Project Assignment #2

COMP 474/6741
Intelligent Systems

Due: April 15, 2024

To: Dr. Rene Witte

Group AK_U_01
By:
Faisal Bari (27517440) - Computer Engineering
Gabriel Giangi (40174314) - Computer Science
Michel Farah (40187552) - Computer Engineering
Omar Debayeh (40100195) - Computer Engineering

https://github.com/gdgiangi/RoboProf-Intelligent-Agent.git

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

Report updates:

Since the original submission, the only change has been to the knowledge base construction script, to which we have added the topic triple generation code, further detailed in the "Knowledge Base Population" section.

Vocabulary:

In constructing the schema for the knowledge base, a mix of commonly used vocabularies and custom-defined vocabularies were used for a comprehensive representation of academic entities and their relationships. RDF (Resource Description Framework), RDFS (RDF Schema), and FOAF (Friend of a Friend) namespaces allowed us to describe resources, relationships, and their corresponding data. The RDF and RDFS namespaces facilitate the definition of classes, properties, and relationships in an interconnected data model. The choice to use RDFS for class definitions and subclassing provides a semantic layer that provides a deeper understanding of the relationships and meanings behind the data. The FOAF vocabulary, known for modeling personal data and social networking connections, was chosen for its widespread acceptance and suitability for representing social entities like students and organizations.

Recognizing the limitations of existing vocabularies to capture the details of Concordia University's course catalog database, vocabulary adaptations were made. The FOCU namespace was created to define domain specific classes and properties, such as University, Course, Lecture, and various content types (Slides, Worksheets, Readings). This allows for a proper representation of Concordia's course academic structure. Additionally, the introduction of a custom namespace, FOCUDATA, allows for the distinction between schema definitions and data instances. The use of WIKIDATA as a namespace for linking to Wikidata entities shows our ability to enrich the knowledge base with external data, connecting information silos.

Classes created include University, Course, Lecture, Content, Topic, Student, CourseEnrollment. These classes are the foundation to the knowledge graph, representing entities like educational institutions, academic courses, individual lectures, educational content, topics covered, students, and course enrollments.

Subclasses under the Content class include Slide, Worksheet, Reading, Other. These provide a categorization of the types of content that can be linked with lectures.

Custom properties were defined to express specific relationships relevant to the academic domain, such as courseSubject, courseNumber, courseCredits, courseDescription, offeredAt, lectureNumber, taughtIn, studentID, enrolledIn, gradeReceived, and hasCompetencyIn. These properties allowed for description of courses, their structure, enrollment details, and the competencies students gain.

The schema further incorporates specific properties to describe academic entities, such as course credits, lecture content, student enrollment details, grades and competencies acquired. By defining properties with explicit domains and ranges, the schema serves as a structured guide. It promotes data consistency by establishing clear standards that must be followed.

Knowledge Base Construction:

The dataset used was taken from Concordia's database. We found two dataset useful for our intelligent agent. The first file contains open data of each course/subject that Concordia offers alongside with its prerequisites, course ID and credits. The second open data file contains the course ID and description of each course that it provides in the first file. We merged the two files to create one big merged file containing all the relevant information about each course provided by Concordia. We also artificially created two extra files containing information about students attending Concordia. The first new created files contain student ID, their email and full names. The second newly created file contains the student ID, the course that they have taken and their grade for that course.

To start populating the knowledge graph we must first import libraries such as rdflib, pandas and anything else that would be useful such as graph, RDF, namespace, URIRef, RDFS and FOAF. The second step is to create an empty graph using the Graph() method. We then import our vocabulary.ttl file written in turtle format by calling the parse() method. We then create the namespaces that are consistent with the ones defined in our vocabulary.ttl file. The namespaces are created and binded to the graph we created by using the bind() method. The namespaces we created are WIKIDATA, FOCU, FOCUDATA. We then reuse FOAF, RDFS and RDF provided by the library. We then create a URI for Concordia and give it some properties, such as FOAF:name and its FOCU:wikidataSource which happens to be Q326342. This instantiation of the Concordia University class is added to the graph by using the add() method.

Through this script we will import the two opendata files from the Concordia database and merge them together to make it easier for us to work with. We will now work with this new file, now named merged.csv. We then create a loop that will go through each row and create a URI by using the URIRef() method. This allows us to uniquely identify the course within the RDF graph. We then use the URI to add to the graph the FOCU.courseSubject, FOCU.courseNumber, FOAF.name, FOCU.courseCredits, FOCU.courseDescription and other information found in the merged file by associating the literals found in each row. The add() method was used to achieve this. We do this for all rows found in the merged.csv file. We then do the exact same thing for the student files created. The student_data file is the same as merged.csv file however, for the grades_data file, there are extra steps that need to be taken, but overall it is very similar to what we mentioned above. Finally, we create a file with all the URIs of the two selected courses (COMP474 & COEN366) we want to add to the knowledge graph. Each row is a URI with its corresponding course number, lecture name, content type and topic names. The script then reads the data from the csv file and processes each row to create a link data entity in the knowledge base.

To create the knowledge base, we can simply run the script, where it will create the merged.csv file from the two raw datasets found on Concordia's open database. It will then start populating the graph from that new merged file as well as add students' information onto the graph. It will then output the graph onto a file called knowledge_base.ttl in turtle format.

Knowledge Base Population:

Document Processing

The first step includes extracting text data from all the documents uploaded in part 1 of the project. One of the problems encountered is that course contents are in different formats (PDF, ppt, docx, HTML). The python code walks through all directories and files from the input directory and creates a corresponding txt file, extracts the text and saves it in the newly created file. All the different files are converted to txt files using Apache Tika which allows SpaCy to extract entities from the uploaded files.

Entity Recognition and Linking

The next step was to recognize entities within the text, extract them and link them with their corresponding Wikidata URI. For this, we used SpaCy to identify entities within the course content, and the library also returned the corresponding wikidata URI. We used *entityfishing* and saved all entities into a CSV for further processing.

Filtering and Validation

After entity recognition and linking, we needed to filter the results as there were some unnecessary results like currency symbols that were extracted as entities. Therefore, we created a script that would remove any entries with less than 3 characters for their topic name. This ensured that we had relevant topics for each entry. We also removed all entries that could not link to WikiData URL. From here, there was some cleaning up of the CSV files so we can use the knowledge base creation script created in part 1 of the project.

Finally, using RDF and RDFLib we merged data from multiple CSV files, creating a graph of RDF triples that model the relationships of the educational entities and linked to external links from Wikidata. Finally, the entire graph is converted to Turtle allowing for data that can be queried for the next part of the project.

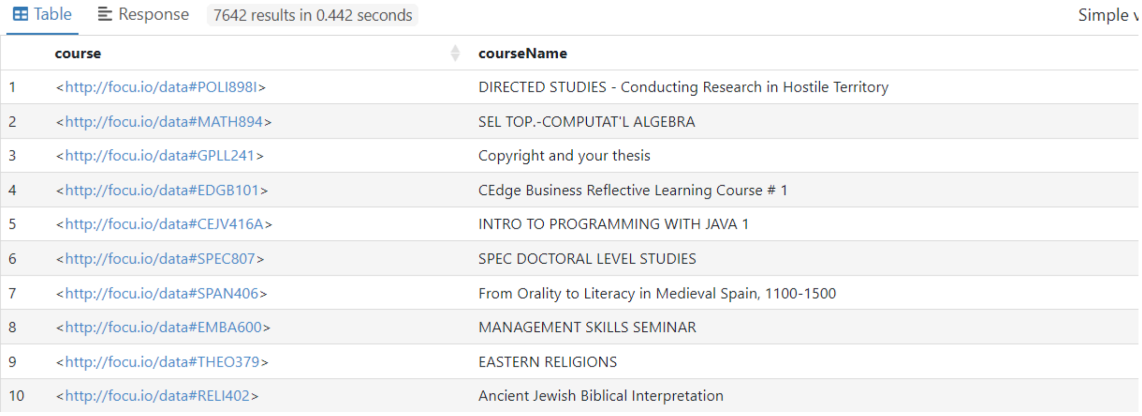
Statistics

Description	**Value**
Triples uploaded	57,217
Number of distinct topics	1,431
Number of topic instances in COMP474	733
Number of topic instances in COEN366	698

Graph Queries:

The queries for the SPARQL will be put in the appendix. The description on how we did it will be written here. Consult the appendix for the proper SPARQL queries.

1. For the first query, we would like to list all the courses available at Concordia University. So we want the course and courseName which are defined in SELECT. We then specify in the WHERE block that we would like all the courses and courseName that is offeredAt Concordia. This will give us a complete list of courses available at Concordia with their corresponding full courseNames.

