Concordia University

Engineering and Computer Science

COMP 6741

Intelligent Systems

Project-1 Report

Roboprof

Submitted To: Prof. Dr. René Witte

Date: 22 March 2024

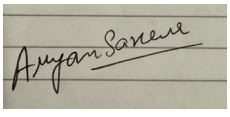
Team: AK_G_04

Aryan Saxena (**40233170**)

Benjamin Douglas (**40264251**)

GitHub Link

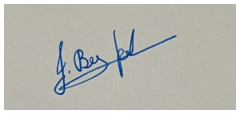
We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality



Aryan Saxena

40233170

22nd March 2024



Benjamin Douglas

40264251

22nd March 2024

Table of Contents

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)

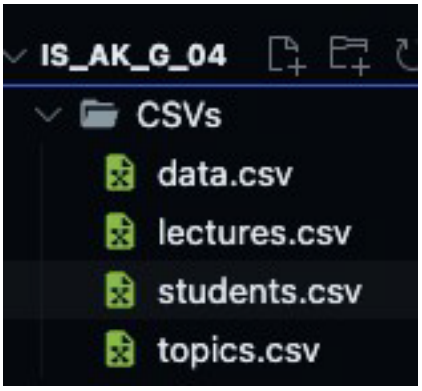


Fig 1: Structure

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,
    ex:subject "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

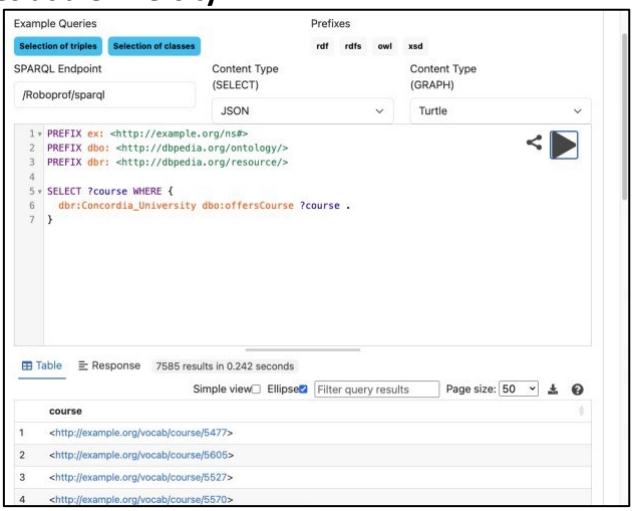
- *Screenshot*:

```
@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.

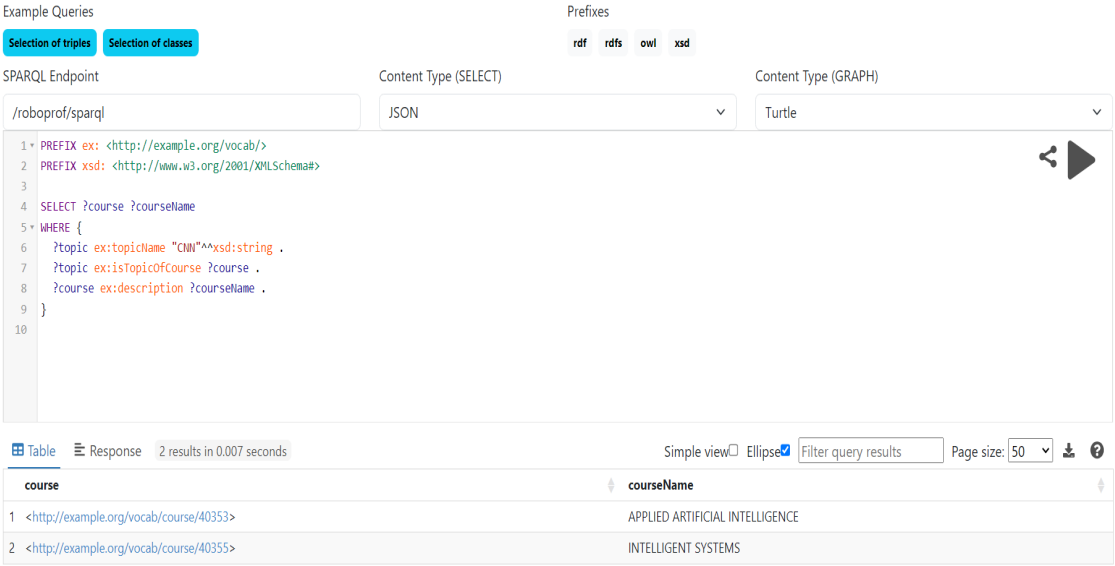
5. Queries

Query 1: Finding Courses at a University



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



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

Prefixes

rdf | rdfs | owl | xsd

SPARQL Endpoint: /roboprof/sparql

Content Type (SELECT): JSON

Content Type (GRAPH): Turtle

```
PREFIX ex: <http://example.org/vocab/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?topicName
WHERE {
  ?topic ex:isTopicOfCourse <http://example.org/vocab/course/40355> ;
         ex:isTopicOfLecture <http://example.org/vocab/lecture/7> ;
         ex:topicName ?topicName .
}
```

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

/Roboprof/sparql | (SELECT) JSON | (GRAPH) Turtle

```
PREFIX ex: <http://example.org/vocab/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?course ?subject ?number
WHERE {
  ?university dbo:offersCourse ?course .
  ?course a ex:Course ;
          ex:subject ?subject ;
          ex:number ?number .
  FILTER(?subject = "COMP" || ?subject = "SOEN")
}
```

Table | Response | 164 results in 0.039 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	course	subject	number
1	<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

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

SELECT DISTINCT ?lectureName
WHERE {
  # Select the specific course by subject and number
  ?course a ex:Course;
          ex:subject "COMP"^^xsd:string;
          ex:number "6741"^^xsd:string.

  # Find topics that are part of this course
  ?topic a ex:Topic;
         ex:isTopicOfCourse ?course;
         ex:topicName "Amazon"^^xsd:string.  # Adjust topic name as necessary

  # Find lectures that include this topic
  ?topic ex:isTopicOfLecture ?lecture.

  # Get the names of these lectures
  ?lecture ex:lectureName ?lectureName.
}
```

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)

