server

VI Reference:

[1] http://protege.stanford.edu/
[2] http://www.w3.org/TR/rdf-sparql-query/
[3] http://www.openrdf.org/doc/sesame/users/ch01.html
[4] http://www.isi.edu/integration/karma
[5] http://dblp.uni-trier.de/search