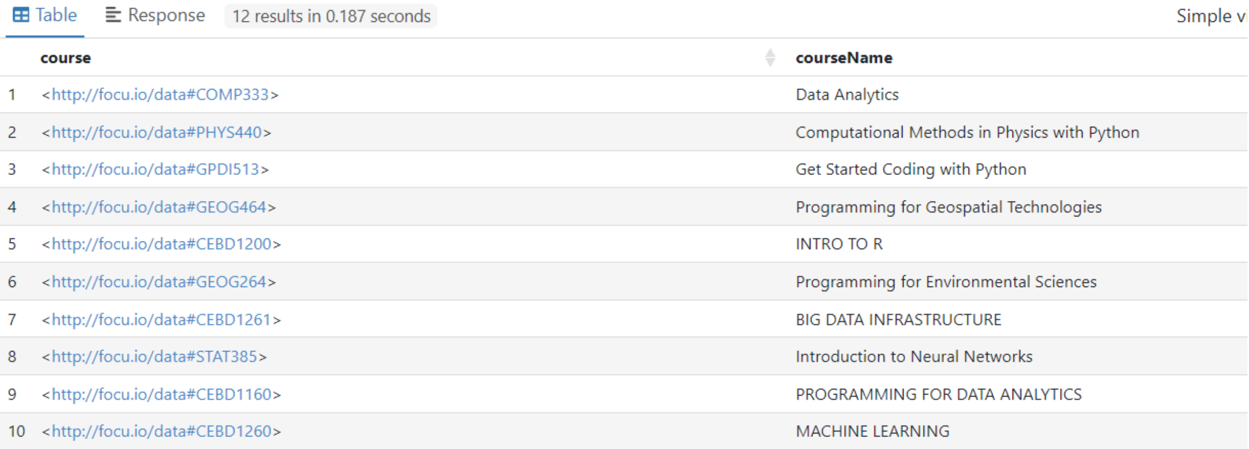


7642 results in 0.442 seconds

	course	courseName
1	<http://focu.io/data#POLI898I>	DIRECTED STUDIES - Conducting Research in Hostile Territory
2	<http://focu.io/data#MATH894>	SEL TOP.-COMPUTAT'L ALGEBRA
3	<http://focu.io/data#GPLL241>	Copyright and your thesis
4	<http://focu.io/data#EDGB101>	CEdge Business Reflective Learning Course # 1
5	<http://focu.io/data#CEJV416A>	INTRO TO PROGRAMMING WITH JAVA 1
6	<http://focu.io/data#SPEC807>	SPEC DOCTORAL LEVEL STUDIES
7	<http://focu.io/data#SPAN406>	From Orality to Literacy in Medieval Spain, 1100-1500
8	<http://focu.io/data#EMBA600>	MANAGEMENT SKILLS SEMINAR
9	<http://focu.io/data#THEO379>	EASTERN RELIGIONS
10	<http://focu.io/data#RELI402>	Ancient Jewish Biblical Interpretation

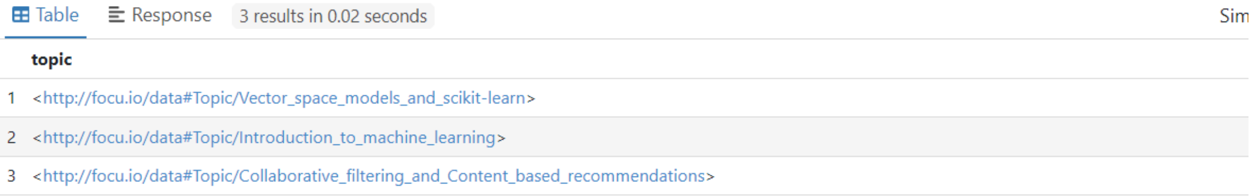
2. In this query, we would like to specify a certain topic and find all courses that mention this topic. To do this we put ?course and ?courseName into SELECT. In the WHERE block we specify that ?course is a FOCU:Course and that is if offeredAt Concordia. It has a courseName and courseDescription. We then use the FILTER, to filter for the topic in

the ?description. This will output and query all courses that have the topic mentioned inside the description. If not, it will simply not query it.



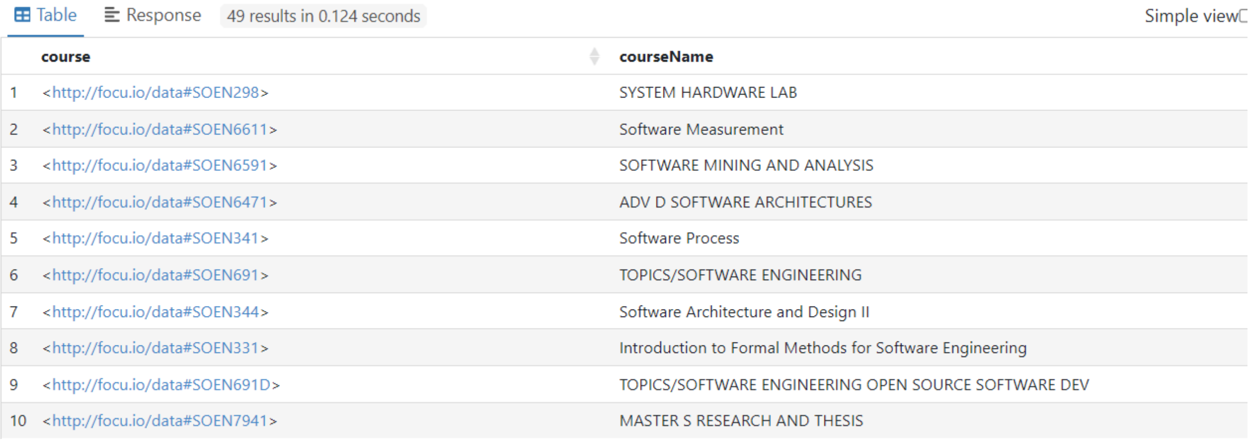
	course	courseName
1	<http://focu.io/data#COMP333>	Data Analytics
2	<http://focu.io/data#PHYS440>	Computational Methods in Physics with Python
3	<http://focu.io/data#GPDI513>	Get Started Coding with Python
4	<http://focu.io/data#GEOG464>	Programming for Geospatial Technologies
5	<http://focu.io/data#CEBD1200>	INTRO TO R
6	<http://focu.io/data#GEOG264>	Programming for Environmental Sciences
7	<http://focu.io/data#CEBD1261>	BIG DATA INFRASTRUCTURE
8	<http://focu.io/data#STAT385>	Introduction to Neural Networks
9	<http://focu.io/data#CEBD1160>	PROGRAMMING FOR DATA ANALYTICS
10	<http://focu.io/data#CEBD1260>	MACHINE LEARNING

3. We want to compile a list of topics covered in a specific course during a particular lecture number. To achieve this, we utilize the variable ?topic in the SELECT statement to retrieve all topics. Within the WHERE block, we specify our search criteria: the lecture content taught in COMP474 (in our example) and the desired lecture number. Subsequently, we retrieve the content and identify the associated topic using FOCU:source.



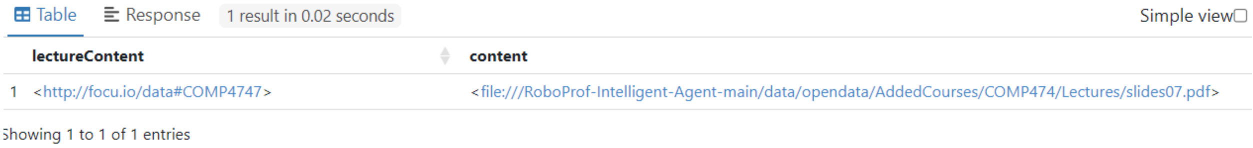
	topic
1	<http://focu.io/data#Topic/Vector_space_models_and_scikit-learn>
2	<http://focu.io/data#Topic/Introduction_to_machine_learning>
3	<http://focu.io/data#Topic/Collaborative_filtering_and_Content_based_recommendations>

4. For this query, we want to list all courses that are offered by Concordia that are within a certain subject. To do this, we SELECT ?course and ?courseName as we want to list all possible courses. In the WHERE block, we specify what we want ?course that is a FOCU:Course with the following properties, FOCU:offeredAt Concordia, it has the FOCU:courseSubject “SOEN” or anything we desire/specify. We then use FOAF:name ?courseName to output all course names that are SOEN or anything we specify for it.



	course	courseName
1	<http://focu.io/data#SOEN298>	SYSTEM HARDWARE LAB
2	<http://focu.io/data#SOEN6611>	Software Measurement
3	<http://focu.io/data#SOEN6591>	SOFTWARE MINING AND ANALYSIS
4	<http://focu.io/data#SOEN6471>	ADV D SOFTWARE ARCHITECTURES
5	<http://focu.io/data#SOEN341>	Software Process
6	<http://focu.io/data#SOEN691>	TOPICS/SOFTWARE ENGINEERING
7	<http://focu.io/data#SOEN344>	Software Architecture and Design II
8	<http://focu.io/data#SOEN331>	Introduction to Formal Methods for Software Engineering
9	<http://focu.io/data#SOEN691D>	TOPICS/SOFTWARE ENGINEERING OPEN SOURCE SOFTWARE DEV
10	<http://focu.io/data#SOEN7941>	MASTER S RESEARCH AND THESIS