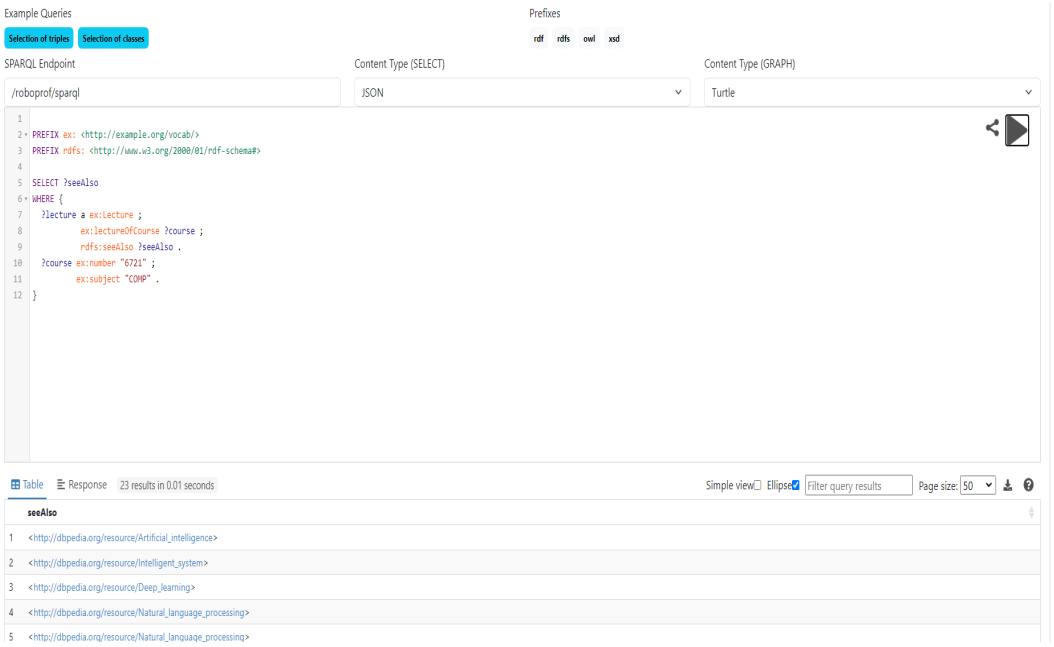
It lists all the required materials for a given topic for the given course

a particular topic within a course, guiding resourceful learning.

Query 6: Credit Value of a Course

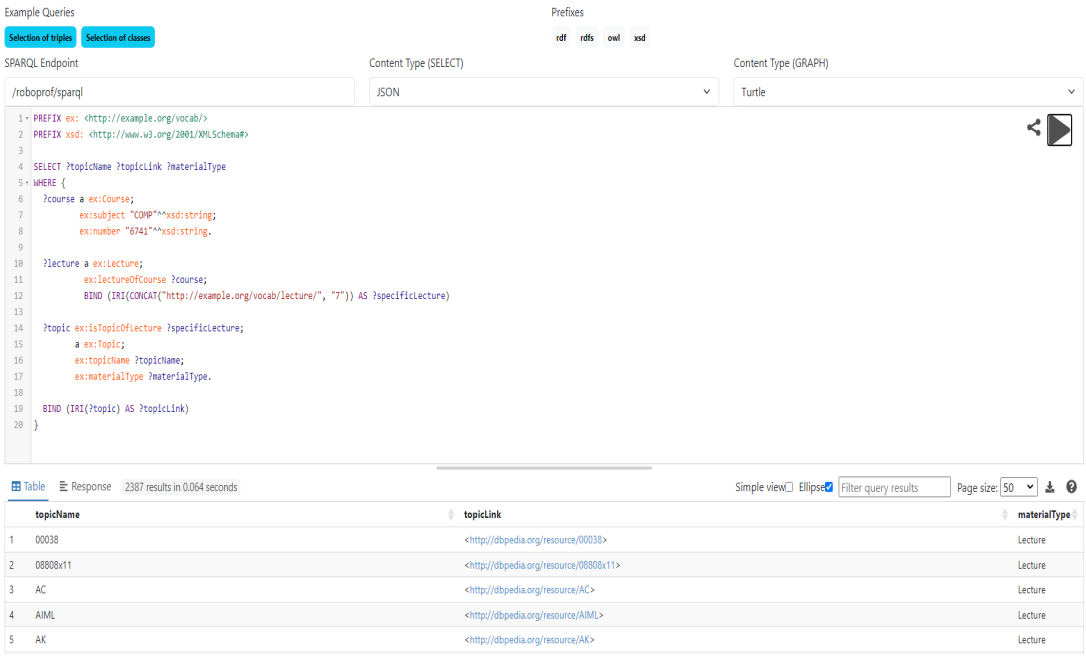
It determines the credit value of a specific course, informing on its academic weight.

Query 7: Additional Resources for a Course



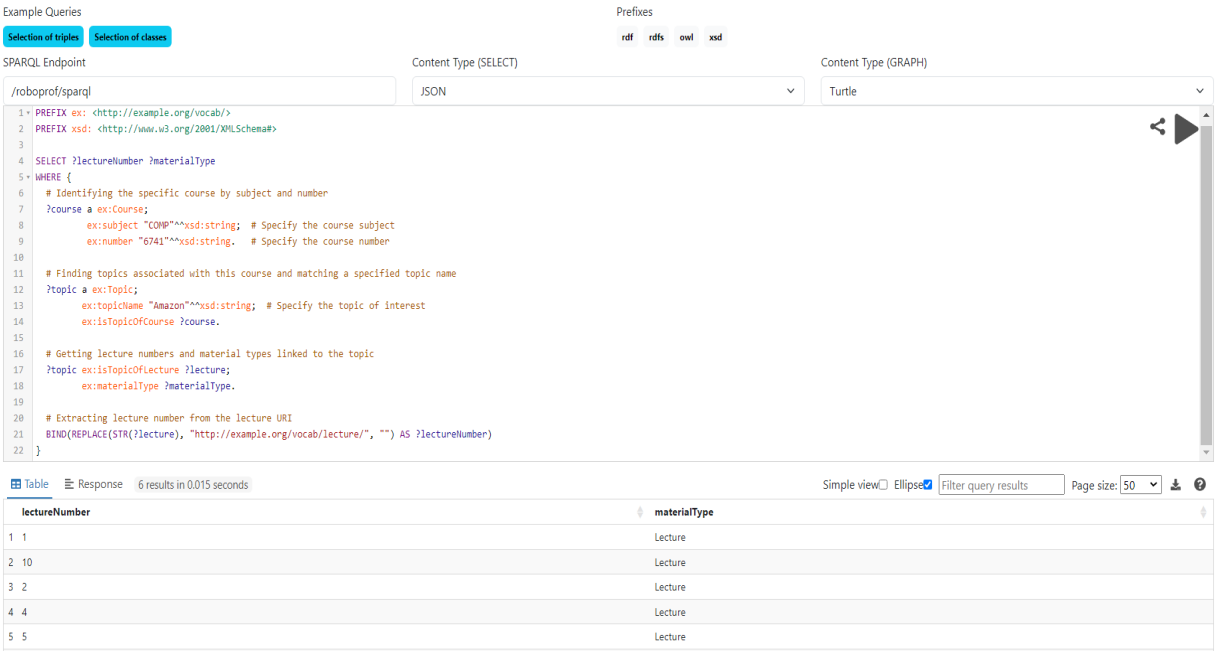
This query finds extra resources, like web links, for a course, providing avenues for extended learning.

