Project Assignment 1

Presented By:

Dhwani Sondhi
40083894

Greeshma Sunil

40092843

Contents

1. 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 can reply 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.

2. Schema Description

2.1 Base University Knowledge Graph Schema

The University graph is stored in the form of RDF schema. It contains five classes (University, Course, Topic, Student and CourseGrade) and four properties(coversCourse, hasPart, studiesAt and tookCourse). These are stored in the form of the classes. Each of these components has attributes that store more information about them. Each component has a Label and Comment for better understanding.

2.1.1 University (Class)

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.

2.1.2 Course (Class)

This contains the name of the course, course subject (like COMP), course number(like 691), the description for the course and additional link from which the course information is taken.

2.1.3 Topic (Class)

This class refers to the topics extracted from course name and description. This contains the name of the topic and the DBpedia URI entry for the topic. Additional, analysis is done to analyze the URI extracted.

2.1.4 Student (Class)

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

2.1.5 Course Grade (Class)

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

2.1.6 coversCourse (Property)

This property links the courses to the university with courses in range and university in domain.

2.1.7 hasPart (Property)

This property links a course with all the topics related to it with course in domain and topics in range.

2.1.8 studiesAt (Property)

This property links a student with the university with student in domain and university in range.

2.1.9 tookCourse (Property)

This property links a student with the courses taken with the grade attained by the student with student in domain and CourseGrade Class in range.

The RDF Schema for the University graph is stored in Turtle format, in the files “universityKG.ttl”, the basic description of the base schema (contains classes and properties listed above) and “DataGraph.ttl” with all the data extracted and linked using triplets with the base schema.

2.2 Vocabularies Used in Schema

2.2.1 Reused Vocabularies

2.2.1.1 RDFS Schema

rdfs: *http://www.w3.org/2000/01/rdf-schema#*

`rdfs:label` : Used to provide a human readable name for all the entities in our KG.

`rdfs:comment` : Used to provide a additional comment all the entities in our KG.

`rdfs:domain` : Used to define the Domain of the property.

`rdfs:range` : Used to define the Range of the property.

`rdfs:Class` : Used to define a class.

`rdfs:seeAlso` : Used to provide additional links for course.

`rdfs:subClassOf` : Used to provide extension of Person Class to the Student Class.

2.2.1.2 RDF Schema

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

`rdf:Property` : Used to define a property.

2.2.1.3 XML Schema

xsd: *http://www.w3.org/2001/XMLSchema#*

`xsd:string` : Used to define a string literal.

`xsd:int` : Used to define an integer literal.

2.2.1.4 FOAF Vocabulary

foaf: *http://xmlns.com/foaf/0.1/*

`foaf:name` : Used to provide names for entities such as course, topic and university.

`foaf:givenName` : Used to provide first name of the student.

`foaf:familyName` : Used to provide last name of the student.

`foaf:mbox` : Used to provide email of the student.

`foaf:Person` : Used to define Student as sub class of the Person Class.

2.2.1.5 DBpedia Property

dbp: *http://dbpedia.org/property/*

`dbp:score` : Used to provide the grade for a student.

`dbp:id` : Used to provide the identity number for a student.

`dbp:termPeriod :` Used to provide the term period of a course.

2.2.1.6 DCMI Metadata Terms

dc: *http://purl.org/dc/elements/1.1/*

`dc:source` : Used to link dbpedia links for University and Topic instances.

`dc:subject` : Used to connect Course(like COMP 691) for a CourseGrade instance(further to a student) and provide the course subject (like COMP) for a Course instance .

`dc:identifier` : Used to provide course number for Course instance.

`dc:description` : Used to provide course description for Course instance.

2.2.2 Our Schema

2.2.2.1 ISP

isp: 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 grades 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.

