Table of Contents

[1. Introduction 2](#_Toc34658713)

[2. Description 3](#_Toc34658714)

[2.1 University Knowledge Graph 3](#_Toc34658715)

[2.1.1 University 3](#_Toc34658716)

[2.1.2 Course 3](#_Toc34658717)

[2.1.3 Topic 3](#_Toc34658718)

[2.1.4 Student 3](#_Toc34658719)

[2.1.5 Course Grade 3](#_Toc34658720)

[2.2 Automated Knowledge Base Construction 3](#_Toc34658721)

[2.3 Knowledge Queries 4](#_Toc34658722)

[3. Architecture 4](#_Toc34658723)

[3.1 Libraries used 4](#_Toc34658724)

[1. Pandas 4](#_Toc34658725)

[2. Rdflib 4](#_Toc34658726)

[3. Spotlight 4](#_Toc34658727)

[4. BeautifulSoup 5](#_Toc34658728)

[3.2 Vocabularies Used 5](#_Toc34658729)

[3.2.1 Reused Vocabularies 5](#_Toc34658730)

[3.2.2 Our Schema 6](#_Toc34658731)

[3.3 SPARQL Queries 8](#_Toc34658732)

[4. Tests 14](#_Toc34658733)

[4.1 Linking analysis 14](#_Toc34658734)

[5. Results 14](#_Toc34658735)

[6. Conclusion 14](#_Toc34658736)

[7. Future Scope and Limitations 14](#_Toc34658737)

# Introduction

Knowledge graphs are information represented in the form of graphs where nodes are entities and the relationships between them are the edges. The fact that they help form better connections between data helps machines to process and connect loads of information for better results. Knowledge graphs are the foundation for artificial intelligence and are the main tool that is required to build intelligent agents such as chatbots that help answer questions based on the user’s input. Each fact in a knowledge graph is represented as a triple (subject, predicate, object) which are interconnected with each other to form the whole graph.

In this project, we create a knowledge graph that stores the information about Universities and its academic details. Here, we concentrate on information about Concordia University alone. The goal of this project is to build a knowledge graph in such a way that it can answer questions about the University through SPARQL queries. This project is the first step of building an intelligent agent that is capable of replying to questions related to the university which would be further done in our next project. Knowledge graphs are created from the data extracted from the web. Here, we take information about the courses, topics related to it and further details regarding it from the Concordia University websites and its open databases.

We have constructed the University knowledge graph using RDF and RDFS standard in turtle format. The graph is built to store information about universities along with its DBpedia entry URI. But since we restrict our project to just one university, the graph stores information about all the courses offered by Concordia University and the topics that are covered in each course. It also stores student information such as name, email etc. For testing purposes, we have used dummy information for ten students and their course history.

# Description

## University Knowledge Graph

The University graph is stored in the form of RDF schema and contains mainly five components, namely, ‘University’, ‘Course’, “Topic’ and ‘Student’ and ‘Course Grade’. These are stored in the form of the classes. Each of these components has attributes that store more information about them.

### University

This contains the name and the DBpedia URI for the given university. Since we use only Concordia University in our project, this has only once instance.

### Course

This contains the name of the course, course identifier and the description for the course.

### Topic

This contains the name of the topic and the DBpedia URI entry for the topic.

### Student

This contains the name of the student; first name and last name, ID number and email address of the student.

### Course Grade

This contains a course class instance along with the grade that the student scored for that course.

The RDF Schema for the University graph is stored in Turtle format, in the files universityKG.ttl and DataGraph.ttl. We first created the universityKG.ttl as the base graph which stores the definitions for the classes described above

## Automated Knowledge Base Construction

In this step, we have first generated .csv files that stores information about the instances.

*Courses.csv* stores the data related to courses and its properties.

*Grades.csv* stores the courses and the grade scores by each student.

*Student.csv* stores information about students such name, email, id etc.

*Topics.csv* stores information about Topics.

*Universities.csv* stores information about universities and its DBpedia entries.

These data was scraped from Concordia University Graduate/Under Graduate websites. We have scraped the data from over 99 URLs which gave information about the courses that are offered by the University, along with its course code and course description. We have found 2742 courses offered by Concordia from all the Departments. We have used Beautiful Soup to extract specific data from the Webpage, specifically under the “courses” section, such as course name, course code and course description. We then used python spotlight library which links to the DBpedia spotlight to find the Topics covered under a topic. This finds the DBpedia URI for the topics which are found inside the course description.