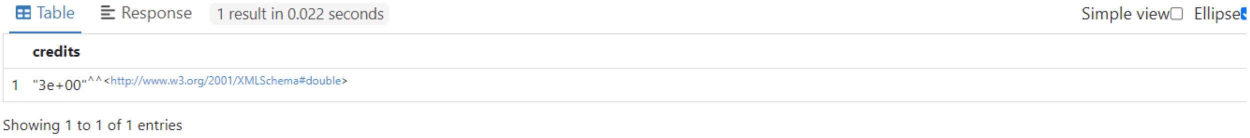
5. For this query, we seek all available materials/slides/readings for a specific course number and a desired topic. We accomplish this by identifying the lectureContent for all the materials in a specific course (COMP474 in our example). Then, we determine the topics of all the content obtained. Finally, we filter these topics for a specific topic we aim to find, such as "Bots" in our example.



	lectureContent	content
1	<http://focu.io/data#COMP4747>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Lectures/slides07.pdf>

Showing 1 to 1 of 1 entries

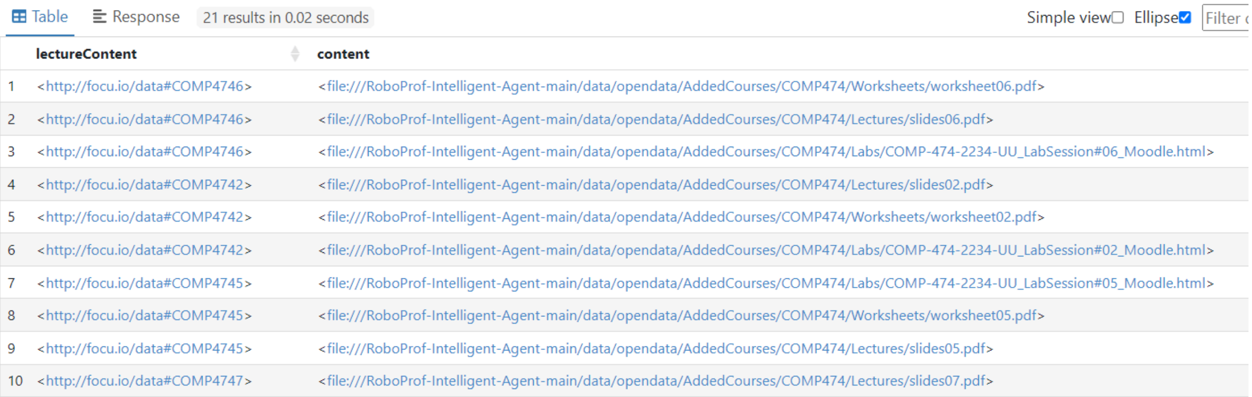
6. We want to know the number of credits for a specific course. So in SELECT we specify we want to see ?credits as the output. In the WHERE block, we specify we want a ?course which is a FOCU:Course, with the properties FOCU:courseNumber “321”, FOCU:courseSubject “SOEN” and we want all FOCU:courseCredits ?credits. This will display the credits for the course that we specify.



	credits
1	"3e+00"^^<http://www.w3.org/2001/XMLSchema#double>

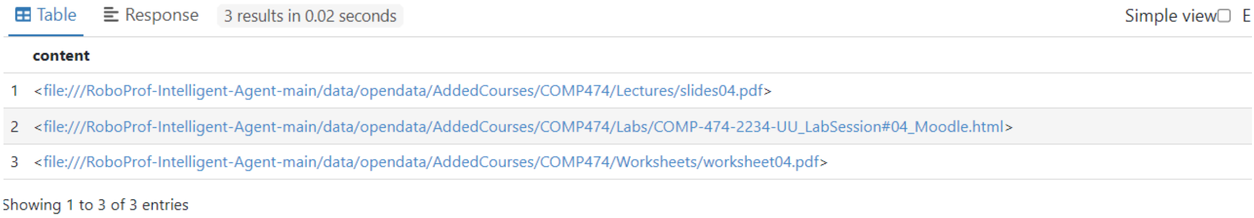
Showing 1 to 1 of 1 entries

7. To achieve this, we aim to identify all additional resources for a specific course. Within the SELECT statement, we utilize ?lectureContent and ?Content to retrieve all resources and lectures pertaining to a particular course (COMP474) as shown in the example. Subsequently, we ascertain the content of all selected lectures, ultimately showcasing all resources associated with the specific course across all its lectures.



	lectureContent	content
1	<http://focu.io/data#COMP4746>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Worksheets/worksheet06.pdf>
2	<http://focu.io/data#COMP4746>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Lectures/slides06.pdf>
3	<http://focu.io/data#COMP4746>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-UU_LabSession#06_Moodle.html>
4	<http://focu.io/data#COMP4742>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Lectures/slides02.pdf>
5	<http://focu.io/data#COMP4742>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Worksheets/worksheet02.pdf>
6	<http://focu.io/data#COMP4742>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-UU_LabSession#02_Moodle.html>
7	<http://focu.io/data#COMP4745>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-UU_LabSession#05_Moodle.html>
8	<http://focu.io/data#COMP4745>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Worksheets/worksheet05.pdf>
9	<http://focu.io/data#COMP4745>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Lectures/slides05.pdf>
10	<http://focu.io/data#COMP4747>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Lectures/slides07.pdf>

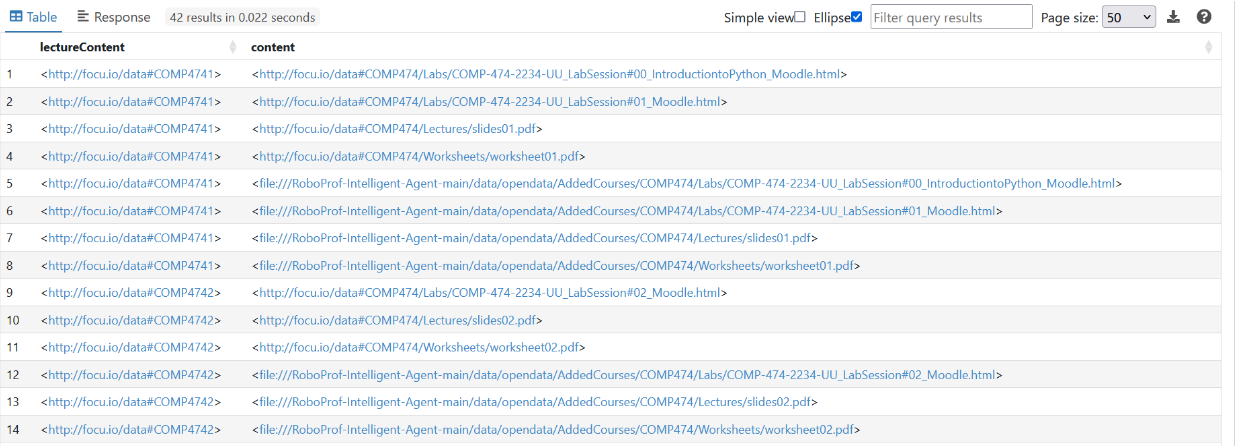
8. We would like the slides, readings, or worksheets that are available for a specific lecture number and course. We begin by including ?content in the SELECT statement to find the content for a chosen lecture number in the query. The lecture is associated with a specific course (COMP474) in our example; therefore, we locate and display the chosen lecture's content.



	content
1	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Lectures/slides04.pdf>
2	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-UU_LabSession#04_Moodle.html>
3	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Worksheets/worksheet04.pdf>

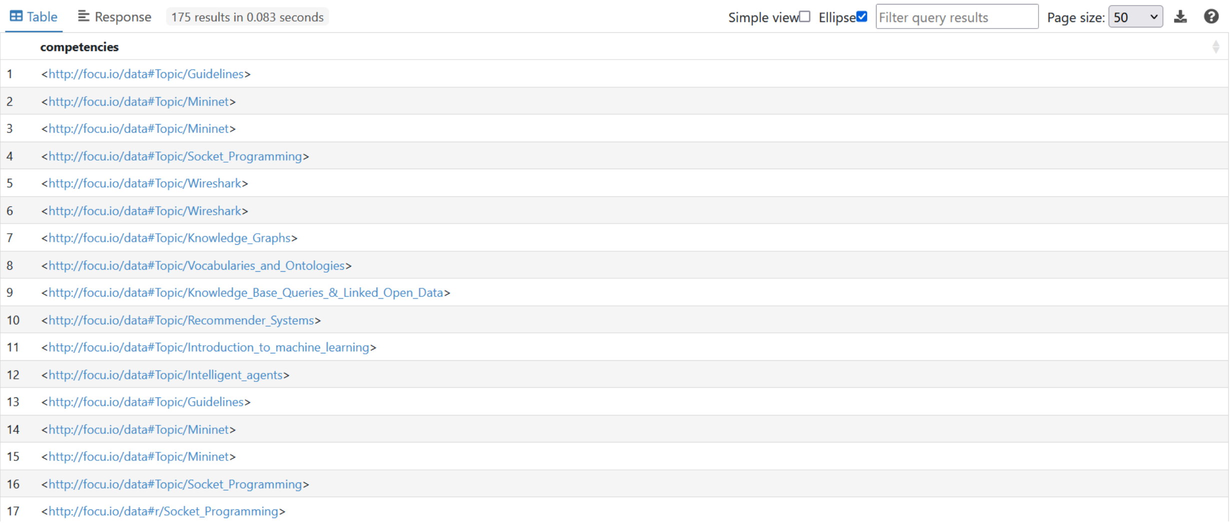
Showing 1 to 3 of 3 entries

9. For this query, we seek all available materials for a specific course number and a desired topic. We accomplish this by identifying the lectureContent for all the materials in a specific course. Then, we determine the topics of all the content obtained. Finally, we filter these topics for a specific topic we aim to find, such as "Bots" in our example. In the SELECT, we put down lectureContent and content as the outputs. In the WHERE block we specify that we want to know the lecture content that is taughtIn FOCUDATA:COMP474. Then we want to know the FOCU:source and filter out a topic from the source.



