



**COMP 6741**

**Intelligent Systems**

**Project-1 Report**

**Roboprof**

Submitted To: Prof. Dr. René Witte

Date: 22 March 2024

**Team: AK\_G\_04**

Aryan Saxena (**40233170**)

Benjamin Douglas (**40264251**)

**GitHub Link**

We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality

A photograph of a handwritten signature in black ink, which appears to be "Aryan Saxena".

Aryan Saxena

40233170

22nd March 2024

A photograph of a handwritten signature in blue ink, which appears to be "J. Ben Douglas".

Benjamin Douglas

40264251

22nd March 2024

## Table of Contents

<b>1. Vocabulary .....</b>	<b>3</b>
A. Classes.....	3
B. Properties .....	3
<b>2. Data (CSVs) .....</b>	<b>4</b>
A. Data.csv.....	4
B. Lectures.csv .....	4
C. Students.csv.....	4
D. Topics.csv .....	4
<b>3. Fetching the Data and Creating Turtle Files.....</b>	<b>5</b>
A. Setting Up the Environment .....	5
B. Initiating RDF Graphs .....	5
C. Populating the University TTL.....	5
D. Populating the Courses TTL .....	5
E. Populating the Students TTL.....	5
F. Populating the Lectures TTL .....	5
G. Populating the Topics TTL.....	6
H. Error Handling and Validation .....	6
I. Serialization .....	6
<b>4. Turtle (.ttl) Files.....</b>	<b>7</b>
A. Universities.ttl .....	7
B. Courses.ttl .....	7
C. Lectures.ttl.....	8
D. Students.ttl.....	8
E. Topics.ttl.....	9
<b>5. Queries .....</b>	<b>11</b>
A. Setting up the SPARQL Endpoint .....	16
B. Crafting Queries.....	16
C. Executing Queries .....	16
D. Saving Results .....	16
E. Iterative Approach .....	16
F. Verification .....	17
<b>6. Updates.....</b>	<b>17</b>
<b>7. Knowledge Base Creation.....</b>	<b>17</b>
A. Converting Documents into readable text files.....	17

B. KB creation script.....	18
C. Query Outputs.....	18
<b>8. Chatbot Design.....</b>	<b>19</b>
A. ROSA framework implementation.....	19
B. Query Examples.....	24

## **1. Vocabulary**

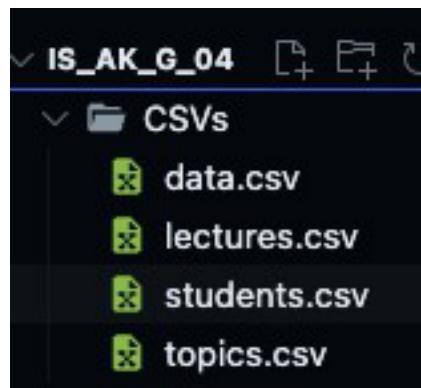
### **A. Classes**

- a. **University, Course, Lecture, Topic, Student:** These are the primary classes, each representing fundamental concepts such as educational institutions, units of teaching, individual instructional sessions, subjects covered, and learners.
- b. **LectureContent (with subclasses Slides, Worksheets, Readings, OtherMaterial):** This hierarchy models different types of materials provided in lectures, offering a structured way to represent educational resources.

### **B. Properties**

- a. **General Properties like hasName, hasLink:** These are used across different classes to denote names and links respectively, applicable to various resources.
- b. **Relationships between Universities and Courses (offersCourse), Courses and their attributes (courseSubject, courseNumber, etc.), Lectures and their specifics (lectureNumber, lectureName, etc.):** These properties define how different entities are interconnected and describe specific attributes of courses and lectures.
- c. **Properties for Student interactions (studentID, completedCourse, courseGrade, hasCompetency):** These capture student-related data, including their identity, courses they've completed, their grades, and competencies in specific topics.
- d. **Linking Topics with Courses and Lectures (isTopicOfLecture, isTopicOfCourse):** These properties explicitly connect topics to the lectures and courses where they are covered.

## **2. Data (CSVs)**



*Fig 1: Structure*

Secondly, Focusing on the datasets we're using as the foundation for creating our .ttl (Turtle) files, which are essential for our RDF framework.

The datasets are structured in CSV format, coming from various educational facets:

- A. [Data.csv](#): Harvested from Concordia's open data portal, this extensive file includes course details such as IDs, titles, units, descriptions, prerequisites, and equivalent courses. This will allow us to define and connect courses within our RDF schema accurately.
- B. [Lectures.csv](#): Contains information about individual lectures linked to their respective courses, including lecture IDs, numbers, names, content, and additional resources.
- C. [Students.csv](#): Provides a snapshot of student information, including their names, ID numbers, email addresses, the courses they've taken, their grades, and areas of competency.
- D. [Topics.csv](#): Lists out topics covered in various lectures and courses, along with IDs, names, and links to external resources for more information.

These CSV files are not just rows and columns of data; they represent the backbone of our semantic data model, allowing for an intricate web of interlinked information reflecting the complex reality of university academic structures.

### **3. Fetching the Data and Creating Turtle Files**

The fourth point in our report delves into the process of **transforming CSV data into Turtle (TTL) files**, which are pivotal for populating our RDF framework.

Fetching the Data and Creating Turtle Files:

#### **A. Setting Up the Environment:**

- Import necessary libraries in Python: pandas for data manipulation, rdflib for RDF operations, SPARQLWrapper for querying.
- Define paths to CSV source files and destination TTL files.

#### **B. Initiating RDF Graphs:**

- Create RDF Graph instances for each TTL file.
- Define and bind the namespaces required for RDF triples such as 'ex', 'foaf', 'rdf', 'rdfs', 'xsd', 'dbo', and 'dbr'.

#### **C. Populating the University TTL:**

- Run a SPARQL query to fetch data about universities from DBpedia.
- Manually add Concordia University if it's not present.
- Serialize and save the university data into 'universities.ttl'.

#### **D. Populating the Courses TTL:**

- Read 'data.csv' containing course information from Concordia's open data.
- Iterate through each row, creating URIs for courses and adding relevant triples (such as course subject, number, description, etc.).
- Serialize and save the course data into 'courses.ttl'.

#### **E. Populating the Students TTL:**

- Read 'students.csv' for student information.
- For each student, create a URI and add triples denoting their name, email, student ID.
- For courses completed by the student, create blank nodes to represent the relationship between the student, the course, and their grade.
- Serialize and save the student data into 'students.ttl'.

#### **F. Populating the Lectures TTL:**

- Read 'lectures.csv' for lecture details.
- For each lecture, create a URI and add triples detailing the lecture's number, name,

- content link, and additional resources.
- Serialize and save the lecture data into 'lectures.ttl'.

#### G. Populating the Topics TTL:

- Read 'topics.csv' for academic topics discussed in lectures.
- For each topic, create a URI and add triples detailing the topic's name, provenance, and link.
- Link each topic to its corresponding lecture and course.
- Serialize and save the topic data into 'topics.ttl'.

#### H. Error Handling and Validation:

- Ensure that the code handles missing data gracefully, such as checking for null values.
- Validate the URIs created and ensure proper encoding to avoid issues in RDF triples.