Query 8: Content Available for a Lecture



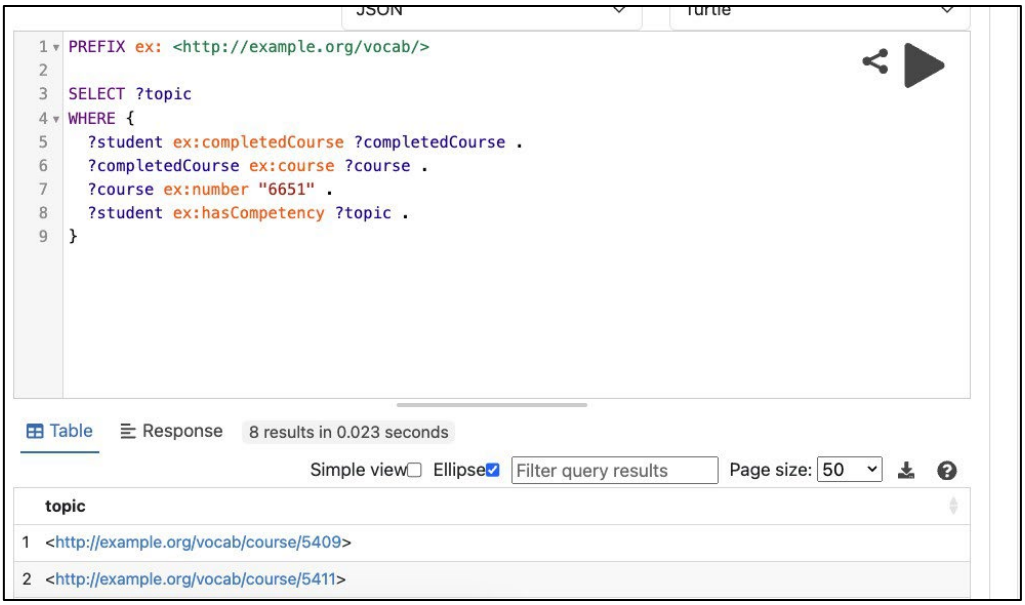
It identifies all the content types available for a specific lecture, giving insights into the lecture's materials.

Query 9: Recommended Reading for a Topic



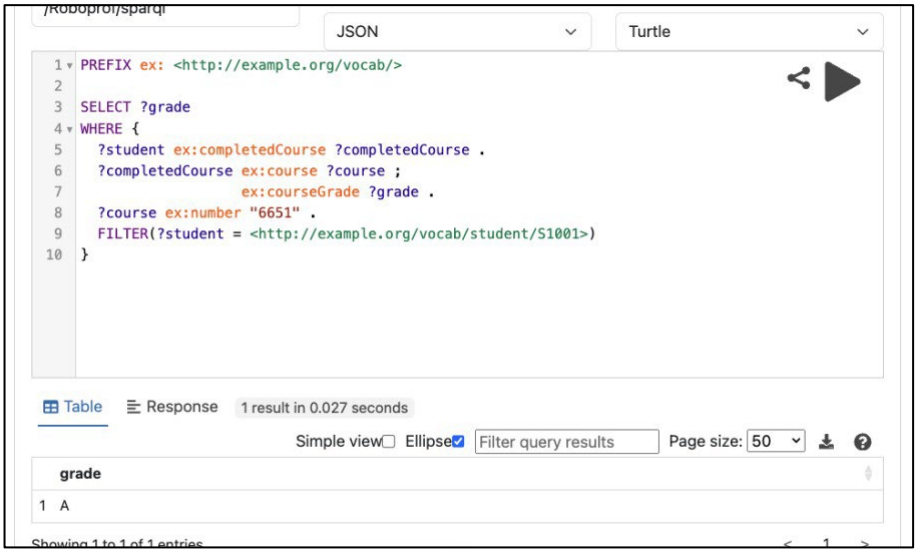
No result because no matches. This query lists reading materials recommended for a topic within a course, aiding in-depth study.

Query 10: Competencies Gained from a Course



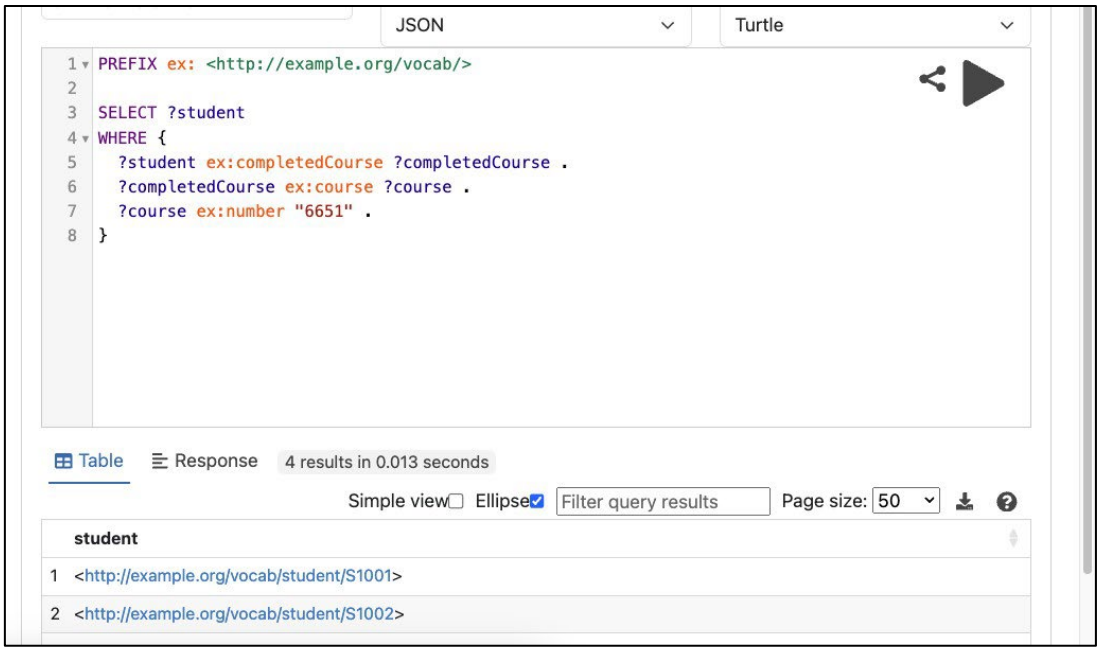
It explores the topics a student becomes proficient in after completing a course, reflecting on the learning outcome.

