COMP 6741 Intelligent Systems (Winter 2024)

Project Assignment #2

Group Details:

Moodle Team Name: AZ_G_06

Team Members:

Shrawan Sai Malyala: 40236492, Masters of Applied Computer Science

Sraddha Bhattacharjee: 40221370, Masters of Applied Computer Science

Project Repository Link:
https://github.com/ShrawanSai/COMP6741_Roboprof

Statement of Originality:

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

Signatures:

Shrawan Sai Malyala, 40236492, 15/04/2024

Sraddha Bhattacharjee, 40221370, 15/04/2024

PART 1

Vocabulary

We used RDF (Resource Description Framework) to construct classes, attributes, and their relationships to describe the knowledge base's schema. Below is a summary of the modeling process for the schema:

- University Definitions
  - ➢ We defined the class 'acad: University' to represent universities.
  - ➢ Properties such as 'acad:universityName', 'acad:universityDBpediaLink', and 'acad:universityWikidataLink' was created to describe university attributes like name and links to DBpedia and Wikidata entries.

- Course Definitions
  - ➢ The class 'acad: Course' represents courses offered by universities.
  - ➢ Properties such as 'acad:courseName', 'acad: courseSubject', 'acad:courseNumber', 'acad:courseCredits', 'acad: courseDescription', 'acad:courseWebpage', and 'acad:courseOutline' provide details about course attributes like name, subject, number, credits, description, webpage link, and course outline.

- Lecture Definitions
  - ➢ The class 'acad: Lecture' represents the lecture components of courses.
  - ➢ Properties like 'acad:lectureNumber', 'acad:lectureName', and 'acad:lectureLink' describe attributes of lectures such as number, name, and additional links.

- Student Definitions:
  - ➢ The class 'acad:Student' represents students enrolled at universities.
  - ➢ Properties like 'acad:studentName', 'acad:studentID', 'acad:studentEmail', and 'acad:studiesAt' describe student attributes such as name, ID, email, and the university they are enrolled in.

- Completed Courses Definitions:
  - ➢ The class 'acad:CompletedCourse' represents the association between completed courses and students.
  - ➢ Properties like 'acad:hasCourse', 'acad:courseGrade', and

‘acad:courseSemester’ describe attributes of completed courses such as the course itself, grade obtained, and the semester completed.

- Subclasses:
  - ➢ Subclasses like ‘acad:Slides’, ‘acad:Worksheet’, ‘acad:Reading’, and ‘acad:otherLectureContent’ were defined to represent different types of lecture content.

- Vocabulary Reuse:
  - ➢ Existing vocabularies such as ‘rdf’, ‘rdfs’, ‘xsd’, ‘vivo’, ‘acad’, ‘acaddata’, and ‘foaf’ were reused for foundational concepts, data types, and describing relationships between entities.
  - ➢ We reused these existing vocabularies to ensure compatibility, interoperability, and adherence to established standards within the Semantic Web community.

Vocabulary Extensions

- ➢ We extended the vocabulary with domain-specific classes and properties (‘acad’ and ‘vivo’ namespaces) to capture information relevant to university courses, lectures, topics, students, and completed courses.
- ➢ This extension allows for more detailed modeling of educational data and effectively facilitates querying and reasoning over the knowledge base.
- ➢ RDF is a scalable and versatile method for expressing data and its relationships, which makes it appropriate for modeling intricate educational domains.
- ➢ Data integration and interoperability are encouraged, and compatibility with current Semantic Web technologies is ensured by reusing established vocabularies.
- ➢ To meet the needs of the university project domain, vocabulary extensions have been developed to enable the specialized representation of educational entities and their attributes.

Knowledge Base Construction

We have implemented Roboprof using the following datasets:

(a) Dataset:

CATALOG.csv :

This dataset is collection of academic course offerings from various departments within Concordia University. Each entry represents a specific course, including details such as the course code, department, program, level (e.g., undergraduate or graduate), course

title, course description, and any additional information
such as prerequisites, course format (e.g., laboratory, fieldwork), and integration type (e.g., course-integrated, work-integrated).

Source URL: https://opendata.concordia.ca/datasets/

studentdata.csv:

This dataset comprises of the ID, Name, Email, Completed Courses, Completed Grade and Semester attempted by us, as well as some sample data. We have constructed this dataset by ourselves.

(b) Process (Detailed in ABKS.ipynb):

- Defined Namespaces and Binded Prefixes:

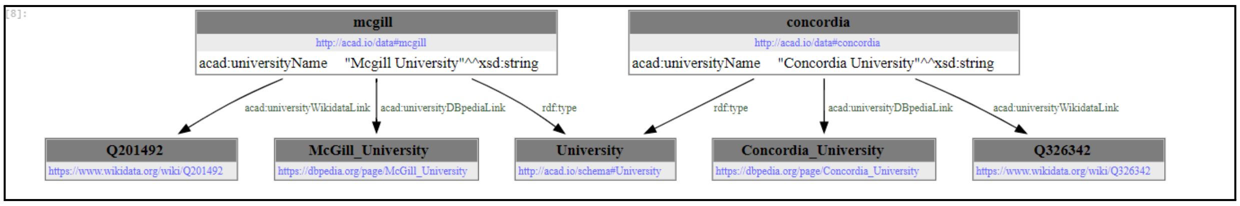
- Created namespace objects for vocabulary terms and RDF namespaces.
- Binded prefixes to these namespaces to simplify the representation of URIs in RDF triples.

- Creating the Vocabulary:

- Defined classes and properties for universities, courses, lectures, topics, students, etc.
- Utilized existing vocabularies and ontologies to capture the information.
- Implemented a binding between the dataset and namespaces using RDFlib in Python.

- Adding University Information:

- Added details about universities Concordia and McGill, including names and locations.
- Utilized the Graphviz Python library to visualize the graph representing university information.


[8]:
mcgill
http://acad.io/data#mcgill
acad:universityName "Mcgill University"^^xsd:string
concordia
http://acad.io/data#concordia
acad:universityName "Concordia University"^^xsd:string
acad:universityWikidataLink
acad:universityDBpediaLink
rdf:type
rdf:type
acad:universityDBpediaLink
acad:universityWikidataLink
Q201492
https://www.wikidata.org/wiki/Q201492
McGill_University
https://dbpedia.org/page/McGill_University
University
http://acad.io/schema#University
Concordia_University
https://dbpedia.org/page/Concordia_University
Q326342
https://www.wikidata.org/wiki/Q326342

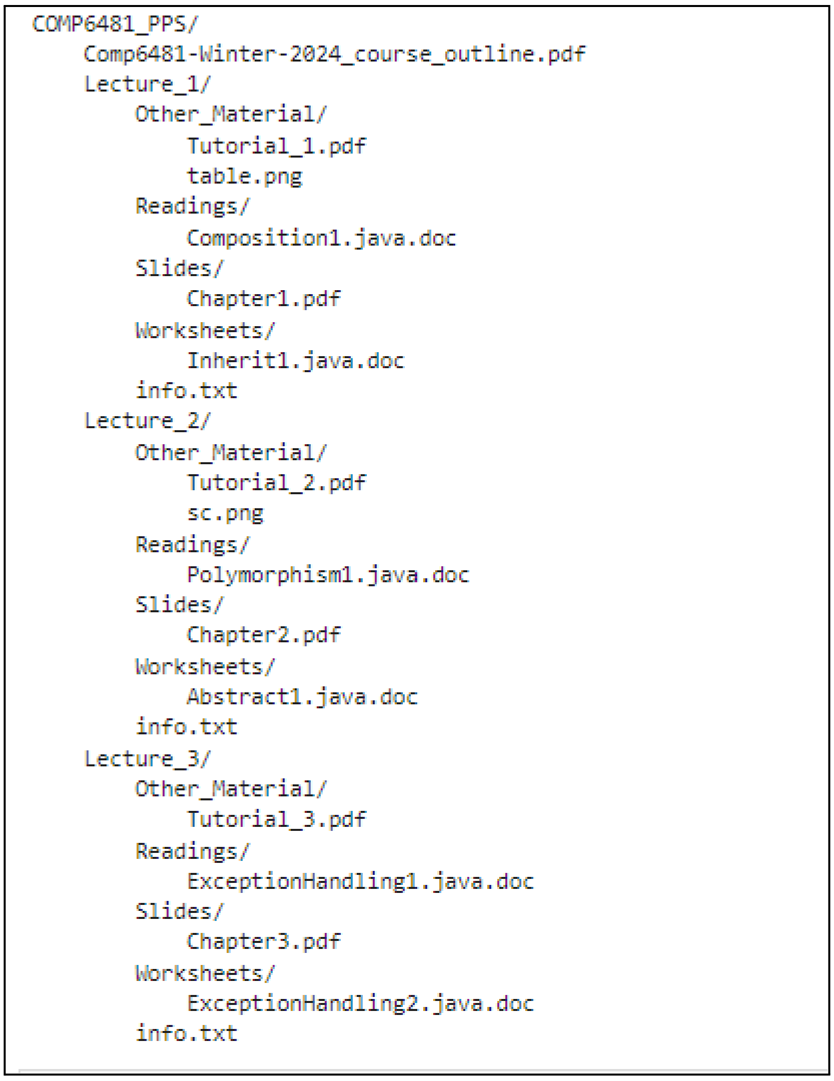
The graph after adding universities

- Adding Courses:

- Added courses COMP 6741 (Intelligent Systems) and COMP 6481 (Programming and Problem Solving).
- Established folder structures for the courses as specified in the project requirements.
- Provided course outlines and saved the file after each step to track progress.
- Leveraged Concordia's open-source dataset Catalog.csv to extract relevant fields.
- Developed a Python script to extract fields from Catalog.csv and integrate them into the knowledge graph.

- Adding Lectures for Courses:

- For the 2 manually added courses, a folder structure as shown below was created. Then a script was developed that goes through the folder structure and extracts path to each file.



```
COMP6481_PPS/
    Comp6481-Winter-2024_course_outline.pdf
    Lecture_1/
        Other_Material/
            Tutorial_1.pdf
            table.png
        Readings/
            Composition1.java.doc
        Slides/
            Chapter1.pdf
        Worksheets/
            Inherit1.java.doc
        info.txt
    Lecture_2/
        Other_Material/
            Tutorial_2.pdf
            sc.png
        Readings/
            Polymorphism1.java.doc
        Slides/
            Chapter2.pdf
        Worksheets/
            Abstract1.java.doc
        info.txt
    Lecture_3/
        Other_Material/
            Tutorial_3.pdf
        Readings/
            ExceptionHandling1.java.doc
        Slides/
            Chapter3.pdf
        Worksheets/
            ExceptionHandling2.java.doc
        info.txt
```

- Then script was made to add lecture numbers, names, and content (slides, worksheets, readings, other material), each with a designated subclass.

- Adding Topics Under Each Lecture:

- Added topics covered in each lecture to enhance the granularity of information within the knowledge graph.
- Extracted data from studentdata.csv, a custom dataset, incorporating ID, name, email, completed courses, and attempts using Python scripts developed for this purpose.

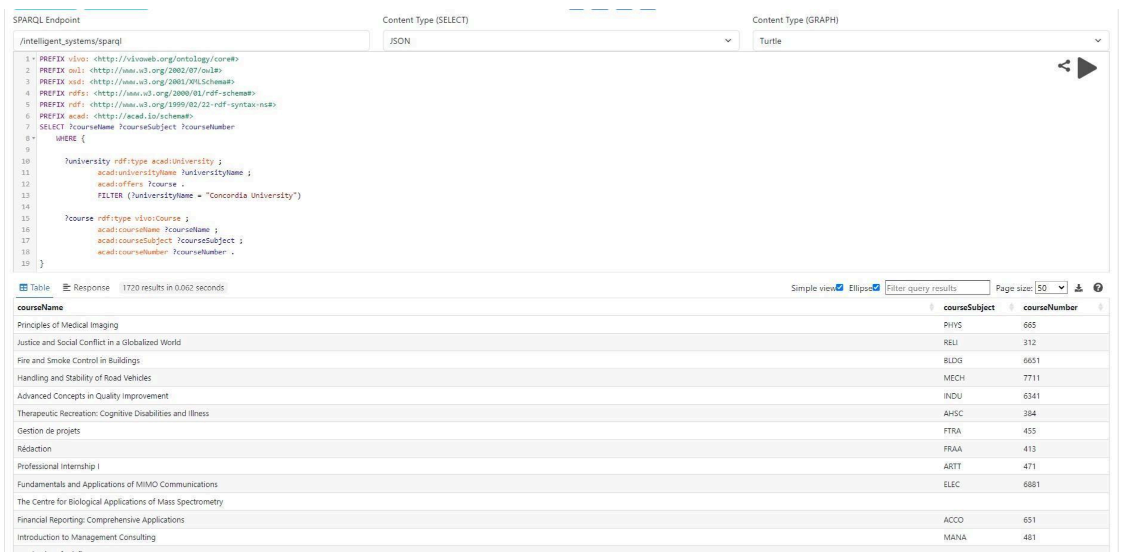
Graph Queries

```
In [1]: import requests
        import json
        # Define the endpoint URL
        endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'
```

1. List all courses offered by [university]

```
In [22]: # Define the SPARQL query
         university_name = "Concordia University"

         sparql_query = f"""
         PREFIX vivo: <http://vivoweb.org/ontology/core#>
          PREFIX owl: <http://www.w3.org/2002/07/owl#>
         PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
         PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
         PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
          PREFIX acad: <http://acad.io/schema#>
          SELECT ?courseName ?courseSubject ?courseNumber
              WHERE {{

         ?university rdf:type acad:University ;
         acad:universityName ?universityName ; acad:offers ?course .
         FILTER (?universityName = "{university_name}")

         ?course rdf:type vivo:Course ;
         acad:courseName ?courseName ;
         acad:courseSubject ?courseSubject ; acad:courseNumber
                                    ?courseNumber .
          }}
          """

          # Define the endpoint URL
         endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

          # Define the payload
         payload = {'query': sparql_query}

          # Send the POST request
         response = requests.post(endpoint_url, data=payload)

          # Print the response
         print(len(response.json()['results']['bindings']))
          print(response.json()['results']['bindings'][:5])
```

```
1720
[{'courseName': {'type': 'literal', 'value': 'Principles of Medical Imaging'},
'cour seSubject': {'type': 'literal', 'value': 'PHYS'}, 'courseNumber': {'type':
'litera
l', 'value': '665'}}, {'courseName': {'type': 'literal', 'value': 'Justice and
Socia l Conflict in a Globalized World'}, 'courseSubject': {'type': 'literal',
'value': 'R ELI'}, 'courseNumber': {'type': 'literal', 'value': '312'}},
{'courseName': {'type': 'literal', 'value': 'Fire and Smoke Control in Buildings'},
'courseSubject': {'typ
e': 'literal', 'value': 'BLDG'}, 'courseNumber': {'type': 'literal', 'value': '665
1'}}, {'courseName': {'type': 'literal', 'value': 'Handling and Stability of Road Ve
hicles'}, 'courseSubject': {'type': 'literal', 'value': 'MECH'}, 'courseNumber': {'t
ype': 'literal', 'value': '7711'}}, {'courseName': {'type': 'literal', 'value':
'Adv anced Concepts in Quality Improvement'}, 'courseSubject': {'type': 'literal',
'valu e': 'INDU'}, 'courseNumber': {'type': 'literal', 'value': '6341'}}]
```



SPARQL Endpoint: /intelligent_systems/sparql

Content Type (SELECT): JSON

Content Type (GRAPH): Turtle

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>
SELECT ?courseName ?courseSubject ?courseNumber
    WHERE {

      ?university rdf:type acad:University ;
              acad:universityName ?universityName ;
              acad:offers ?course .
              FILTER (?universityName = "Concordia University")

      ?course rdf:type vivo:Course ;
              acad:courseName ?courseName ;
              acad:courseSubject ?courseSubject ;
              acad:courseNumber ?courseNumber .
}
```

Table | Response | 1720 results in 0.062 seconds | Simple view ✓ | Ellipse ✓ | Filter query results | Page size: 50

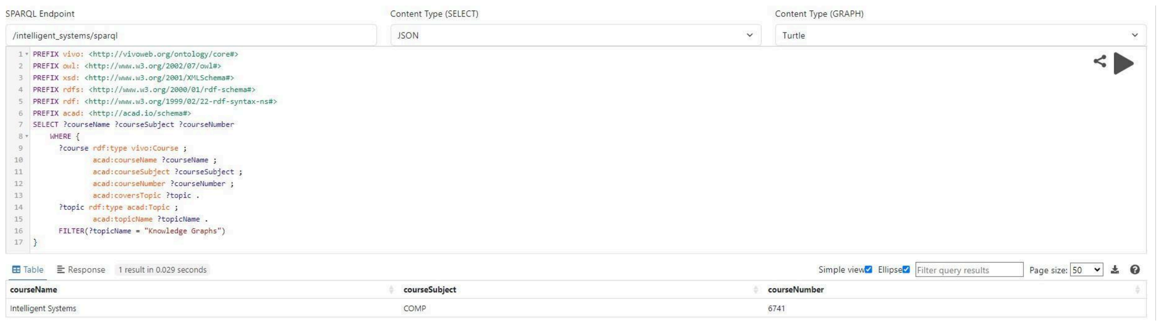
courseName	courseSubject	courseNumber
Principles of Medical Imaging	PHYS	665
Justice and Social Conflict in a Globalized World	RELI	312
Fire and Smoke Control in Buildings	BLDG	6651
Handling and Stability of Road Vehicles	MECH	7711
Advanced Concepts in Quality Improvement	INDU	6341
Therapeutic Recreation: Cognitive Disabilities and Illness	AHSC	384
Gestion de projets	FTRA	455
Rédaction	FRAA	413
Professional Internship I	ARTT	471
Fundamentals and Applications of MIMO Communications	ELEC	6881
The Centre for Biological Applications of Mass Spectrometry		
Financial Reporting: Comprehensive Applications	ACCO	651
Introduction to Management Consulting	MANA	481

2. In which courses is [topic] discussed?

In [3]:
```
# Define the SPARQL query
topic_name = "Knowledge Graphs"

sparql_query = f"""
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>
SELECT ?courseName ?courseSubject ?courseNumber
    WHERE {{
          ?course rdf:type vivo:Course ;
              acad:courseName ?courseName ;
                          acad:courseSubject ?courseSubject ;
                          acad:courseNumber ?courseNumber ;
              acad:coversTopic ?topic .
      ?topic rdf:type acad:Topic ;
              acad:topicName ?topicName .
      FILTER(?topicName = "{topic_name}")
}}
```

```python
"""

# Define the endpoint URL
endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

# Define the payload
payload = {'query': sparql_query}

# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
print(json.dumps(response.json(), indent=1))
```

```
{
 "head": {
  "vars": [
   "courseName",
   "courseSubject",
   "courseNumber"
  ]
 },
 "results": {
  "bindings": [
   {
    "courseName": {
     "type": "literal",
     "value": "Intelligent Systems"
    },
    "courseSubject": {
     "type": "literal",
     "value": "COMP"
    },
    "courseNumber": {
     "type": "literal",
     "value": "6741"
    }
   }
  ]
 }
}
```



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

```sparql
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>
SELECT ?courseName ?courseSubject ?courseNumber
    WHERE {
      ?course rdf:type vivo:Course ;
              acad:courseName ?courseName ;
              acad:courseSubject ?courseSubject ;
              acad:courseNumber ?courseNumber ;
              acad:coversTopic ?topic .
      ?topic rdf:type acad:Topic ;
              acad:topicName ?topicName .
      FILTER(?topicName = "Knowledge Graphs")
}
```

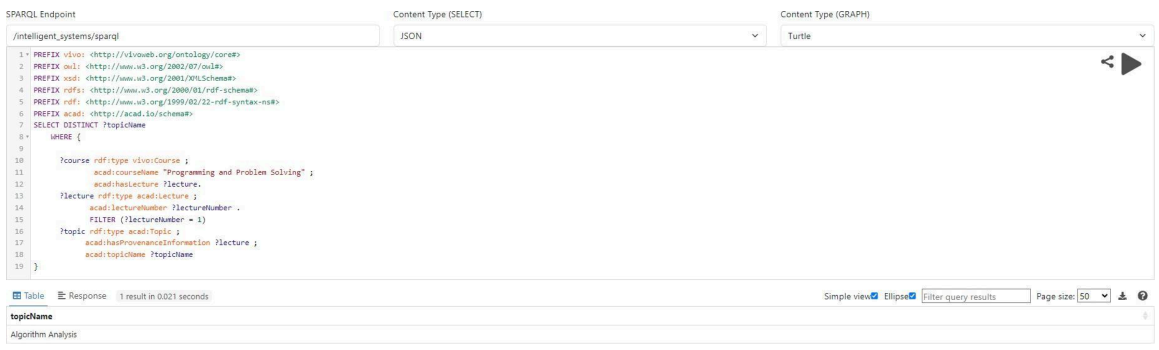
Table | Response | 1 result in 0.029 seconds | Simple view ✓ | Ellipse ✓ | Page size: 50

courseName	courseSubject	courseNumber
Intelligent Systems	COMP	6741

3. Which [topics] are covered in [course] during [lecture number]?