#### I. Serialization:

- Convert the populated graphs to Turtle format.
- Serialize and save the graphs to the defined paths, creating the TTL files for use in RDF stores or other applications.

Each step ensures the data is **correctly structured and relationships** are defined following the RDF standards, allowing the semantic data to be queried and understood in a machine-readable format.

## 4. Turtle (.ttl) Files

For the fourth part of our report, we'll present visual documentation of the Turtle (TTL) files generated. Here's how it will be structured for inclusion in the Word document:

### A. Universities.ttl

- *Screenshot:*

```
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix ex: <http://example.org/vocab/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

dbr:Bay_River_College a dbo:University ;
    rdfs:label "Bay River College"@en .

dbr:Beedie_School_of_Business a dbo:University ;
    rdfs:label "Beedie School of Business"@en .

dbr:Bishop_Feild_College a dbo:University ;
    rdfs:label "Bishop Feild College"@en .

dbr:Booth_University_College a dbo:University ;
    rdfs:label "Booth University College"@en .

dbr:Bora_Laskin_Faculty_of_Law a dbo:University ;
    rdfs:label "Bora Laskin Faculty of Law"@en .
```

- **Description:** Captures RDF data of various universities, including Concordia, each with its own URI and a label in English.

### B. Courses.ttl

- *Screenshot:*

```
@prefix ex: <http://example.org/vocab/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://example.org/vocab/course/1000> a ex:Course ;
    ex:credits 3.0 ;
    ex:description "Understanding Myths"^^xsd:string ;
    ex:number "307"^^xsd:string ;
    ex:preRequisiteDescription "Course Prerequisite: ANTH202 or equivalent AND minimum 3 more credits 200-level ANTH"^^xsd:string ;
    ex:subject "ANTH"^^xsd:string .

<http://example.org/vocab/course/10005> a ex:Course ;
    ex:credits 3.0 ;
    ex:description "Probability and Statistics in Engineering"^^xsd:string ;
    ex:number "371"^^xsd:string ;
    ex:preRequisiteDescription "Course Prerequisite: ENGR213; ENGR233"^^xsd:string ;
    ex:subject "ENGR"^^xsd:string .

<http://example.org/vocab/course/10009> a ex:Course ;
    ex:credits 3.0 ;
    ex:description "Numerical Methods in Engineering"^^xsd:string ;
    ex:equivalentCourses "EMAT 391 = ENGR 391"^^xsd:string ;
    ex:number "391"^^xsd:string ;
    ex:preRequisiteDescription "Course Prerequisite: ENGR213; ENGR 233; You must complete 1 of the following courses COMP248, COEN243, ENGR"^^xsd:string .
```

- **Description:** Outlines course offerings with details like subject and prerequisites, structured into RDF format.

### C. Lectures.ttl

- *Screenshot:*

```

@prefix ex: <http://example.org/vocab/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://example.org/vocab/lecture/1> a ex:Lecture ;
    ex:lectureName "Introduction to AI: Overview & History"^^xsd:string,
        "Introduction to Intelligent Systems"^^xsd:string ;
    ex:lectureOfCourse <http://example.org/vocab/course/40353>,
        <http://example.org/vocab/course/40355> ;
    ex:lectureTopic <http://example.org/vocab/topic/Artificial_intelligence>,
        <http://example.org/vocab/topic/Intelligent_Systems> ;
    rdfs:seeAlso <http://dbpedia.org/resource/Artificial_intelligence>,
        <http://dbpedia.org/resource/Intelligent_system> .

<http://example.org/vocab/lecture/10> a ex:Lecture ;
    ex:lectureName "Introduction to Deep Learning"^^xsd:string,
        "Introduction to Natural Language Processing (NLP)"^^xsd:string ;
    ex:lectureOfCourse <http://example.org/vocab/course/40353>,
        <http://example.org/vocab/course/40355> ;
    ex:lectureTopic <http://example.org/vocab/topic/Deep_learning>,
        <http://example.org/vocab/topic/Natural_language_processing> ;
    rdfs:seeAlso <http://dbpedia.org/resource/Deep_learning>,
        <http://dbpedia.org/resource/Natural_language_processing> .

```

- **Description:** Each lecture is detailed with a unique URI, showing its connection to courses and the topics discussed.

### D. Students.ttl

- *Screenshot:*

```

@prefix ex: <http://example.org/vocab/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://example.org/vocab/student/S1001> a ex:Student ;
    ex:completedCourse [ a ex:CompletedCourse ;
        ex:course <http://example.org/vocab/course/5570> ;
        ex:courseGrade "A"^^xsd:string ],
        [ a ex:CompletedCourse ;
            ex:course <http://example.org/vocab/course/5605> ;
            ex:courseGrade "A"^^xsd:string ],
        [ a ex:CompletedCourse ;
            ex:course <http://example.org/vocab/course/5477> ;
            ex:courseGrade "C+"^^xsd:string ],
        [ a ex:CompletedCourse ;
            ex:course <http://example.org/vocab/course/5527> ;
            ex:courseGrade "A-"^^xsd:string ] ;
    ex:hasCompetency <http://example.org/vocab/course/5409>,
        <http://example.org/vocab/course/5411> ;
    ex:studentID "S1001"^^xsd:string ;
    foaf:mbox "aryan.saxena@concordia.ca"^^xsd:string ;
    foaf:name "Aryan Saxena"^^xsd:string .

```

- **Description:** Student profiles are represented here, linking them to the courses they've taken and their performance.

## E. Topics.ttl

- **Screenshot:**

```

@prefix ex: <http://example.org/vocab/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://dbpedia.org/resource/00038> a ex:Topic ;
    ex:entityType "PERSON"^^xsd:string ;
    ex:isTopicOfCourse <http://example.org/vocab/course/40353> ;
    ex:isTopicOfLecture <http://example.org/vocab/lecture/7> ;
    ex:materialType "Lecture"^^xsd:string ;
    ex:topicName "00038"^^xsd:string .

<http://dbpedia.org/resource/055524> a ex:Topic ;
    ex:entityType "ORG"^^xsd:string ;
    ex:isTopicOfCourse <http://example.org/vocab/course/40355> ;
    ex:isTopicOfLecture <http://example.org/vocab/lecture/9> ;
    ex:materialType "Lecture"^^xsd:string ;
    ex:topicName "055524"^^xsd:string .

<http://dbpedia.org/resource/08808x11> a ex:Topic ;
    ex:entityType "ORG"^^xsd:string ;
    ex:isTopicOfCourse <http://example.org/vocab/course/40353> ;
    ex:isTopicOfLecture <http://example.org/vocab/lecture/7> ;
    ex:materialType "Lecture"^^xsd:string ;
    ex:topicName "08808x11"^^xsd:string .

```

- **Description:** Defines the academic topics taught, linking them to the relevant lectures and courses.

## 5. Queries

### Query 1: Finding Courses at a University

The screenshot shows a SPARQL query interface. The query is:

```
1 v PREFIX ex: <http://example.org/ns#>
2 v PREFIX dbo: <http://dbpedia.org/ontology/>
3 v PREFIX dbr: <http://dbpedia.org/resource/>
4 v
5 v SELECT ?course WHERE {
6 v   dbr:Concordia_University dbo:offersCourse ?course .
7 }
```

The results table shows four courses:

course
1 <http://example.org/vocab/course/5477>
2 <http://example.org/vocab/course/5605>
3 <http://example.org/vocab/course/5527>
4 <http://example.org/vocab/course/5570>

This query searches for all the courses offered by a specific university, showcasing the educational opportunities available there.