Query 11: Student's Grades in a Course



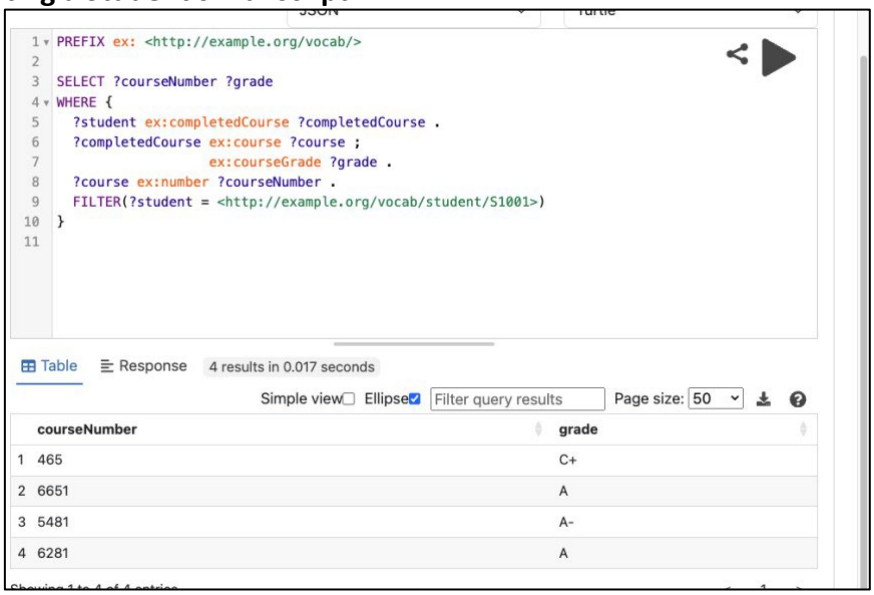
This query fetches the grades a student achieved in a course, reflecting their academic performance.

Query 12: Students Who Completed a Course



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

Query 13: Printing a Student's Transcript



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

```yaml
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

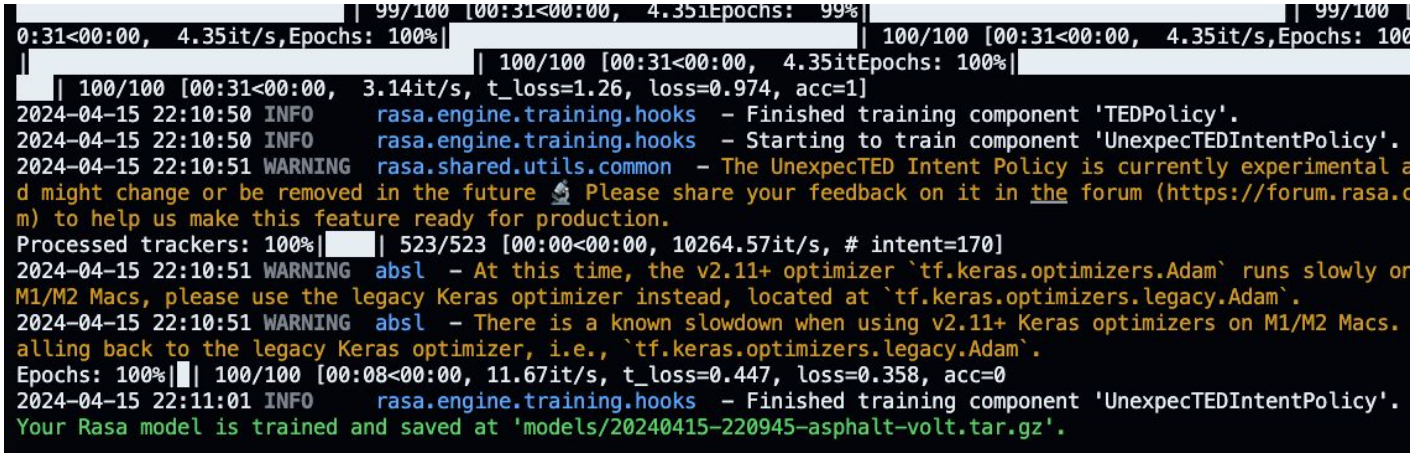


Fig: Rasa training status

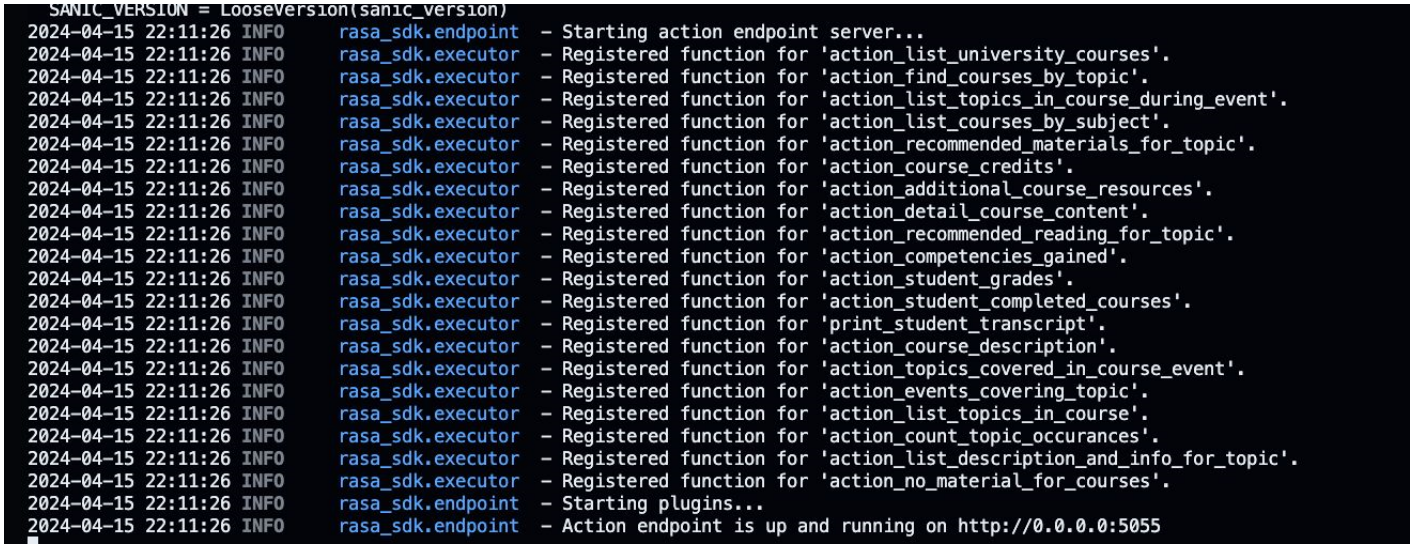
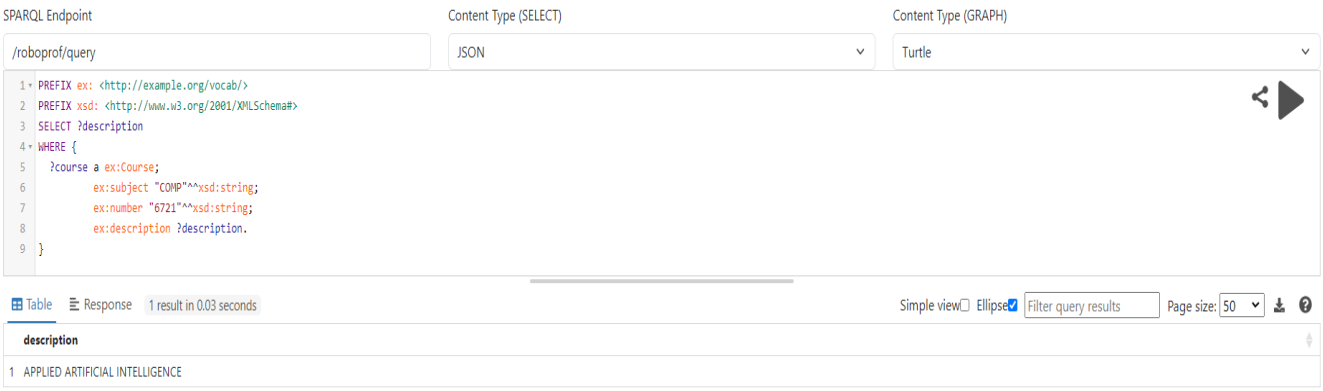


Fig: Rasa Run Actions

C. Query Outputs

QUERY 1: What is the <course> about?



It gives the description for the given queried course.

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

/roboprof/query | JSON | Turtle

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

SELECT ?topicName ?topicLink
WHERE {
  ?course a ex:Course;
          ex:subject "COMP"^^xsd:string;
          ex:number "6721"^^xsd:string.

  ?lecture a ex:Lecture;
           ex:lectureNumber 2 ;
           ex:lectureOfCourse ?course.

  ?topic a ex:Topic;
         ex:topicName ?topicName;
         ex:isTopicOfLecture ?lecture.
  BIND (STR(?topic) AS ?topicLink)
}
```