	lectureContent	content
1	<http://focu.io/data#COMP4741>	<http://focu.io/data#COMP474/Labs/COMP-474-2234-UU_LabSession#00_IntroductiontoPython_Moodle.html>
2	<http://focu.io/data#COMP4741>	<http://focu.io/data#COMP474/Labs/COMP-474-2234-UU_LabSession#01_Moodle.html>
3	<http://focu.io/data#COMP4741>	<http://focu.io/data#COMP474/Lectures/slides01.pdf>
4	<http://focu.io/data#COMP4741>	<http://focu.io/data#COMP474/Worksheets/worksheet01.pdf>
5	<http://focu.io/data#COMP4741>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-UU_LabSession#00_IntroductiontoPython_Moodle.html>
6	<http://focu.io/data#COMP4741>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-UU_LabSession#01_Moodle.html>
7	<http://focu.io/data#COMP4741>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Lectures/slides01.pdf>
8	<http://focu.io/data#COMP4741>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Worksheets/worksheet01.pdf>
9	<http://focu.io/data#COMP4742>	<http://focu.io/data#COMP474/Labs/COMP-474-2234-UU_LabSession#02_Moodle.html>
10	<http://focu.io/data#COMP4742>	<http://focu.io/data#COMP474/Lectures/slides02.pdf>
11	<http://focu.io/data#COMP4742>	<http://focu.io/data#COMP474/Worksheets/worksheet02.pdf>
12	<http://focu.io/data#COMP4742>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-UU_LabSession#02_Moodle.html>
13	<http://focu.io/data#COMP4742>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Lectures/slides02.pdf>
14	<http://focu.io/data#COMP4742>	<file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Worksheets/worksheet02.pdf>

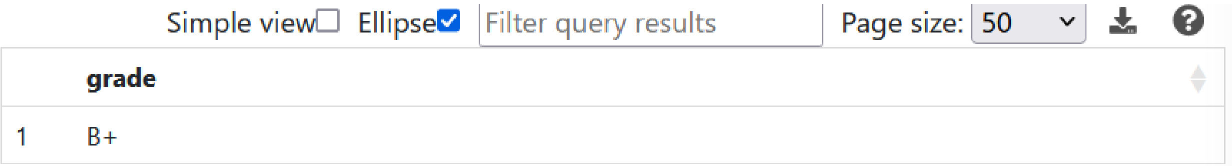
10. We want to know the competencies of the student after completing a certain course. In SELECT we put down competencies as the output. We specify the course as COEN for the subject and 366 for the courseNumber. Then we want to know which lectures are taught in the course by using the property taughtIn. Then we find the source of the lecture topics.



175 results in 0.083 seconds

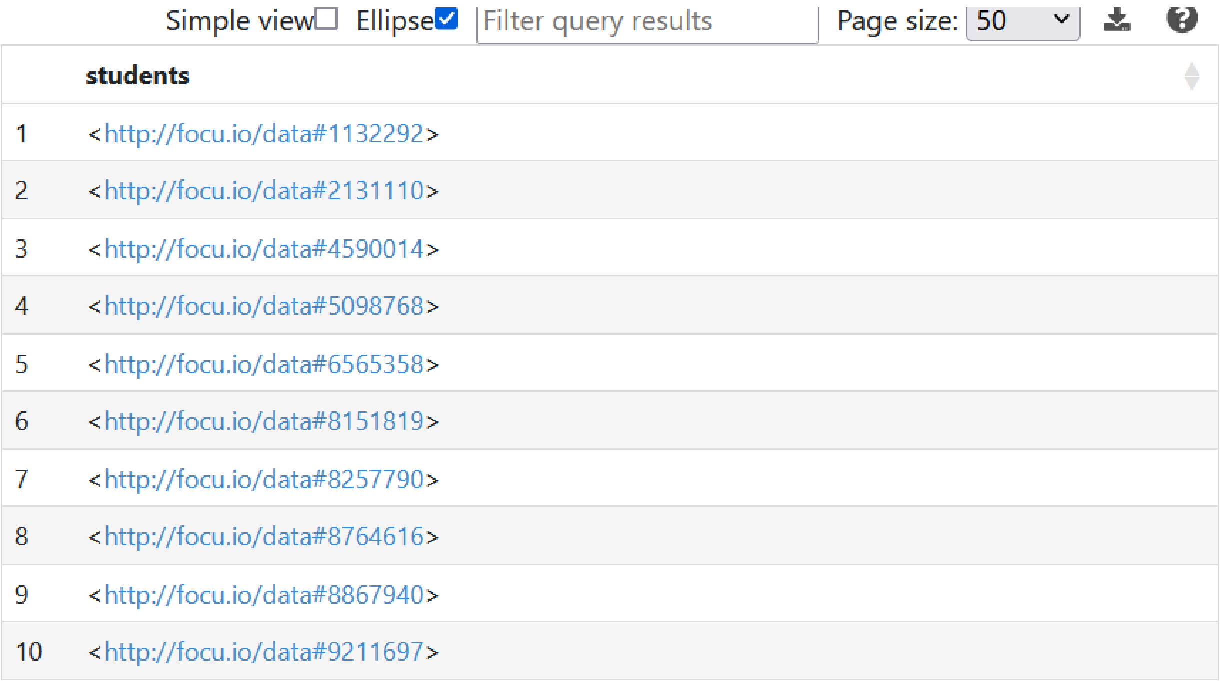
	competencies
1	<http://focu.io/data#Topic/Guidelines>
2	<http://focu.io/data#Topic/Mininet>
3	<http://focu.io/data#Topic/Mininet>
4	<http://focu.io/data#Topic/Socket_Programming>
5	<http://focu.io/data#Topic/Wireshark>
6	<http://focu.io/data#Topic/Wireshark>
7	<http://focu.io/data#Topic/Knowledge_Graphs>
8	<http://focu.io/data#Topic/Vocabularies_and_Ontologies>
9	<http://focu.io/data#Topic/Knowledge_Base_Queries_&_Linked_Open_Data>
10	<http://focu.io/data#Topic/Recommender_Systems>
11	<http://focu.io/data#Topic/Introduction_to_machine_learning>
12	<http://focu.io/data#Topic/Intelligent_agents>
13	<http://focu.io/data#Topic/Guidelines>
14	<http://focu.io/data#Topic/Mininet>
15	<http://focu.io/data#Topic/Mininet>
16	<http://focu.io/data#Topic/Socket_Programming>
17	<http://focu.io/data#r/Socket_Programming>

11. We want to know the grade of a student for a particular course. In SELECT we specify the grade as the output. In the WHERE block we put our desired course that we are interested in, the student we would like to know about, then in the FOCU:courseEnrollment, we specify the student, course and output all the grades.



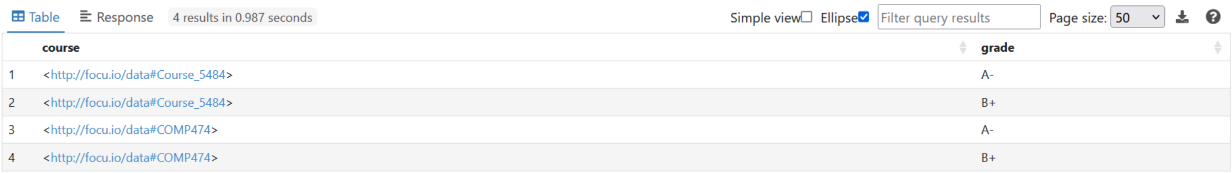
	grade
1	B+

12. We want to know all the students that have completed a specific course. We put the student in the SELECT block. In the WHERE block we say, we specify the course number. This will output all the students that took this particular course.


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

	students
1	<http://focu.io/data#1132292>
2	<http://focu.io/data#2131110>
3	<http://focu.io/data#4590014>
4	<http://focu.io/data#5098768>
5	<http://focu.io/data#6565358>
6	<http://focu.io/data#8151819>
7	<http://focu.io/data#8257790>
8	<http://focu.io/data#8764616>
9	<http://focu.io/data#8867940>
10	<http://focu.io/data#9211697>

13. For this last query, we want to know everything about this particular student. We want to print its transcript, which is all the courses and grades they have taken. Grades and courses will be put into the SELECT block. In the WHERE block we will want the FOCU:course offeredAt Concordia and the gradeReceived within the CourseEnrollment class.