## Knowledge Queries

We use SPARQL queries to ask and get information from the RDF Knowledge graph. We have tested our knowledge graph for various queries. The queries are run on python using rdflib but we have also tested our queries on Apache Jena Fuseki.

# Architecture

In this section, we describe the architecture of our RDF schema in details.

## Libraries used

### Pandas

We use pandas here to process the University data stored in the csv files.

### Rdflib

Rdlib is used in our project to create and parse the Knowledge graph.

### Spotlight

We use spotlight library to find access the DBpedia spotlight to identify entities from the course description to extract the topics covered under a course.

### BeautifulSoup

Beautiful library is used for scraoing data from the Concordia University URLs to get academic data such as courses provided by the university with the course description, course code and course name.

## Vocabularies Used

### Reused Vocabularies

#### RDFS Schema

*rdfs:*[*http://www.w3.org/2000/01/rdf-schema#*](http://www.w3.org/2000/01/rdf-schema)*:*

rdfs:label This is used to provide a human readable name for all the entities in our KG.

rdfs:comment A description of the subject resource.

rdfs:domain Domain of the property defined

rdfs:range Range of the property defined

rdfs:Class A class

#### RDF Schema

*rdf:* [*http://www.w3.org/1999/02/22-rdf-syntax-ns#*](http://www.w3.org/1999/02/22-rdf-syntax-ns)

rdf:Property

#### XML Schema

*xsd:* [*http://www.w3.org/2001/XMLSchema#*](http://www.w3.org/2001/XMLSchema)

*xsd:string*

#### FOAF Vocabulary

*foaf:* [*http://xmlns.com/foaf/0.1/*](http://xmlns.com/foaf/0.1/)

*foaf:name* This is used for giving name for entities such as course, topic and university.

*foaf:givenName* This

*foaf:familyName*

*foaf:mbox*

*foaf:Person*

#### DBpedia Property

*dbp:* [*http://dbpedia.org/property/*](http://dbpedia.org/property/)

*dbp:score* Connects the Grade for a student as a string literal

*dbp:id* Connects the Identity number for a student

#### DCMI Metadata Terms

*dc:* [*http://purl.org/dc/elements/1.1/*](http://purl.org/dc/elements/1.1/)

*dc:source*

*dc:subject*

*dc:identifier*

*dc:description*

### Our Schema

#### ISP

*isp:* [*http://intelligentsystemproj1.io/schema#*](http://intelligentsystemproj1.io/schema)