Table | Response | 193 results in 0.025 seconds | Simple view | Ellipse | Filter query results | Page size: 50

	topicName	topicLink
1	A12	http://dbpedia.org/resource/A12
2	A2	http://dbpedia.org/resource/A2
3	AI	http://dbpedia.org/resource/AI
4	ANJA	http://dbpedia.org/resource/ANJA
5	API	http://dbpedia.org/resource/API
6	Alice	http://dbpedia.org/resource/Alice
7	Amazon	http://dbpedia.org/resource/Amazon
8	Amitav	http://dbpedia.org/resource/Amitav
9	Annotated	http://dbpedia.org/resource/Annotated
10	Auteur	http://dbpedia.org/resource/Auteur

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

Query 3: Which course events cover <Topic>?

/roboprof/query | JSON | Turtle

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

SELECT ?course ?event ?materialType ?eventName
WHERE {
  ?topic ex:topicName "CNN"^^xsd:string;
         ex:isTopicOfLecture ?lecture;
         ex:materialType ?materialType.

  # Retrieve lecture details
  ?lecture a ex:Lecture;
           ex:lectureName ?eventName;
           ex:lectureOfCourse ?course.

  # Retrieve course details
  ?course a ex:Course;
          ex:description ?courseDescription.
}
```

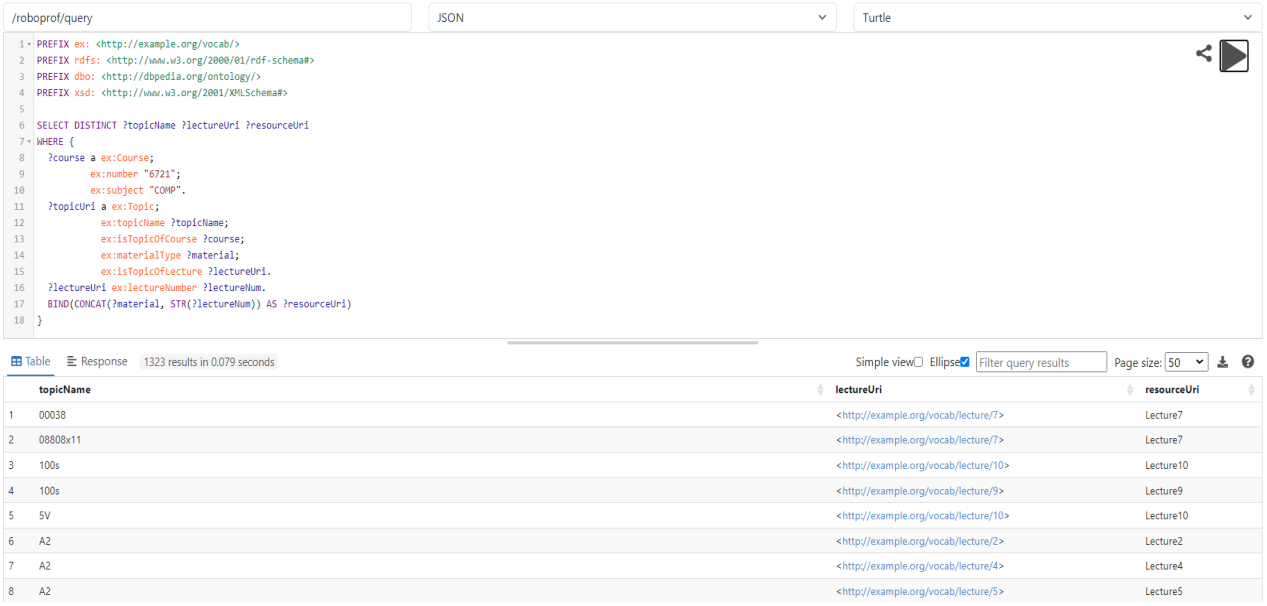
Table | Response | 34 results in 0.019 seconds | Simple view | Ellipse | Filter query results | Page size: 50

	course	event	materialType	eventName
1	<http://example.org/vocab/course/40353>		Lecture	Introduction to Deep Learning
2	<http://example.org/vocab/course/40355>		Lecture	Introduction to Deep Learning
3	<http://example.org/vocab/course/40353>		Lecture	Introduction to Natural Language Processing (NLP)
4	<http://example.org/vocab/course/40355>		Lecture	Introduction to Natural Language Processing (NLP)
5	<http://example.org/vocab/course/40353>		Worksheet	Introduction to Deep Learning
6	<http://example.org/vocab/course/40355>		Worksheet	Introduction to Deep Learning
7	<http://example.org/vocab/course/40353>		Worksheet	Introduction to Natural Language Processing (NLP)
8	<http://example.org/vocab/course/40355>		Worksheet	Introduction to Natural Language Processing (NLP)
9	<http://example.org/vocab/course/40353>		Lecture	Deep Learning for Intelligent Systems
10	<http://example.org/vocab/course/40355>		Lecture	Deep Learning for Intelligent Systems

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.



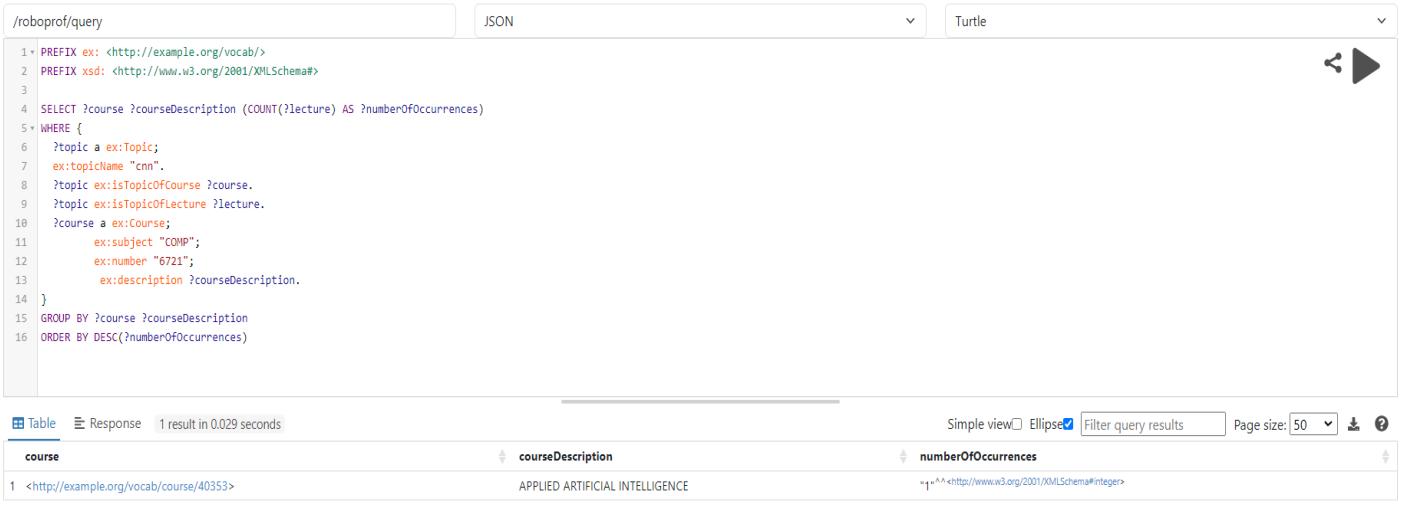
/roboprof/query | JSON | Turtle

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

SELECT DISTINCT ?topicName ?lectureUri ?resourceUri
WHERE {
  ?course a ex:Course;
          ex:number "6721";
          ex:subject "COMP".
  ?topicUri a ex:Topic;
            ex:topicName ?topicName;
            ex:isTopicOfCourse ?course;
            ex:materialType ?material;
            ex:isTopicOfLecture ?lectureUri.
  ?lectureUri ex:lectureNumber ?lectureNum.
  BIND(CONCAT(?material, STR(?lectureNum)) AS ?resourceUri)
}
```