2.3 Schema Description in Image

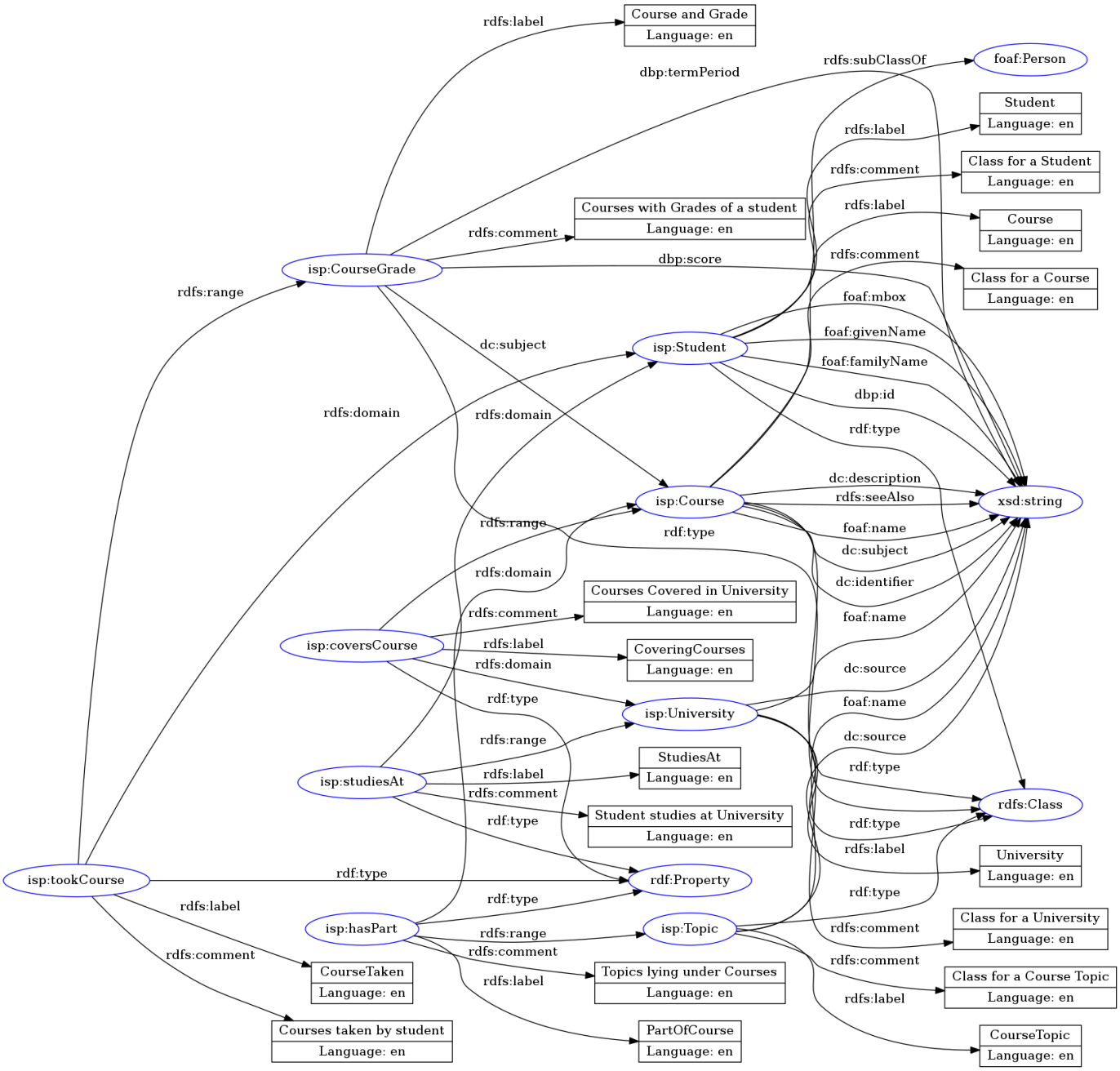

Figure 1: Shows the RDF graph visualisation for the base knowledge graph.

3. Chatbot's Knowledge Base Construction

3.1 Dataset Created

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 as name, email, id etc.
Topics.csv stores information and DBpedia links for Topics for each course.
Universities.csv stores information about universities and its DBpedia entries.

The data was scraped from Concordia University Graduate websites. We have scraped the data from over 99 URLs (graduate), 203 URLs (undergraduate) and open data (Concordia University) which gave information about the courses that are offered by the University, along with its course code and course description. We have found 2742 graduate courses, 3272 undergraduate courses and 17099 topics offered by Concordia from all the Departments.

3.2 Tools/Libraries Used

Pandas

We use pandas here to process all the 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 name and description to extract the topics with their DBpedia links for a course.

BeautifulSoup

BeautifulSoup library is used for scraping 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.

3.3 Tools Created

We have created 3 tools to build the knowledge graph and 1 tool to run the 6 queries given. All the tools are developed in python.