*isp:University* Class for a University

*isp:Course* Class for a Course

*isp:Topic* Class for a Topic

*isp:Student* Class for a student

*isp:CourseGrade* Class for courses with it’s grade for a student

*isp:studiesAt* Property that links the university that the student studies at

*isp:tookCourse* Property that links the Courses completed by a student

*isp:hasPart* Property that links the Topics covered for a Course

*isp:coversCourse* Property that links the Courses covered in a University

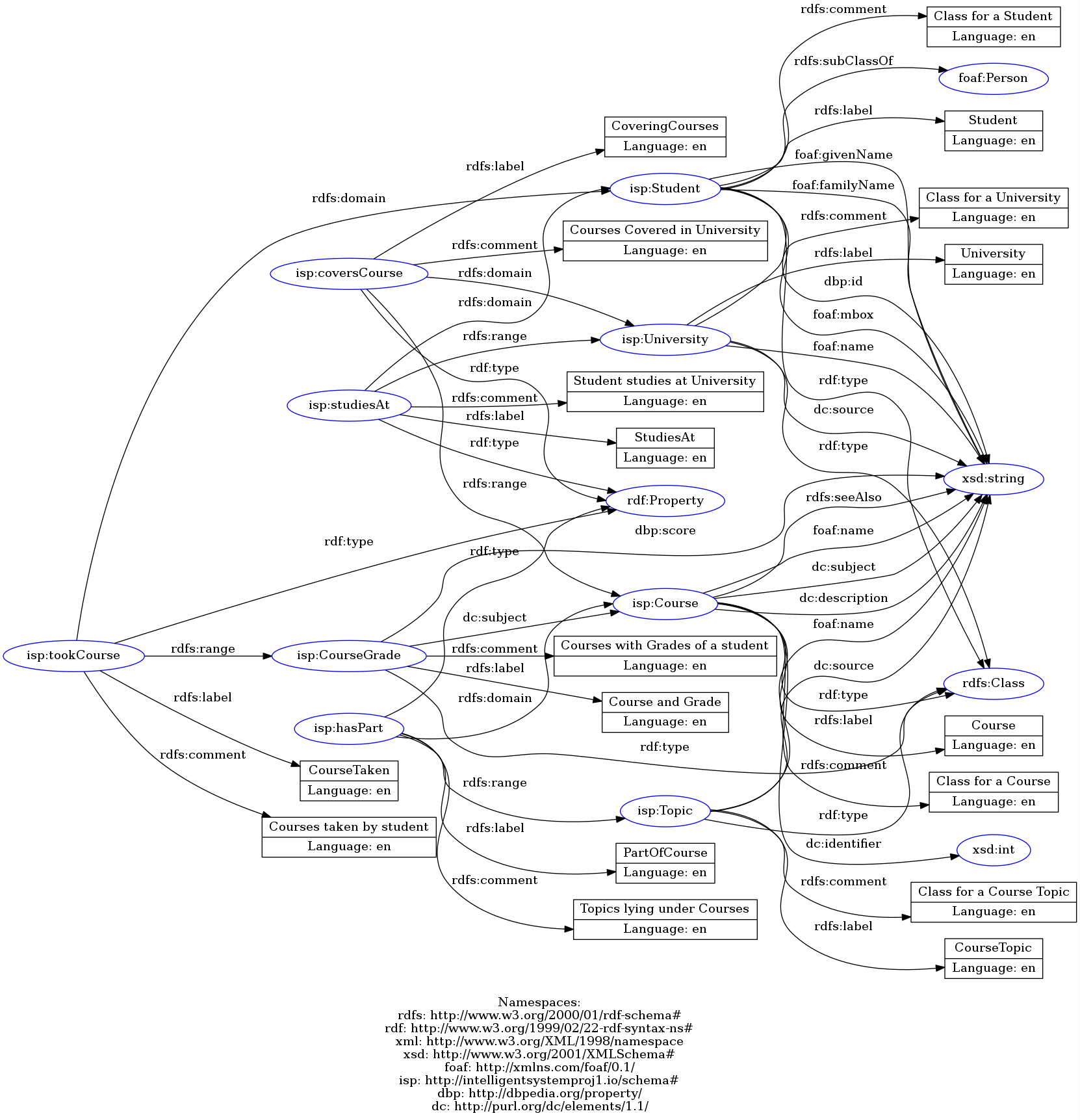


Figure 1: Shows the RDF graph visualisation for the base knowledge graph.

## SPARQL Queries

PREFIX dbr: <http://dbpedia.org/resource/>

PREFIX db: <http://dbpedia.org/>

PREFIX is: <http://purl.org/ontology/is/core#>

prefix dbp: <http://dbpedia.org/property/>

prefix dc: <http://purl.org/dc/elements/1.1/>

prefix isp: <http://intelligentsystemproj1.io/schema#>

prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

prefix xsd: <http://www.w3.org/2001/XMLSchema#>

Query 1: Total number of triples in the KB.

SELECT (COUNT(\*) as ?triples) WHERE { ?s ?p ?o .}

This query matches all the triples and ‘?s ?p ?o’ which represents every single triple present in the graph and counts it.

Output

Top of Form

Bottom of Form

Top of Form

Bottom of Form

|  |  |
| --- | --- |
|  | Total number of triples: 46254 |

Query 2 : Total number of students, courses, and topics

|  |
| --- |
| We tried two queries: |
|  | a. Following query was tried first. Because of the huge database, this was taking a lot of time while running in python. Thus, we tried another query listed in b part. |
|  |  |
|  | SELECT |
|  | (COUNT(DISTINCT ?student) as ?scount) |
|  | (COUNT(DISTINCT ?course) as ?ccount) |
|  | (COUNT(DISTINCT ?topic) as ?tcount) |
|  | WHERE{ |
|  | ?student rdf:type isp:Student . |
|  | ?course rdf:type isp:Course . |
|  | ?topic rdf:type isp:Topic . |
|  | }  This query matches all the triples that are of the type isp:Student, isp:Course and isp:Topic and stores them in the variables ?students, ?courses, and ?topics. The COUNT value for each of these variables gives the number of students, courses and topics |
|  |  |
|  | b. This query was taking very less time comparatively. |
|  |  |
|  | SELECT ?scount ?ccount ?tcount |
|  | WHERE{ |
|  | {SELECT (COUNT(DISTINCT ?student) as ?scount) WHERE { ?student rdf:type isp:Student . } } |
|  | UNION {SELECT (COUNT(DISTINCT ?course) as ?ccount) WHERE { ?course rdf:type isp:Course .} } |
|  | UNION {SELECT (COUNT(DISTINCT ?topic) as ?tcount) WHERE { ?topic rdf:type isp:Topic .} } |
|  | } |

Here, we took nested SELECT queries which took much less time.