4 results in 0.987 seconds

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

	course	grade
1	<http://focu.io/data#Course_5484>	A-
2	<http://focu.io/data#Course_5484>	B+
3	<http://focu.io/data#COMP474>	A-
4	<http://focu.io/data#COMP474>	B+

Triplestore and SPARQL Endpoint Setup:

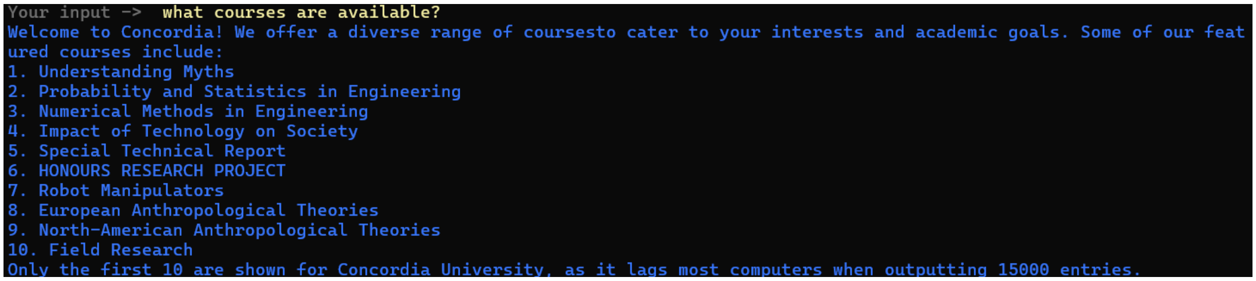
For the setting up of the apache server, we downloaded their official software from their website. We installed it and connected to the localserver port 3030. We then uploaded the knowledge graph file as a turtle file onto the server until it had specified that 53996 triples had been uploaded. There was no issue when installing the software of uploading the knowledge graph.

Roboprof Chatbot

From the previous part, all 13 queries largely stay the same. No major changes were made. The 4 SPARQL queries were written and stored in the Query folder in .txt file as well as their outputs stored in a .csv file.

The roboprof chatbot can answer all 13 questions from part 1 of the project as well as answer the 3 questions, what is the <course> about?, which topics are covered in <course event>?, and which course events cover <topic>?. It is all written within the RASA framework, inside the actions.py file. The Rasa chatbot framework is a powerful tool used to build conversational AI assistants which are capable of handling NLU, natural language understanding. It has many .yml files which are used to customize the functionality of the chatbot. The first and important file is the NLU.yml file which is used to store the intent for each sample question. The stories.yml link the intent to the action. This will allow us to know which query code to execute based on the type of question the user inputs. The domain.yml file helps us understand entities and what they mean and represent. This allows us to give these entities properties and help the chatbot understand the important keywords of a question. Lastly, we have the actions.py file, which lists down all 13 + 3 actions/queries that we will be executing based on the question at hand.

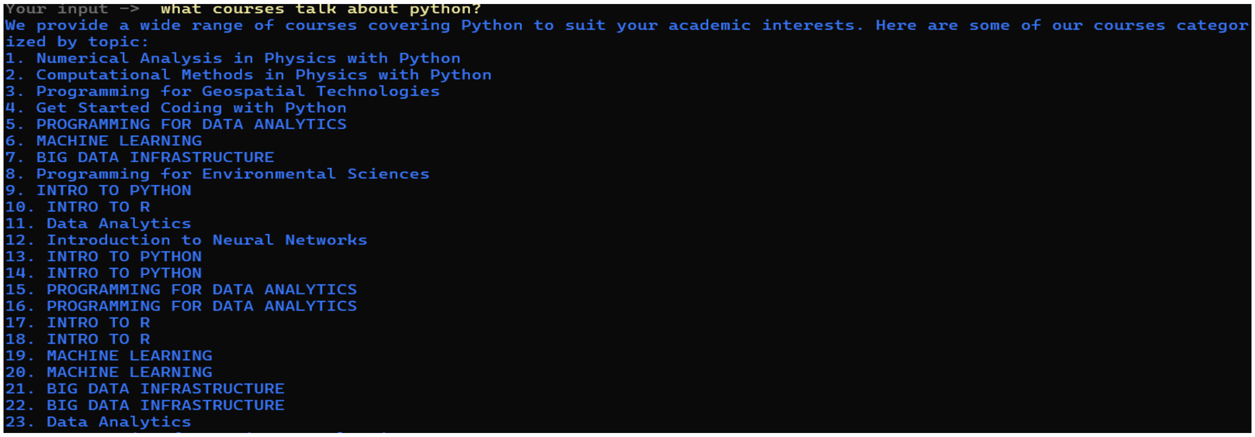
Query 1:



```
Your input ->  what courses are available?
Welcome to Concordia! We offer a diverse range of coursesto cater to your interests and academic goals. Some of our feat
ured courses include:
1. Understanding Myths
2. Probability and Statistics in Engineering
3. Numerical Methods in Engineering
4. Impact of Technology on Society
5. Special Technical Report
6. HONOURS RESEARCH PROJECT
7. Robot Manipulators
8. European Anthropological Theories
9. North-American Anthropological Theories
10. Field Research
Only the first 10 are shown for Concordia University, as it lags most computers when outputting 15000 entries.
```

This is an example of the first query. The question asks for information about courses available at Concordia. Here the chatbot knows, we are asking about Concordia, as we listed the University as an entity and put the default value as Concordia. Therefore, if we ask a question without saying the university type, it should know by default it is Concordia. This triggers the first action and it points to the first query. The query is slightly modified by the entities by using python's f-string functionality. This ensures we can query the correct set of outputs. The query is then sent to the SPARQL Fuseki server. The python code will then recover the output and naturalize the response, by using static templates we have created. Lookup tables were used in almost every query.

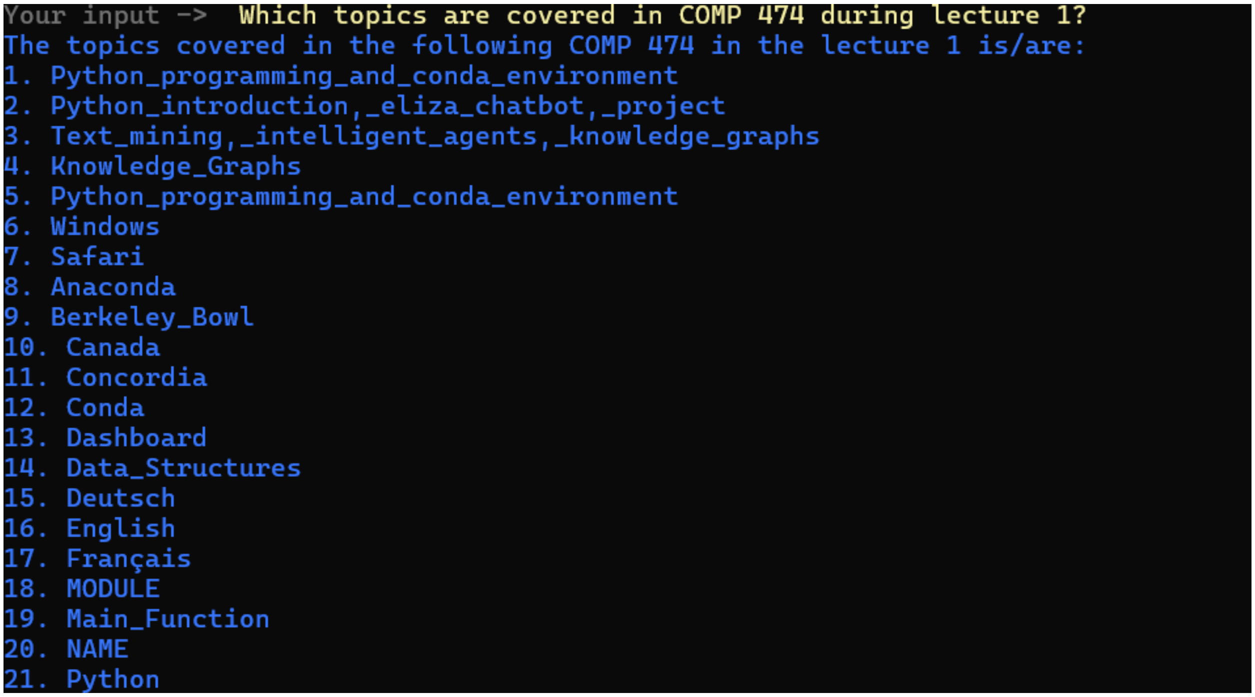
Query 2:



```
Your input ->  what courses talk about python?
We provide a wide range of courses covering Python to suit your academic interests. Here are some of our courses categor
ized by topic:
1. Numerical Analysis in Physics with Python
2. Computational Methods in Physics with Python
3. Programming for Geospatial Technologies
4. Get Started Coding with Python
5. PROGRAMMING FOR DATA ANALYTICS
6. MACHINE LEARNING
7. BIG DATA INFRASTRUCTURE
8. Programming for Environmental Sciences
9. INTRO TO PYTHON
10. INTRO TO R
11. Data Analytics
12. Introduction to Neural Networks
13. INTRO TO PYTHON
14. INTRO TO PYTHON
15. PROGRAMMING FOR DATA ANALYTICS
16. PROGRAMMING FOR DATA ANALYTICS
17. INTRO TO R
18. INTRO TO R
19. MACHINE LEARNING
20. MACHINE LEARNING
21. BIG DATA INFRASTRUCTURE
22. BIG DATA INFRASTRUCTURE
23. Data Analytics
```

This is an example of the second query. Here we specify the topic, and we get a list of courses that mention this topic.We use lookup tables to say which topics are valid. The topics that are valid are, Bots, Python, machine learning and deep learning. We could add more to make the chatbot more flexible, but so far these are the topics of interest. This information is sent to the actions.py file which will then activate the appropriate query code in the fuseki server. It will then retrieve the code from the database and process it using a static template to naturalize the sentence and output onto the screen.