```
In [4]: # Define the SPARQL query
        course_name = "Programming and Problem Solving"
         lecture_number = 1

        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
         PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
         PREFIX acad: <http://acad.io/schema#>
         SELECT DISTINCT ?topicName
             WHERE {{

        ?course rdf:type vivo:Course ;
        acad:courseName "{course_name}" ;
        acad:hasLecture ?lecture.
        ?lecture rdf:type acad:Lecture ;
        acad:lectureNumber ?lectureNumber .
        FILTER (?lectureNumber = {lecture_number})
        ?topic rdf:type acad:Topic ;
        acad:hasProvenanceInformation ?lecture ; acad:topicName
                     ?topicName
         }}
         """

         # Define the endpoint URL
        endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

         # Define the payload
        payload = {'query': sparql_query}

         # Send the POST request
        response = requests.post(endpoint_url, data=payload)

         # Print the response
        print(json.dumps(response.json(), indent=1))
```

```
       {
        "head": {
         "vars": [
          "topicName"
      ]
     },
        "results": {
         "bindings": [
          {
           "topicName": {
            "type": "literal",
            "value": "Algorithm Analysis"
           }
          }
         ]
        }
       }
```


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

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>
SELECT DISTINCT ?topicName
    WHERE {

      ?course rdf:type vivo:Course ;
              acad:courseName "Programming and Problem Solving" ;
              acad:hasLecture ?lecture.
      ?lecture rdf:type acad:Lecture ;
             acad:lectureNumber ?lectureNumber .
             FILTER (?lectureNumber = 1)
      ?topic rdf:type acad:Topic ;
            acad:hasProvenanceInformation ?lecture ;
            acad:topicName ?topicName
}
```

Table | Response | 1 result in 0.021 seconds

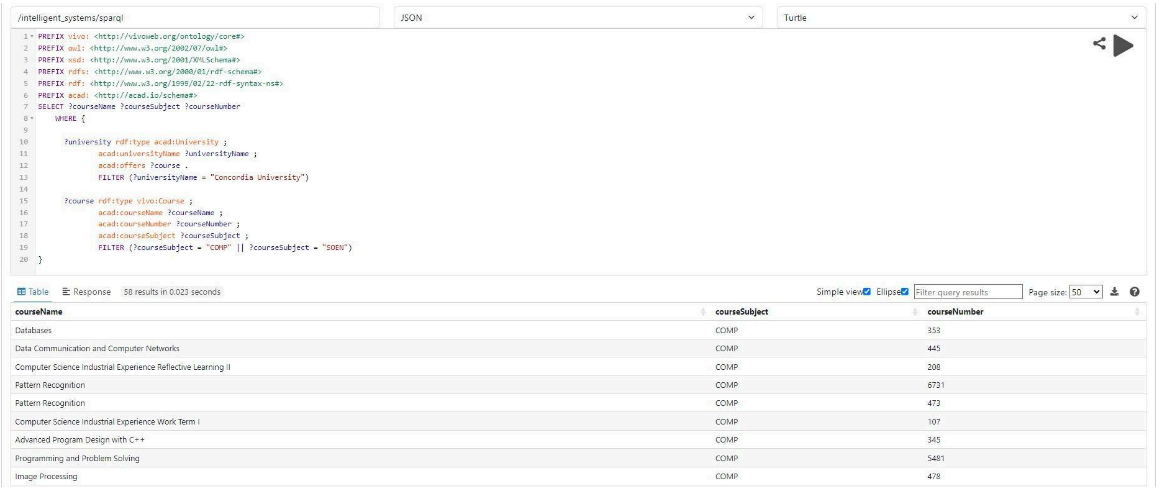
topicName
Algorithm Analysis

4. List all [courses] offered by [university] within the [subject] (e.g., \COMP", \SOEN").

In [23]:
```
# Define the SPARQL query
course_subject1 = "COMP"
course_subject2 = "SOEN"
university_name = "Concordia University"
lecture_number = 1

sparql_query = f"""
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>
SELECT ?courseName ?courseSubject ?courseNumber
    WHERE {{

          ?university rdf:type acad:University ;
                           acad:universityName ?universityName ;
                           acad:offers ?course .
              FILTER (?universityName = "{university_name}")

          ?course rdf:type vivo:Course ;
              acad:courseName ?courseName ;
                           acad:courseNumber ?courseNumber ;
              acad:courseSubject ?courseSubject ;
              FILTER (?courseSubject = "{course_subject1}" || ?courseSubject = "{co
}}
"""

# Define the endpoint URL
endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

# Define the payload
payload = {'query': sparql_query}

# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
```

```python
# Print the response
print(len(response.json()['results']['bindings']))
print(response.json()['results']['bindings'][:5])
```

```
58
[{'courseName': {'type': 'literal', 'value': 'Databases'}, 'courseSubject': {'type':
'literal', 'value': 'COMP'}, 'courseNumber': {'type': 'literal', 'value': '353'}},
{'courseName':  {'type':  'literal',  'value':  'Data  Communication  and  Computer
Network s'}, 'courseSubject': {'type': 'literal', 'value': 'COMP'}, 'courseNumber':
{'type':
'literal', 'value': '445'}}, {'courseName': {'type': 'literal', 'value': 'Computer
S cience Industrial Experience Reflective Learning II'}, 'courseSubject': {'type':
'li teral', 'value': 'COMP'}, 'courseNumber': {'type': 'literal', 'value': '208'}},
{'co    urseName':    {'type':    'literal',    'value':    'Pattern    Recognition'},
'courseSubject': {'t ype': 'literal', 'value': 'COMP'}, 'courseNumber': {'type':
'literal', 'value': '673 1'}}, {'courseName': {'type': 'literal', 'value': 'Pattern
Recognition'},   'courseSub   ject':   {'type':   'literal',   'value':   'COMP'},
'courseNumber': {'type': 'literal', 'va lue': '473'}}]
```



/intelligent_systems/sparql JSON Turtle

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>
SELECT ?courseName ?courseSubject ?courseNumber
    WHERE {

      ?university rdf:type acad:University ;
              acad:universityName ?universityName ;
              acad:offers ?course .
              FILTER (?universityName = "Concordia University")

      ?course rdf:type vivo:Course ;
              acad:courseName ?courseName ;
              acad:courseNumber ?courseNumber ;
              acad:courseSubject ?courseSubject ;
              FILTER (?courseSubject = "COMP" || ?courseSubject = "SOEN")
}
```

courseName	courseSubject	courseNumber
Databases	COMP	353
Data Communication and Computer Networks	COMP	445
Computer Science Industrial Experience Reflective Learning II	COMP	208
Pattern Recognition	COMP	6731
Pattern Recognition	COMP	473
Computer Science Industrial Experience Work Term I	COMP	107
Advanced Program Design with C++	COMP	345
Programming and Problem Solving	COMP	5481
Image Processing	COMP	478

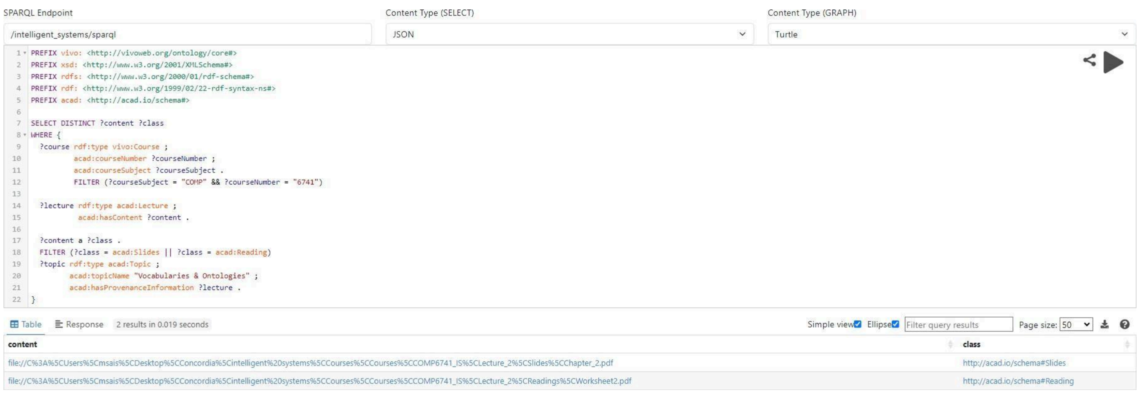
5. What [materials] (slides, readings) are recommended for [topic] in [course] [number]?

In [6]:

```python
# Define the SPARQL query
course_subject = "COMP"
course_number = "6741"
topic_name = "Vocabularies & Ontologies"

sparql_query = f"""
PREFIX ac: <http://umbel.org/umbel/ac/>
PREFIX prefix: <http://prefix.cc/>
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?content ?class
WHERE {{
  ?course rdf:type vivo:Course ;
```

```
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject .
          FILTER (?courseSubject = "{course_subject}" && ?courseNumber = "{course_n

  ?lecture rdf:type acad:Lecture ;
           acad:hasContent ?content .

  ?content a ?class .
  FILTER (?class = acad:Slides || ?class = acad:Reading)
  ?topic rdf:type acad:Topic ;
         acad:topicName "{topic_name}" ;
         acad:hasProvenanceInformation ?lecture .
}}
"""

# Define the endpoint URL
endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

# Define the payload
payload = {'query': sparql_query}

# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
print(json.dumps(response.json(), indent=1))
```

```
{
 "head": {
  "vars": [
   "content",
   "class"
  ]
 },
 "results": {
  "bindings": [
   {
    "content": {
     "type": "uri",
     "value": "file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20sys
tems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_2%5CSlides%5CChapter_2.pdf"
    },
    "class": {
     "type": "uri",
     "value": "http://acad.io/schema#Slides"
 }
},
   {
    "content": {
     "type": "uri",
     "value": "file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20sys
tems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_2%5CReadings%5CWorksheet2.pdf"
    },
    "class": {
     "type": "uri",
     "value": "http://acad.io/schema#Reading"
    }
   }
  ]
 }
}
```



```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?content ?class
WHERE {
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject .
          FILTER (?courseSubject = "COMP" && ?courseNumber = "6741")

  ?lecture rdf:type acad:Lecture ;
           acad:hasContent ?content .

  ?content a ?class .
  FILTER (?class = acad:Slides || ?class = acad:Reading)
  ?topic rdf:type acad:Topic ;
         acad:topicName "Vocabularies & Ontologies" ;
         acad:hasProvenanceInformation ?lecture .
}
```

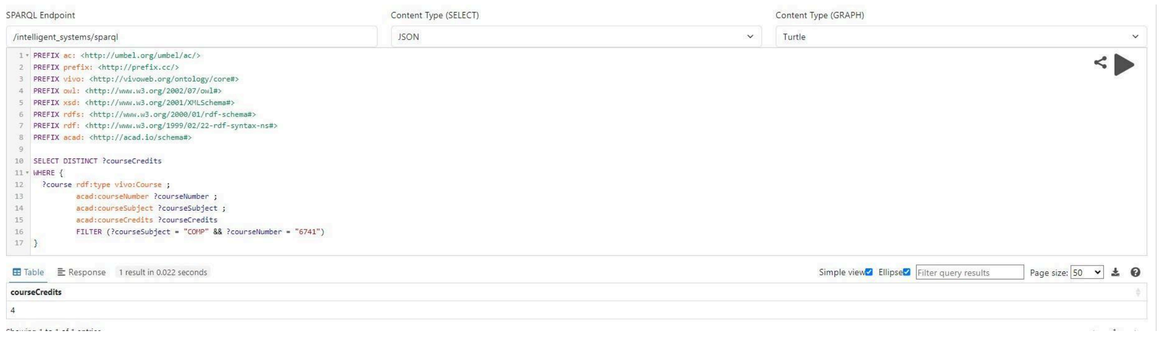
content	class
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_2%5CSlides%5CChapter_2.pdf	http://acad.io/schema#Slides
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_2%5CReadings%5CWorksheet2.pdf	http://acad.io/schema#Reading

6. How many credits is [course] [number] worth?

In [7]:
```python
# Define the SPARQL query
course_subject = "COMP"
course_number = "6741"

sparql_query = f"""
```

```
PREFIX ac: <http://umbel.org/umbel/ac/>
PREFIX prefix: <http://prefix.cc/>
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?courseCredits
WHERE {{
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject ;
          acad:courseCredits ?courseCredits
          FILTER (?courseSubject = "{course_subject}" && ?courseNumber = "{course_n
}}
"""

# Define the endpoint URL
endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

# Define the payload
payload = {'query': sparql_query}

# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
print(json.dumps(response.json(), indent=1))
```

```
{
 "head": {
  "vars": [
   "courseCredits"
  ]
 },
 "results": {
  "bindings": [
   {
    "courseCredits": {
     "type": "literal",
     "value": "4"
    }
   }
  ]
 }
}
```



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

```
PREFIX ac: <http://umbel.org/umbel/ac/>
PREFIX prefix: <http://prefix.cc/>
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?courseCredits
WHERE {
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject ;
          acad:courseCredits ?courseCredits
          FILTER (?courseSubject = "COMP" && ?courseNumber = "6741")
}
```

Table | Response | 1 result in 0.022 seconds

Simple view ☑ Ellipse ☑ Filter query results Page size: 50

courseCredits
4

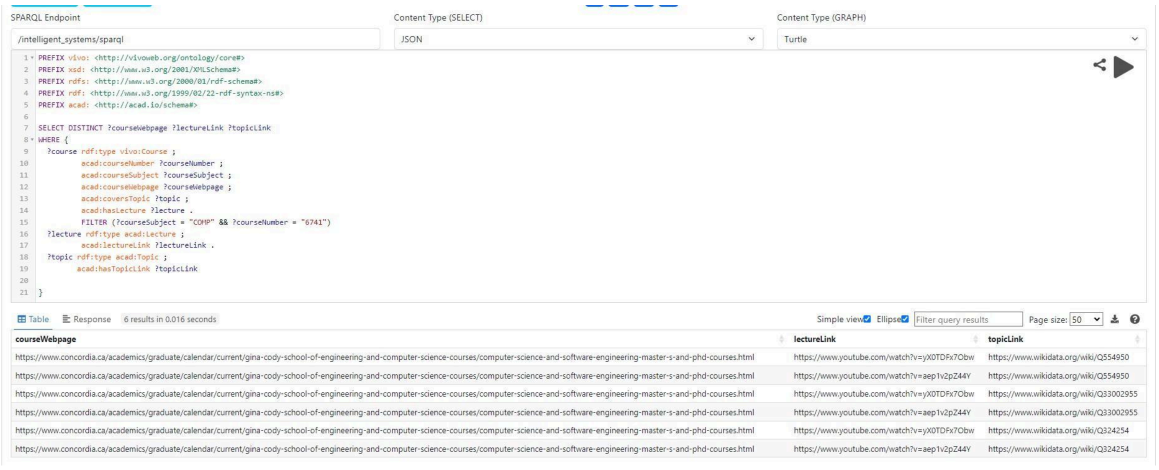
7. For [course] [number], what additional resources (links to web pages) are available

In [24]:

```python
# Define the SPARQL query
course_subject = "COMP"
course_number = "6741"

sparql_query = f"""
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?courseWebpage ?lectureLink ?topicLink
WHERE {{
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject  ;
          acad:courseWebpage  ?courseWebpage  ;
          acad:coversTopic ?topic ;
          acad:hasLecture ?lecture .
          FILTER (?courseSubject = "{course_subject}" && ?courseNumber = "{course_n
  ?lecture rdf:type acad:Lecture ;
          acad:lectureLink ?lectureLink .
  ?topic rdf:type acad:Topic ;
         acad:hasTopicLink ?topicLink

}}
"""

# Define the endpoint URL
endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

# Define the payload
payload = {'query': sparql_query}

# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
# Print the response
```

```python
print(len(response.json()['results']['bindings']))
print(response.json()['results']['bindings'][:5])
```

```
6
[{'courseWebpage':              {'type':             'literal',             'value':
'https://www.concordia.ca/academics/
graduate/calendar/current/gina-cody-school-of-engineering-and-computer-science-cour
s     es/computer-science-and-software-engineering-master-s-and-phd-courses.html'},
'lectu          reLink':          {'type':           'literal',           'value':
'https://www.youtube.com/watch?v=yX0TDFx7Ob
w'},          'topicLink':          {'type':          'literal',          'value':
'https://www.wikidata.org/wiki/Q55495  0'}}, {'courseWebpage': {'type': 'literal',
'value':                                      'https://www.concordia.ca/acade
mics/graduate/calendar/current/gina-cody-school-of-engineering-and-computer-science
-
courses/computer-science-and-software-engineering-master-s-and-phd-courses.html'},
'lectureLink':              {'type':              'literal',              'value':
'https://www.youtube.com/watch?v=aep1v2p  Z44Y'},  'topicLink':  {'type':  'literal',
'value':  'https://www.wikidata.org/wiki/Q55  4950'}},  {'courseWebpage':  {'type':
'literal',                  'value':                  'https://www.concordia.ca/ac
ademics/graduate/calendar/current/gina-cody-school-of-engineering-and-computer-scie
n ce-courses/computer-science-and-software-engineering-master-s-and-phd-courses.htm
l'}, 'lectureLink': {'type': 'literal', 'value':
'https://www.youtube.com/watch?v=yX 0TDFx7Obw'}, 'topicLink': {'type': 'literal',
'value': 'https://www.wikidata.org/wik i/Q33002955'}}, {'courseWebpage': {'type':
'literal', 'value': 'https://www.concordi
a.ca/academics/graduate/calendar/current/gina-cody-school-of-engineering-and-comput
e
r-science-courses/computer-science-and-software-engineering-master-s-and-phd-course
s.html'}, 'lectureLink': {'type': 'literal', 'value': 'https://www.youtube.com/watc
h?v=aep1v2pZ44Y'}, 'topicLink': {'type': 'literal', 'value':
'https://www.wikidata.o rg/wiki/Q33002955'}}, {'courseWebpage': {'type': 'literal',
'value': 'https://www.co
ncordia.ca/academics/graduate/calendar/current/gina-cody-school-of-engineering-and-
c
omputer-science-courses/computer-science-and-software-engineering-master-s-and-phd-
c  ourses.html'}, 'lectureLink': {'type': 'literal', 'value':
'https://www.youtube.com/ watch?v=yX0TDFx7Obw'}, 'topicLink': {'type': 'literal',
'value': 'https://www.wikida ta.org/wiki/Q324254'}}]
```



SPARQL Endpoint: /intelligent_systems/sparql

Content Type (SELECT): JSON

Content Type (GRAPH): Turtle