#### OUTPUT

|  |  |
| --- | --- |
|  | Total number of students : 15 |
|  | Total number of courses : 2742 |
|  | Total number of topics : 6702 |

Query 3 : For a course c, list all covered topics using their (English) labels and their link to DBpedia

|  |
| --- |
| SELECT DISTINCT ?name ?link |
|  | WHERE{ |
|  | ?course rdf:type isp:Course . |
|  | ?course dc:subject "MAST" . |
|  | ?course dc:identifier 830 . |
|  | ?course isp:hasPart ?topic . |
|  | ?topic dc:source ?link . |
|  | ?topic foaf:name ?name . |
|  | } |

Here, we match all the nodes of the type isp:Course that has course subject :MAST and Course number 830 (example instance). We match those course nodes to the topics covered by that course, given by isp:hasPart. We then display the names (foaf:name) of these topics and their dbpedia links ( given by dc:source).

#### OUTPUT

|  |
| --- |
| For COMP 7251, |
|  |  |
|  | Please enter the Course Subject(like COMP): COMP |
|  | Please enter the Course Number(like 691): 7251 |
|  |  |
|  | TopicName TopicLink |
|  | wireless http://dbpedia.org/resource/Wireless\_LAN |
|  | topology http://dbpedia.org/resource/Network\_topology |
|  | MAN http://dbpedia.org/resource/Metropolitan\_area\_network |
|  | sensor networks http://dbpedia.org/resource/Wireless\_sensor\_network |
|  | PAN http://dbpedia.org/resource/Personal\_area\_network |
|  | mobile computing http://dbpedia.org/resource/Mobile\_computing |
|  |  |
|  |  |
|  | -------------------------------------------------------------------------------------- |
|  |  |
|  |  |
|  | For MAST 830, |
|  |  |
|  | Please enter the Course Subject(like COMP): MAST |
|  | Please enter the Course Number(like 691): 830 |
|  |  |
|  | TopicName TopicLink |
|  | Gauss http://dbpedia.org/resource/Carl\_Friedrich\_Gauss |
|  | class number http://dbpedia.org/resource/Ideal\_class\_group |
|  | L-series http://dbpedia.org/resource/L-function |
|  | Cyclotomic Fields http://dbpedia.org/resource/Cyclotomic\_field |
|  | theorem http://dbpedia.org/resource/Theorem |

Query 4: For a given student, list all courses this student completed, together with the grade

|  |
| --- |
| SELECT ?courseSub ?courseNum ?courseName ?grade |
|  | WHERE{ |
|  | ?student rdf:type isp:Student . |
|  | ?student dbp:id "40083902" . |
|  | ?student isp:tookCourse ?courseGrade . |
|  | ?courseGrade dbp:score ?grade . |
|  | ?courseGrade dc:subject ?course . |
|  | ?course foaf:name ?courseName. |
|  | ?course dc:subject ?courseSub . |
|  | ?course dc:identifier ?courseNum . |
|  | } |

In this Query, we start by matching all the nodes of the type isp:student and has “40083902” (example instance) as id (dbp:id). We then take the courses with the grades scored by the student by using isp:tookCourse. Dbp:score gives the grade and foaf:name of dc:subject gives the name of the course.

#### OUTPUT

|  |
| --- |
| For student id 40083902 |
|  | Please enter the Student id: 40083902 |
|  |  |
|  | CourseSubject CourseNumber CourseName Grade |
|  | COMP 6321 Machine Learning (4 credits) F |
|  | COMP 6341 Computer Vision (\*) (4 credits) A+ |
|  | COMP 6331 Advanced Game Development (\*) (4 credits) B+ |
|  |  |
|  | --------------------------------------------------------------------------------------------------------------------- |
|  |  |
|  | For student id 40083898 |
|  | Please enter the Student id: 40083898 |
|  |  |
|  | CourseSubject CourseNumber CourseName Grade |
|  | COMP 7251 Mobile Computing and Wireless Networks (4 credits) A+ |
|  | COMP 6281 Parallel Programming (\*) (4 credits) F |
|  | COMP 7241 Parallel Algorithms and Architectures (4 credits) B+ |