### Query 2: Discovering Courses Covering a Topic

The screenshot shows a SPARQL query interface. The query is:

```
1 v PREFIX ex: <http://example.org/vocab/>
2 v PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3 v
4 v SELECT ?course ?courseName
5 v WHERE {
6 v   ?topic ex:topicName "CNN"^^xsd:string .
7 v   ?topic ex:isTopicOfCourse ?course .
8 v   ?course ex:description ?courseName .
9 }
10
```

The results table shows two courses with their names:

course	courseName
1 <http://example.org/vocab/course/40353>	APPLIED ARTIFICIAL INTELLIGENCE
2 <http://example.org/vocab/course/40355>	INTELLIGENT SYSTEMS

It identifies which courses discuss a particular topic, highlighting the academic exploration of specific subjects.

### Query 3: Exploring Topics in a Lecture

Example Queries

**Selection of triples** **Selection of classes**

SPARQL Endpoint /roboprof/sparql Content Type (SELECT) JSON Content Type (GRAPH) Turtle

Prefixes: rdf rdfs owl xsd

```

1 v PREFIX ex: <http://example.org/vocab/>
2 v PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3
4 v SELECT ?topicName
5 v WHERE {
6 v   ?topic ex:isTopicOfCourse <http://example.org/vocab/course/40355> ;
7 v     ex:isTopicOfLecture <http://example.org/vocab/lecture/> ;
8 v     ex:topicName ?topicName .
9 v }

```

Table Response 152 results in 0.016 seconds

Simple view Ellipse Filter query results Page size: 50

topicName
1 AC
2 AIML
3 AK
4 ALICE
5 ANNIE

This query reveals what topics are covered in a specific lecture of a course, offering a peek into the lecture's focus.

#### Query 4: Listing Subject-Specific Courses at a University

(SELECT) (GRAPH)

/Roboprof/sparql JSON Turtle

```

1 v PREFIX ex: <http://example.org/vocab/>
2 v PREFIX dbo: <http://dbpedia.org/ontology/>
3
4 v SELECT ?course ?subject ?number
5 v WHERE {
6 v   ?university dbo:offersCourse ?course .
7 v   ?course a ex:Course ;
8 v     ex:subject ?subject ;
9 v     ex:number ?number .
10 v    FILTER(?subject = "COMP" || ?subject = "SOEN")
11 v }

```

Table Response 164 results in 0.039 seconds

Simple view Ellipse Filter query results Page size: 50

course	subject	number
<http://example.org/vocab/course/5477>	COMP	465

It lists all courses under certain subjects offered by a university, pointing out specialized educational paths.

#### Query 5: Materials Recommended for a Topic in a Course

```

1 v PREFIX ex: <http://example.org/vocab/>
2 v PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 v SELECT DISTINCT ?lectureName
5 v WHERE {
6 v   # Select the specific course by subject and number
7 v   ?course a ex:Course ;
8 v     ex:subject "COMP"^^xsd:string;
9 v     ex:number "6741"^^xsd:string.
10 v
11 v   # Find topics that are part of this course
12 v   ?topic a ex:Topic ;
13 v     ex:isTopicOfCourse ?course;
14 v     ex:topicName "Amazon"^^xsd:string. # Adjust topic name as necessary
15 v
16 v   # Find lectures that include this topic
17 v   ?topic ex:isTopicOfLecture ?lecture.
18 v
19 v   # Get the names of these lectures
20 v   ?lecture ex:lectureName ?lectureName.
21 v }

```

Table Response 11 results in 0.01 seconds

Simple view Ellipse Filter query results Page size: 50

lectureName
1 Introduction to AI: Overview & History
2 Introduction to Intelligent Systems
3 Introduction to Deep Learning
4 Introduction to Natural Language Processing (NLP)

It lists all the required materials for a given topic for the given course

a particular topic within a course, guiding resourceful learning.

### Query 6: Credit Value of a Course

The screenshot shows a SPARQL query interface with the following details:

- Endpoint:** /Roboprof/sparql
- Content Type (SELECT):** JSON
- Content Type (GRAPH):** Turtle
- Query:**

```
1 v PREFIX ex: <http://example.org/vocab/>
2 v
3 v SELECT ?credits
4 v WHERE {
5 v   ?course a ex:Course ;
6 v     ex:subject "COMP" ;
7 v     ex:number "6651" ;
8 v     ex:credits ?credits .
9 v
10 }
```

- Results:** 1 result in 0.021 seconds
- Table:** credits
- Response:** 1 \*4.0\*^<<http://www.w3.org/2001/XMLSchema#decimal>>

It determines the credit value of a specific course, informing on its academic weight.

### Query 7: Additional Resources for a Course

The screenshot shows a SPARQL query interface with the following details:

- Example Queries:** Selection of triples (selected), Selection of classes
- SPARQL Endpoint:** /roboprof/sparql
- Content Type (SELECT):** JSON
- Content Type (GRAPH):** Turtle
- Prefixes:** rdf, rdfs, owl, xsd
- Query:**

```
1 v
2 v PREFIX ex: <http://example.org/vocab/>
3 v PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4 v
5 v SELECT ?seeAlso
6 v WHERE {
7 v   ?lecture a ex:lecture ;
8 v     ex:lectureOfCourse ?course ;
9 v     rdfs:seeAlso ?seeAlso .
10 v  ?course ex:number "6721" ;
11 v    ex:subject "COMP" .
12 v }
```

- Results:** 23 results in 0.01 seconds
- Table:** seeAlso
- Response:** 1 <[http://dbpedia.org/resource/Artificial\\_intelligence](http://dbpedia.org/resource/Artificial_intelligence)>  
2 <[http://dbpedia.org/resource/Intelligent\\_system](http://dbpedia.org/resource/Intelligent_system)>  
3 <[http://dbpedia.org/resource/Deep\\_learning](http://dbpedia.org/resource/Deep_learning)>  
4 <[http://dbpedia.org/resource/Natural\\_language\\_processing](http://dbpedia.org/resource/Natural_language_processing)>  
5 <[http://dbpedia.org/resource/Natural\\_language\\_processing](http://dbpedia.org/resource/Natural_language_processing)>

This query finds extra resources, like web links, for a course, providing avenues for extended learning.