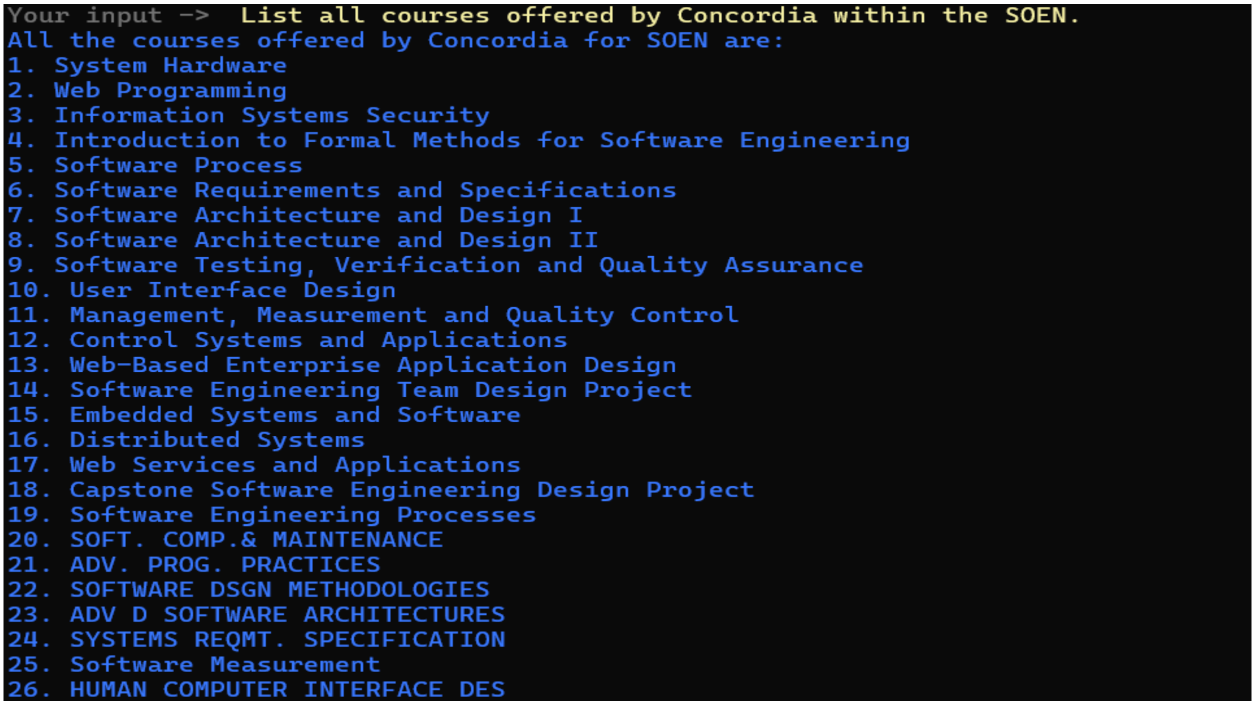
Query 3:



```
Your input ->  Which topics are covered in COMP 474 during lecture 1?
The topics covered in the following COMP 474 in the lecture 1 is/are:
1. Python_programming_and_conda_environment
2. Python_introduction,_eliza_chatbot,_project
3. Text_mining,_intelligent_agents,_knowledge_graphs
4. Knowledge_Graphs
5. Python_programming_and_conda_environment
6. Windows
7. Safari
8. Anaconda
9. Berkeley_Bowl
10. Canada
11. Concordia
12. Conda
13. Dashboard
14. Data_Structures
15. Deutsch
16. English
17. Français
18. MODULE
19. Main_Function
20. NAME
21. Python
```

This is an example of query 3. The process is essentially the same as the above 2 queries. The lookup table focused on the course name, course number and lecture number, we created a variety of possibilities to ensure we have the important parts covered. It can identify the lecture number, course name and course number to ensure good query results.

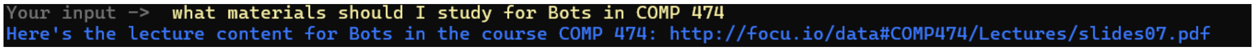
Query 4:



```
Your input ->  List all courses offered by Concordia within the SOEN.
All the courses offered by Concordia for SOEN are:
1. System Hardware
2. Web Programming
3. Information Systems Security
4. Introduction to Formal Methods for Software Engineering
5. Software Process
6. Software Requirements and Specifications
7. Software Architecture and Design I
8. Software Architecture and Design II
9. Software Testing, Verification and Quality Assurance
10. User Interface Design
11. Management, Measurement and Quality Control
12. Control Systems and Applications
13. Web-Based Enterprise Application Design
14. Software Engineering Team Design Project
15. Embedded Systems and Software
16. Distributed Systems
17. Web Services and Applications
18. Capstone Software Engineering Design Project
19. Software Engineering Processes
20. SOFT. COMP.& MAINTENANCE
21. ADV. PROG. PRACTICES
22. SOFTWARE DSGN METHODOLOGIES
23. ADV D SOFTWARE ARCHITECTURES
24. SYSTEMS REQMT. SPECIFICATION
25. Software Measurement
26. HUMAN COMPUTER INTERFACE DES
```

Query 4 looks at the courses offered at Concordia based on the subject. We ensured there was a sufficient amount of sample questions and lookup words to ensure the correct results. It can identify SOEN as the entity subject.

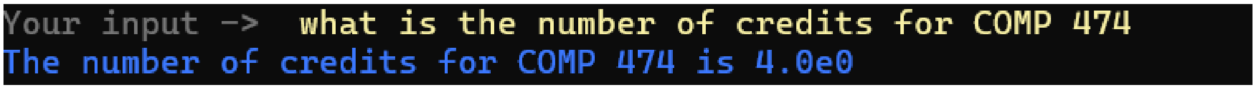
Query 5:



```
Your input ->  what materials should I study for Bots in COMP 474
Here's the lecture content for Bots in the course COMP 474: http://focu.io/data#COMP474/Lectures/slides07.pdf
```

For query 5, it can identify the topic of interest, in this case Bots. It is a entity of Topic. It can also identify COMP as the course name and 474 as the course number. These entities will help better identify which query to select, in this case it is query 5.

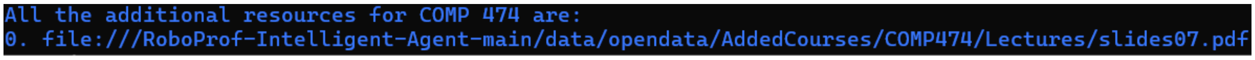
Query 6:



```
Your input ->  what is the number of credits for COMP 474
The number of credits for COMP 474 is 4.0e0
```

For query 6, we identify the COMP as the course name and 474 as the course subject. The chatbot uses lookup tables to know that COEN 366 is also valid as its input.

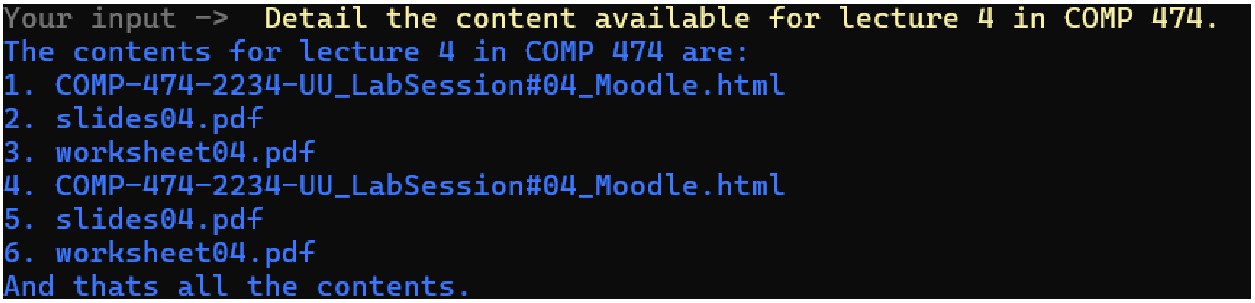
Query 7:



```
All the additional resources for COMP 474 are:
0. file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Lectures/slides07.pdf
```

This query is simple and follows the same steps as above queries. The course name COMP and course number 474 are entities and have lookup tables.