Table | Response | 1323 results in 0.079 seconds | Simple view | Ellipse | Filter query results | Page size: 50

	topicName	lectureUri	resourceUri
1	00038	<http://example.org/vocab/lecture/7>	Lecture7
2	08808x11	<http://example.org/vocab/lecture/7>	Lecture7
3	100s	<http://example.org/vocab/lecture/10>	Lecture10
4	100s	<http://example.org/vocab/lecture/9>	Lecture9
5	5V	<http://example.org/vocab/lecture/10>	Lecture10
6	A2	<http://example.org/vocab/lecture/2>	Lecture2
7	A2	<http://example.org/vocab/lecture/4>	Lecture4
8	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.



/roboprof/query | JSON | Turtle

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

SELECT ?course ?courseDescription (COUNT(?lecture) AS ?numberOfOccurrences)
WHERE {
  ?topic a ex:Topic;
  ex:topicName "cnn".
  ?topic ex:isTopicOfCourse ?course.
  ?topic ex:isTopicOfLecture ?lecture.
  ?course a ex:Course;
        ex:subject "COMP";
        ex:number "6721";
         ex:description ?courseDescription.
}
GROUP BY ?course ?courseDescription
ORDER BY DESC(?numberOfOccurrences)
```

Table | Response | 1 result in 0.029 seconds | Simple view | Ellipse | Filter query results | Page size: 50

	course	courseDescription	numberOfOccurrences
1	<http://example.org/vocab/course/40353>	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: /roboprof/query | Content Type (SELECT): JSON | Content Type (GRAPH): Turtle

```
PREFIX ex: <http://example.org/vocab/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?courseDescription ?lectureFormat ?lectureNumber
WHERE {
  ?topicUri a ex:Topic;
            ex:isTopicOfCourse ?courseUri;
            ex:isTopicOfLecture ?lectureUri;
            ex:topicName "cnn";
            ex:materialType ?material.
  ?courseUri ex:description ?courseDescription.
  ?lectureUri ex:lectureNumber ?lectureNumber.

  # Adjust the output format for lecture/material type.
  BIND (IF(CONTAINS(STR(?material), "Lecture"), CONCAT("Lecture ", STR(?lectureNumber)),
          IF(CONTAINS(STR(?material), "Worksheet"), CONCAT("Worksheet ", STR(?lectureNumber)), STR(?material))) AS ?lectureFormat)
}
```

Table | Response | 1 result in 0.019 seconds | Simple view | Ellipse | Filter query results | Page size: 50

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

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: /roboprof/query | Content Type (SELECT): JSON | Content Type (GRAPH): Turtle

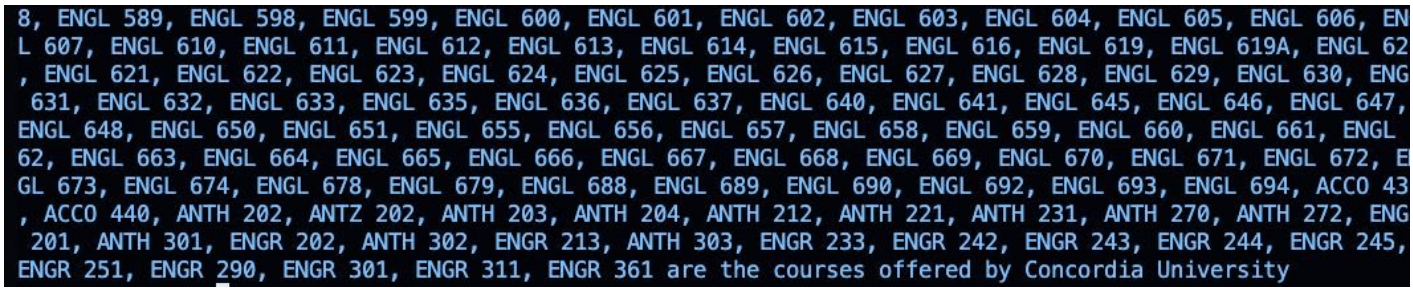
```
PREFIX ex: <http://example.org/vocab/>

SELECT ?courseCode
WHERE {
  ?courseUri a ex:Course;
             ex:number ?number;
             ex:subject ?subject.