```sparql
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?courseWebpage ?lectureLink ?topicLink
WHERE {
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject ;
          acad:courseWebpage ?courseWebpage ;
          acad:coversTopic ?topic ;
          acad:hasLecture ?lecture .
          FILTER (?courseSubject = "COMP" && ?courseNumber = "6741")
  ?lecture rdf:type acad:Lecture ;
          acad:lectureLink ?lectureLink .
  ?topic rdf:type acad:Topic ;
         acad:hasTopicLink ?topicLink

}
```

Table | Response | 6 results in 0.016 seconds

Simple view ✓ Ellipse ✓ Page size: 50

courseWebpage	lectureLink	topicLink
https://www.concordia.ca/academics/graduate/calendar/current/gina-cody-school-of-engineering-and-computer-science-courses/computer-science-and-software-engineering-master-s-and-phd-courses.html	https://www.youtube.com/watch?v=yX0TDFx7Obw	https://www.wikidata.org/wiki/Q554950
https://www.concordia.ca/academics/graduate/calendar/current/gina-cody-school-of-engineering-and-computer-science-courses/computer-science-and-software-engineering-master-s-and-phd-courses.html	https://www.youtube.com/watch?v=aep1v2pZ44Y	https://www.wikidata.org/wiki/Q554950
https://www.concordia.ca/academics/graduate/calendar/current/gina-cody-school-of-engineering-and-computer-science-courses/computer-science-and-software-engineering-master-s-and-phd-courses.html	https://www.youtube.com/watch?v=yX0TDFx7Obw	https://www.wikidata.org/wiki/Q33002955
https://www.concordia.ca/academics/graduate/calendar/current/gina-cody-school-of-engineering-and-computer-science-courses/computer-science-and-software-engineering-master-s-and-phd-courses.html	https://www.youtube.com/watch?v=aep1v2pZ44Y	https://www.wikidata.org/wiki/Q33002955
https://www.concordia.ca/academics/graduate/calendar/current/gina-cody-school-of-engineering-and-computer-science-courses/computer-science-and-software-engineering-master-s-and-phd-courses.html	https://www.youtube.com/watch?v=yX0TDFx7Obw	https://www.wikidata.org/wiki/Q324254
https://www.concordia.ca/academics/graduate/calendar/current/gina-cody-school-of-engineering-and-computer-science-courses/computer-science-and-software-engineering-master-s-and-phd-courses.html	https://www.youtube.com/watch?v=aep1v2pZ44Y	https://www.wikidata.org/wiki/Q324254

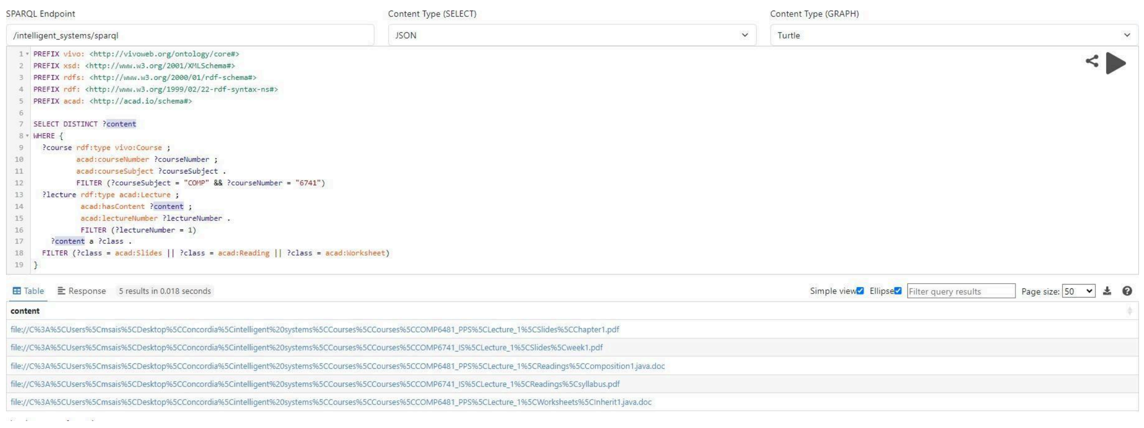
8. Detail the content (slides, worksheets, readings) available for [lecture number] in [course] [number].

In [9]:

```python
# Define the SPARQL query
course_subject = "COMP"
course_number = "6741"
```

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?content
WHERE {{
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject .
          FILTER (?courseSubject = "{course_subject}" && ?courseNumber = "{course_n
  ?lecture rdf:type acad:Lecture ;
           acad:hasContent ?content ;
                   acad:lectureNumber ?lectureNumber .
                   FILTER (?lectureNumber = {lecture_number})
    ?content a ?class .
  FILTER (?class = acad:Slides || ?class = acad:Reading || ?class = acad:Worksheet)
}}
"""

# Define the endpoint URL
endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

# Define the payload
payload = {'query': sparql_query}

# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
print(json.dumps(response.json(), indent=1))
```

```
{
 "head": {
  "vars": [
   "content"
  ]
 },
 "results": {
  "bindings": [
   {
    "content": {
     "type": "uri",
     "value": "file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20sys
tems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_1%5CSlides%5CChapter1.pdf"
    }
   },
   {
    "content": {
     "type": "uri",
     "value": "file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20sys
tems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_1%5CSlides%5Cweek1.pdf"
    }
   },
   {
    "content": {
     "type": "uri",
     "value":
"file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20sys
tems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_1%5CReadings%5CComposition1.java.
d oc"
    }
   },
   {
    "content": {
     "type": "uri",
     "value": "file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20sys
tems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_1%5CReadings%5Csyllabus.pdf"
    }
   },
   {
    "content": {
     "type": "uri",
     "value":
"file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20sys
tems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_1%5CWorksheets%5CInherit1.java.
do  c"
    }
   }
  ]
 }
}
```


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

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?content
WHERE {
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject .
          FILTER (?courseSubject = "COMP" && ?courseNumber = "6741")
  ?lecture rdf:type acad:Lecture ;
           acad:hasContent ?content ;
           acad:lectureNumber ?lectureNumber .
           FILTER (?lectureNumber = 1)
    ?content a ?class .
  FILTER (?class = acad:Slides || ?class = acad:Reading || ?class = acad:Worksheet)
}
```

Table | Response | 5 results in 0.018 seconds

Simple view ✓ Ellipse ✓ Filter query results Page size: 50

content
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_1%5CSlides%5CChapter1.pdf
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_1%5CSlides%5Cweek1.pdf
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_1%5CReadings%5CComposition1.java.doc
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_1%5CReadings%5Csyllabus.pdf
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_1%5CWorksheets%5CInherit1.java.doc

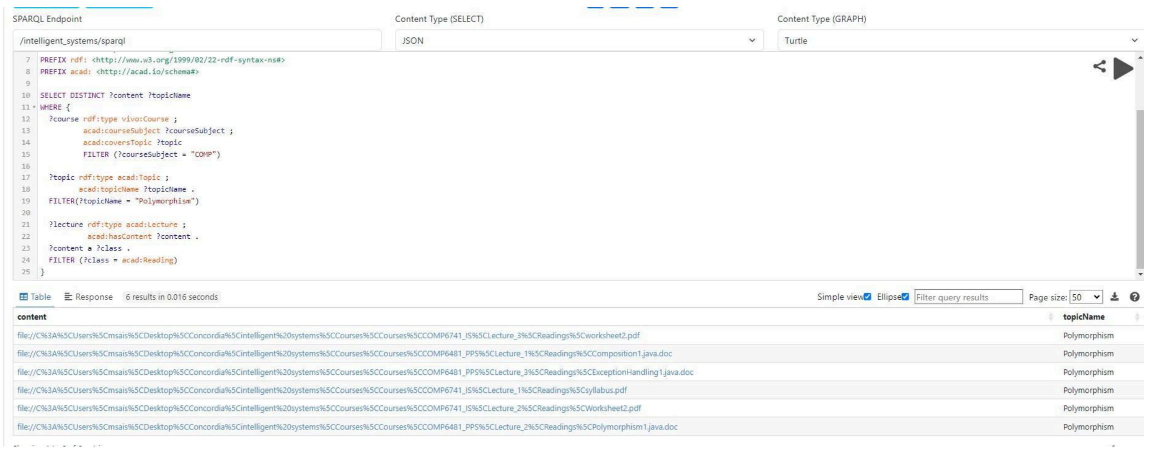
9. What reading materials are recommended for studying [topic] in [course]?

In [10]:

```
# Define the SPARQL query
course_subject = "COMP"
topic_name = "Polymorphism"

sparql_query = f"""
PREFIX ac: <http://umbel.org/umbel/ac/>
PREFIX prefix: <http://prefix.cc/>
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?content ?topicName
WHERE {{
  ?course rdf:type vivo:Course ;
          acad:courseSubject ?courseSubject ;
          acad:coversTopic ?topic
          FILTER (?courseSubject = "{course_subject}")

  ?topic rdf:type acad:Topic ;
         acad:topicName ?topicName .
  FILTER(?topicName = "{topic_name}")

  ?lecture rdf:type acad:Lecture ;
           acad:hasContent ?content .
  ?content a ?class .
  FILTER (?class = acad:Reading)
}}
"""

# Define the endpoint URL
endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

# Define the payload
payload = {'query': sparql_query}
```

```python
# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
print(response.json())
```

```
{'head': {'vars': ['content', 'topicName']}, 'results': {'bindings': [{'content':
{'type':                          'uri',                                   'value':
'file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintell
igent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_3%5CReadings%5Cworkshee
t 2.pdf'}, 'topicName': {'type': 'literal', 'value': 'Polymorphism'}}, {'content':
{'t                    ype':                 'uri',                        'value':
'file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintellige
nt%20systems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_1%5CReadings%5CCompositio
n   1.java.doc'},   'topicName':   {'type':   'literal',   'value':   'Polymorphism'}},
{'conten
t':                    {'type':                    'uri',                    'value':
'file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cin
telligent%20systems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_3%5CReadings%5CExc
e ptionHandling1.java.doc'}, 'topicName': {'type': 'literal', 'value': 'Polymorphis
m'}},           {'content':           {'type':           'uri',           'value':
'file://C%3A%5CUsers%5Cmsais%5CDesktop%5C
Concordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_1%5CR
e    adings%5Csyllabus.pdf'},    'topicName':    {'type':    'literal',    'value':
'Polymorphism'}},
{'content': {'type': 'uri', 'value': 'file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcor
dia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_2%5CReading
s%5CWorksheet2.pdf'}, 'topicName': {'type': 'literal', 'value': 'Polymorphism'}},
{'content':              {'type':               'uri',                       'value':
'file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcor
dia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_2%5CReadin
g   s%5CPolymorphism1.java.doc'},   'topicName':   {'type':   'literal',   'value':
'Polymorphis m'}}]}}
```


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

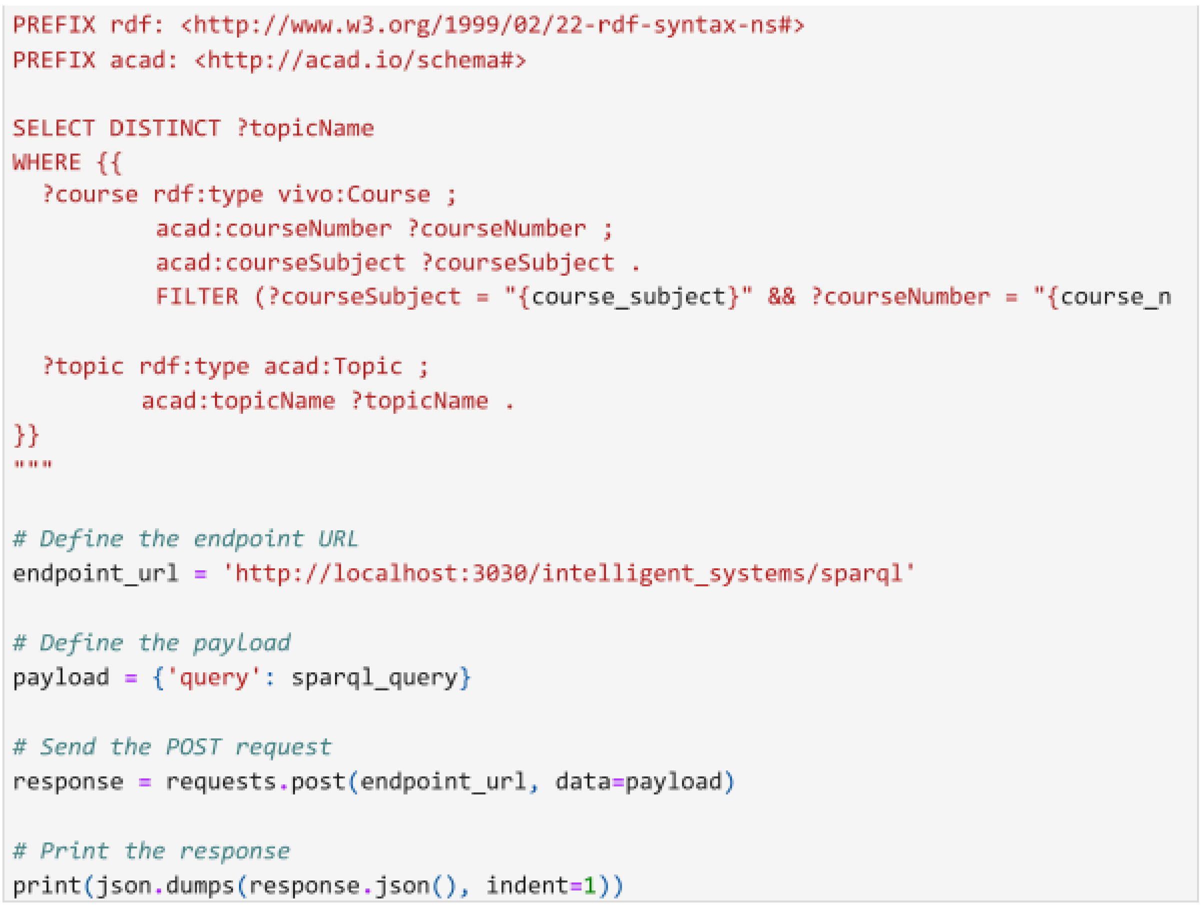
```sparql
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?content ?topicName
WHERE {
  ?course rdf:type vivo:Course ;
          acad:courseSubject ?courseSubject ;
          acad:coversTopic ?topic
          FILTER (?courseSubject = "COMP")

  ?topic rdf:type acad:Topic ;
         acad:topicName ?topicName .
  FILTER(?topicName = "Polymorphism")

  ?lecture rdf:type acad:Lecture ;
           acad:hasContent ?content .
  ?content a ?class .
  FILTER (?class = acad:Reading)
}
```

Table | Response | 6 results in 0.016 seconds | Simple view ✓ | Ellipse ✓ | Page size: 50

content	topicName
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_3%5CReadings%5Cworksheet2.pdf	Polymorphism
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_1%5CReadings%5CComposition1.java.doc	Polymorphism
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_3%5CReadings%5CExceptionHandling1.java.doc	Polymorphism
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_1%5CReadings%5Csyllabus.pdf	Polymorphism
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_2%5CReadings%5CWorksheet2.pdf	Polymorphism
file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6481_PPS%5CLecture_2%5CReadings%5CPolymorphism1.java.doc	Polymorphism

10. What competencies [topics] does a student gain after completing [course] [number]?

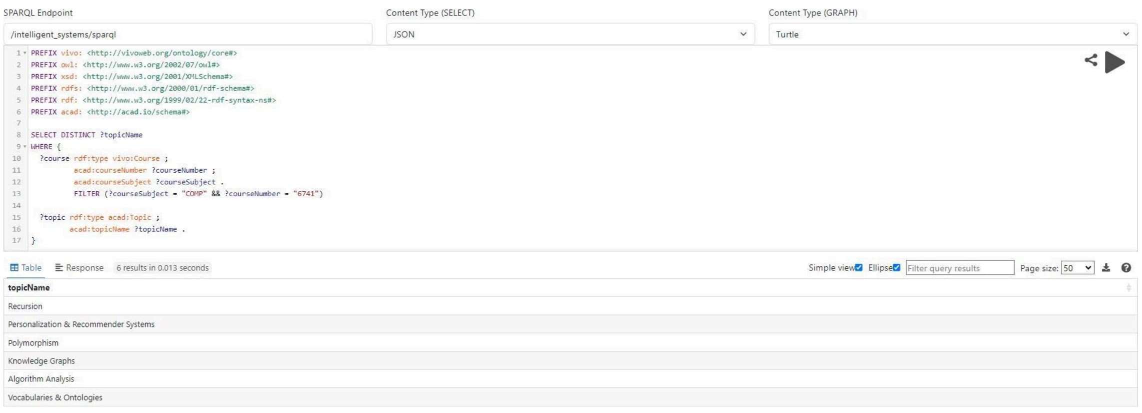
In [11]:
```python
# Define the SPARQL query
course_subject = "COMP"
course_number = "6741"

sparql_query = f"""
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?topicName
WHERE {{
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject .
          FILTER (?courseSubject = "{course_subject}" && ?courseNumber = "{course_n

  ?topic rdf:type acad:Topic ;
         acad:topicName ?topicName .
}}
"""

# Define the endpoint URL
endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

# Define the payload
payload = {'query': sparql_query}

# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
print(json.dumps(response.json(), indent=1))
```

```
   {
   "head": {
    "vars": [
     "topicName"
 ]
},
   "results": {
    "bindings": [
     {
      "topicName": {
       "type": "literal",
       "value": "Recursion"
   }
  },
     {
      "topicName": {
       "type": "literal",
       "value": "Personalization & Recommender Systems"
   }
  },
     {
      "topicName": {
       "type": "literal",
       "value": "Polymorphism"
   }
  },
     {
      "topicName": {
       "type": "literal",
       "value": "Knowledge Graphs"
   }
  },
     {
```

```
    "topicName": {
     "type": "literal",
     "value": "Algorithm Analysis"
  }
 },
   {
    "topicName": {
     "type": "literal",
     "value": "Vocabularies & Ontologies"
    }
   }
  ]
 }
}
```


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

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?topicName
WHERE {
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject .
          FILTER (?courseSubject = "COMP" && ?courseNumber = "6741")

  ?topic rdf:type acad:Topic ;
         acad:topicName ?topicName .
}
```

Table | Response | 6 results in 0.013 seconds — Simple view ✓ Ellipse ✓ Filter query results Page size: 50

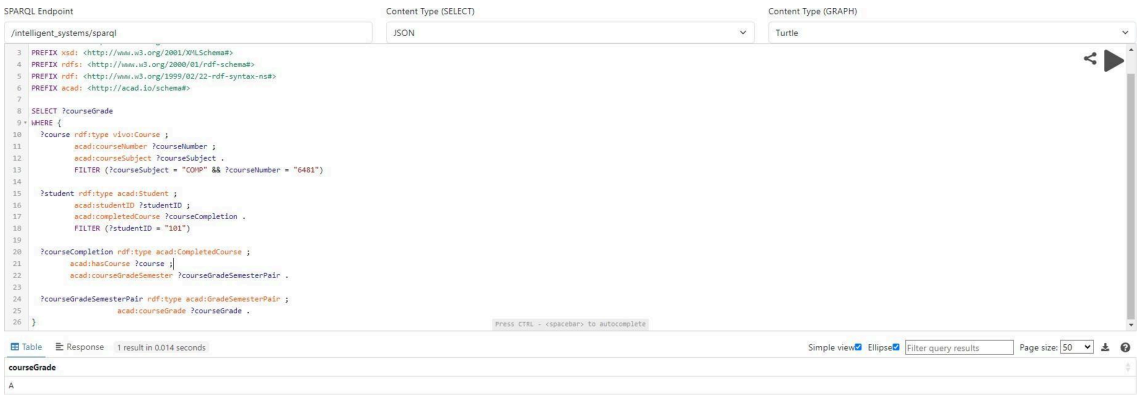
topicName
Recursion
Personalization & Recommender Systems
Polymorphism
Knowledge Graphs
Algorithm Analysis
Vocabularies & Ontologies

11. What grades did [student] achieve in [course] [number]?

In [12]:
```
# Define the SPARQL query
course_subject = "COMP"
course_number = "6741"
student_id = "101"

sparql_query = f"""
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT ?courseGrade
WHERE {{
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject .
          FILTER (?courseSubject = "{course_subject}" && ?courseNumber = "{course_n

  ?student rdf:type acad:Student ;
          acad:studentID ?studentID ;
                  acad:completedCourse ?courseCompletion .
                  FILTER (?studentID = "{student_id}")

  ?courseCompletion rdf:type acad:CompletedCourse ;
         acad:hasCourse ?course ;
                 acad:courseGradeSemester ?courseGradeSemesterPair .

  ?courseGradeSemesterPair rdf:type acad:GradeSemesterPair ;
```

```
# Define the payload
payload = {'query': sparql_query}

# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
print(json.dumps(response.json(), indent=1))
```

```
{
 "head": {
  "vars": [
   "courseGrade"
  ]
 },
 "results": {
  "bindings": [
   {
    "courseGrade": {
     "type": "literal",
     "value": "A+"
    }
   }
  ]
 }
}
```



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

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT ?courseGrade
WHERE {
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject .
          FILTER (?courseSubject = "COMP" && ?courseNumber = "6481")

  ?student rdf:type acad:Student ;
          acad:studentID ?studentID ;
          acad:completedCourse ?courseCompletion .
          FILTER (?studentID = "101")

  ?courseCompletion rdf:type acad:CompletedCourse ;
        acad:hasCourse ?course ;
        acad:courseGradeSemester ?courseGradeSemesterPair .