Query 5: For a given topic, list all students that are familiar with the topic

|  |
| --- |
| SELECT DISTINCT ?id (CONCAT(?firstName, " ", ?lastName) as ?name) |
|  | WHERE{ ?student rdf:type isp:Student . |
|  | ?student dbp:id ?id . |
|  | ?student foaf:givenName ?firstName . |
|  | ?student foaf:familyName ?lastName . |
|  | ?student isp:tookCourse ?courseGrade. |
|  | ?courseGrade dbp:score ?grade . |
|  | ?courseGrade dc:subject ?course . |
|  | ?course isp:hasPart ?topic . |
|  | ?topic foaf:name "Game engine" . |
|  | FILTER(?grade < "F") } |

We take all the nodes related to a topic identified by a topics name (foaf:name), say for example, “CORBA” by matching all the Course node(isp:tookCouse) taken by a student(isp:student) which has a topic(isp:hasPart) under the course named “CORBA”. We then check if the student has passed in this course by filtering out the fail grades using FILTER. This says that the student is indeed familiar with this topic.

#### OUTPUT

|  |
| --- |
| For "Game engine", |
|  | Please enter the topic name: Game engine |
|  |  |
|  | StudentID StudentName |
|  | 40083901 Mary Smith |
|  | 40083902 Maria Hernandez |
|  |  |
|  | --------------------------------------------------------------------------------- |
|  |  |
|  | For "Computer Science", |
|  |  |
|  | Please enter the topic name: Computer Science |
|  |  |
|  | StudentID StudentName |
|  | 40083895 James Smith |
|  |  |
|  | --------------------------------------------------------------------------------- |
|  |  |
|  | For "design patterns", no students are there. |
|  | Please enter the topic name: design patterns |
|  |  |
|  | StudentID StudentName |

Query 6: For a student, list all topics (no duplicates) that this student is familiar with

|  |
| --- |
| SELECT DISTINCT ?tName |
|  | WHERE{ |
|  | ?student rdf:type isp:Student . |
|  | ?student dbp:id "40083895" . |
|  | ?student isp:tookCourse ?courseGrade. |
|  | ?courseGrade dbp:score ?grade . |
|  | ?courseGrade dc:subject ?course . |
|  | ?course isp:hasPart ?topic . |
|  | ?topic foaf:name ?tName . |
|  | FILTER(?grade < "F") . |
|  | } |

We find the student nodes linked to a specific student id ( dbp:id or we can use any other identification attribute such as foaf:name or foaf:mbox), say “40083895”. We then retrieve the courses take by the student using isp:tookCourse and through that we take the topics covered under it using dc:subject. We then check if the grade(dbp:score) for those courses are >F by using FILTER.

#### OUTPUT

|  |
| --- |
| For student id 40083895, |
|  |  |
|  | Please enter the Student id: 40083895 |
|  |  |
|  | TopicName |
|  | --------- |
|  | server |
|  | fault tolerance |
|  | concurrency |
|  | remote procedure call |
|  | distributed computing |
|  | scalability |
|  | CORBA |
|  | concurrency control |
|  | Client-server |
|  | communication |
|  | fault-tolerant |
|  | interprocess communication |
|  | distributed systems |
|  | COMP |
|  | Computer Science |
|  |  |
|  | ---------------------------------------------------------------------------------------- |
|  |  |
|  | For student id 40083902, |
|  | Please enter the Student id: 40083902 |
|  |  |
|  | TopicName |
|  | --------- |
|  | computer |
|  | 3D |
|  | Artificial Intelligence |
|  | AI |
|  | mobile gaming |
|  | Game engine |
|  | collision detection |
|  | pathfinding |
|  | OpenCV |
|  | COMP |
|  | perceptual organization |
|  | Computer Vision |
|  |  |
|  | -------------------------------------------------------------------------------------- |
|  |  |
|  | For student id 40083904, he has only one subject(F grade), so no topics listed. |
|  | Please enter the Student id: 40083904 |
|  |  |
|  | TopicName |
|  | --------- |

# Tests

## Linking analysis

We have used linking analysis to calculate the accuracy for DBpedia spotlight entity detection to find the URI entry in DBpedia for the topics that are from the course description text. We have tested for any ambiguity or errors in the entities identified from 100 links detected by python spotlight. It was found that our project showed xx% success in linking analysis

# Results

# Conclusion

# Future Scope and Limitations