### Query 8: Content Available for a Lecture

Example Queries

**Selection of triples** **Selection of classes**

SPARQL Endpoint: /roboprof/sparql

Content Type (SELECT): JSON

Prefixes: rdf rdfs owl xsd

Content Type (GRAPH): Turtle

```

1+ PREFIX ex: <http://example.org/vocab/>
2+ PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3+
4+ SELECT ?topicName ?topicLink ?materialType
5+ WHERE {
6+   ?course a ex:Course;
7+   ex:subject "COMP"^^xsd:string;
8+   ex:number "6741"^^xsd:string.
9+
10+  ?lecture a ex:Lecture;
11+  ex:lectureOfCourse ?course;
12+  BIND (IRI(CONCAT("http://example.org/vocab/lecture/", "?")) AS ?specificLecture)
13+
14+  ?topic ex:isTopicOfLecture ?specificLecture;
15+  a ex:Topic;
16+  ex:topicName ?topicName;
17+  ex:materialType ?materialType.
18+
19+  BIND (IRI(?topic) AS ?topicLink)
20+ }

```

Table Response 2387 results in 0.064 seconds

Simple view Ellipse Filter query results Page size: 50

topicName	topicLink	materialType
00038	<http://dbpedia.org/resource/00038>	Lecture
08800x11	<http://dbpedia.org/resource/08800x11>	Lecture
AC	<http://dbpedia.org/resource/AC>	Lecture
AIML	<http://dbpedia.org/resource/AIML>	Lecture
AK	<http://dbpedia.org/resource/AK>	Lecture

It identifies all the content types available for a specific lecture, giving insights into the lecture's materials.

## Query 9: Recommended Reading for a Topic

Example Queries

**Selection of triples** **Selection of classes**

SPARQL Endpoint: /roboprof/sparql

Content Type (SELECT): JSON

Prefixes: rdf rdfs owl xsd

Content Type (GRAPH): Turtle

```

1+ PREFIX ex: <http://example.org/vocab/>
2+ PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3+
4+ SELECT ?lectureNumber ?materialType
5+ WHERE {
6+   # Identifying the specific course by subject and number
7+   ?course a ex:Course;
8+   ex:subject "COMP"^^xsd:string; # Specify the course subject
9+   ex:number "6741"^^xsd:string. # Specify the course number
10+
11+  # Finding topics associated with this course and matching a specified topic name
12+  ?topic a ex:Topic;
13+  ex:topicName "Amazon"^^xsd:string; # Specify the topic of interest
14+  ex:isTopicOfCourse ?course.
15+
16+  # Getting lecture numbers and material types linked to the topic
17+  ?topic ex:isTopicOfLecture ?lecture;
18+  ex:materialType ?materialType.
19+
20+  # Extracting lecture number from the lecture URI
21+  BIND(REPLACE(STR(?lecture), "http://example.org/vocab/lecture/", "") AS ?lectureNumber)
22+ }

```

Table Response 6 results in 0.015 seconds

Simple view Ellipse Filter query results Page size: 50

lectureNumber	materialType
1	Lecture
10	Lecture
2	Lecture
4	Lecture
5	Lecture

No result because no matches. This query lists reading materials recommended for a topic within a course, aiding in-depth study.

## Query 10: Competencies Gained from a Course

The screenshot shows a SPARQL query editor with the following details:

- Query:**

```

1 v PREFIX ex: <http://example.org/vocab/>
2
3 SELECT ?topic
4 v WHERE {
5   ?student ex:completedCourse ?completedCourse .
6   ?completedCourse ex:course ?course .
7   ?course ex:number "6651" .
8   ?student ex:hasCompetency ?topic .
9 }

```
- Results:**
  - Table view: 8 results in 0.023 seconds.
  - Response: Shows two rows of results under the column "topic".

topic
1 <http://example.org/vocab/course/5409>
2 <http://example.org/vocab/course/5411>
- Controls:** Includes "Simple view", "Ellipsis", "Filter query results", "Page size: 50", download, and help icons.

It explores the topics a student becomes proficient in after completing a course, reflecting on the learning outcome.

### Query 11: Student's Grades in a Course

The screenshot shows a SPARQL query editor with the following details:

- Query:**

```

1 v PREFIX ex: <http://example.org/vocab/>
2
3 SELECT ?grade
4 v WHERE {
5   ?student ex:completedCourse ?completedCourse .
6   ?completedCourse ex:course ?course ;
7     ex:courseGrade ?grade .
8   ?course ex:number "6651" .
9   FILTER(?student = <http://example.org/vocab/student/S1001>)
10 }

```
- Results:**
  - Table view: 1 result in 0.027 seconds.
  - Response: Shows one row of results under the column "grade".

grade
1 A
- Controls:** Includes "Simple view", "Ellipsis", "Filter query results", "Page size: 50", download, and help icons.

This query fetches the grades a student achieved in a course, reflecting their academic performance.

### Query 12: Students Who Completed a Course

The screenshot shows a SPARQL query editor with the following details:

- Query:**

```

1 PREFIX ex: <http://example.org/vocab/>
2
3 SELECT ?student
4 WHERE {
5   ?student ex:completedCourse ?completedCourse .
6   ?completedCourse ex:course ?course .
7   ?course ex:number "6651" .
8 }
```
- Results:**

Table view (4 results in 0.013 seconds):

student
1 <http://example.org/vocab/student/S1001>
2 <http://example.org/vocab/student/S1002>

It identifies students who have completed a specific course, showing who has advanced through that academic challenge.

### Query 13: Printing a Student's Transcript

The screenshot shows a SPARQL query editor with the following details:

- Query:**

```

1 PREFIX ex: <http://example.org/vocab/>
2
3 SELECT ?courseNumber ?grade
4 WHERE {
5   ?student ex:completedCourse ?completedCourse .
6   ?completedCourse ex:course ?course ;
7     ex:courseGrade ?grade .
8   ?course ex:number ?courseNumber .
9   FILTER(?student = <http://example.org/vocab/student/S1001>)
10 }
11 
```
- Results:**

Table view (4 results in 0.017 seconds):

courseNumber	grade
1 465	C+
2 6651	A
3 5481	A-
4 6281	A

This query compiles a student's courses and grades into a transcript, summarizing their academic journey.

## Running Procedure



Fig: File Structure

The queries executed using our local Fuseki server, which provides a SPARQL endpoint for interacting with RDF data. Here's the breakdown of our process:

### A. Setting up the SPARQL Endpoint:

- Initialized a SPARQL Wrapper with the endpoint URL pointing to our local Fuseki server.

### B. Crafting Queries:

- Wrote SPARQL queries tailored to extract specific information from the RDF data.
- Each query was designed to target a distinct aspect of our dataset, ranging from course details offered by universities to lecture materials and student grades.

### C. Executing Queries:

- Set each query to the SPARQL endpoint through the SPARQL Wrapper.
- Configured the return format to CSV for easy data manipulation and review.

### D. Saving Results:

- Executed the queries and saved the results directly to corresponding CSV files, named after the specific query number, such as "q13.csv" for the 13th query.
- This structured approach allowed us to keep our results organized and accessible for further analysis.

### E. Iterative Approach:

- Repeated this process for a series of queries, incrementally building up our collection of CSV files.
- This iterative method allowed for flexibility and the ability to refine queries as needed.

#### F. Verification:

- After running each query, we verified the output by checking the CSV files, ensuring the results matched our expectations.

The queries served as a powerful tool to dissect and understand our RDF data, bringing insights and clarity to the academic structure we've modeled. The systematic approach ensured a consistent, clear, and documented process, reflecting the depth and interconnectivity of our semantic dataset.

## **6. UPDATES**

After the initial assessment, it was found the necessary data was not provided in the knowledge graph to give correct outputs for queries 5,8,9. This was rectified by correctly providing data and building the knowledge graph to include these data in it.