1LoadCoursesStudentsGrades.py

This tool is used to create student and course data. It uses pandas and Beautiful Soup. The data for courses such as course name, course number and course description are extracted by doing web scraping of 99 Concordia University URLs with Beautiful Soup. With these links, we were able to attain 2742 graduate courses. The student data includes 15 students' names hard coded. The courses are randomly allotted to the students with various grades. The data extracted is saved using pandas in respective csvs.

2LoadLinkTopics.py

This tool is used to get the topics with the dbpedia links for a course name and description. It uses pandas and Spotlight. This tool extracts the courses data from courses.csv. The course name and description are joined to make a call to spotlight annotate function. The link "https://api.dbpediaspotlight.org/en/annotate" , 0.5 confidence, support=20 and the course data is send to get the topics(surfaceFrom) and dbpedia links. Various confidence levels were tried started from 0.3, which was further increased and chose 0.5.

3CreateKnowledgeGraph.py

This tool is used to extract the data from the course, university, course, grade and topics csvs with the help of pandas library. Further, using rdflib is used to create the knowledge graph which is further saved in "DataGraph.ttl" file.

4RunQueries.py

This tool is used to run the 6 queries given in the assignment with various inputs taken from console.

3.4 How to use the tools to create knowledge graph for the ChatBot

- Run *1LoadCoursesStudentsGrades.py* which takes input the "universityKG.ttl" file and saves the data in the respective files in CSV folder.
- Run *2LoadLinkTopics*.py which takes input the Courses.csv and saves the data in respective files in CSV folder.
- Run *3CreateKnowledgeGraph*.py which takes input the "universityKG.ttl" file and the files in CSV folder and saves the triplets in "DataGraph.ttl". Steps 1-3 creates the Knowledge Graph.
- Run *4RunQueries*.py to ask test SPARQL queries to the knowledge base if needed.

4. Queries from assignment 1

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.

Following prefixes are used or all 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#>
```

4.1 Query 1

The assignment asked to get the total number of triples in knowledge graph in this query. Following query was designed to solve this:

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

QUERY EXPLAINED

The query iterates all the triples and filter using the statement in the where clause and count them using count function.

RESULT

We were able to attain following result:

```
Total number of triples: 46254
```

4.2 Query 2

The assignment asked to get total number of students, courses, and topics. We tried two queries:

a. Following query was tried first. But, 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 .
}
```

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 .} }
}
```

QUERY EXPLAINED

The query iterates all the triples and filter using the statements in the where clause and calculates the number of students, courses and topics. The second query use Union which helps in reducing the time for the query.

RESULT

We were able to attain following results:

```
Total number of students : 15
Total number of courses : 6014
Total number of topics : 17099
```

4.3 Query 3

The assignment asked to list all covered topics using their (English) labels and their link to DBpedia for a course c. Following query was designed to solve this:

```
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 .
}
```

QUERY EXPLAINED

The query iterates all the triples and filter using the statements in the where clause. Here, we match all the nodes of the type isp:Course that has given course subject (like MAST) and Course number (like 830), these are the example values. 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).

RESULT

We were able to attain following results:

1. For COMP 7251, we got

```
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
```

2. For MAST 830, we got

```
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
```

4.4 Query 4

The assignment asked to list all courses this student completed, together with the grade for a given student. Following query was designed to solve this:

```
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 .
}
```

QUERY EXPLAINED

The query iterates all the triples and filter using the statements in the where clause. 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 dc:subject gives the course whose name is extracted using foaf:name, subject using dc:subject and course number using dc:identifier. The details of the course with grade is outputted.

RESULT

We were able to attain following result:

1. For Student id 40083902, we got:

```
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+
```

2. For Student id 40083898, we got:

```
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+
```

4.4 Query 5

The assignment asked to list all students that are familiar with the topic (i.e., took, and did not fail, a course that covered the topic) for a given topic. Following query was designed to solve this:

```
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") }
```

QUERY EXPLAINED

The query iterates all the triples and filter using the statements in the where clause. 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 course with topic(isp:hasPart) 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.

RESULT

We were able to attain following result:

1. For "Game engine", we got:

```
Please enter the topic name: Game engine

StudentID            StudentName
40083901             Mary Smith
40083902             Maria Hernandez
```

2. For "Computer Science", we got:

```
Please enter the topic name: Computer Science

StudentID            StudentName
40083895             James Smith
```

4.4 Query 6

The assignment asked to list all topics (no duplicates) that this student is familiar with (based on the completed courses for this student that are better than an “F” grade) for a given student. Following query was designed to solve this:

```
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") .
}
```

QUERY EXPLAINED

The query iterates all the triples and filter using the statements in the where clause. 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 taken 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 better than F by using FILTER.

RESULT

We were able to attain following result:

1. 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
```

2. 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
```

5. Chatbot Script

5.1 Method of Input Translation

We have used ***Regular Expressions*** to translate the given input into SPARQL queries. The python tool created is in *5Chatbot.py.* To use in python, we have used re library. Following are the steps used:

- We take the input from the user.
- The input is searched with various patterns listed in below sections.
- For searching, we have used *re.search()* function, in which we pass a parameter as re.IGNORECASE to avoid case sensitivity.
- Example pattern: *'^what(\s)*is(\s)*the(\s)*(?P<course>.*\b\w*\b)(\s)*about(\s)*\?$'*
- *?P<course>* captures the matching group into "course" which can be extracted using *group()* function.

5.2 Question 1: "What is the <course> about?"

Given a course (<course>), give the description of the course.

REGULAR EXPRESSION: r'^what(\s)*is(\s)*the(\s)*(?P<course>.*\b\w*\b)(\s)*about(\s)*\?$'

QUERY

```
SELECT ?desp
WHERE{
      ?course rdf:type isp:Course .
      ?course dc:subject "MAST" .
      ?course dc:identifier 830 .
      ?course dc:description ?desp .
}
```

QUERY EXPLANATION

The query iterates all the triples and filter using the statements in the where clause. We take all the nodes related to a course (isp:Course) identified by a course subject (dc:subject) and course number(dc:identifier). Then we give the output as the description of the course node using dc:description.

RESULT

We were able to attain following results:

1. ```
   Please enter the query (Please type 0 to exit):
   What is the COMP 7251 about?

   COMP 7251 has following description:
   Prerequisite: COMP 6461. Introduction to mobile computing and
   wireless networks:local (LAN), personal (PAN) and metropolitan
   (MAN). Mobile ad hoc networks and sensor networks. Algorithms and
   protocols for medium access, routing, topology control, and reliable
   transport. A project is required.
   ```

2. ```
   Please enter the query (Please type 0 to exit):
   What is the FLIT 300 about?

   FLIT 300 has following description:

   Aperçu général de la littérature française du Moyen Âge, de la
   Renaissance et du XVIIe siècle, et des contextes historiques,
   sociaux et culturels qui permettent de mieux comprendre les œuvres.
   ```

5.3 Question 2: “Which courses did <Student> take?"

Given a student <student>, list all the courses the student has taken with grade and term.

REGULAR EXPRESSION : r'^which(\s)*courses(\s)*did(\s)*(?P<student>.*\b\w*\b)(\s)*take(\s)*\?$

Here, we have taken 2 queries for the input can either be student id or student name.

QUERY1

```
SELECT ?subject ?number ?cname ?grade ?term
WHERE{
?student rdf:type isp:Student .
?student dbp:id "40083895" .
?student isp:tookCourse ?courseGrade.
?courseGrade dbp:termPeriod ?term .
?courseGrade dbp:score ?grade .
?courseGrade dc:subject ?course .
?course dc:subject ?subject .
?course dc:identifier ?number .
?course foaf:name ?cname .
}
```

QUERY 2