  ?courseGradeSemesterPair rdf:type acad:GradeSemesterPair ;
                acad:courseGrade ?courseGrade .
}
```

Press CTRL - <spacebar> to autocomplete

Table | Response | 1 result in 0.014 seconds

Simple view ✓ Ellipse ✓ Filter query results Page size: 50

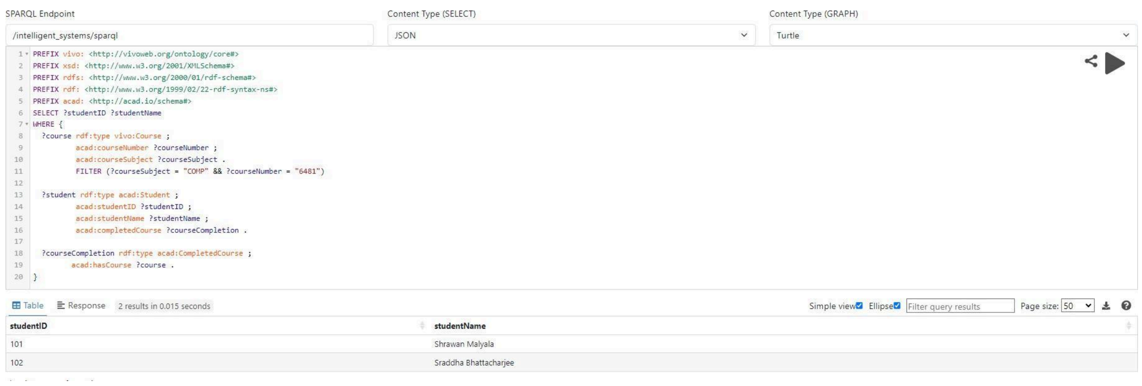
courseGrade
A

12. Which [students] have completed [course] [number]?

```
In [13]: # Define the SPARQL query
         course_subject = "COMP"
         course_number = "6741"

         sparql_query = f"""
         PREFIX vivo: <http://vivoweb.org/ontology/core#>
         PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
         PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
         PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
         PREFIX acad: <http://acad.io/schema#>
```

```
SELECT ?studentID ?studentName
WHERE {{
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject .
          FILTER (?courseSubject = "{course_subject}" && ?courseNumber = "{course_n

  ?student rdf:type acad:Student ;
          acad:studentID ?studentID ;
          acad:studentName ?studentName ;
                  acad:completedCourse ?courseCompletion .

  ?courseCompletion rdf:type acad:CompletedCourse ;
         acad:hasCourse ?course .
}}

"""

# Define the endpoint URL
endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

# Define the payload
payload = {'query': sparql_query}

# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
print(json.dumps(response.json(), indent=1))
```

```
{
 "head": {
  "vars": [
   "studentID",
   "studentName"
  ]
 },
 "results": {
  "bindings": [
   {
    "studentID": {
     "type": "literal",
     "value": "101"
    },
    "studentName": {
     "type": "literal",
     "value": "Shrawan Malyala"
 }
},
   {
    "studentID": {
     "type": "literal",
     "value": "102"
    },
    "studentName": {
     "type": "literal",
     "value": "Sraddha Bhattacharjee"
    }
   }
  ]
 }
}
```



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

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>
SELECT ?studentID ?studentName
WHERE {
  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject .
          FILTER (?courseSubject = "COMP" && ?courseNumber = "6481")

  ?student rdf:type acad:Student ;
          acad:studentID ?studentID ;
          acad:studentName ?studentName ;
          acad:completedCourse ?courseCompletion .

  ?courseCompletion rdf:type acad:CompletedCourse ;
         acad:hasCourse ?course .
}
```

Table | Response | 2 results in 0.015 seconds

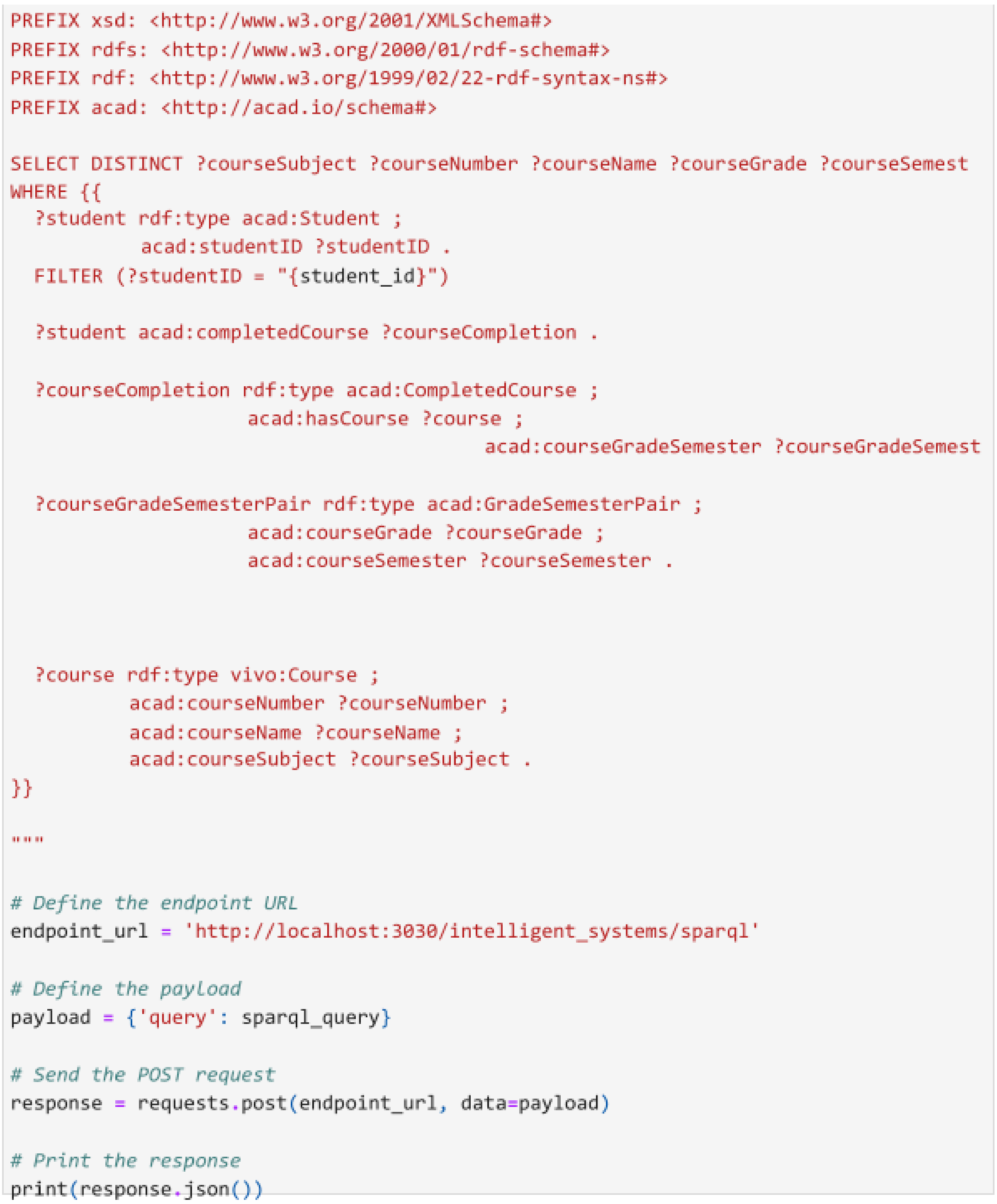
Simple view ✓ Ellipse ✓ Filter query results Page size: 50

studentID	studentName
101	Shrawan Malyala
102	Sraddha Bhattacharjee

13. Print a transcript for a [student], listing all the course taken with their grades

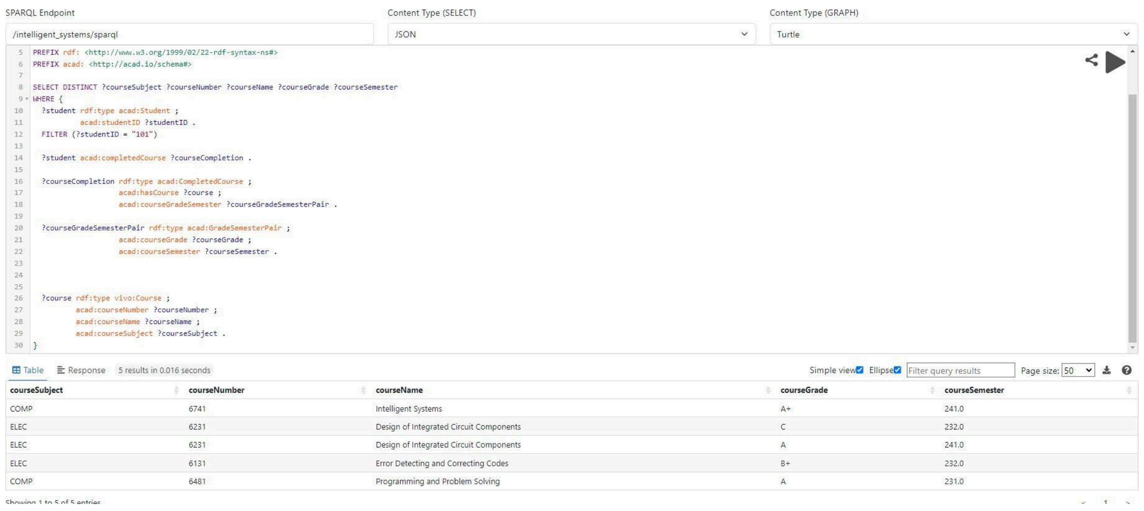
In [14]:

```
# Define the SPARQL query
student_id = "101"

sparql_query = f"""
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```



```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?courseSubject ?courseNumber ?courseName ?courseGrade ?courseSemest
WHERE {{
  ?student rdf:type acad:Student ;
           acad:studentID ?studentID .
  FILTER (?studentID = "{student_id}")

  ?student acad:completedCourse ?courseCompletion .

  ?courseCompletion rdf:type acad:CompletedCourse ;
                    acad:hasCourse ?course ;
                                        acad:courseGradeSemester ?courseGradeSemest

  ?courseGradeSemesterPair rdf:type acad:GradeSemesterPair ;
                    acad:courseGrade ?courseGrade ;
                    acad:courseSemester ?courseSemester .



  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseName ?courseName ;
          acad:courseSubject ?courseSubject .
}}

"""

# Define the endpoint URL
endpoint_url = 'http://localhost:3030/intelligent_systems/sparql'

# Define the payload
payload = {'query': sparql_query}

# Send the POST request
response = requests.post(endpoint_url, data=payload)

# Print the response
print(response.json())
```

```
{'head':  {'vars':  ['courseSubject',  'courseNumber',  'courseName',  'courseGrade',  'co
urseSemester']}, 'results': {'bindings': [{'courseSubject': {'type': 'literal', 'val
ue': 'COMP'}, 'courseNumber': {'type': 'literal', 'value': '6741'}, 'courseName':
{'type':  'literal',  'value':  'Intelligent  Systems'},  'courseGrade':  {'type':  'litera  l',
'value': 'A+'}, 'courseSemester': {'type': 'literal', 'value': '241.0'}}, {'cour
seSubject': {'type': 'literal', 'value': 'ELEC'}, 'courseNumber': {'type': 'litera
l',  'value':  '6231'},  'courseName':  {'type':  'literal',  'value':  'Design  of  Integrat  ed
Circuit  Components'},  'courseGrade':  {'type':  'literal',  'value':  'C'},  'courseSe  mester':
{'type': 'literal', 'value': '232.0'}}, {'courseSubject': {'type': 'litera
l',  'value':  'ELEC'},  'courseNumber':  {'type':  'literal',  'value':  '6231'},  'courseN  ame':
{'type':  'literal',  'value':  'Design  of  Integrated  Circuit  Components'},  'cour  seGrade':
{'type': 'literal', 'value': 'A'}, 'courseSemester': {'type': 'literal',
'value': '241.0'}}, {'courseSubject': {'type': 'literal', 'value': 'ELEC'}, 'courseN
umber':  {'type':  'literal',  'value':  '6131'},  'courseName':  {'type':  'literal',  'val  ue':
'Error  Detecting  and  Correcting  Codes'},  'courseGrade':  {'type':  'literal',  'va  lue':  'B+'},
'courseSemester': {'type': 'literal', 'value': '232.0'}}, {'courseSubje
ct': {'type': 'literal', 'value': 'COMP'}, 'courseNumber': {'type': 'literal', 'valu e':
'6481'}, 'courseName': {'type': 'literal', 'value': 'Programming and Problem Sol ving'},
'courseGrade': {'type': 'literal', 'value': 'A'}, 'courseSemester': {'type': 'literal',
'value': '231.0'}}]}}
```



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?courseSubject ?courseNumber ?courseName ?courseGrade ?courseSemester
WHERE {
  ?student rdf:type acad:Student ;
           acad:studentID ?studentID .
  FILTER (?studentID = "101")

  ?student acad:completedCourse ?courseCompletion .

  ?courseCompletion rdf:type acad:CompletedCourse ;
                    acad:hasCourse ?course ;
                    acad:courseGradeSemester ?courseGradeSemesterPair .

  ?courseGradeSemesterPair rdf:type acad:GradeSemesterPair ;
                    acad:courseGrade ?courseGrade ;
                    acad:courseSemester ?courseSemester .



  ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseName ?courseName ;
          acad:courseSubject ?courseSubject .
}
```

courseSubject	courseNumber	courseName	courseGrade	courseSemester
COMP	6741	Intelligent Systems	A+	241.0
ELEC	6231	Design of Integrated Circuit Components	C	232.0
ELEC	6231	Design of Integrated Circuit Components	A	241.0
ELEC	6131	Error Detecting and Correcting Codes	B+	232.0
COMP	6481	Programming and Problem Solving	A	231.0

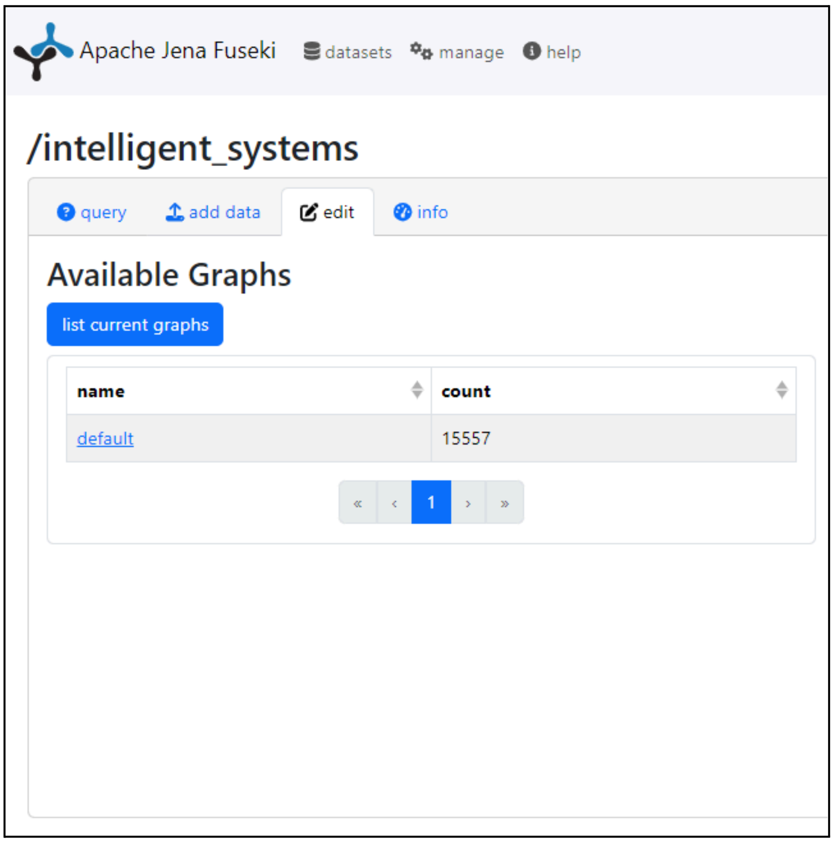
Showing 1 to 5 of 5 entries

Triplestore and SPARQL Endpoint Setup:

- Setting up Fuseki: Downloaded the Apache Fuseki binary distribution and unpacked it. Made the server script executable and launched the server.

- Accessing the Web Interface: Navigated to http://localhost:3030/ in browser to access Fuseki's web interface.

- Uploading Triples: Uploaded triples to the Fuseki server from GraphData.ttl.



name	count
default	15557

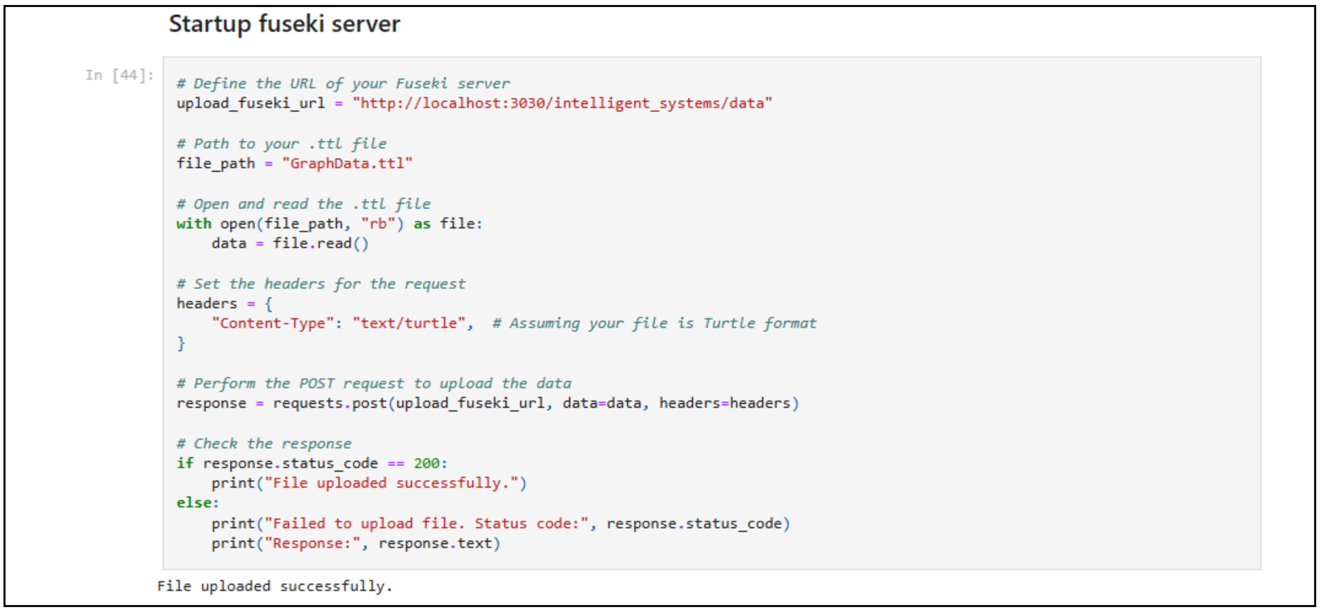
- Sending SPARQL Queries: From the query tab, queried the data using SPARQL queries.