## **7. Knowledge Base Creation**

### **A. Document processing:**

The initial step in processing educational content involves converting various types of course materials—such as PDF files, PowerPoint presentations (PPT), and other document formats—into plain text. This conversion is crucial because it standardizes the format of the input data, ensuring that the subsequent analysis, such as natural language processing (NLP), can be applied uniformly across all documents.

### **B. Linking to Linked Open Data**

Utilized the power of spaCy based custom entity linker and organizes this data into a structured format. It iteratively reads text files from specified directories, cleans and segments the text, identifies relevant entities, and stores the results in a pandas Data Frame, which is then saved as a CSV file.

### **C. Triple Creation**

Once the entities within the text have been successfully identified and linked to appropriate URIs from Linked Open Data (LOD) sources like DBpedia or Wikidata, the next crucial step in semantic data processing involves constructing RDF (Resource Description Framework) triples. RDF is a standard model for data interchange on the web, and triples are the fundamental structures used in RDF, each consisting of a subject, predicate, and object, forming a statement about resources in the form of “subject-predicate-object.”

Statistics	Value / Count
Total Number of Triples	58492
Number of distinct topics	1178

## 8. Chatbot Design

RASA is an open-source framework designed for developing conversational AI applications, such as chatbots. RASA involves two primary components: RASA NLU for understanding user inputs and RASA Core for managing dialogues.

### A. RASA Frame Work Implementation

For this project ROSA chatbot framework, you can focus on creating a simple, interactive robot that responds to specific commands or questions. The following steps were followed.

1. Latest version of ROSA was installed and added into the workspace
2. Defined intents on the NLU training data in nlu.yml, that help the bot understand what users mean.

```
# 1
- intent: list_university_courses
  examples: |
    - List all courses offered by [Concordia University](university).
    - Show me all courses available at [Stanford](university).
    - What courses does [MIT](university) offer?
    - List all courses at [Harvard](university).
    - Show me all courses at [McGill University](university).

# 2
- intent: find_courses_by_topic
  examples: |
    - In which courses is [Graph Theory](topic) discussed?
    - Where can I find information on [machine learning](topic)?
    - Is [data science](topic) covered in any courses?
    - Do any courses discuss [cryptography](topic)?
    - Can you tell me about courses dealing with [neural networks](topic)?
```

3. Defined the dialogue management in stories.yml, which are sample conversations that teach the bot how to respond. They link intents to responses and actions.

```

- story: University courses listing
  steps:
    - intent: list_university_courses
    - action: action_list_university_courses

- story: Find courses by topic
  steps:
    - intent: find_courses_by_topic
    - action: action_find_courses_by_topic

- story: List topics in a course
  steps:
    - intent: list_topics_in_course
    - action: action_list_topics_in_course

- story: Courses by subject listing
  steps:
    - intent: list_courses_by_subject
    - action: action_list_courses_by_subject

```

4. Added the necessary rule for a smooth workflow in rules.yml.

```

# Rule for listing university courses
- rule: List University Courses
  steps:
    - intent: list_university_courses
    - action: action_list_university_courses

# Rule for finding courses by topic
- rule: Find Courses by Topic
  steps:
    - intent: find_courses_by_topic
    - action: action_find_courses_by_topic

# Rule for listing topics in a course
- rule: List Topics in a Course
  steps:
    - intent: list_topics_in_course
    - action: action_list_topics_in_course

```

5. Define domain which ties together intents, entities, actions, templates, and slots (variables to store conversation state) in domain.yml

```
intents:  
- greet  
- goodbye  
- affirm  
- deny  
- mood_great  
- mood_unhappy  
- bot_challenge  
- list_university_courses  
- find_courses_by_topic  
- list_topics_in_course  
- list_courses_by_subject  
- recommended_materials_for_topic  
- course_credits  
- additional_course_resources  
- detail_course_content  
- recommended_reading_for_topic  
- competencies_gained  
- student_grades  
- students_completed_course  
- print_student_transcript  
- course_description  
- query_topics_in_course_event  
- query_events_covering_topic  
  
entities:  
- university  
- topic  
- course_number  
- event  
- subject  
- person
```

6. Training the model on both the NLU and dialogue management models using the provided examples and stories. These python scripts are stores in actions/actions.py
7. Testing and Interacting with the bot using 'rasa shell' command

```

| 99/100 [00:31<00:00,  4.35it/s,Epochs: 100%] | 100/100 [00:31<00:00,  4.35it/s,Epochs: 100%]
| 100/100 [00:31<00:00,  3.14it/s, t_loss=1.26, loss=0.974, acc=1]
2024-04-15 22:10:50 INFO rasa.engine.training.hooks - Finished training component 'TEDPolicy'.
2024-04-15 22:10:50 INFO rasa.engine.training.hooks - Starting to train component 'Unexp TEDIntentPolicy'.
2024-04-15 22:10:51 WARNING rasa.shared.utils.common - The Unexp TED Intent Policy is currently experimental and might change or be removed in the future. Please share your feedback on it in the forum (https://forum.rasa.com) to help us make this feature ready for production.
Processed trackers: 100%|██████████| 523/523 [00:00<00:00, 10264.57it/s, # intent=170]
2024-04-15 22:10:51 WARNING absl - At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2 Macs, please use the legacy Keras optimizer instead, located at `tf.keras.optimizers.legacy.Adam`.
2024-04-15 22:10:51 WARNING absl - There is a known slowdown when using v2.11+ Keras optimizers on M1/M2 Macs. falling back to the legacy Keras optimizer, i.e., `tf.keras.optimizers.legacy.Adam`.
Epochs: 100%|██████████| 100/100 [00:08<00:00, 11.67it/s, t_loss=0.447, loss=0.358, acc=0
2024-04-15 22:11:01 INFO rasa.engine.training.hooks - Finished training component 'Unexp TEDIntentPolicy'.
Your Rasa model is trained and saved at 'models/20240415-220945-asphalt-volt.tar.gz'.

```

Fig: Rasa training status

```

SANIC_VERSION = LooseVersion(sanic_version)
2024-04-15 22:11:26 INFO rasa_sdk.endpoint - Starting action endpoint server...
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_list_university_courses'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_find_courses_by_topic'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_list_topics_in_course_during_event'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_list_courses_by_subject'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_recommended_materials_for_topic'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_course_credits'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_additional_course_resources'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_detail_course_content'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_recommended_reading_for_topic'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_competencies_gained'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_student_grades'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_student_completed_courses'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'print_student_transcript'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_course_description'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_topics_covered_in_course_event'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_events_covering_topic'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_list_topics_in_course'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_count_topic_occurrences'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_list_description_and_info_for_topic'.
2024-04-15 22:11:26 INFO rasa_sdk.executor - Registered function for 'action_no_material_for_courses'.
2024-04-15 22:11:26 INFO rasa_sdk.endpoint - Starting plugins...
2024-04-15 22:11:26 INFO rasa_sdk.endpoint - Action endpoint is up and running on http://0.0.0.0:5055

```

Fig: Rasa Run Actions

## C. Query Outputs

**QUERY 1:** What is the <course> about?

The screenshot shows a SPARQL endpoint interface with the following details:

- SPARQL Endpoint:** /roboprof/query
- Content Type (SELECT):** JSON
- Content Type (GRAPH):** Turtle
- Query:**

```

1 PREFIX ex: <http://example.org/vocab/>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3 SELECT ?description
4 WHERE {
5   ?course a ex:Course;
6   ex:subject "COMP"^^xsd:string;
7   ex:number "6721"^^xsd:string;
8   ex:description ?description.
9 }

```
- Results:**

description
1 APPLIED ARTIFICIAL INTELLIGENCE
- Bottom Navigation:** Simple view □ Ellipse ⚙ Filter query results □ Page size: 50 □

It gives the description for the given queried course.