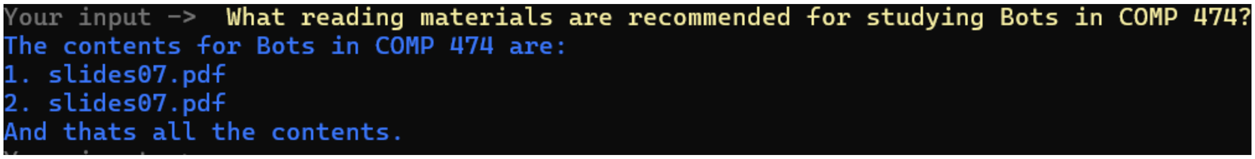
Query 8:



```
Your input ->  Detail the content available for lecture 4 in COMP 474.
The contents for lecture 4 in COMP 474 are:
1. COMP-474-2234-UU_LabSession#04_Moodle.html
2. slides04.pdf
3. worksheet04.pdf
4. COMP-474-2234-UU_LabSession#04_Moodle.html
5. slides04.pdf
6. worksheet04.pdf
And thats all the contents.
```

For query 8, we have the lecture number 4 as an entity the chatbot can recognize as well as the course name and course number. This is passed into the actions class which then executes the 8th query. Recognizing the lecture number and course names is crucial for ensuring the SPAQRL server can output the correct values for our chatbot.

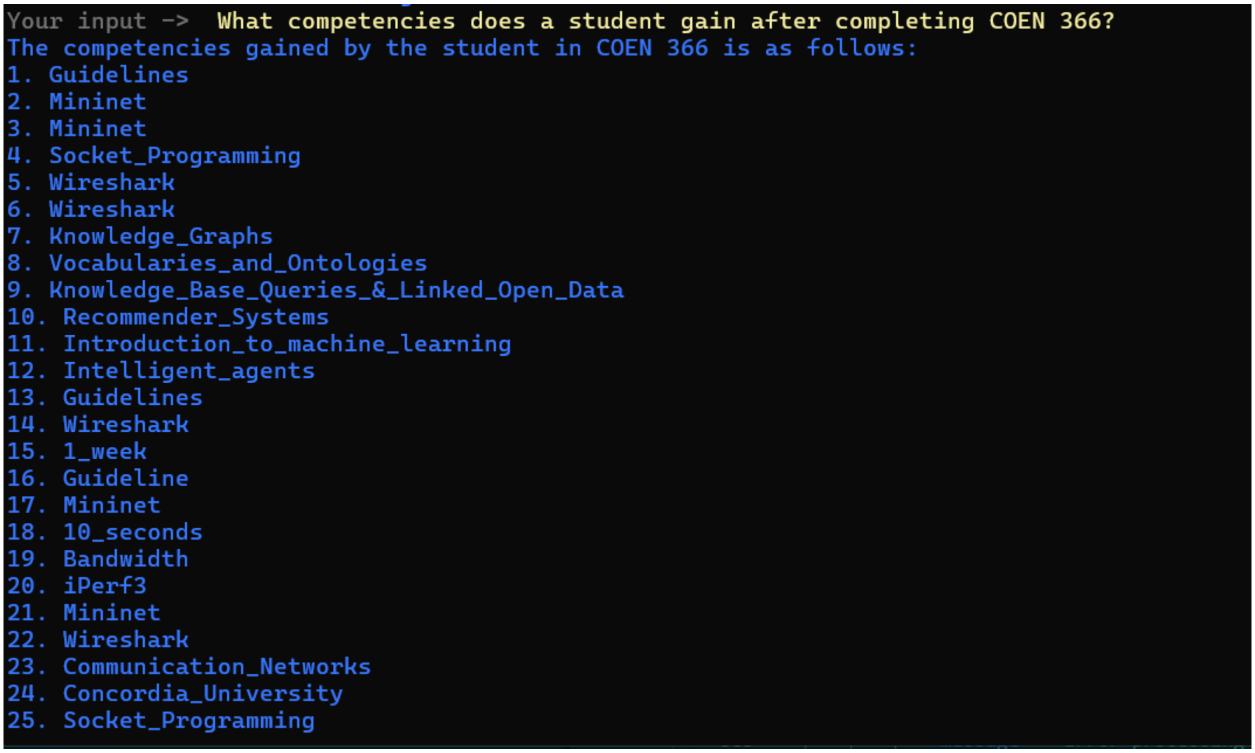
Query 9:



```
Your input ->  What reading materials are recommended for studying Bots in COMP 474?
The contents for Bots in COMP 474 are:
1. slides07.pdf
2. slides07.pdf
And thats all the contents.
```

Query 9 focuses on the recommend reading materials on Bots in COMP 474. Bots is an entity Topic which the user chooses. The chatbot can understand that Bots is a topic and choose the query code accordingly.

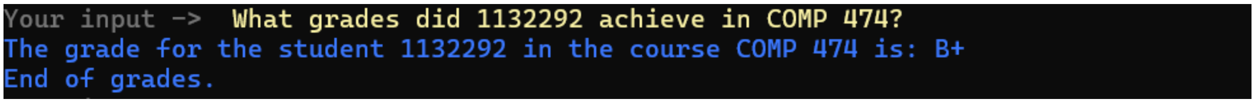
Query 10:



```
Your input ->  What competencies does a student gain after completing COEN 366?
The competencies gained by the student in COEN 366 is as follows:
1. Guidelines
2. Mininet
3. Mininet
4. Socket_Programming
5. Wireshark
6. Wireshark
7. Knowledge_Graphs
8. Vocabularies_and_Ontologies
9. Knowledge_Base_Queries_&_Linked_Open_Data
10. Recommender_Systems
11. Introduction_to_machine_learning
12. Intelligent_agents
13. Guidelines
14. Wireshark
15. 1_week
16. Guideline
17. Mininet
18. 10_seconds
19. Bandwidth
20. iPerf3
21. Mininet
22. Wireshark
23. Communication_Networks
24. Concordia_University
25. Socket_Programming
```

In this query we list the competencies based on the desired course name and number selected by the user. The process is essentially the same as above queries that we explained.

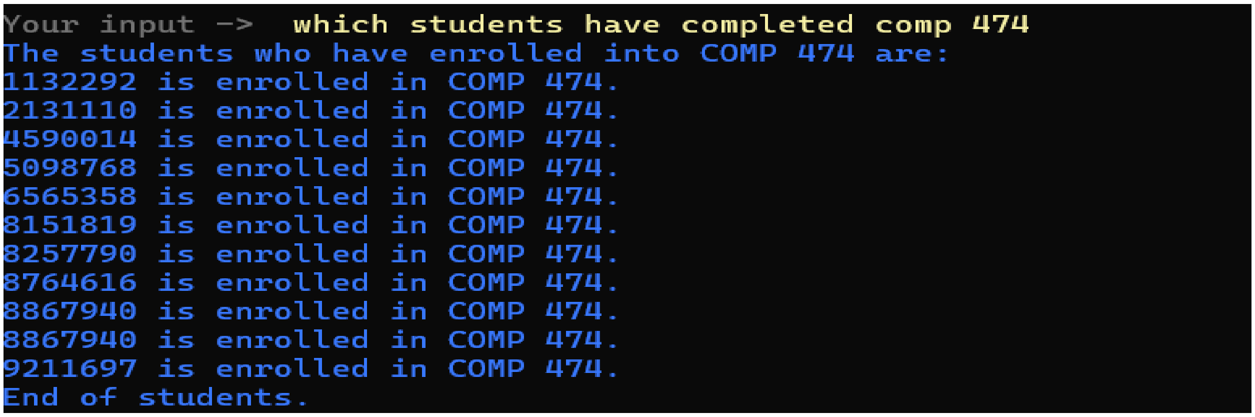
Query 11:



```
Your input ->  What grades did 1132292 achieve in COMP 474?
The grade for the student 1132292 in the course COMP 474 is: B+
End of grades.
```

Query 11 essentially retrieves information about a student when given an ID of the student. The ID is an entity that identifies a student. This will be passed as one of the inputs to the SPARQL server.

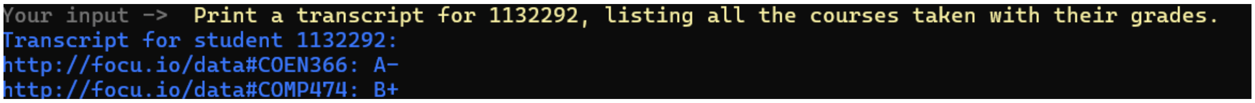
Query 12:



```
Your input ->  which students have completed comp 474
The students who have enrolled into COMP 474 are:
1132292 is enrolled in COMP 474.
2131110 is enrolled in COMP 474.
4590014 is enrolled in COMP 474.
5098768 is enrolled in COMP 474.
6565358 is enrolled in COMP 474.
8151819 is enrolled in COMP 474.
8257790 is enrolled in COMP 474.
8764616 is enrolled in COMP 474.
8867940 is enrolled in COMP 474.
8867940 is enrolled in COMP 474.
9211697 is enrolled in COMP 474.
End of students.
```

Query 12 outputs to the screen the number of students enrolled into 474. The inputs here are comp 474, and both are entities that are identifiable by the chatbot.