```
SELECT ?subject ?number ?cname ?grade ?term
WHERE{
    ?student rdf:type isp:Student .
    ?student foaf:givenName "James" .
    ?student foaf:familyName "Smith" .
    ?student isp:tookCourse ?courseGrade .
    ?courseGrade dbp:score ?grade .
    ?courseGrade dbp:termPeriod ?term .
    ?courseGrade dc:subject ?course .
    ?course dc:subject ?subject .
    ?course dc:identifier ?number .
    ?course foaf:name ?cname .
}
```

QUERY EXPLANATION

The query iterates all the triples and filter using the statements in the where clause. We find the student nodes linked to a specific student id (dbp:id), say “40083895” for query 1 or using first name(foaf:givenName) and last name(foaf:familyName) for query 2. We then retrieve the courseGrade linked with the student using isp:tookCourse. We give the grade and term from courseGrade using dbp:score and dbp:termPeriod. And course details from course taken from courseGrade(dc:subject) using dc:subject for course subject, dc:identifier for course number and foaf:name for the course name.

RESULTS

We were able to attain following results:

1. Please enter the query (Please type 0 to exit):
   Which courses did 40083900 take?

   40083900 took following courses:

   COMP 6321 Machine Learning (4 credits) got a grade A+ in Fall 2019.
   COMP 7251 Mobile Computing and Wireless Networks (4 credits) got a grade F in Summer 2019.
   COMP 6311 Animation for Computer Games (*) (4 credits) got a grade B+ in Winter 2019.

2. Please enter the query (Please type 0 to exit):
   Which courses did Maria Rodriguez take?

   Maria Rodriguez took following courses:

   COMP 6321 Machine Learning (4 credits) got a grade A+ in Fall 2019.
   COMP 7251 Mobile Computing and Wireless Networks (4 credits) got a grade F in Summer 2019.
   COMP 6311 Animation for Computer Games (*) (4 credits) got a grade B+ in Winter 2019.

5.4 Question 3: "Which courses cover <Topic>?"

Given a Topic <Topic>, list all the courses that cover the topic.

REGULAR EXPRESSION : r'^which(\s)*courses(\s)*cover(\s)*(?P<topic>.*\b\w*\b)(\s)*\?$'

QUERY

```
SELECT  ?subject  ?number  ?name
WHERE{
?course rdf:type isp:Course .
?course dc:subject ?subject .
?course dc:identifier ?number .
?course foaf:name ?name .
?course isp:hasPart ?topic .
?topic foaf:name "Game engine" .
}
```

QUERY EXPLANATION

The query iterates all the triples and filter using the statements in the where clause. We take all the course nodes that are related to a topic by isp:hasPart with topic name identified using foaf:name. The subject, number and name of the course is given using dc:subject, dc:identifier and foaf:name respectively.

RESULTS

We were able to attain following results:

1. *Please enter the query (Please type 0 to exit):*
   Which courses cover Client-server?

   Client-server is covered in following courses:

   COMP 6231 Distributed System Design (4 credits)

2. *Please enter the query (Please type 0 to exit):*
   Which courses cover distributed computing?

   distributed computing is covered in following courses:

   COMP 6231 Distributed System Design (4 credits)
   SOEN 423 Distributed Systems

5.5 Question 4: "Who is familiar with <Topic>?"

For a given topic, list all students that are familiar with the topic.

REGULAR EXPRESSION : r'^who(\s)*is(\s)*familiar(\s)*with(\s)*(?P<topic>.*\b\w*\b)(\s)*\?$'

QUERY

```
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") }
```

QUERY EXPLAINED

The query iterates all the triples and filter using the statements in the where clause. 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 course with topic(isp:hasPart) 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.

RESULTS

We were able to attain following results:

1. *Please enter the query (Please type 0 to exit):*

   Who is familiar with Game engine?

   Following students are familiar with Game engine:

   40083901 Mary Smith

   40083902 Maria Hernandez

2. *Please enter the query (Please type 0 to exit):*

   Who is familiar with Computer Science?

   Following students are familiar with Computer Science :

   40083895 James Smith

5.6 Question 5: "What does <Student> know?"

For a student, list all topics that this student is familiar with.

REGULAR EXPRESSION : r'^what(\s)*does(\s)*(?P<student>.*\b\w*\b)(\s)*know(\s)*\?$'

Here, we have taken 2 queries for the input can either be student id or student name.

QUERY 1

```
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") .
}
```

QUERY 2

```
SELECT DISTINCT ?tName
WHERE{
?student rdf:type isp:Student .
?student foaf:givenName "James" .
?student foaf:familyName "Smith" .
?student isp:tookCourse ?courseGrade.
?courseGrade dbp:score ?grade .
?courseGrade dc:subject ?course .
?course isp:hasPart ?topic .
?topic foaf:name ?tName .
FILTER(?grade < "F") .
}
```

QUERY EXPLANATION

The query iterates all the triples and filter using the statements in the where clause. We find the student nodes linked to a specific student id (dbp:id), say "40083895" for query 1 or using first name(foaf:givenName) and last name(foaf:familyName) for query 2. We then retrieve the courses taken by the student using isp:tookCourse and through that we take the topics covered under it in course using dc:subject and further isp:hasPart. We then check if the grade(dbp:score) for those courses are better than F by using FILTER.

RESULTS

We were able to attain following results:

1.
```
Please enter the query (Please type 0 to exit):
What does 40083902 know?