## Query 2: Which topics are covered in <course event>?

The screenshot shows a SPARQL query interface with the following details:

- Query Editor:** /roboprof/query
- Result Format:** JSON (selected), Turtle (available)
- Code:**

```

1 PREFIX ex: <http://example.org/vocab/>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 SELECT ?topicName ?topicLink
5 WHERE {
6   ?course a ex:Course;
7   ex:subject "COMP"^^xsd:string;
8   ex:number "6721"^^xsd:string.
9
10  ?lecture a ex:Lecture;
11    ex:lectureNumber 2 ;
12    ex:lectureOfCourse ?course.
13
14  ?topic a ex:Topic;
15    ex:topicName ?topicName;
16    ex:isTopicOfLecture ?lecture.
17  BIND (STR(?topic) AS ?topicLink)
18 }
```
- Results:**

topicName	topicLink
A12	http://dbpedia.org/resource/A12
A2	http://dbpedia.org/resource/A2
AI	http://dbpedia.org/resource/AI
ANJA	http://dbpedia.org/resource/ANJA
API	http://dbpedia.org/resource/API
Alice	http://dbpedia.org/resource/Alice
Amazon	http://dbpedia.org/resource/Amazon
Amitav	http://dbpedia.org/resource/Amitav
Annotated	http://dbpedia.org/resource/Annotated
Auteur	http://dbpedia.org/resource/Auteur
- Metrics:** 193 results in 0.025 seconds
- Buttons:** Simple view, Ellipse, Filter query results, Page size: 50, etc.

It give the list of topics covered in a particular course.

## Query 3: Which course events cover <Topic>?

The screenshot shows a SPARQL query interface with the following details:

- Query Editor:** /roboprof/query
- Result Format:** JSON (selected), Turtle (available)
- Code:**

```

1 PREFIX ex: <http://example.org/vocab/>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 SELECT ?course ?event ?materialType ?eventName
5 WHERE {
6   ?topic ex:topicName "COMP"^^xsd:string;
7   ex:isTopicOfLecture ?lecture;
8   ex:materialType ?materialType.
9
10  # Retrieve lecture details
11  ?lecture a ex:Lecture;
12    ex:lectureName ?eventName;
13    ex:lectureOfCourse ?course.
14
15  # Retrieve course details
16  ?course a ex:Course;
17    ex:description ?courseDescription.
18 }
```
- Results:**

course	event	materialType	eventName
<http://example.org/vocab/course/40353>	Lecture		Introduction to Deep Learning
<http://example.org/vocab/course/40355>	Lecture		Introduction to Deep Learning
<http://example.org/vocab/course/40353>	Lecture		Introduction to Natural Language Processing (NLP)
<http://example.org/vocab/course/40355>	Lecture		Introduction to Natural Language Processing (NLP)
<http://example.org/vocab/course/40353>	Worksheet		Introduction to Deep Learning
<http://example.org/vocab/course/40355>	Worksheet		Introduction to Deep Learning
<http://example.org/vocab/course/40353>	Worksheet		Introduction to Natural Language Processing (NLP)
<http://example.org/vocab/course/40355>	Worksheet		Introduction to Natural Language Processing (NLP)
<http://example.org/vocab/course/40353>	Lecture		Deep Learning for Intelligent Systems
<http://example.org/vocab/course/40355>	Lecture		Deep Learning for Intelligent Systems
- Metrics:** 34 results in 0.019 seconds
- Buttons:** Simple view, Ellipse, Filter query results, Page size: 50, etc.

It gives on which lab/worksheet the specific topic was covered in the course.

## Query 4: For a course c, list all covered topics t, printing out their English labels and their

DBpedia/Wikidata URI, together with the course event URI (e.g., 'lab3') and resource URI (e.g., 'slides10') where they appeared. Filter out duplicates.

The screenshot shows a SPARQL query interface with the following details:

- Query:**

```

1 PREFIX ex: <http://example.org/vocab/>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX dbo: <http://dbpedia.org/ontology/>
4 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
5
6 SELECT DISTINCT ?topicName ?lectureUri ?resourceUri
7 WHERE {
8   ?course a ex:Course;
9   ex:number "6721";
10  ex:subject "COMP".
11  ?topicUri a ex:Topic;
12  ex:topicName ?topicName;
13  ex:isTopicOfCourse ?course;
14  ex:materialType ?material;
15  ex:isTopicOfLecture ?lectureUri.
16  ?lectureUri ex:lectureNumber ?lectureNum.
17  BIND(CONCAT(?material, STR(?lectureNum)) AS ?resourceUri)
18 }

```
- Results:**

topicName	lectureUri	resourceUri
00038	<http://example.org/vocab/lecture/7>	Lecture7
08808x11	<http://example.org/vocab/lecture/7>	Lecture7
100s	<http://example.org/vocab/lecture/10>	Lecture10
100s	<http://example.org/vocab/lecture/9>	Lecture9
5V	<http://example.org/vocab/lecture/10>	Lecture10
A2	<http://example.org/vocab/lecture/2>	Lecture2
A2	<http://example.org/vocab/lecture/4>	Lecture4
A2	<http://example.org/vocab/lecture/5>	Lecture5

**Query 5:** For a given topic t (DBpedia or Wikidata URI), list all courses c and their events e where the given topic t appears, along with the count of occurrences, with the results sorted by this count in descending order.

The screenshot shows a SPARQL query interface with the following details:

- Query:**

```

1 PREFIX ex: <http://example.org/vocab/>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 SELECT ?course ?courseDescription (COUNT(?lecture) AS ?numberOfOccurrences)
5 WHERE {
6   ?topic a ex:Topic;
7   ex:topicName "cnn".
8   ?topic ex:isTopicOfCourse ?course.
9   ?topic ex:isTopicOfLecture ?lecture.
10  ?course a ex:Course;
11    ex:subject "COMP";
12    ex:number "6721";
13    ex:description ?courseDescription.
14 }
15 GROUP BY ?course ?courseDescription
16 ORDER BY DESC(?numberOfOccurrences)

```
- Results:**

course	courseDescription	numberOfOccurrences
<http://example.org/vocab/course/4053>	APPLIED ARTIFICIAL INTELLIGENCE	+1+^<http://www.w3.org/2001/XMLSchema#integer>

**Query 6:** For a given topic t, list the precise course URI, course event URI and corresponding

resource URI where the topic is covered (e.g., \NLP" is covered in COMP474 ! Lecture 10! Lab 10 Notes).