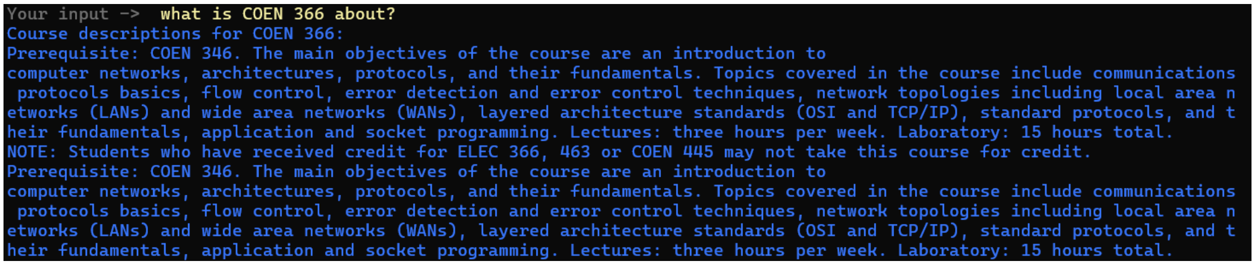
Query 13:



```
Your input ->  Print a transcript for 1132292, listing all the courses taken with their grades.
Transcript for student 1132292:
http://focu.io/data#COEN366: A-
http://focu.io/data#COMP474: B+
```

Query 12 is similar to 12 except we are retrieving from the Fuseki server the courses and grades the student took. The chatbot can identify the long list of numbers as the entity student.

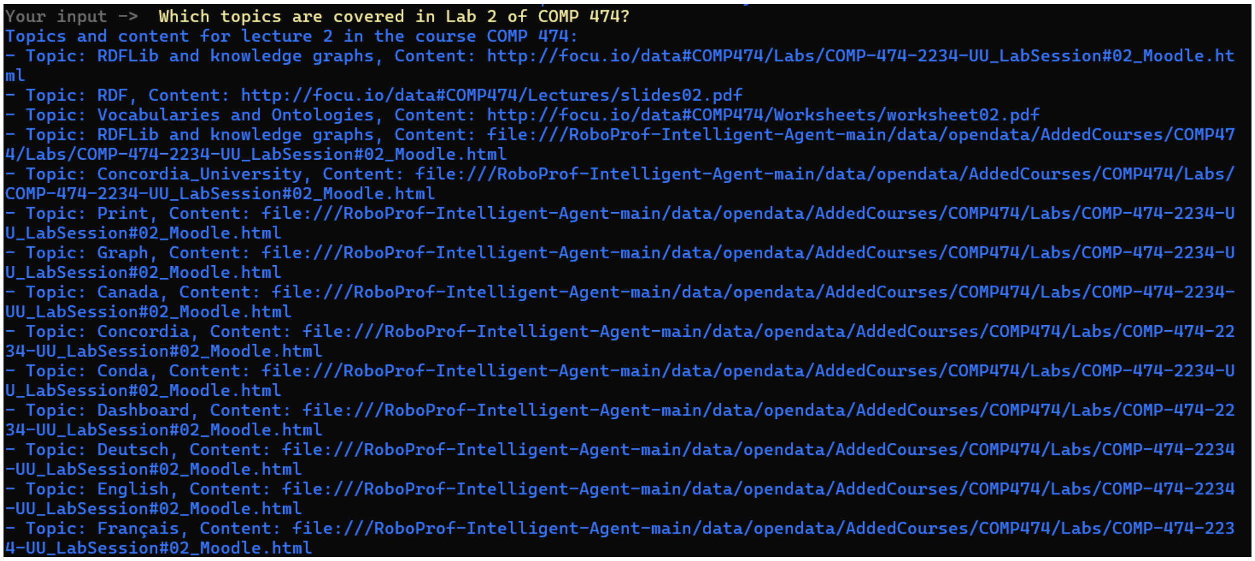
Query 14:



```
Your input ->  what is COEN 366 about?
Course descriptions for COEN 366:
Prerequisite: COEN 346. The main objectives of the course are an introduction to
computer networks, architectures, protocols, and their fundamentals. Topics covered in the course include communications
 protocols basics, flow control, error detection and error control techniques, network topologies including local area n
etworks (LANs) and wide area networks (WANs), layered architecture standards (OSI and TCP/IP), standard protocols, and t
heir fundamentals, application and socket programming. Lectures: three hours per week. Laboratory: 15 hours total.
NOTE: Students who have received credit for ELEC 366, 463 or COEN 445 may not take this course for credit.
Prerequisite: COEN 346. The main objectives of the course are an introduction to
computer networks, architectures, protocols, and their fundamentals. Topics covered in the course include communications
 protocols basics, flow control, error detection and error control techniques, network topologies including local area n
etworks (LANs) and wide area networks (WANs), layered architecture standards (OSI and TCP/IP), standard protocols, and t
heir fundamentals, application and socket programming. Lectures: three hours per week. Laboratory: 15 hours total.
```

In this query, the chatbot can identify the course name and course number. This query was a bit tricky and we had to make sure we had the right amount of sample questions for this to work effectively. Lookup tables were used to ensure the user can get the correct output for COEN 366 and COMP 474 specifically as they are the main focus of our chatbot.

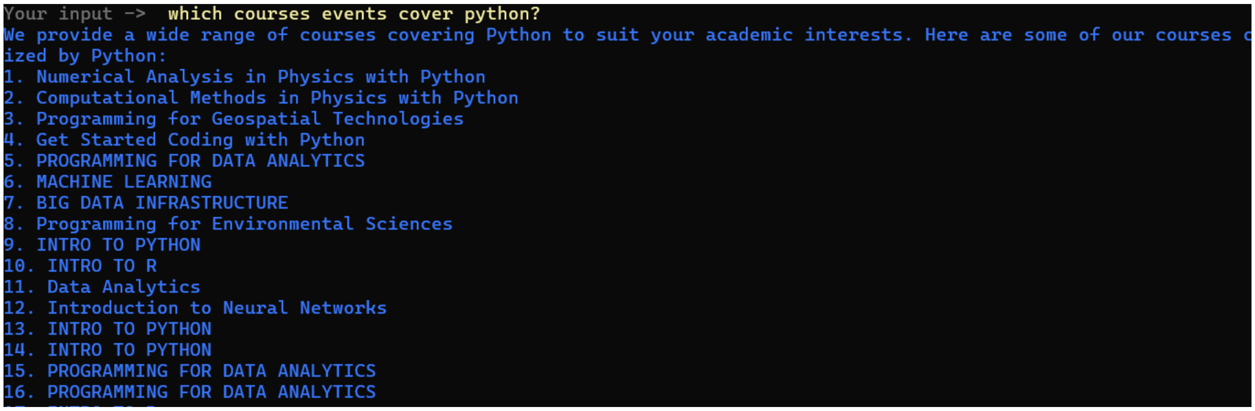
Query 15:



```
Your input ->  Which topics are covered in Lab 2 of COMP 474?
Topics and content for lecture 2 in the course COMP 474:
- Topic: RDFLib and knowledge graphs, Content: http://focu.io/data#COMP474/Labs/COMP-474-2234-UU_LabSession#02_Moodle.ht
ml
- Topic: RDF, Content: http://focu.io/data#COMP474/Lectures/slides02.pdf
- Topic: Vocabularies and Ontologies, Content: http://focu.io/data#COMP474/Worksheets/worksheet02.pdf
- Topic: RDFLib and knowledge graphs, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP47
4/Labs/COMP-474-2234-UU_LabSession#02_Moodle.html
- Topic: Concordia_University, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/
COMP-474-2234-UU_LabSession#02_Moodle.html
- Topic: Print, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-U
U_LabSession#02_Moodle.html
- Topic: Graph, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-U
U_LabSession#02_Moodle.html
- Topic: Canada, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-
UU_LabSession#02_Moodle.html
- Topic: Concordia, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-22
34-UU_LabSession#02_Moodle.html
- Topic: Conda, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234-U
U_LabSession#02_Moodle.html
- Topic: Dashboard, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-22
34-UU_LabSession#02_Moodle.html
- Topic: Deutsch, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234
-UU_LabSession#02_Moodle.html
- Topic: English, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-2234
-UU_LabSession#02_Moodle.html
- Topic: Français, Content: file:///RoboProf-Intelligent-Agent-main/data/opendata/AddedCourses/COMP474/Labs/COMP-474-223
4-UU_LabSession#02_Moodle.html
```

Query 15 essentially lists the topics covered in either lab/lecture for a specific course. The entities here are the lecture type, course name and course number. These are important entities, the chatbot needs to distinguish in order to ask the server for the right query.

Query 16:



```
Your input ->  which courses events cover python?
We provide a wide range of courses covering Python to suit your academic interests. Here are some of our courses c
ized by Python:
1. Numerical Analysis in Physics with Python
2. Computational Methods in Physics with Python
3. Programming for Geospatial Technologies
4. Get Started Coding with Python
5. PROGRAMMING FOR DATA ANALYTICS
6. MACHINE LEARNING
7. BIG DATA INFRASTRUCTURE
8. Programming for Environmental Sciences
9. INTRO TO PYTHON
10. INTRO TO R
11. Data Analytics
12. Introduction to Neural Networks
13. INTRO TO PYTHON
14. INTRO TO PYTHON
15. PROGRAMMING FOR DATA ANALYTICS
16. PROGRAMMING FOR DATA ANALYTICS
```

Lastly for query 16, the user can ask for course events that cover an entity Topic, which in this case is python. We used a lookup table to ensure the user can ask for other topics as well such as deep learning, bots, machine learning and python. The correct action is chosen in which the correct query code is chosen and sent to the Fuseki server. In which the query code is slightly modified by the entities to query the correct information. It is then sent to the chatbot, where we naturalize it by using a static template and then it is outputted to the screen.