40083902 knows:

perceptual organization
Computer Vision
OpenCV
COMP
computer
collision detection
Game engine
3D
Artificial Intelligence
AI
pathfinding
mobile gaming
```

2. Please enter the query (Please type 0 to exit):
   What does Maria Hernandez know?

   Maria Hernandez knows:

   perceptual organization
   Computer Vision
   OpenCV
   COMP
   computer
   collision detection
   Game engine
   3D
   Artificial Intelligence
   AI
   pathfinding
   mobile gaming

6. Tests

Link analysis

We have used link 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 name and description text. We have tested for any ambiguity or errors in the entities identified from 100 links detected by python spotlight and provided correct URL in case of wrong URIs. The data for link analysis is present in “LinkAnalysis.csv” file. The analysis showed 96% success.

	Course Su	Course Nu	Topic	URI Generated	Correct/Not Correct	Correct URI
1	COMP	691	Computer Science	http://dbpedia.org/resource/Computer_science	Correct	
2	SOEN	6011	Version control	http://dbpedia.org/resource/Version_control	Correct	
3	SOEN	6011	code review	http://dbpedia.org/resource/Code_review	Correct	
4	SOEN	6011	Continuous integration	http://dbpedia.org/resource/Continuous_integration	Correct	
5	SOEN	6021	software metrics	http://dbpedia.org/resource/Software_metric	Correct	
6	COMP	6231	distributed computing	http://dbpedia.org/resource/Distributed_computing	Correct	
7	COMP	6231	scalability	http://dbpedia.org/resource/Scalability	Correct	
8	COMP	6231	concurrency	http://dbpedia.org/resource/Concurrent_computing	Correct	
9	COMP	6231	fault tolerance	http://dbpedia.org/resource/Fault_tolerance	Correct	
10	COMP	6231	server	http://dbpedia.org/resource/Web_server	Incorrect	https://en.wikipedia.org/wiki/Server_(computing)
11	COMP	6281	heterogeneous computing	http://dbpedia.org/resource/Heterogeneous_computing	Correct	
12	COMP	6281	Parallel programming	http://dbpedia.org/resource/Parallel_computing	Correct	
13	SOEN	6311	software life-cycle	http://dbpedia.org/resource/Software_release_life_cycle	Correct	
14	SOEN	6311	Software development	http://dbpedia.org/resource/Software_development	Correct	
15	COMP	6321	logistic regression	http://dbpedia.org/resource/Logistic_regression	Correct	
16	COMP	6321	neural networks	http://dbpedia.org/resource/Artificial_neural_network	Correct	
17	COMP	6321	Unsupervised learning	http://dbpedia.org/resource/Unsupervised_learning	Correct	
18	COMP	6321	k-means	http://dbpedia.org/resource/K-means_clustering	Correct	
19	COMP	6321	Reinforcement learning	http://dbpedia.org/resource/Reinforcement_learning	Correct	
20	COMP	6331	Artificial Intelligence	http://dbpedia.org/resource/Artificial_intelligence	Correct	
21	COMP	6331	AI	http://dbpedia.org/resource/Artificial_intelligence	Correct	
22	COMP	6341	Computer Vision	http://dbpedia.org/resource/Computer_vision	Correct	
23	COMP	6341	OpenCV	http://dbpedia.org/resource/OpenCV	Correct	
24	COMP	6351	partial differential equations	http://dbpedia.org/resource/Partial_differential_equation	Correct	
25	COMP	6361	Numerical Analysis	http://dbpedia.org/resource/Numerical_analysis	Correct	
26	COMP	6381	reverse engineering	http://dbpedia.org/resource/Reverse_engineering	Correct	
27	SOEN	6431	software maintenance	http://dbpedia.org/resource/Software_maintenance	Correct	
28	SOEN	6441	unit tests	http://dbpedia.org/resource/Unit_testing	Correct	
29	SOEN	6441	multi-threading	http://dbpedia.org/resource/Thread_(computing)	Incorrect	https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)
30	SOEN	6441	code reuse	http://dbpedia.org/resource/Code_reuse	Correct	
31	COMP	6461	multicasting	http://dbpedia.org/resource/Multicast	Correct	
32	COMP	6461	UDP	http://dbpedia.org/resource/User_Datagram_Protocol	Correct	
33	COMP	6461	Network security	http://dbpedia.org/resource/Network_security	Correct	