SPARQL Endpoint      Content Type (SELECT)      Content Type (GRAPH)

/roboprof/query      JSON      Turtle

```

1+ PREFIX ex: <http://example.org/vocab/>
2+ PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3+
4+ SELECT DISTINCT ?courseDescription ?lectureFormat ?lectureNumber
5+ WHERE {
6+   ?topicUri a ex:Topic;
7+   ex:isTopicOfCourse ?courseUri;
8+   ex:isTopicOfLecture ?lectureUri;
9+   ex:topicName "cnn";
10+  ex:materialType ?material.
11+  ?courseUri ex:description ?courseDescription.
12+  ?lectureUri ex:lectureNumber ?lectureNumber.
13+
14+ # Adjust the output format for lecture/material type.
15+ BIND (IF(CONTAINS(STR(?material), "Lecture"), CONCAT("Lecture ", STR(?lectureNumber)),
16+           IF(CONTAINS(STR(?material), "Worksheet"), CONCAT("Worksheet ", STR(?lectureNumber)), STR(?material))) AS ?lectureFormat)
17+

```

Table Response 1 result in 0.019 seconds

courseDescription	lectureFormat	lectureNumber
APPLIED ARTIFICIAL INTELLIGENCE	Lecture 2	*2^^<http://www.w3.org/2001/XMLSchema#integer>

Simple view Ellipse Filter query results Page size: 50

**Query 7:** Write a SPARQL query to identify any course events or resources within a specific course c that do not have any associated topic entities

SPARQL Endpoint      Content Type (SELECT)      Content Type (GRAPH)

/roboprof/query      JSON      Turtle

```

1+ PREFIX ex: <http://example.org/vocab/>
2+
3+ SELECT ?courseCode
4+ WHERE {
5+   ?courseUri a ex:Course;
6+   ex:number ?number;
7+   ex:subject ?subject.
8+
9+   # Construct course code from subject and number
10+  BIND(CONCAT(?subject, " ", ?number) AS ?courseCode)
11+
12+  # Check if there is no topic linked to this course
13+  FILTER NOT EXISTS {
14+    ?topicUri a ex:Topic;
15+    ex:isTopicOfCourse ?courseUri.
16+  }
17+

```

Table Response 7646 results in 0.541 seconds

courseCode
COMP 6281
COMP 6651
COMP 465
COMP 5481
COMP 352
COMP 354
COMP 5461
COMP 353

Simple view Ellipse Filter query results Page size: 50

## Rasa Query Outputs:

### Query 1:

```
8, ENGL 589, ENGL 598, ENGL 599, ENGL 600, ENGL 601, ENGL 602, ENGL 603, ENGL 604, ENGL 605, ENGL 606, ENGL 607, ENGL 610, ENGL 611, ENGL 612, ENGL 613, ENGL 614, ENGL 615, ENGL 616, ENGL 619, ENGL 619A, ENGL 621, ENGL 622, ENGL 623, ENGL 624, ENGL 625, ENGL 626, ENGL 627, ENGL 628, ENGL 629, ENGL 630, ENGL 631, ENGL 632, ENGL 633, ENGL 635, ENGL 636, ENGL 637, ENGL 640, ENGL 641, ENGL 645, ENGL 646, ENGL 647, ENGL 648, ENGL 650, ENGL 651, ENGL 655, ENGL 656, ENGL 657, ENGL 658, ENGL 659, ENGL 660, ENGL 661, ENGL 662, ENGL 663, ENGL 665, ENGL 666, ENGL 667, ENGL 668, ENGL 669, ENGL 670, ENGL 671, ENGL 672, ENGL 673, ENGL 674, ENGL 678, ENGL 679, ENGL 688, ENGL 690, ENGL 692, ENGL 693, ENGL 694, ACCO 431, ACCO 440, ANTH 202, ANTZ 202, ANTH 203, ANTH 204, ANTH 212, ANTH 221, ANTH 231, ANTH 270, ANTH 272, ENGR 201, ANTH 301, ENGR 202, ANTH 302, ENGR 213, ANTH 303, ENGR 233, ENGR 242, ENGR 243, ENGR 244, ENGR 245, ENGR 251, ENGR 290, ENGR 301, ENGR 311, ENGR 361 are the courses offered by Concordia University
```

It gives the list of all the courses available at Concordia University

### Query 4:

```
Subject: COMP, Number: 6751, Description: NATURAL LANGUAGE ANALYSIS
Subject: COMP, Number: 6761, Description: ADVANCED COMPUTER GRAPHICS
Subject: COMP, Number: 6771, Description: Image Processing
Subject: COMP, Number: 6781, Description: COMP 6781 Statistical Natural Language Proces
Subject: COMP, Number: 6791, Description: INFO. RETRIEVAL & WEB SEARCH
Subject: COMP, Number: 6811, Description: BIOINFORMATICS ALGORITHMS
Subject: COMP, Number: 690, Description: COMP.ANAL.NATURAL LANG.TEST
Subject: COMP, Number: 6961, Description: GRADUATE SEMINAR-COMP.SC.
Subject: COMP, Number: 7241, Description: PARALLEL ALGORITHMS+ARCHIT E
Subject: COMP, Number: 7531, Description: DB SYSTEMS PRINCIPLES
Subject: COMP, Number: 7651, Description: ADV. ANALYSIS OF ALOGRITHMS
Subject: COMP, Number: 7781, Description: ADV IMAGE PROC
Subject: COMP, Number: 790, Description: TOPICS/COMPUTER SCIENCE II
Subject: COMP, Number: 791, Description: TOPICS IN COMP. SCIENCE II
Subject: COMP, Number: 792, Description: MASTER S RESEARCH AND THESIS
Subject: COMP, Number: 890, Description: DOCTORAL RESEARCH AND THESIS
```

It gives all the course available in the University within a particular department in this case "COMP"

### Query 5:

```
Your input -> Materials for CNN in COMP 6721?
No lectures were found for the topic amazon.
```

It returns the materials like lectures, worksheet, slides etc. that are recommended for a particular topic in a specific course. This particular case doesn't give any output because this course doesn't have any materials linked to it.

**Query 5 Rasa side action:**

```
Subject: COMP, Number: None, Lecture Number: lecture 1
Subject: COMP, Number: None, Lecture Number: 1
Running function: action_list_courses_by_subject
Subject: COMP
Running function: action_recommended_materials_for_topic
Topic: amazon, Subject: COMP, Number: 6721
Running function: action_recommended_materials_for_topic
Topic: amazon, Subject: COMP, Number: 6721
□
```

This shows how the Rasa queries the above given query in the backend.

**Query 6:**

```
No lectures were found for the topic amazon.
Your input -> How many credits is COMP 6721?
The course COMP 6721 is worth 4.0 credits.
Your input -> □
```

It gives how many credits each course is worth.

**Query 10:**

```
Your input -> What competencies do students gain after completing COMP 6651?
Upon completion of the course COMP 6651, you will have gained the following competencies: COMP 352, COMP 354, COMP 352, COMP 354, COMP 353, COMP 472, COMP 371, COMP 6321
□
```

It gives what are the particular competencies does the student gains after studying a particular course.

**Query 10 Rasa side action:**

```
Subject: COMP, Number: 6651
Running function: action_competencies_gained
Subject: COMP, Number: 6651
□
```

This shows how the Rasa queries the above given query in the backend.

**Query 11:**

```
Your input -> What grades did Aryan Saxena achieve in COMP 6651?  
The grades for Aryan Saxena in course COMP 6651 are: A  
Your input -> [REDACTED]
```

It gives what are the grade did the student obtain in a particular course.