- Used SPARQL Server from Python to receive JSON responses of the corresponding data.

PART 2

1. Updates made from Part 1:

- Created a script to do the setup of the database in Fuseki



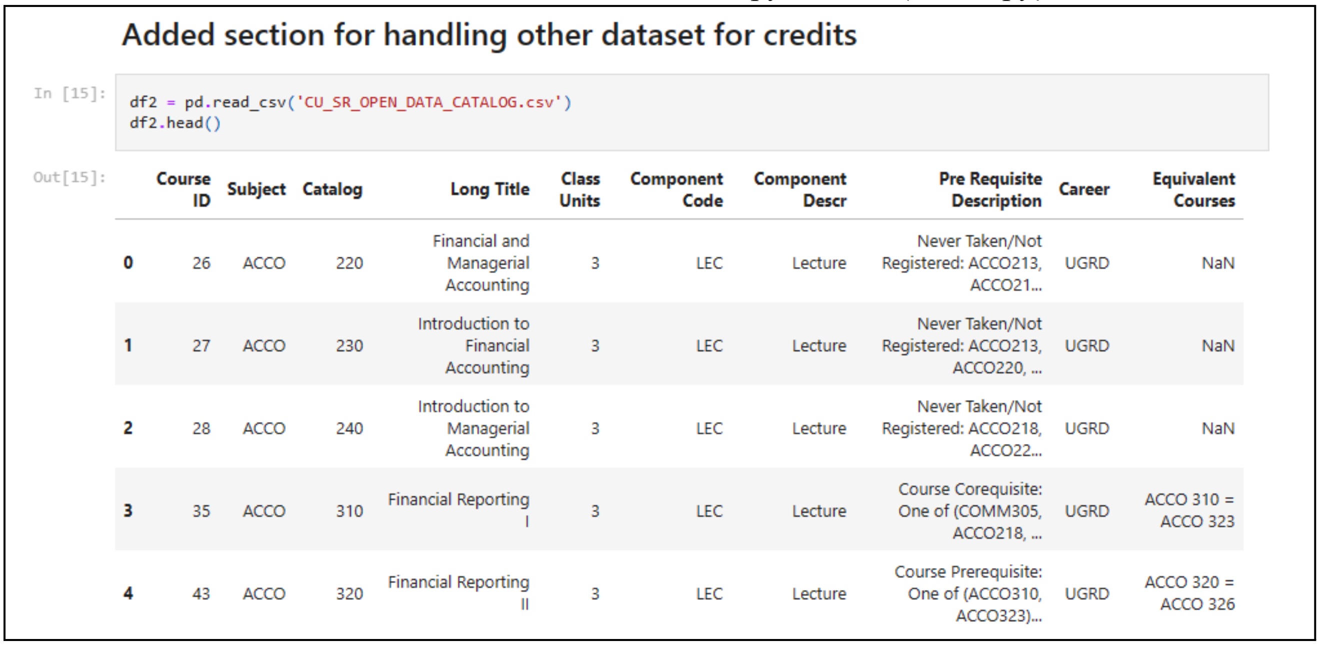
Startup fuseki server

In [44]:

```
# Define the URL of your Fuseki server
upload_fuseki_url = "http://localhost:3030/intelligent_systems/data"

# Path to your .ttl file
file_path = "GraphData.ttl"

# Open and read the .ttl file
with open(file_path, "rb") as file:
    data = file.read()

# Set the headers for the request
headers = {
    "Content-Type": "text/turtle",  # Assuming your file is Turtle format
}

# Perform the POST request to upload the data
response = requests.post(upload_fuseki_url, data=data, headers=headers)

# Check the response
if response.status_code == 200:
    print("File uploaded successfully.")
else:
    print("Failed to upload file. Status code:", response.status_code)
    print("Response:", response.text)
```

```
File uploaded successfully.
```

- Used public data available on https://opendata.concordia.ca/datasets/ to add the real credits for the different courses in the main python file (ABKS.py)



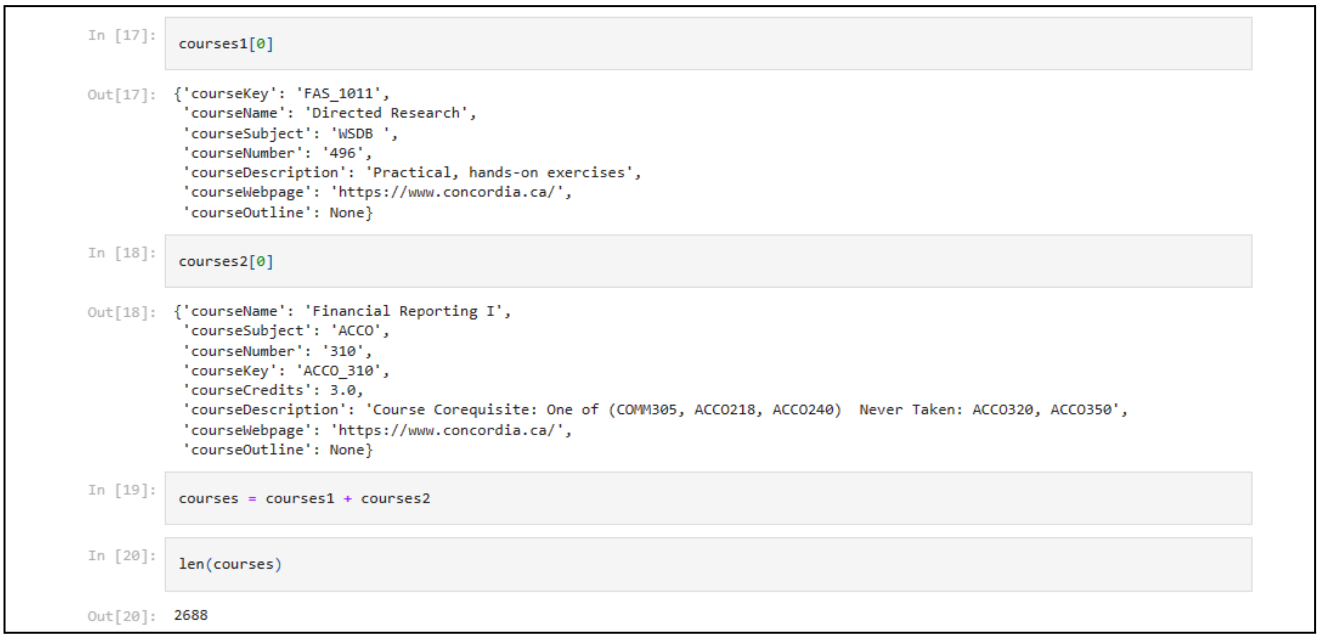
Added section for handling other dataset for credits

In [15]:

```
df2 = pd.read_csv('CU_SR_OPEN_DATA_CATALOG.csv')
df2.head()
```

Out[15]:

	Course ID	Subject	Catalog	Long Title	Class Units	Component Code	Component Descr	Pre Requisite Description	Career	Equivalent Courses
0	26	ACCO	220	Financial and Managerial Accounting	3	LEC	Lecture	Never Taken/Not Registered: ACCO213, ACCO21...	UGRD	NaN
1	27	ACCO	230	Introduction to Financial Accounting	3	LEC	Lecture	Never Taken/Not Registered: ACCO213, ACCO220, ...	UGRD	NaN
2	28	ACCO	240	Introduction to Managerial Accounting	3	LEC	Lecture	Never Taken/Not Registered: ACCO218, ACCO22...	UGRD	NaN
3	35	ACCO	310	Financial Reporting I	3	LEC	Lecture	Course Corequisite: One of (COMM305, ACCO218, ...	UGRD	ACCO 310 = ACCO 323
4	43	ACCO	320	Financial Reporting II	3	LEC	Lecture	Course Prerequisite: One of (ACCO310, ACCO323)...	UGRD	ACCO 320 = ACCO 326



```
In [17]: courses1[0]

Out[17]: {'courseKey': 'FAS_1011',
          'courseName': 'Directed Research',
          'courseSubject': 'WSDB ',
          'courseNumber': '496',
          'courseDescription': 'Practical, hands-on exercises',
          'courseWebpage': 'https://www.concordia.ca/',
          'courseOutline': None}

In [18]: courses2[0]

Out[18]: {'courseName': 'Financial Reporting I',
          'courseSubject': 'ACCO',
          'courseNumber': '310',
          'courseKey': 'ACCO_310',
          'courseCredits': 3.0,
          'courseDescription': 'Course Corequisite: One of (COMM305, ACCO218, ACCO240)  Never Taken: ACCO320, ACCO350',
          'courseWebpage': 'https://www.concordia.ca/',
          'courseOutline': None}

In [19]: courses = courses1 + courses2

In [20]: len(courses)

Out[20]: 2688
```

2. Creating Topic Triples for Knowledge Base

Document Processing: The first step is to collect and process documents relevant to the knowledge base's domain. These documents can include slides, worksheets and other material. We wrote a python script to extract topics for the two selected courses we populated initially. Using a recursive code, we parse each pdf one at a time and text process them using apache tika for tasks such as text parsing, entity recognition, and relationship extraction.

Entity Recognition and Linking: In this step, entities mentioned in the documents were identified and linked to their corresponding entries in LOD. The LOD provides a vast repository of structured data, including information about entities, their properties, and relationships. We used DBpedia Spotlight for entity recognition and linking. These tools map textual mentions of entities to unique identifiers in LOD.

Relationship Extraction: Once entities are linked to LOD, the next step is to extract relationships between these entities. For our case, the relationships are static as they are all related to lectures and the material the lectures were parsed from

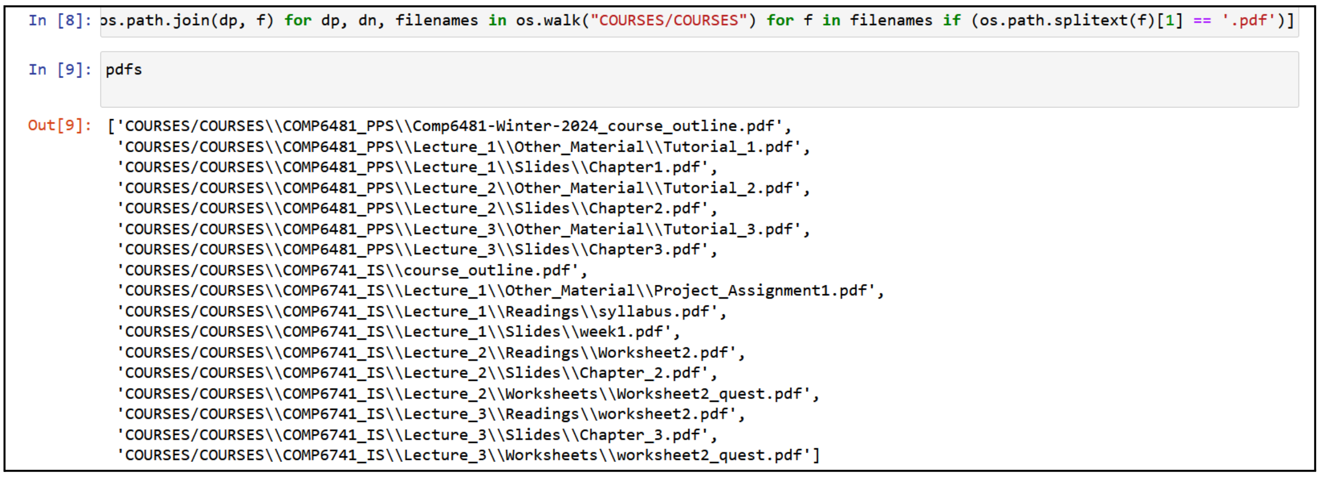
Triple Formation: With entities and relationships identified, the final step is to form triples representing the knowledge extracted from the documents. Triples consist of subject-predicate-object tuples, where the subject and object are entities, and the predicate represents the relationship between them. These triples are typically represented in RDF format, which is a standard for encoding metadata and data on the web. A code snippet was added to parse the extracted topics and link them to the related courses/ lectures

Filtering and Validation: Not all extracted information may be relevant or accurate. Therefore, it's essential to apply filters to ensure the quality of the triples. This can involve removing duplicates, filtering out irrelevant information, and validating the extracted relationships against known knowledge sources or ontologies.

Population of the Knowledge Base: Once the triples are formed and validated, they can be used to populate the knowledge base. The triples serve as the building blocks of the knowledge base, capturing the structured representation of domain-specific knowledge extracted from textual sources. The populated knowledge base can then be queried. A code snippet was added to feed in all the created triples into the knowledge graph.

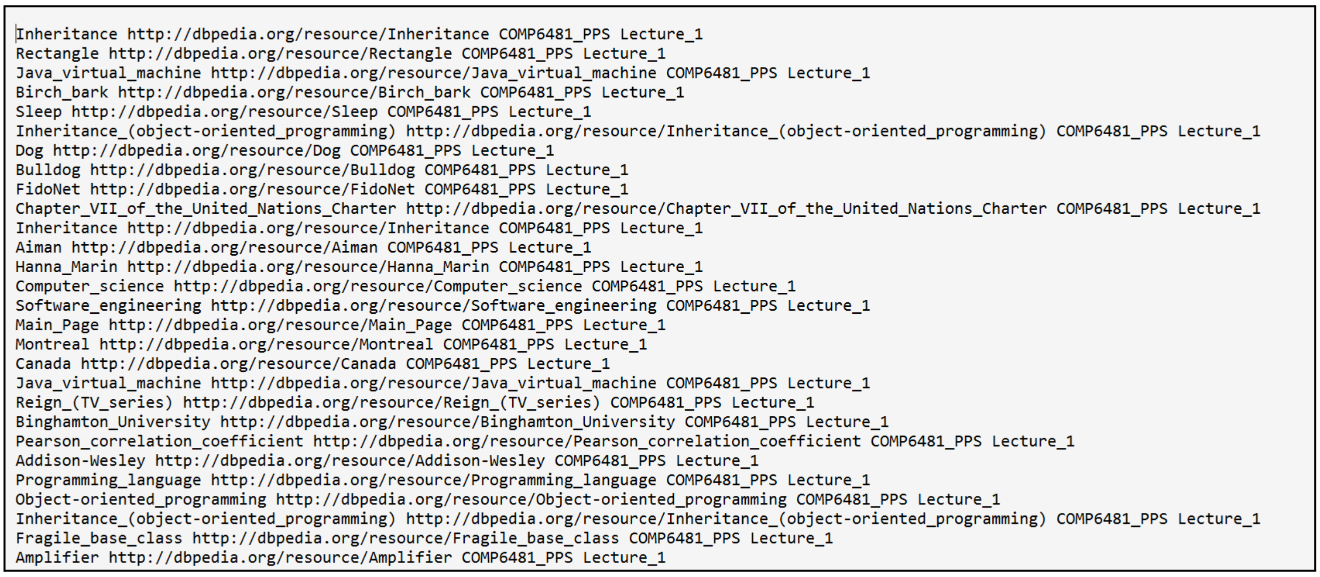
We have written a python script to perform the above steps.

- ❖ Namespaces, Dataset, and a graph known as KG have all been defined. We bound namespaces and created a dataset.
- ❖ Next, we imported the vocabulary.ttl file, which contained the externally defined schema, into the dataset's default (context) graph .
- ❖ To locate and interpret documents, we make use of Apache Tika. The PDFs for every course we have inside the COURSES/COURSES folder are then obtained recursively.



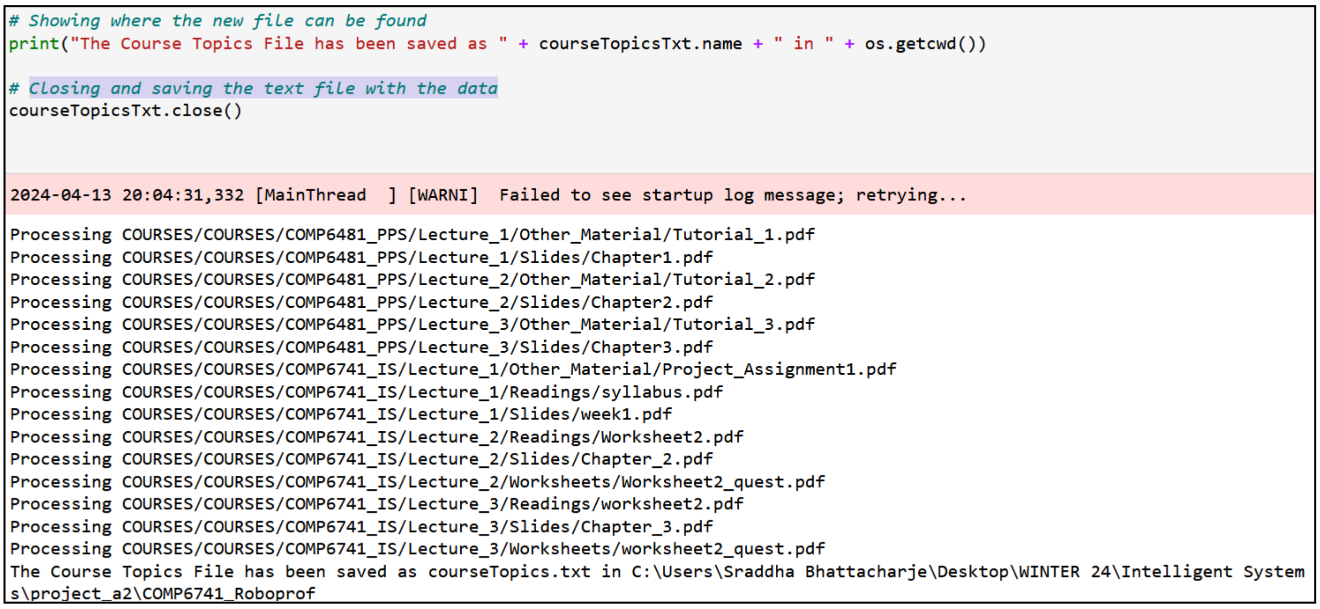
```
In [8]: os.path.join(dp, f) for dp, dn, filenames in os.walk("COURSES/COURSES") for f in filenames if (os.path.splitext(f)[1] == '.pdf')]

In [9]: pdfs

Out[9]: ['COURSES/COURSES\\COMP6481_PPS\\Comp6481-Winter-2024_course_outline.pdf',
         'COURSES/COURSES\\COMP6481_PPS\\Lecture_1\\Other_Material\\Tutorial_1.pdf',
         'COURSES/COURSES\\COMP6481_PPS\\Lecture_1\\Slides\\Chapter1.pdf',
         'COURSES/COURSES\\COMP6481_PPS\\Lecture_2\\Other_Material\\Tutorial_2.pdf',
         'COURSES/COURSES\\COMP6481_PPS\\Lecture_2\\Slides\\Chapter2.pdf',
         'COURSES/COURSES\\COMP6481_PPS\\Lecture_3\\Other_Material\\Tutorial_3.pdf',
         'COURSES/COURSES\\COMP6481_PPS\\Lecture_3\\Slides\\Chapter3.pdf',
         'COURSES/COURSES\\COMP6741_IS\\course_outline.pdf',
         'COURSES/COURSES\\COMP6741_IS\\Lecture_1\\Other_Material\\Project_Assignment1.pdf',
         'COURSES/COURSES\\COMP6741_IS\\Lecture_1\\Readings\\syllabus.pdf',
         'COURSES/COURSES\\COMP6741_IS\\Lecture_1\\Slides\\week1.pdf',
         'COURSES/COURSES\\COMP6741_IS\\Lecture_2\\Readings\\Worksheet2.pdf',
         'COURSES/COURSES\\COMP6741_IS\\Lecture_2\\Slides\\Chapter_2.pdf',
         'COURSES/COURSES\\COMP6741_IS\\Lecture_2\\Worksheets\\Worksheet2_quest.pdf',
         'COURSES/COURSES\\COMP6741_IS\\Lecture_3\\Readings\\worksheet2.pdf',
         'COURSES/COURSES\\COMP6741_IS\\Lecture_3\\Slides\\Chapter_3.pdf',
         'COURSES/COURSES\\COMP6741_IS\\Lecture_3\\Worksheets\\worksheet2_quest.pdf']
```