34	SOEN	6461	object-oriented	http://dbpedia.org/resource/Object-oriented_design	Correct	
35	SOEN	6461	design patterns	http://dbpedia.org/resource/Software_design_pattern	Correct	
36	SOEN	6471	cloud computing	http://dbpedia.org/resource/Cloud_computing	Correct	
37	SOEN	6481	requirements engineering	http://dbpedia.org/resource/Requirements_engineering	Correct	
38	SOEN	6481	formal languages	http://dbpedia.org/resource/Formal_language	Correct	
39	SOEN	6481	quality assurance	http://dbpedia.org/resource/Quality_assurance	Correct	
40	SOEN	6481	UML	http://dbpedia.org/resource/Unified_Modeling_Language	Correct	
41	COMP	6521	optimization	http://dbpedia.org/resource/Query_optimization	Correct	
42	COMP	6521	Data warehouse	http://dbpedia.org/resource/Data_warehouse	Correct	
43	COMP	6521	Data mining	http://dbpedia.org/resource/Data_mining	Correct	
44	COMP	6521	XML	http://dbpedia.org/resource/XML	Correct	
45	COMP	6521	multimedia	http://dbpedia.org/resource/Multimedia	Correct	
46	COMP	6591	first-order logic	http://dbpedia.org/resource/First-order_logic	Correct	
47	COMP	6591	relational algebra	http://dbpedia.org/resource/Relational_algebra	Correct	
48	COMP	6591	relational calculus	http://dbpedia.org/resource/Relational_calculus	Correct	
49	COMP	6641	Turing machines	http://dbpedia.org/resource/Turing_machine	Correct	
50	COMP	6641	recursive functions	http://dbpedia.org/resource/Computable_function	Incorrect	https://en.wikipedia.org/wiki/Recursion_(computer_science)
51	COMP	6651	algorithm	http://dbpedia.org/resource/Algorithm	Correct	
52	COMP	6651	complexity analysis	http://dbpedia.org/resource/Analysis_of_algorithms	Correct	
53	COMP	6651	NP-complete	http://dbpedia.org/resource/NP-completeness	Correct	
54	COMP	6711	point location	http://dbpedia.org/resource/Point_location	Correct	
55	COMP	6711	robot	http://dbpedia.org/resource/Robotics	Correct	
56	COMP	6711	motion planning	http://dbpedia.org/resource/Motion_planning	Correct	
57	COMP	6711	computer graphics	http://dbpedia.org/resource/Computer_graphics	Correct	
58	COMP	6731	Pattern Recognition	http://dbpedia.org/resource/Pattern_recognition	Correct	
59	COMP	6731	Feature extraction	http://dbpedia.org/resource/Feature_extraction	Correct	
60	COMP	6751	natural language processing	http://dbpedia.org/resource/Natural-language_processing	Incorrect	https://en.wikipedia.org/wiki/Natural_language_processing
61	COMP	6751	text mining	http://dbpedia.org/resource/Text_mining	Correct	
62	SOEN	6751	User interface design	http://dbpedia.org/resource/User_interface_design	Correct	
63	SOEN	6751	User interface	http://dbpedia.org/resource/User_interface	Correct	
64	COMP	6761	software engineering	http://dbpedia.org/resource/Software_engineering	Correct	
65	COMP	6761	collision detection	http://dbpedia.org/resource/Collision_detection	Correct	
66	COMP	6761	physics	http://dbpedia.org/resource/Game_physics	Correct	
67	COMP	6771	frequency domain	http://dbpedia.org/resource/Frequency_domain	Correct	
68	COMP	6771	image segmentation	http://dbpedia.org/resource/Image_segmentation	Correct	
69	COMP	6771	Hough transform	http://dbpedia.org/resource/Hough_transform	Correct	
70	COMP	6771	edge detection	http://dbpedia.org/resource/Edge_detection	Correct	
71	COMP	6781	N-gram	http://dbpedia.org/resource/N-gram	Correct	
72	COMP	6781	part-of-speech tagging	http://dbpedia.org/resource/Part-of-speech_tagging	Correct	
73	COMP	6781	parsing	http://dbpedia.org/resource/Parsing	Correct	
74	COMP	6791	Information Retrieval	http://dbpedia.org/resource/Information_retrieval	Correct	
75	COMP	6791	vector space	http://dbpedia.org/resource/Vector_space	Correct	
76	COMP	6791	Tokenization	http://dbpedia.org/resource/Lexical_analysis	Correct	
77	COMP	6811	dynamic programming	http://dbpedia.org/resource/Dynamic_programming	Correct	
78	COMP	6821	database	http://dbpedia.org/resource/Database	Correct	