**Query 11 Rasa side action:**

```
Running function: action_student_grades  
Person: Aryan Saxena, Subject: COMP, Number: 6651  
[REDACTED]
```

This shows how the Rasa queries the above given query in the backend.

**Query 14:**

```
student_transcript  
Your input -> What is COMP 6651 about?  
The course COMP 6651 is described as: ALGORITHM DESIGN TECHNIQUES  
Your input -> [REDACTED]
```

It gives a brief description about the particular course that is queries.

**Query 14 Rasa side action:**

```
[action found for name 'action_print_student_  
Running function: action_course_description  
Subject: COMP, Number: 6651  
[REDACTED]
```

This shows how the Rasa queries the above given query in the backend.

**Query 15:**

```
Your input -> Topics covered in Lecture 5 of COMP 6721?  
The topics covered in the event lecture 5 of course COMP 6721 are:  
Topic: A2, Link: http://dbpedia.org/resource/A2  
Topic: AC, Link: http://dbpedia.org/resource/AC  
Topic: AHSHSA, Link: http://dbpedia.org/resource/AHSHSA  
Topic: AI, Link: http://dbpedia.org/resource/AI  
Topic: ANTHONY, Link: http://dbpedia.org/resource/ANTHONY  
Topic: AP, Link: http://dbpedia.org/resource/AP  
Topic: APN, Link: http://dbpedia.org/resource/APN  
Topic: ARACHNOCENTRIC, Link: http://dbpedia.org/resource/ARACHNOCENTRIC  
Topic: Alan, Link: http://dbpedia.org/resource/Alan  
Topic: Alison, Link: http://dbpedia.org/resource/Alison  
Topic: Amazon, Link: http://dbpedia.org/resource/Amazon  
Topic: Anthony, Link: http://dbpedia.org/resource/Anthony  
Topic: B15, Link: http://dbpedia.org/resource/B15  
Topic: BRUTUS, Link: http://dbpedia.org/resource/BRUTUS
```

It lists all the topics that are covered in a particular lecture that is queried.

#### Query 16:

```
Your input -> which course events cover cnn?
The following events cover the topic amazon:
Course: http://example.org/vocab/course/40353, Material Type: Worksheet, Name: Deep Learning for Intelligent Systems
Course: http://example.org/vocab/course/40355, Material Type: Worksheet, Name: Deep Learning for Intelligent Systems
Course: http://example.org/vocab/course/40353, Material Type: Worksheet, Name: Deep Learning for NLP
Course: http://example.org/vocab/course/40353, Material Type: Worksheet, Name: Introduction to Deep Learning
Course: http://example.org/vocab/course/40355, Material Type: Worksheet, Name: Introduction to Deep Learning
Course: http://example.org/vocab/course/40353, Material Type: Worksheet, Name: Introduction to Natural Language Processing (NLP)
Course: http://example.org/vocab/course/40355, Material Type: Worksheet, Name: Introduction to Natural Language Processing (NLP)
Course: http://example.org/vocab/course/40353, Material Type: Worksheet, Name: Knowledge Graphs & Intelligent Agents (I)
Course: http://example.org/vocab/course/40355, Material Type: Worksheet, Name: Knowledge Graphs & Intelligent Agents (I)
Course: http://example.org/vocab/course/40353, Material Type: Worksheet, Name: Knowledge Graphs & Intelligent Agents (II)
Course: http://example.org/vocab/course/40355, Material Type: Worksheet, Name: Knowledge Graphs & Intelligent Agents (II)
Course: http://example.org/vocab/course/40353, Material Type: Worksheet, Name: NLP Applications & Text Mining
Course: http://example.org/vocab/course/40355, Material Type: Worksheet, Name: NLP Applications & Text Mining
Course: http://example.org/vocab/course/40353, Material Type: Worksheet, Name: NLP: Applications, Vector Space Models
Course: http://example.org/vocab/course/40355, Material Type: Worksheet, Name: NLP: Applications, Vector Space Models
```

It gives which course event covers a particular topic in a course

#### Query 17:

```
Topic: NO, Lecture: http://example.org/vocab/lecture/9, R
Your input -> list all topics covered in COMP 6651.
No topics were found for the course COMP 6651.
Your input -> list all topics covered in 6721.
```

```
Topic: pickupstackHA, Lecture: http://example.org/vocab/lecture/2, Resource: Lecture2
Topic: pltfigurefigsize10, Lecture: http://example.org/vocab/lecture/3, Resource: Worksheet3
Topic: positionsfoof, Lecture: http://example.org/vocab/lecture/5, Resource: Worksheet5
Topic: positionssec, Lecture: http://example.org/vocab/lecture/3, Resource: Lecture3
Topic: printdfsent0dotdfsent1, Lecture: http://example.org/vocab/lecture/4, Resource: Worksheet3
Topic: printkmeanslabels, Lecture: http://example.org/vocab/lecture/3, Resource: Worksheet3
Topic: printrigrams3, Lecture: http://example.org/vocab/lecture/2, Resource: Worksheet2
Topic: pronoun, Lecture: http://example.org/vocab/lecture/1, Resource: Lecture1
Topic: pronoun, Lecture: http://example.org/vocab/lecture/10, Resource: Lecture10
Topic: rect0, Lecture: http://example.org/vocab/lecture/3, Resource: Worksheet3
Topic: sec, Lecture: http://example.org/vocab/lecture/2, Resource: Lecture2
Topic: sent2, Lecture: http://example.org/vocab/lecture/11, Resource: Lecture11
Topic: sent2, Lecture: http://example.org/vocab/lecture/4, Resource: Lecture4
```

It gives for a particular course, list of topics that is covered. This particular query doesn't give any output because these particular data doesn't have any data available for it.

**Query 20:**

Your input -> **For which courses are there no materials available?**

The following courses have no material associated with them: COMP 6281, COMP 6651, COMP 465, COMP 5481, COMP 352, COMP 354, COMP 5461, COMP 353, COMP 472, COMP 5531, COMP 6481, COMP 371, COMP 632 , COMP 346, ANTH 307, ENGR 371, ENGR 391, ENGR 392, ENGR 411, ENGR 412, ENGR 472, ANTH 311, ANTH 12, ANTH 315, ANTH 322, SOCI 322, ENGR 6121, ENGR 6141, ENGR 6161, ENGR 6191, ENGR 6201, ENGR 622 , ENGR 6231, ANTH 324, ENGR 6241, ENGR 6251, ENGR 6261, ENGR 6281, ENGR 6291, ENGR 6301, ANTH 325 ENGR 6311, ANTH 326, ENGR 6411, ENGR 6421, ANTH 332, ENGR 6471, ANTH 345, ENGR 6531, ENGR 6601, NTH 361, ENGR 6811, ANTH 363, SOCI 363, ENGR 691, ENGR 6971, ENGR 6981, ENGR 6991, ENGR 7011, ENGR 7121, ENGR 7131, ENGR 7181, ENGR 7331, ENGR 7401, ENGR 7461, ENGR 7961, ANTH 377, ANTH 379, SOCI

It gives all the list of courses that doesn't have any materials linked to it. This query helps in verifying that all relevant educational materials have been adequately linked to topics in the knowledge base, ensuring completeness