- ❖ We then created the text file ‘CourseTopicsTxt.txt’ to save the extracted topics.
- ❖ We opened the pdfs and linked their contents to the DBpedia resource using Spotlight, and filtered out the duplicates.
- ❖ In the opened text file we store the following information about each topic:
  - ➢ Topic Name
  - ➢ The DBpedia URI of the topic
  - ➢ The course the topic was extracted from
  - ➢ The lecture material the topic was extracted from
- ❖ “TopicLabel topic_dbpedia_URI PDF_URI Course_Component”. Below is an extract from the populated text file.



```
Inheritance http://dbpedia.org/resource/Inheritance COMP6481_PPS Lecture_1
Rectangle http://dbpedia.org/resource/Rectangle COMP6481_PPS Lecture_1
Java_virtual_machine http://dbpedia.org/resource/Java_virtual_machine COMP6481_PPS Lecture_1
Birch_bark http://dbpedia.org/resource/Birch_bark COMP6481_PPS Lecture_1
Sleep http://dbpedia.org/resource/Sleep COMP6481_PPS Lecture_1
Inheritance_(object-oriented_programming) http://dbpedia.org/resource/Inheritance_(object-oriented_programming) COMP6481_PPS Lecture_1
Dog http://dbpedia.org/resource/Dog COMP6481_PPS Lecture_1
Bulldog http://dbpedia.org/resource/Bulldog COMP6481_PPS Lecture_1
FidoNet http://dbpedia.org/resource/FidoNet COMP6481_PPS Lecture_1
Chapter_VII_of_the_United_Nations_Charter http://dbpedia.org/resource/Chapter_VII_of_the_United_Nations_Charter COMP6481_PPS Lecture_1
Inheritance http://dbpedia.org/resource/Inheritance COMP6481_PPS Lecture_1
Aiman http://dbpedia.org/resource/Aiman COMP6481_PPS Lecture_1
Hanna_Marin http://dbpedia.org/resource/Hanna_Marin COMP6481_PPS Lecture_1
Computer_science http://dbpedia.org/resource/Computer_science COMP6481_PPS Lecture_1
Software_engineering http://dbpedia.org/resource/Software_engineering COMP6481_PPS Lecture_1
Main_Page http://dbpedia.org/resource/Main_Page COMP6481_PPS Lecture_1
Montreal http://dbpedia.org/resource/Montreal COMP6481_PPS Lecture_1
Canada http://dbpedia.org/resource/Canada COMP6481_PPS Lecture_1
Java_virtual_machine http://dbpedia.org/resource/Java_virtual_machine COMP6481_PPS Lecture_1
Reign_(TV_series) http://dbpedia.org/resource/Reign_(TV_series) COMP6481_PPS Lecture_1
Binghamton_University http://dbpedia.org/resource/Binghamton_University COMP6481_PPS Lecture_1
Pearson_correlation_coefficient http://dbpedia.org/resource/Pearson_correlation_coefficient COMP6481_PPS Lecture_1
Addison-Wesley http://dbpedia.org/resource/Addison-Wesley COMP6481_PPS Lecture_1
Programming_language http://dbpedia.org/resource/Programming_language COMP6481_PPS Lecture_1
Object-oriented_programming http://dbpedia.org/resource/Object-oriented_programming COMP6481_PPS Lecture_1
Inheritance_(object-oriented_programming) http://dbpedia.org/resource/Inheritance_(object-oriented_programming) COMP6481_PPS Lecture_1
Fragile_base_class http://dbpedia.org/resource/Fragile_base_class COMP6481_PPS Lecture_1
Amplifier http://dbpedia.org/resource/Amplifier COMP6481_PPS Lecture_1
```

- We showed where the new file can be found and closed and saved the text file with the data.



```
# Showing where the new file can be found
print("The Course Topics File has been saved as " + courseTopicsTxt.name + " in " + os.getcwd())

# Closing and saving the text file with the data
courseTopicsTxt.close()
```

```
2024-04-13 20:04:31,332 [MainThread  ] [WARNI]  Failed to see startup log message; retrying...

Processing COURSES/COURSES/COMP6481_PPS/Lecture_1/Other_Material/Tutorial_1.pdf
Processing COURSES/COURSES/COMP6481_PPS/Lecture_1/Slides/Chapter1.pdf
Processing COURSES/COURSES/COMP6481_PPS/Lecture_2/Other_Material/Tutorial_2.pdf
Processing COURSES/COURSES/COMP6481_PPS/Lecture_2/Slides/Chapter2.pdf
Processing COURSES/COURSES/COMP6481_PPS/Lecture_3/Other_Material/Tutorial_3.pdf
Processing COURSES/COURSES/COMP6481_PPS/Lecture_3/Slides/Chapter3.pdf
Processing COURSES/COURSES/COMP6741_IS/Lecture_1/Other_Material/Project_Assignment1.pdf
Processing COURSES/COURSES/COMP6741_IS/Lecture_1/Readings/syllabus.pdf
Processing COURSES/COURSES/COMP6741_IS/Lecture_1/Slides/week1.pdf
Processing COURSES/COURSES/COMP6741_IS/Lecture_2/Readings/Worksheet2.pdf
Processing COURSES/COURSES/COMP6741_IS/Lecture_2/Slides/Chapter_2.pdf
Processing COURSES/COURSES/COMP6741_IS/Lecture_2/Worksheets/Worksheet2_quest.pdf
Processing COURSES/COURSES/COMP6741_IS/Lecture_3/Readings/worksheet2.pdf
Processing COURSES/COURSES/COMP6741_IS/Lecture_3/Slides/Chapter_3.pdf
Processing COURSES/COURSES/COMP6741_IS/Lecture_3/Worksheets/worksheet2_quest.pdf
The Course Topics File has been saved as courseTopics.txt in C:\Users\Sraddha Bhattacharje\Desktop\WINTER 24\Intelligent System
s\project_a2\COMP6741_Roboprof
```

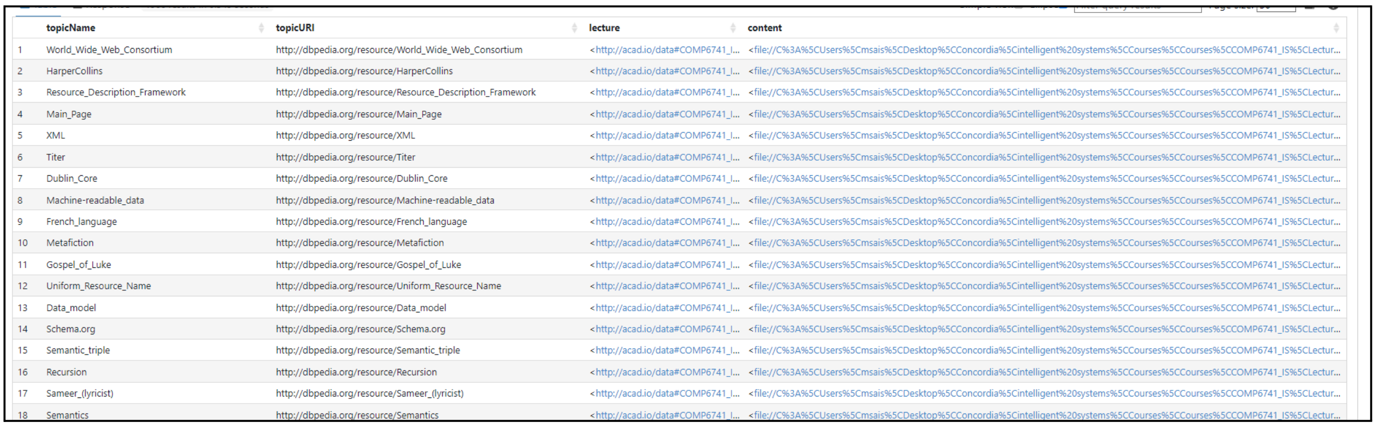
SPARQL queries with result screenshots that can answer given questions:

1. 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.

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT DISTINCT ?topicName ?topicURI ?lecture ?content
WHERE {
   ?course rdf:type vivo:Course ;
          acad:courseNumber ?courseNumber ;
          acad:courseSubject ?courseSubject ;
          acad:hasLecture ?lecture ;
          FILTER(?courseNumber = "6741" && ?courseSubject = "COMP")

   ?lecture rdf:type acad:Lecture ;
           acad:lectureNumber ?lectureNumber ;
           acad:hasContent ?content .

   ?topic rdf:type acad:Topic ;

         acad:hasProvenanceInformation ?lecture ;
```

```
            acad:topicName ?topicName ;
            acad:hasTopicLink ?topicURI .


}
```

Result:



	topicName	topicURI	lecture	content
1	World_Wide_Web_Consortium	http://dbpedia.org/resource/World_Wide_Web_Consortium	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
2	HarperCollins	http://dbpedia.org/resource/HarperCollins	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
3	Resource_Description_Framework	http://dbpedia.org/resource/Resource_Description_Framework	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
4	Main_Page	http://dbpedia.org/resource/Main_Page	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
5	XML	http://dbpedia.org/resource/XML	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
6	Titer	http://dbpedia.org/resource/Titer	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
7	Dublin_Core	http://dbpedia.org/resource/Dublin_Core	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
8	Machine-readable_data	http://dbpedia.org/resource/Machine-readable_data	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
9	French_language	http://dbpedia.org/resource/French_language	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
10	Metafiction	http://dbpedia.org/resource/Metafiction	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
11	Gospel_of_Luke	http://dbpedia.org/resource/Gospel_of_Luke	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
12	Uniform_Resource_Name	http://dbpedia.org/resource/Uniform_Resource_Name	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
13	Data_model	http://dbpedia.org/resource/Data_model	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
14	Schema.org	http://dbpedia.org/resource/Schema.org	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
15	Semantic_triple	http://dbpedia.org/resource/Semantic_triple	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
16	Recursion	http://dbpedia.org/resource/Recursion	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
17	Sameer_(lyricist)	http://dbpedia.org/resource/Sameer_(lyricist)	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...
18	Semantics	http://dbpedia.org/resource/Semantics	<http://acad.io/data#COMP6741_l...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLectur...

2. 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

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>

SELECT ?course ?lecture (COUNT(?topic) AS ?occurrences)
WHERE {
    ?course rdf:type vivo:Course ;
            acad:hasLecture ?lecture .
    ?lecture rdf:type acad:Lecture ;
             acad:hasContent ?content .
    ?topic rdf:type acad:Topic ;
           acad:hasProvenanceInformation ?lecture ;
           acad:topicName ?topicName .
    FILTER(?topicName = "Compiler")
}
GROUP BY ?course ?lecture
ORDER BY DESC(?occurrences)
```

Result:

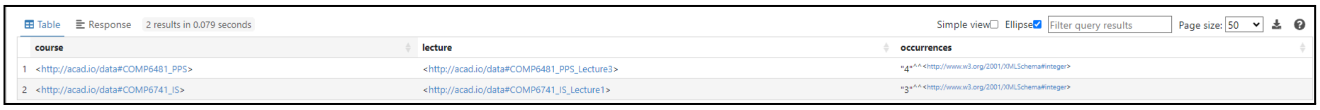


Table   Response   2 results in 0.079 seconds   Simple view☐ Ellipse☑ Filter query results Page size: 50

	course	lecture	occurrences
1	<http://acad.io/data#COMP6481_PPS>	<http://acad.io/data#COMP6481_PPS_Lecture3>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
2	<http://acad.io/data#COMP6741_IS>	<http://acad.io/data#COMP6741_IS_Lecture1>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>

3. 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).

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acad: <http://acad.io/schema#>
SELECT ?courseSubject ?courseNumber ?lectureNumber ?course ?lecture ?content
   WHERE {

   ?course rdf:type vivo:Course ;
            acad:courseNumber ?courseNumber ;
            acad:courseSubject ?courseSubject ;
            acad:hasLecture ?lecture.
   ?lecture rdf:type acad:Lecture ;
         acad:lectureNumber ?lectureNumber ;
         acad:hasContent ?content .
   ?topic rdf:type acad:Topic ;
             acad:hasProvenanceInformation ?lecture ;
             acad:topicName ?topicName .
FILTER(?topicName = "PHP")
}
```

Result:

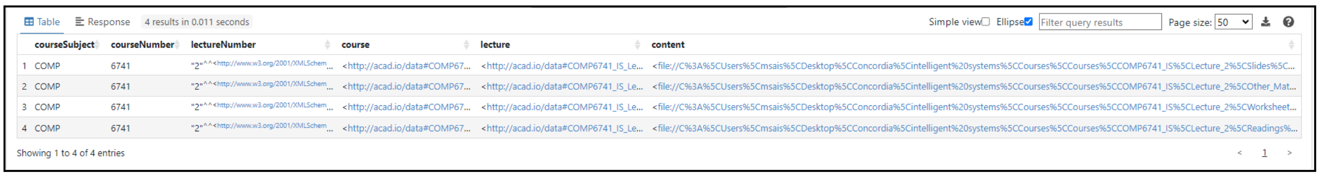


Table   Response   4 results in 0.011 seconds   Simple view☐ Ellipse☑ Filter query results Page size: 50

	courseSubject	courseNumber	lectureNumber	course	lecture	content
1	COMP	6741	"2"^^<http://www.w3.org/2001/XMLSchem...	<http://acad.io/data#COMP67...	<http://acad.io/data#COMP6741_IS_Le...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_2%5CSlides%5C...
2	COMP	6741	"2"^^<http://www.w3.org/2001/XMLSchem...	<http://acad.io/data#COMP67...	<http://acad.io/data#COMP6741_IS_Le...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_2%5COther_Mat...
3	COMP	6741	"2"^^<http://www.w3.org/2001/XMLSchem...	<http://acad.io/data#COMP67...	<http://acad.io/data#COMP6741_IS_Le...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_2%5CWorksheet...
4	COMP	6741	"2"^^<http://www.w3.org/2001/XMLSchem...	<http://acad.io/data#COMP67...	<http://acad.io/data#COMP6741_IS_Le...	<file://C%3A%5CUsers%5Cmsais%5CDesktop%5CConcordia%5Cintelligent%20systems%5CCourses%5CCourses%5CCOMP6741_IS%5CLecture_2%5CReadings%...

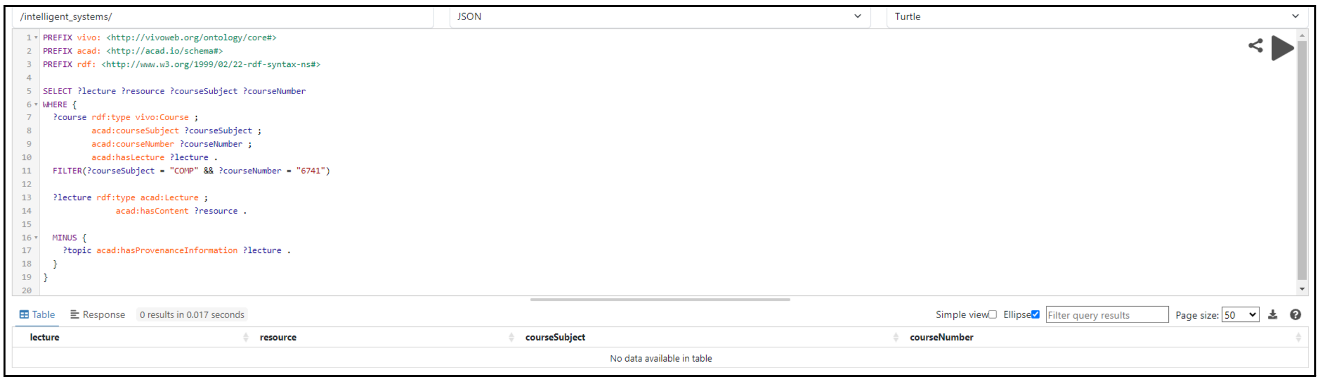
Showing 1 to 4 of 4 entries   < 1 >

4. Write a SPARQL query to identify any course events or resources within a specific course c that do not have any associated topic entities. This query helps in verifying that all relevant educational materials have been adequately linked to topics in the knowledge base, ensuring completeness and providing insights into potential areas for improvement.

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX acad: <http://acad.io/schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?lecture ?resource ?courseSubject ?courseNumber
WHERE {
 ?course rdf:type vivo:Course ;
        acad:courseSubject ?courseSubject ;
        acad:courseNumber ?courseNumber ;
```

```
                    acad:hasLecture ?lecture .
  FILTER(?courseSubject = "COMP" && ?courseNumber = "6741")

  ?lecture rdf:type acad:Lecture ;
           acad:hasContent ?resource .

  MINUS {
    ?topic acad:hasProvenanceInformation ?lecture .
  }
}
```

Result:



/intelligent_systems/ | JSON | Turtle

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX acad: <http://acad.io/schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?lecture ?resource ?courseSubject ?courseNumber
WHERE {
  ?course rdf:type vivo:Course ;
          acad:courseSubject ?courseSubject ;
          acad:courseNumber ?courseNumber ;
          acad:hasLecture ?lecture .
  FILTER(?courseSubject = "COMP" && ?courseNumber = "6741")

  ?lecture rdf:type acad:Lecture ;
           acad:hasContent ?resource .