79	SOEN	6841	risk management	http://dbpedia.org/resource/Risk_management	Correct	
80	SOEN	6861	Service Oriented Architecture	http://dbpedia.org/resource/Service-oriented_architecture	Correct	
81	SOEN	6861	Web services	http://dbpedia.org/resource/Web_service	Correct	
82	SOEN	6861	SOAP	http://dbpedia.org/resource/SOAP	Correct	
83	COMP	7251	mobile computing	http://dbpedia.org/resource/Mobile_computing	Correct	
84	COMP	7251	PAN	http://dbpedia.org/resource/Personal_area_network	Correct	
85	COMP	7251	MAN	http://dbpedia.org/resource/Metropolitan_area_network	Correct	
86	COMP	7251	topology	http://dbpedia.org/resource/Network_topology	Correct	
87	COMP	7451	distributed systems	http://dbpedia.org/resource/Distributed_computing	Correct	
88	COMP	7521	cryptographic	http://dbpedia.org/resource/Cryptography	Correct	
89	COMP	7521	authentication	http://dbpedia.org/resource/Authentication	Correct	
90	COMP	7521	data integrity	http://dbpedia.org/resource/Data_integrity	Correct	
91	COMP	7521	confidentiality	http://dbpedia.org/resource/Confidentiality	Correct	
92	COMP	7521	access control	http://dbpedia.org/resource/Access_control	Correct	
93	SOEN	6481	IEEE	http://dbpedia.org/resource/Institute_of_Electrical_and_Electronics_Engineers	Correct	
94	SOEN	6481	ISO	http://dbpedia.org/resource/International_Organization_for_Standardization	Correct	
95	SOEN	6481	Agile	http://dbpedia.org/resource/Agile_software_development	Correct	
96	SOEN	6481	user stories	http://dbpedia.org/resource/User_story	Correct	
97	SOEN	6481	formal specification	http://dbpedia.org/resource/Formal_specification	Correct	
98	SOEN	6861	RESTful	http://dbpedia.org/resource/Representational_state_transfer	Correct	
99	SOEN	6491	Refactoring	http://dbpedia.org/resource/Code_refactoring	Correct	
100	SOEN	6211	Semantic Web	http://dbpedia.org/resource/Semantic_Web	Correct	

7. Conclusion

We have successfully created a system that scrapes data from Concordia University websites and stores the data in a Knowledge Graph. We have implemented a program that answers questions about our database through SPARQL queries. All the queries implemented in our program gave fast and accurate results. It was found by linking analysis of spotlight generated URI's of the entities, our project correctly identifies 96% of the DBpedia entities linked to the entities in our knowledge graph. Our project qualifies for the properties:

Scalable: The knowledge graph created can be expanded to as many numbers of courses or universities as needed. It can be expanded to contain information about other Universities or Departments too.

Flexible: Since we are taking all our University data from the university websites, any changes that occur in a course attributes or an addition of a new course would get updated in our knowledge graph every time we run the code. This ensures flexibility of the data in the KG.

Convenience: The program has a pleasant user interface which makes it easier to understand the knowledge graph and the queries related to it.

Accuracy: We have eliminated all duplicate entries and therefore data is stored only once.

8. Future Scope and Limitations

In this project we have limited our database to Concordia University alone. We could expand our knowledge graph to many Universities for our future scope. We could also store more information about a university such as the services offered by the universities, information about the faculties and their course and research topics, schedules of the courses, locations of the buildings, contact information of the departments etc. As we grow our knowledge graph, it also calls for us to optimise our code to handle much larger data so that the code doesn't take too long to run for bigger databases.

9. References

- https://www.regular-expressions.info/named.html
- https://regex101.com/