  # Construct course code from subject and number
  BIND(CONCAT(?subject, " ", ?number) AS ?courseCode)

  # Check if there is no topic linked to this course
  FILTER NOT EXISTS {
    ?topicUri a ex:Topic;
              ex:isTopicOfCourse ?courseUri.
  }
}
```

Table | Response | 7646 results in 0.541 seconds | Simple view | Ellipse | Filter query results | Page size: 50

	courseCode
1	COMP 6281
2	COMP 6651
3	COMP 465
4	COMP 5481
5	COMP 352
6	COMP 354
7	COMP 5461
8	COMP 353

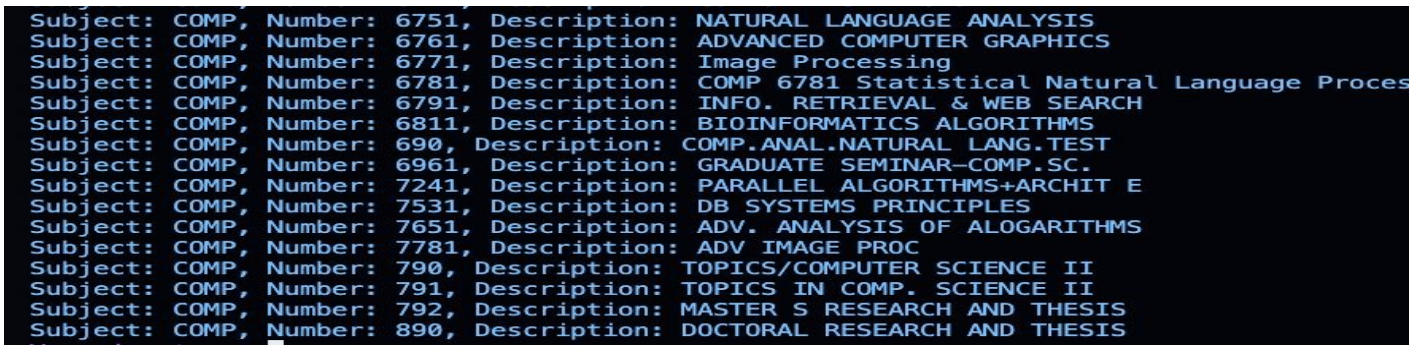
Rasa Query Outputs:

Query 1:



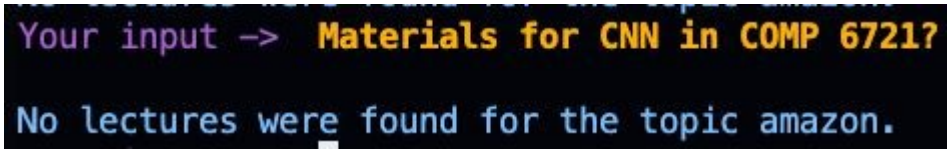
It gives the list of all the courses available at Concordia Univeristy

Query 4:



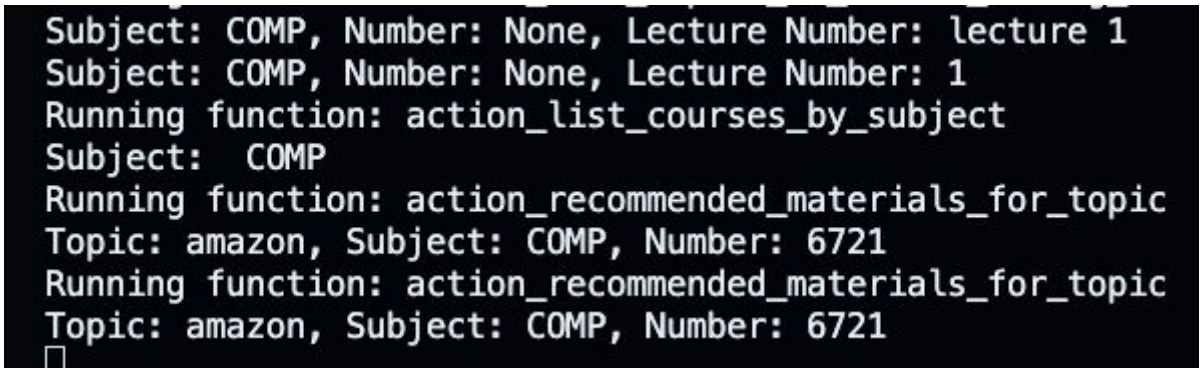
It gives all the course available in the University within a particular department in this case “COMP”

Query 5:



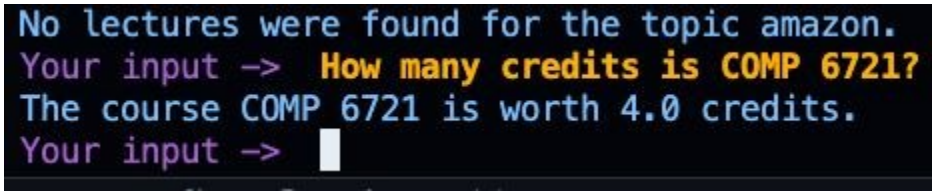
It returns the materials like lectures, worksheet, slides etc. that are recommended for a particular topic in a specific course. This particular case doesn’t gives any output because this course doesn’t have any martials linked to it.

Query 5 Rasa side action:



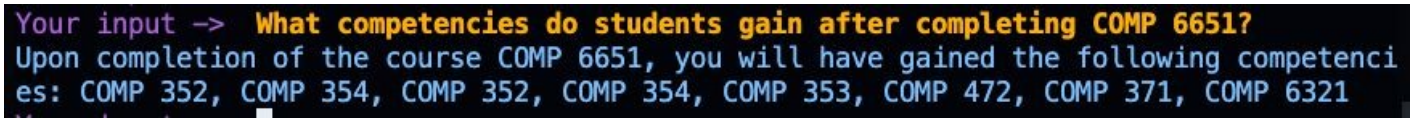
This shows show how the Rasa queries the above given query in the backend.