  MINUS {
    ?topic acad:hasProvenanceInformation ?lecture .
  }
}
```

Table | Response | 0 results in 0.017 seconds

Simple view ☐ Ellipse ☑ Filter query results Page size: 50

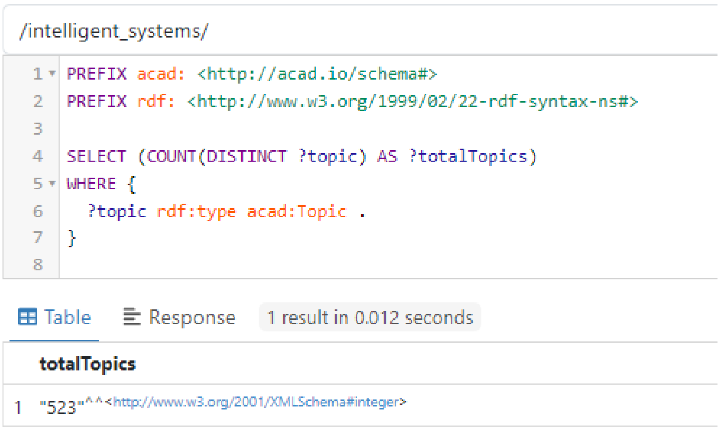
lecture	resource	courseSubject	courseNumber
No data available in table			

Statistics of the Knowledge Graph

S No	Item	Statistics
1.	Total number of triples	23473
2.	Total number of distinct topics	523
3.	Total number of distinct courses	2464
4.	Total number of resource files	23

Queries used for populating above table are

1. Topic count



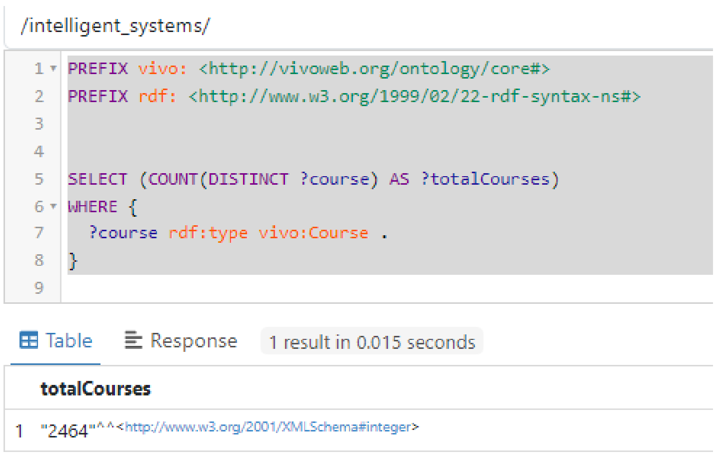
/intelligent_systems/

```
PREFIX acad: <http://acad.io/schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT (COUNT(DISTINCT ?topic) AS ?totalTopics)
WHERE {
  ?topic rdf:type acad:Topic .
}
```

Table | Response | 1 result in 0.012 seconds

	totalTopics
1	"523"^^<http://www.w3.org/2001/XMLSchema#integer>

2. Course count



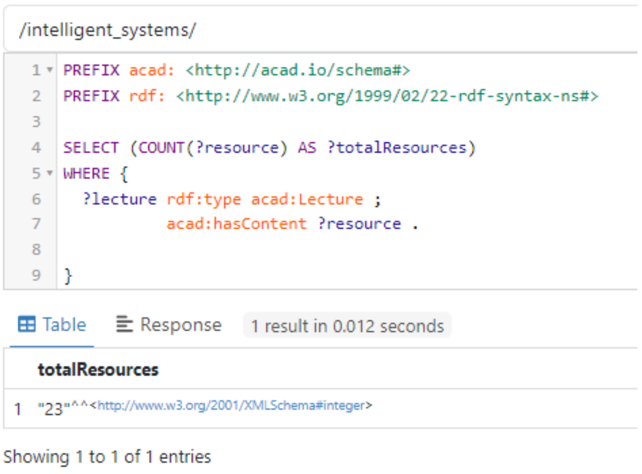
/intelligent_systems/

```
PREFIX vivo: <http://vivoweb.org/ontology/core#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>


SELECT (COUNT(DISTINCT ?course) AS ?totalCourses)
WHERE {
  ?course rdf:type vivo:Course .
}

```

Table | Response | 1 result in 0.015 seconds

	totalCourses
1	"2464"^^<http://www.w3.org/2001/XMLSchema#integer>

```
3. Resource Files
```



/intelligent_systems/

```
PREFIX acad: <http://acad.io/schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT (COUNT(?resource) AS ?totalResources)
WHERE {
  ?lecture rdf:type acad:Lecture ;
           acad:hasContent ?resource .

}
```

Table | Response | 1 result in 0.012 seconds

	totalResources
1	"23"^^<http://www.w3.org/2001/XMLSchema#integer>

Showing 1 to 1 of 1 entries

3. Chatbot Design

RoboProf Chatbot

This section details the design and implementation of a Rasa chatbot aimed at answering educational inquiries. The system leverages a Fuseki server loaded with a knowledge graph (KG) to provide information about courses, students, and universities.

Intents and Entities

The Rasa core utilizes various intents to categorize user queries. These intents include:

```
- greet
- goodbye
- affirm
- deny
- bot_challenge
- about_course
```

```
  - about_course_lecture_topics
  - about_course_topic
  - courses_offeredby_uni
  - topics_coveredby_course_in_lecturenumber
  - courses_offeredby_uni_within_course
  - materials_recommendation_for_topic_in_coursenumber
  - credits_for_course
  - additional_resources_for_course
  - content_for_course_in_lecturenumber
  - competencies_after_course_completion
  - grades_for_student_in_course
  - students_completed_course
  - transcript_for_student
```

Additionally, the system employs entities to extract specific details from user queries. **These entities encompass:**

```
courseSubject
courseNumber
course
lectureNumber
topic
university
studentID
```

The defined entities precisely match the information required to answer all potential queries.

Stories and Rules

A set of stories and rules govern the conversation flow within the Rasa core. Here's a breakdown of some key interactions:

- **Greetings:** Upon user greetings, the bot responds with a welcome message using the utter_greet action.
- **Goodbyes:** When the user bids farewell, the utter_goodbye action is triggered, signifying the end of the conversation.
- **Bot Challenges**: If the user inquires about the bot's identity (e.g., "Are you a bot?"), the utter_iamabot action clarifies that it is indeed a chatbot.
- **Out-of-Scope and NLU Fallback**: The utter_out_of_scope action handles out-of-scope questions or queries with low NLU confidence, prompting the user to rephrase.
- **Educational Inquiries**: Specific stories are defined for various educational inquiries, mapping each intent to a corresponding action (e.g., action_about_course for about_course intent). These actions retrieve information from the knowledge graph.

```
rules:
```

```yaml
- rule: Say goodbye anytime the user says goodbye
  steps:
  - intent: goodbye
  - action: utter_goodbye

- rule: Say 'I am a bot' anytime the user challenges
  steps:
  - intent: bot_challenge
  - action: utter_iamabot

- rule: Greet the user anytime they say hello
  steps:
  - intent: greet
  - action: utter_greet

- rule: out-of-scope
  steps:
  - intent: out_of_scope
  - action: utter_out_of_scope

- rule: Ask the user to rephrase whenever they send a message with
low NLU confidence
  steps:
  - intent: nlu_fallback
  - action: utter_out_of_scope

stories:

- story: q2_1_about_course
  steps:
    - intent: about_course
    - action: action_about_course

- story: q2_2_about_course_lecture_topics
  steps:
    - intent: about_course_lecture_topics
    - action: action_about_course_lecture_topics

- story: q2_3_about_course_topic
  steps:
    - intent: about_course_topic
    - action: action_about_course_topic
- story: q1_1_courses_offeredby_uni
  steps:
    - intent: courses_offeredby_uni
    - action: action_courses_offeredby_uni

- story: q1_3_topics_coveredby_course_in_lecturenumber
  steps:
    - intent: topics_coveredby_course_in_lecturenumber
    - action: action_topics_coveredby_course_in_lecturenumber

- story: q1_4_courses_offeredby_uni_within_course
  steps:
    - intent: courses_offeredby_uni_within_course
    - action: action_courses_offeredby_uni_within_course

- story: q1_5_materials_recommendation_for_topic_in_coursenumber
  steps:
    - intent: materials_recommendation_for_topic_in_coursenumber
    - action:
action_materials_recommendation_for_topic_in_coursenumber

- story: q1_6_credits_for_course
  steps:
    - intent: credits_for_course
    - action: action_credits_for_course

- story: q1_7_additional_resources_for_course
  steps:
    - intent: additional_resources_for_course
    - action: action_additional_resources_for_course

- story: q1_8_content_for_course_in_lecturenumber
  steps:
    - intent: content_for_course_in_lecturenumber
    - action: action_content_for_course_in_lecturenumber

- story: q1_10_competencies_after_course_completion
  steps:
    - intent: competencies_after_course_completion
    - action: action_competencies_after_course_completion

- story: q1_11_grades_for_student_in_course
  steps:
  - intent: grades_for_student_in_course
  - action: action_grades_for_student_in_course

- story: q1_12_students_completed_course
  steps:
  - intent: students_completed_course
  - action: action_students_completed_course

- story: q1_13_transcript_for_student
  steps:
  - intent: transcript_for_student
  - action: action_transcript_for_student
```

Actions and Integration with Fuseki Server

For each inquiry, a unique action is implemented.
Here is the actions.py file:

```
import requests
import json
import re

from rdflib import Graph, Literal, RDF, URIRef, Namespace, Dataset

from typing import Any, Text, Dict, List

from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
from langchain.llms import OpenAI
from langchain import PromptTemplate

def rewrite_with_llm(preface_text, answers):
    llm = OpenAI(model_name="gpt-3.5-turbo-instruct",
openai_api_key='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
')
    result_found = True
    if isinstance(answers, list):
        if len(answers) == 0:
            result_found = False
    if isinstance(answers, str ):
        if len(answers) == 0:
            result_found = False
    if not result_found:
        template = """
        description_of_answers: {preface_text}
        Looking at what the description_of_answers is saying,
generate a message saying you could not find the requested
information
        """

        prompt = PromptTemplate(

        input_variables=["preface_text"],

        template=template,
        )

        final_prompt = prompt.format(preface_text=preface_text,)
        return llm(final_prompt)

    try:
        template = """
        description_of_answers: {preface_text}
        answers = {answers}
        Based on the description_of_answers, formulate the answers
for the user to read and understand in a human-readable manner. Do
not add any new information. Make sure you convey the whole of the
answers. If answers is long, consider using a list.
        If there is a path in the answers, make sure to include the
full paths in your response
        If answers is empty, then make a message that says you are
sorry and could not find what was asked in the
description_of_answers. Do not give any more information
        """

        prompt = PromptTemplate(

        input_variables=["preface_text", "answers"],

        template=template,
        )

        final_prompt = prompt.format(preface_text=preface_text,
answers = answers )

        return llm(final_prompt)
    except Exception as e:
        if isinstance(answers, list):
            return_message = f"{preface_text} \n"
            for i in answers:
                return_message += str(i) + "\n"
        else:
            return_message = f"{preface_text} \n {answers}"
        return return_message



def make_fuseki_server_request(sparql_query):
    # Define the endpoint URL
    endpoint_url =
'http://localhost:3030/intelligent_systems/sparql'

    # Define the payload
    payload = {'query': sparql_query}

    # Send the POST request
    response = requests.post(endpoint_url, data=payload)
    return response


# Q1) List all courses offered by the [university]
class CoursesOfferedbyUni(Action):

    def name(self) -> Text:
        return "action_courses_offeredby_uni"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        university = tracker.slots['university']

        if "concordia" in university.lower() or "university" in
university.lower():
            university = "Concordia University"

        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>
        SELECT ?courseName ?courseSubject ?courseNumber
            WHERE {{

            ?university rdf:type acad:University ;
                    acad:universityName ?universityName ;
                    acad:offers ?course .
                    FILTER (?universityName = "{university}")

            ?course rdf:type vivo:Course ;
                    acad:courseName ?courseName ;
                    acad:courseSubject ?courseSubject ;
                    acad:courseNumber ?courseNumber .
        }}
        """

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)

        y = y['results']['bindings']

        course_info = []
        for course in y:
            course_name = course['courseName']['value']
            course_subject = course['courseSubject']['value']
            course_number = course['courseNumber']['value']
            course_info.append([course_name, course_subject,
course_number])

        response = rewrite_with_llm(f"Here are the courses offered
by {university}: \n", course_info)

        dispatcher.utter_message(response)

# Q3) Which [topics] are covered in [course] during [lecture
number]?
class TopicsCoveredByCourseInLecture(Action):

    def name(self) -> Text:
        return "action_topics_coveredby_course_in_lecturenumber"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
```

```
        course_name = tracker.slots['course']

        lecture_number = tracker.slots['lectureNumber']
        print(lecture_number)
        lecture_number = re.search(r'\d+', lecture_number).group()
        print(lecture_number)
        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>
        SELECT DISTINCT ?topicName
            WHERE {{

            ?course rdf:type vivo:Course ;
                    acad:courseName "{course_name}" ;
                    acad:hasLecture ?lecture.
            ?lecture rdf:type acad:Lecture ;
                    acad:lectureNumber ?lectureNumber .
                    FILTER (?lectureNumber = {lecture_number})
            ?topic rdf:type acad:Topic ;
                    acad:hasProvenanceInformation ?lecture ;
                    acad:topicName ?topicName
        }}
        """

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)

        y = y['results']['bindings']

        topic_info = []
        for topic in y:
            topic_name = topic['topicName']['value']
            topic_info.append(topic_name)

        response = rewrite_with_llm(f"The topics covered in
{course_name} during lecture {lecture_number} are: \n", topic_info)
        dispatcher.utter_message(response)
```

```
# Q4) List all [courses] offered by [university] within the
[subject] (e.g., \COMP", \SOEN").
class CoursesOfferedbyUniWithinCourse(Action):

    def name(self) -> Text:
        return "action_courses_offeredby_uni_within_course"
    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        university_name = tracker.slots['university']

        if "concordia" in university_name.lower() or "university"
in university_name.lower():
            university = "Concordia University"

        course_subject1 = tracker.slots['courseSubject']


        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>
        SELECT ?courseName ?courseSubject ?courseNumber
            WHERE {{

            ?university rdf:type acad:University ;
                    acad:universityName ?universityName ;
                    acad:offers ?course .
                    FILTER (?universityName = "{university_name}")

            ?course rdf:type vivo:Course ;
                    acad:courseName ?courseName ;
                    acad:courseNumber ?courseNumber ;
                    acad:courseSubject ?courseSubject ;
                    FILTER (?courseSubject = "{course_subject1}")
        }}
        """

        response = make_fuseki_server_request(sparql_query)
```

```python
        y = json.loads(response.text)

        y = y['results']['bindings']

        course_info = []
        for course in y:
            course_name = course['courseName']['value']
            course_subject = course['courseSubject']['value']
            course_number = course['courseNumber']['value']
            course_info.append([course_name, course_subject,
course_number])

        response = rewrite_with_llm(f"Here are the courses covered
in {university_name} for the subject {course_subject1}: \n",
course_info)

        dispatcher.utter_message(response)

# Q5) What [materials] (slides, readings) are recommended for
[topic] in [course] [number]?
class MaterialsRecommendedForTopicInCourse(Action):

    def name(self) -> Text:
        return
"action_materials_recommendation_for_topic_in_coursenumber"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        course_subject = tracker.slots['courseSubject']
        topic_name = tracker.slots['topic']
        course_number = tracker.slots['courseNumber']

        sparql_query = f"""
        PREFIX ac: <http://umbel.org/umbel/ac/>
        PREFIX prefix: <http://prefix.cc/>
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>
        SELECT DISTINCT ?content ?class
        WHERE {{
        ?course rdf:type vivo:Course ;
                acad:courseNumber ?courseNumber ;
                acad:courseSubject ?courseSubject .
                FILTER (?courseSubject = "{course_subject}" &&
?courseNumber = "{course_number}")

        ?lecture rdf:type acad:Lecture ;
                acad:hasContent ?content .

        ?content a ?class .
        FILTER (?class = acad:Slides || ?class = acad:Reading)
        ?topic rdf:type acad:Topic ;
                acad:topicName "{topic_name}" ;
                acad:hasProvenanceInformation ?lecture .
        }}
        """

        response = make_fuseki_server_request(sparql_query)
        material_info = []

        y = json.loads(response.text)
        y = y['results']['bindings']
        material_info = []
        for item in y:
            content = item['content']['value']
            material_class = item['class']['value'].split('#')[1]
            material_info.append((content, material_class))

        response = rewrite_with_llm(f"The materials recommended for
{topic_name} in {course_subject} {course_number} are: \n",
material_info)

        dispatcher.utter_message(response)

# Q6) How many credits is [course] [number] worth?
class CreditsWorthOfCourse(Action):

    def name(self) -> Text:
        return "action_credits_for_course"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        course_subject = tracker.slots['courseSubject']
        course_number = tracker.slots['courseNumber']

        #print(course_subject, course_number)

        sparql_query = f"""
        PREFIX ac: <http://umbel.org/umbel/ac/>
        PREFIX prefix: <http://prefix.cc/>
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>

        SELECT DISTINCT ?courseCredits
        WHERE {{
        ?course rdf:type vivo:Course ;
                acad:courseNumber ?courseNumber ;
                acad:courseSubject ?courseSubject ;
                acad:courseCredits ?courseCredits
                FILTER (?courseSubject = "{course_subject}" &&
?courseNumber = "{course_number}")
        }}
        """

        print(sparql_query)

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)

        y = y['results']['bindings']

        course_credits = []
        for course in y:
            course_credits = course['courseCredits']['value']

        print(y)
        print(course_credits)

        response = rewrite_with_llm(f"The number of credits awarded
for completing {course_subject} {course_number} is: \n",
course_credits)

        dispatcher.utter_message(response)

# Q7) For [course] [number], what additional resources (links to
web pages) are available
class AdditionalResourcesForCourse(Action):

    def name(self) -> Text:
        return "action_additional_resources_for_course"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        course_subject = tracker.slots['courseSubject']
        course_number = tracker.slots['courseNumber']

        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>

        SELECT DISTINCT ?courseWebpage ?lectureLink ?topicLink
        WHERE {{
        ?course rdf:type vivo:Course ;
                acad:courseNumber ?courseNumber ;
                acad:courseSubject ?courseSubject ;
                acad:courseWebpage ?courseWebpage ;
                acad:coversTopic ?topic ;
                acad:hasLecture ?lecture .
                FILTER (?courseSubject = "{course_subject}" &&
?courseNumber = "{course_number}")
        ?lecture rdf:type acad:Lecture ;
                acad:lectureLink ?lectureLink .
        ?topic rdf:type acad:Topic ;
                acad:hasTopicLink ?topicLink

        }}
        """
        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)

        y = y['results']['bindings']

        additional_resources = []
        for item in y:
            course_webpage = item['courseWebpage']['value']
            lecture_link = item['lectureLink']['value']
            topic_link = item['topicLink']['value']
            additional_resources.append(course_webpage)
            additional_resources.append(lecture_link)
            additional_resources.append(topic_link)

        # Remove duplicates from the list
        additional_resources = list(set(additional_resources))

        response = rewrite_with_llm(f"Additional resources for
{course_subject} {course_number} are: \n", additional_resources)

        dispatcher.utter_message(response)

# Q8) Detail the content (slides, worksheets, readings) available
for [lecture number] in [course] [number].
class ContentAvailableForLectureInCourse(Action):

    def name(self) -> Text:
        return "action_content_for_course_in_lecturenumber"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        course_subject = tracker.slots['courseSubject']
        course_number = tracker.slots['courseNumber']
        lecture_number = tracker.slots['lectureNumber']
        lecture_number = re.search(r'\d+', lecture_number).group()

        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>

        SELECT DISTINCT ?content
        WHERE {{
        ?course rdf:type vivo:Course ;
                acad:courseNumber ?courseNumber ;
                acad:courseSubject ?courseSubject .
                FILTER (?courseSubject = "{course_subject}" &&
?courseNumber = "{course_number}")
        ?lecture rdf:type acad:Lecture ;
                acad:hasContent ?content ;
                acad:lectureNumber ?lectureNumber .
                FILTER (?lectureNumber = {lecture_number})
            ?content a ?class .
        FILTER (?class = acad:Slides || ?class = acad:Reading ||
?class = acad:Worksheet)
        }}
        """

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)
        content_info = [item['content']['value'] for item in
y['results']['bindings']]

        response = rewrite_with_llm(f"The content available for
lecture {lecture_number} in {course_subject} {course_number} is:
\n", content_info)

        dispatcher.utter_message(response)

# Q10) What competencies [topics] does a student gain after
completing [course] [number]?
class CompetenciesGainedForCourse(Action):

    def name(self) -> Text:
        return "action_competencies_after_course_completion"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        course_subject = tracker.slots['courseSubject']
        course_number = tracker.slots['courseNumber']
        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>

        SELECT DISTINCT ?topicName
        WHERE {{
        ?course rdf:type vivo:Course ;
                acad:courseNumber ?courseNumber ;
                acad:courseSubject ?courseSubject .
                FILTER (?courseSubject = "{course_subject}" &&
?courseNumber = "{course_number}")

        ?topic rdf:type acad:Topic ;
                acad:topicName ?topicName .
        }}
        """

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)

        y = y['results']['bindings']

        competency_info = []
        for competency in y:
            competency_name = competency['topicName']['value']
            competency_info.append(competency_name)

        response = rewrite_with_llm(f"The competencies gained after
completing {course_subject} {course_number} are: \n",
competency_info)

        dispatcher.utter_message(response)

# Q11) What grades did [student] achieve in [course] [number]?
class GradesAchievedForStudentInCourse(Action):

    def name(self) -> Text:
        return "action_grades_for_student_in_course"
    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        student_id = tracker.slots['studentID']
        course_subject = tracker.slots['courseSubject']
        course_number = tracker.slots['courseNumber']

        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>

        SELECT ?courseGrade
        WHERE {{
        ?course rdf:type vivo:Course ;
                acad:courseNumber ?courseNumber ;
                acad:courseSubject ?courseSubject .
                FILTER (?courseSubject = "{course_subject}" &&
?courseNumber = "{course_number}")

        ?student rdf:type acad:Student ;
                acad:studentID ?studentID ;
                acad:completedCourse ?courseCompletion .
                FILTER (?studentID = "{student_id}")

        ?courseCompletion rdf:type acad:CompletedCourse ;
                acad:hasCourse ?course ;
                acad:courseGradeSemester ?courseGradeSemesterPair .