Query 6:



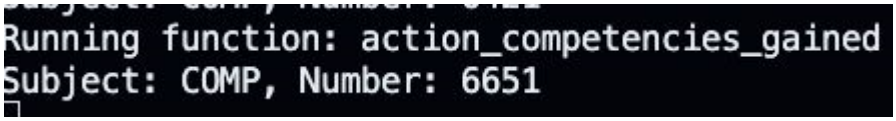
It gives how many credits each course is worth.

Query 10:



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

Query 10 Rasa side action:

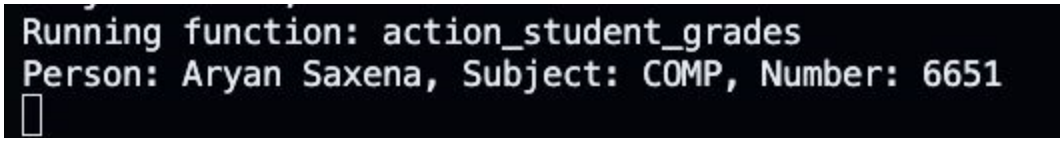


This shows show how the Rasa queries the above given query in the backend.

Query 11:

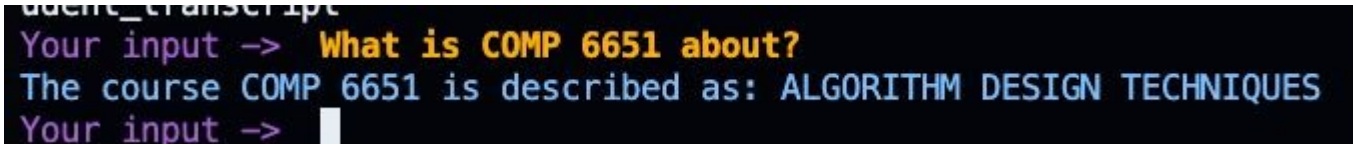
It gives what are the grade did the student obtain in a particular course.

Query 11 Rasa side action:



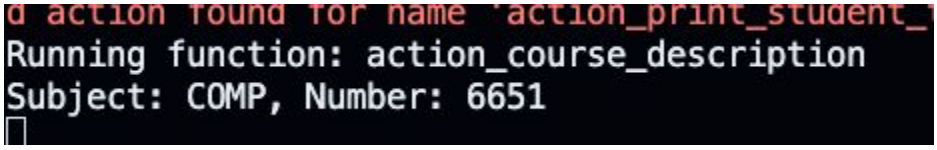
This shows show how the Rasa queries the above given query in the backend.

Query 14:



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

Query 14 Rasa side action:

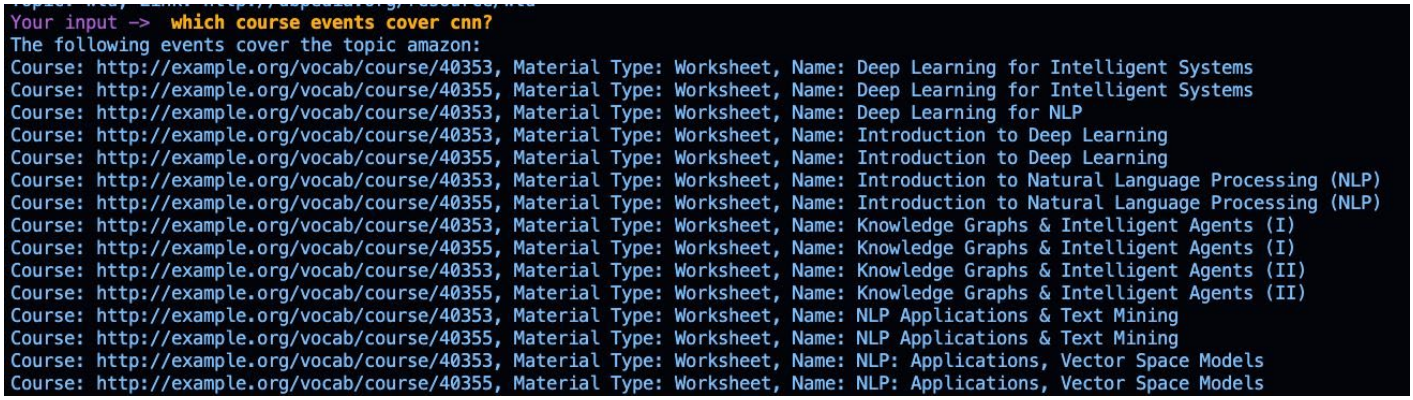


This shows show how the Rasa queries the above given query in the backend.

Query 15:

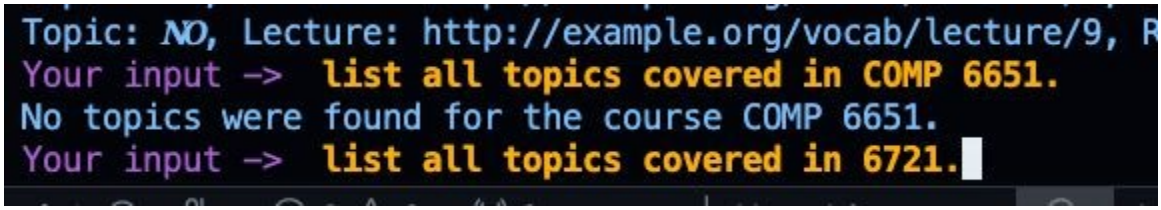
It list all the topic that are covered in a particular lecture that is queried.

Query 16:



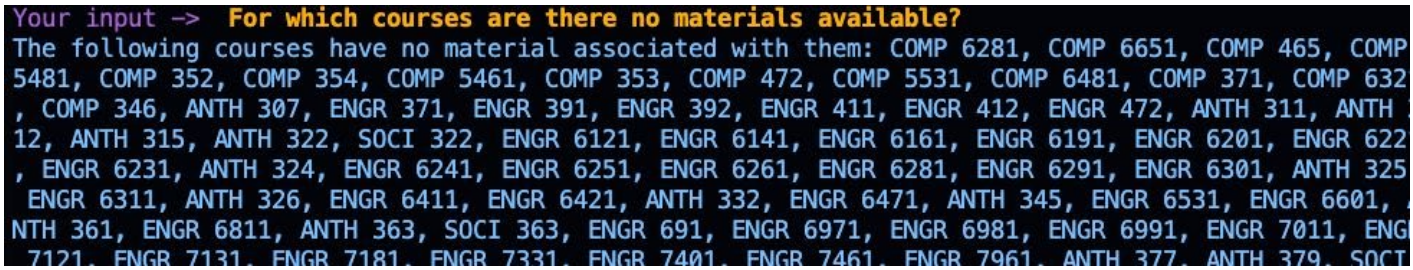
It gives which course event covers a particular topic in a course

Query 17:



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:



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