        ?courseGradeSemesterPair rdf:type acad:GradeSemesterPair ;
                            acad:courseGrade ?courseGrade .
        }}

        """

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)
        y = y['results']['bindings']

        grades_info = []
        for grade in y:
            grade_value = grade['courseGrade']['value']
            grades_info.append(grade_value)

        response = rewrite_with_llm(f"The grades achieved by
{student_id} in {course_subject} {course_number} are: \n",
grades_info)

        dispatcher.utter_message(response)

# Q12) Which [students] have completed [course] [number]?
class StudentsCompletedCourse(Action):

    def name(self) -> Text:
        return "action_students_completed_course"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        course_subject = tracker.slots['courseSubject']
        course_number = tracker.slots['courseNumber']

        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>
        SELECT ?studentID ?studentName
        WHERE {{
        ?course rdf:type vivo:Course ;
                acad:courseNumber ?courseNumber ;
                acad:courseSubject ?courseSubject .
                FILTER (?courseSubject = "{course_subject}" &&
?courseNumber = "{course_number}")

        ?student rdf:type acad:Student ;
                acad:studentID ?studentID ;
                acad:studentName ?studentName ;
                acad:completedCourse ?courseCompletion .
        ?courseCompletion rdf:type acad:CompletedCourse ;
                acad:hasCourse ?course .
        }}

        """

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)

        y = y['results']['bindings']

        students_info = []
        for student in y:
            students_info.append([student['studentID']['value'],
student['studentName']['value']])

        response = rewrite_with_llm(f"The students who have
completed {course_subject} {course_number} are: \n", students_info)

        dispatcher.utter_message(response)

# Q13) Print a transcript for a [student], listing all the course
taken with their grades
class TranscriptForStudent(Action):

    def name(self) -> Text:
        return "action_transcript_for_student"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        student_id = tracker.slots['studentID']

        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>
        SELECT DISTINCT ?courseSubject ?courseNumber ?courseName
?courseGrade ?courseSemester
        WHERE {{
        ?student rdf:type acad:Student ;
                acad:studentID ?studentID .
        FILTER (?studentID = "{student_id}")

        ?student acad:completedCourse ?courseCompletion .

        ?courseCompletion rdf:type acad:CompletedCourse ;
                            acad:hasCourse ?course ;
                            acad:courseGradeSemester
?courseGradeSemesterPair .

        ?courseGradeSemesterPair rdf:type acad:GradeSemesterPair ;
                            acad:courseGrade ?courseGrade ;
                            acad:courseSemester ?courseSemester .



        ?course rdf:type vivo:Course ;
                acad:courseNumber ?courseNumber ;
                acad:courseName ?courseName ;
                acad:courseSubject ?courseSubject .
        }}

        """

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)

        y = y['results']['bindings']

        transcript_info = []
        for course in y:
            for course in y:
                course_subject = course['courseSubject']['value']
                course_number = course['courseNumber']['value']
                course_grade = course['courseGrade']['value']
                course_semester = "Course taken in Semester: " +
course['courseSemester']['value']
                transcript_info.append([course_subject,
course_number, course_grade, course_semester])
```

```
        response = rewrite_with_llm(f"The transcript for
{student_id} is: \n", transcript_info)

        dispatcher.utter_message(response)

# Q2-1) What is the <course> about?
class CourseDescription(Action):

    def name(self) -> Text:
        return "action_about_course"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        course_subject = tracker.slots['courseSubject']
        course_number = tracker.slots['courseNumber']


        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>
        SELECT ?courseDescription
            WHERE {{

            ?course rdf:type vivo:Course ;
                        acad:courseNumber ?courseNumber ;
                        acad:courseSubject ?courseSubject ;
                        acad:courseDescription ?courseDescription
                        FILTER (?courseSubject = "{course_subject}"
&& ?courseNumber = "{course_number}")
        }}
        """

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)

        y = y['results']['bindings']
        course_description = []
        for course in y:
            course_description = 
course['courseDescription']['value']

        response = rewrite_with_llm(f"The course {course_subject} 
{course_number} is about: \n", course_description)

        dispatcher.utter_message(response)

# Q2-2) "Which topics are covered in event of course?
class TopicsCoveredByCourseEvent(Action):

    def name(self) -> Text:
        return "action_about_course_lecture_topics"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        course_subject = tracker.slots['courseSubject']
        course_number = tracker.slots['courseNumber']

        lecture_number = tracker.slots['lectureNumber']
        lecture_number = re.search(r'\d+', lecture_number).group()
        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>
        SELECT DISTINCT ?topicName
            WHERE {{

            ?course rdf:type vivo:Course ;
                    acad:courseNumber ?courseNumber ;
                    acad:courseSubject ?courseSubject ;
                    acad:hasLecture ?lecture.
                    FILTER (?courseSubject = "{course_subject}" && 
?courseNumber = "{course_number}")
            ?lecture rdf:type acad:Lecture ;
                    acad:lectureNumber ?lectureNumber .
                    FILTER (?lectureNumber = {lecture_number})
            ?topic rdf:type acad:Topic ;
                    acad:hasProvenanceInformation ?lecture ;
                    acad:topicName ?topicName
        }}
        """

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)

        y = y['results']['bindings']

        topic_info = []
        for topic in y:
            topic_name = topic['topicName']['value']
            topic_info.append(topic_name)

        response = rewrite_with_llm(f"The topics covered in
{course_subject} {course_number} during lecture {lecture_number}
are: \n ", topic_info)

        dispatcher.utter_message(response)

# Q2-3) Which course events cover <Topic>?
class CourseEventsCoveringTopic(Action):

    def name(self) -> Text:
        return "action_about_course_topic"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        topic_name = tracker.slots['topic']

        sparql_query = f"""
        PREFIX vivo: <http://vivoweb.org/ontology/core#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX acad: <http://acad.io/schema#>
        SELECT ?courseName ?courseSubject ?courseNumber
```

```
?lectureName ?lectureNumber
            WHERE {{

            ?course rdf:type vivo:Course ;
                        acad:courseNumber ?courseNumber ;
                        acad:courseSubject ?courseSubject ;
                        acad:courseName ?courseName ;
                        acad:hasLecture ?lecture.
            ?lecture rdf:type acad:Lecture ;
                    acad:lectureNumber ?lectureNumber ;
                    acad:lectureName ?lectureName .
            ?topic rdf:type acad:Topic ;
                            acad:hasProvenanceInformation ?lecture
;
                            acad:topicName ?topicName .
        FILTER(?topicName = "{topic_name}")
        }}
        """
        print(sparql_query)

        response = make_fuseki_server_request(sparql_query)

        y = json.loads(response.text)

        y = y['results']['bindings']

        events = []
        for course in y:
            course_name = "Course Name: " +
str(course['courseName']['value'])
            course_subject = "Course Subject: " +
course['courseSubject']['value']
            course_number = "Course Number: " +
course['courseNumber']['value']
            lecture_name = "Lecture Name: " +
course['lectureName']['value']
            lecture_number = "Lecture Number: " +
course['lectureNumber']['value']
            events.append([course_name, course_subject,
course_number, lecture_number, lecture_name])


        response = rewrite_with_llm(f"The course events covering
{topic_name} are: \n ",events)
```

```
    dispatcher.utter_message(response)
```

Steps followed to make a response:

- Extract relevant entities from the user query.
- Construct a SPARQL query template, inserting the extracted entities.
- Send a POST request to the Fuseki server containing the completed SPARQL query.
- Parse the JSON response received from the Fuseki server.
- The SPARQL queries act as templates, allowing for retrieval of specific information based on the extracted entities.

Handling OOV Words and Error Cases

The Rasa chatbot incorporates mechanisms to address out-of-vocabulary (OOV) words and errors:

- **Featurizers and Entity Synonym Mappers:** The Rasa configuration employs these components to manage words not explicitly included in the training data. This helps handle complex terms and variations in user phrasing.
- **FallbackClassifier:** The NLU pipeline incorporates a FallbackClassifier. This classifier identifies abnormal or off-topic questions, directing them to the utter_out_of_scope response, informing the user that the question is beyond the bot's scope.
- **Knowledge Graph Null Response:** If the user inquires about a valid entity that doesn't exist in the knowledge graph, the Fuseki server returns a null response. The Rasa Langchain Language Model (LLM) is employed to interpret this null response and provide a "search failed" message to the user.

Translating SPARQL Query Results

The LLM integration plays a crucial role in translating the raw SPARQL query results into human-readable responses. This process involves:

- **Prompt Template**: A predefined template is used to structure the LLM's interpretation of the results. This template specifies placeholders for answer descriptions and the actual answers retrieved.
- **Answer Formulation:** Based on the template, the LLM formulates clear and concise answers for the user, ensuring all retrieved information is conveyed without introducing extraneous content.
- **Empty Answer Handling:** If no answers are found, the LLM generates a message informing the user that the requested information could not be located within the knowledge graph.
- In cases where the knowledge graph returns a null response, separate LLM templates are used to craft appropriate messages for the user.

We have then integrated Slack with Rasa to create the Chatbot UI.

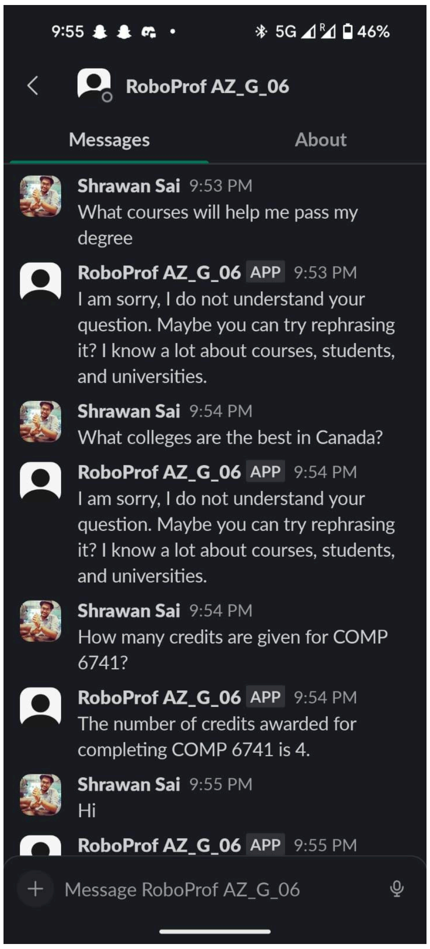
Chatbot UI : Slack Integration with Rasa.

Below are the steps we used to do this:

- Setting Up Rasa: We installed Rasa and configured it on our system.
- Creating a Slack App: On the Slack API website, we created a new Slack app. This provided us with an API token that needed to communicate with Slack.
- Configuring Slack Events API: In the Slack app settings, we configured the Events API to listen for messages sent to ourbot. We specified a Request URL where Slack will send events.
- Setting Up a Webhook: We created a webhook endpoint in our Rasa application that Slack can send messages to. This endpoint receives messages from Slack and sends them to Rasa for processing.
- 5. Handling Slack Events : In our Rasa application, we implemented logic to handle incoming messages from Slack. This involved parsing the messages, extracting user intents, and generating responses.
- Sending Responses to Slack: Once Rasa generated a response, we sent it back to Slack using the Slack API. We used the API token provided by Slack to authenticate our requests.
- Testing and Iterating: We thoroughly tested the integration to ensure that messages are being sent and received correctly between Slack and Rasa. We iterated on our bot's responses and behavior to improve its performance.

❖ **Chatbot Queries** :

1. Part #1: "How many credits are given for <Course>?" and also incorrect queries example



Shrawan Sai 9:53 PM
What courses will help me pass my degree

RoboProf AZ_G_06 APP 9:53 PM
I am sorry, I do not understand your question. Maybe you can try rephrasing it? I know a lot about courses, students, and universities.

Shrawan Sai 9:54 PM
What colleges are the best in Canada?

RoboProf AZ_G_06 APP 9:54 PM
I am sorry, I do not understand your question. Maybe you can try rephrasing it? I know a lot about courses, students, and universities.

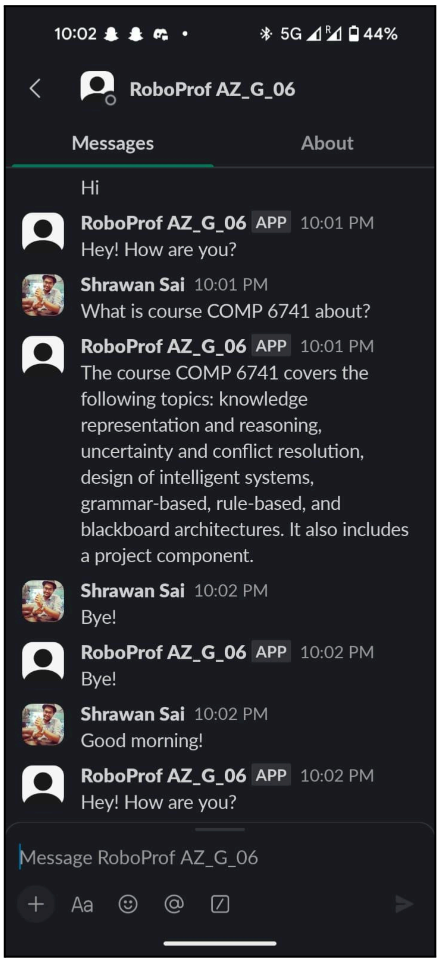
Shrawan Sai 9:54 PM
How many credits are given for COMP 6741?

RoboProf AZ_G_06 APP 9:54 PM
The number of credits awarded for completing COMP 6741 is 4.

Shrawan Sai 9:55 PM
Hi

RoboProf AZ_G_06 APP 9:55 PM

2. Part #2: "What is the <course> about?"



10:02 5G 44%

RoboProf AZ_G_06

Messages About

Hi

RoboProf AZ_G_06 APP 10:01 PM
Hey! How are you?

Shrawan Sai 10:01 PM
What is course COMP 6741 about?

RoboProf AZ_G_06 APP 10:01 PM
The course COMP 6741 covers the following topics: knowledge representation and reasoning, uncertainty and conflict resolution, design of intelligent systems, grammar-based, rule-based, and blackboard architectures. It also includes a project component.

Shrawan Sai 10:02 PM
Bye!

RoboProf AZ_G_06 APP 10:02 PM
Bye!

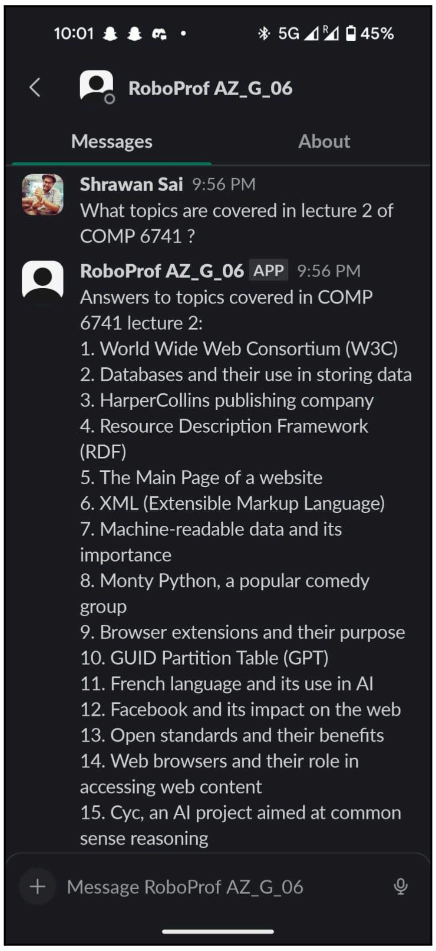
Shrawan Sai 10:02 PM
Good morning!

RoboProf AZ_G_06 APP 10:02 PM
Hey! How are you?

Message RoboProf AZ_G_06

Aa

3. Part #2: "Which topics are covered in <course event> ?"


10:01 5G 45%

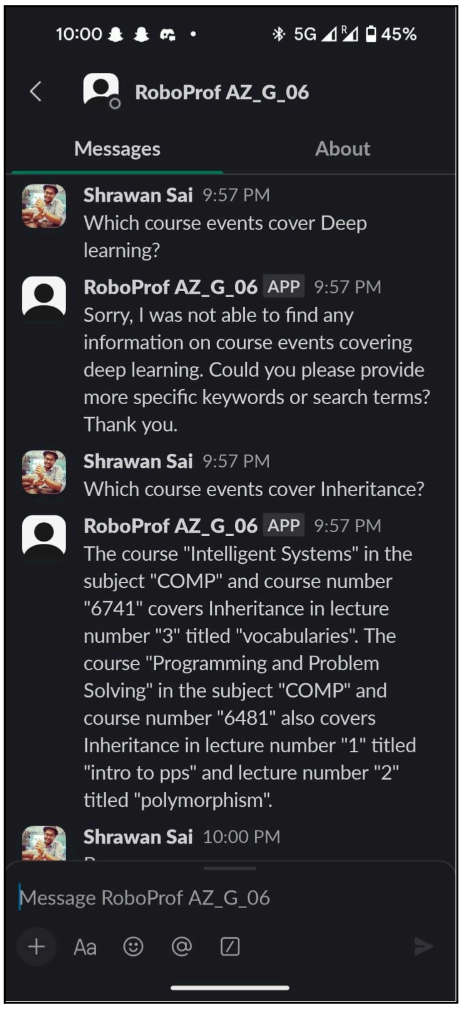
RoboProf AZ_G_06

Messages About

Shrawan Sai 9:56 PM
What topics are covered in lecture 2 of COMP 6741 ?

RoboProf AZ_G_06 APP 9:56 PM
Answers to topics covered in COMP 6741 lecture 2:
1. World Wide Web Consortium (W3C)
2. Databases and their use in storing data
3. HarperCollins publishing company
4. Resource Description Framework (RDF)
5. The Main Page of a website
6. XML (Extensible Markup Language)
7. Machine-readable data and its importance
8. Monty Python, a popular comedy group
9. Browser extensions and their purpose
10. GUID Partition Table (GPT)
11. French language and its use in AI
12. Facebook and its impact on the web
13. Open standards and their benefits
14. Web browsers and their role in accessing web content
15. Cyc, an AI project aimed at common sense reasoning

4. Part #2: “Which course events cover <Topic>?”



10:00 5G 45%

RoboProf AZ_G_06

Messages About

Shrawan Sai 9:57 PM
Which course events cover Deep learning?

RoboProf AZ_G_06 APP 9:57 PM
Sorry, I was not able to find any information on course events covering deep learning. Could you please provide more specific keywords or search terms? Thank you.

Shrawan Sai 9:57 PM
Which course events cover Inheritance?

RoboProf AZ_G_06 APP 9:57 PM
The course "Intelligent Systems" in the subject "COMP" and course number "6741" covers Inheritance in lecture number "3" titled "vocabularies". The course "Programming and Problem Solving" in the subject "COMP" and course number "6481" also covers Inheritance in lecture number "1" titled "intro to pps" and lecture number "2" titled "polymorphism".

Shrawan Sai 10:00 PM

References:

1. OpenAI. "ChatGPT." 2024. OpenAI. https://openai.com/chatgpt.
2. Rasa. (n.d.). Home. Retrieved, from https://rasa.com/
3. Slack. (n.d.). API Documentation. Retrieved, from https://api.slack.com/
4. LangChain. (n.d.). Home. Retrieved, from https://www.langchain.com/