COMP 6741
INTELLIGENT SYSTEMS
PROJECT ASSIGNMENT #2

WINTER 2021

Instructor: Dr. René Witte,

SUBMITTED BY:

Piyush Kumar (40119786)

Table of Content

1. COMPETENCY QUESTIONS

1. Which topics are covered in [course]?

2. Which topics are covered in [lecture_no] for [course]?

3. What lecture content was covered in [lecture no.] for [course]?

-lecture content here means, all the contents of lecture like slides, worksheets, readings and other materials.

4. What lab content was covered in [lab no.] for [course]?

5. What is the web link for a [course]?

6. What is the name of course with [course number] of [subject]?

7. What is course description for [course]?

8. What is the course outline for [course]?

9. Give slides for [lecture no] for [course]?

10. Give worksheets for [lecture no] for [course]?

11. Give lab Content for [lecture no] for [course]?

2. VOCABULARY DESIGN

For the design of the vocabularies we have the following major entities –

- University
- Courses
- Lectures
- Labs/Tutorials
- Topics

We made a new schema- sch-<http://example.org/schema#/>

1. For **University**(sch:University), we had two data members- name and link to dbpedia page.

a) name-for name, a new property called sch:uniName was created with domain as University and range as String.

b) link to dbpedia- we used OWL.sameAs for this linking.

2. For **Courses**(sch:Course), we had six data members- name , subject, number, description, link to webpage and outline.

a) name- a new property called sch:courseName was created with domain as sch:Course and range as xsd:String.

b) subject- a new property called sch:subject was created with domain as sch:Course and range as xsd:String.

c) number- a new property called sch:number was created with domain as sch:Course and range as xsd:int.

d) description- a new property called courseDesc was created with domain as sch:Course and range as xsd:String.

e) link to webpage- rdfs:seeAlso was used to link to course webpage.

f) outline- a new property called sch:outline was created with domain as course and range as rdf:resource.

3) For **Lecture**(sch:lecture), we had six data members- number, name , content ,link to webpage, link to course and link to topics covered with resource.

a) number- a new property called sch:lectNumber was created with domain as sch:lecture and range as xsd:int.

b) name- a new property called sch:lectName was created with domain as sch:lecture and range as xsd:String.

c) content- for content, we first created individual classes named as sch:Slide, sch:Worksheet, sch:Readings , sch:Others and then created a new class called sch:lectContent with new properties named as sch:hasSlides, sch:hasWorksheets, sch:hasReadings, sch:hasOthers with domain as sch:lectContent and range as those individual classes accordingly.

Now, we have a sch:lectContent with contents but not linked to lecture.

To link to a lecture, we created a new property sch:hasContent with domain as lecture and range as sch:lectContent.

In this way, we had a property in lecture sch:hasContent containing the sch:lectContent which in turn had slides, worksheets, readings and other materials in it.

d) link to webpage- rdfs:seeAlso was used to link to lecture webpage.

e) link to course-Now, we also wanted to link lecture and course for which we created a new property called sch:forCourse with domain sch:lecture and range as sch:Course.

f) Link to topics covered- a new class sch:topic was made the subclass of lecture which had two properties hasTopic linked to sch:Topic and sch:resource linked to the resource.

4) For **Lab**, we had four data members- number, name , content, link to course and link to topics covered with resource.

a) number- a new property called sch:labNumber was created with domain as sch:lab and range as xsd:int.

b) name- a new property called sch:labName was created with domain as sch:lab and range as xsd:String.

c) content- for content, we created a new class called sch:labContent with new properties named as sch:Slides, sch:Worksheets, sch:Readings, sch:Others with domain as sch:Content and range as individual classes for slides, worksheets, readings, other materials accordingly.

Now, we have a sch:labContent with contents but not linked to lab. To link to a lab we created a new property sch:Content with domain as lab and range as sch:labContent.

In this way, we had a property in lab sch:Content containing the sch:labContent which in turn had slides,worksheets,readings and other materials in it.

We also had to link lab to a lecture.

To link to a lecture, we made this sch:labContent class a subclass of sch:lecture using rdfs:subClassOf.

d) link to course-Now, we also wanted to link lab and lecture for which we created a new property called sch:forLecture with domain sch:lab and range as sch:lecture.

e) Link to topics covered- a new class sch:topic was made the subclass of lab which had two properties hasTopic linked to sch:Topic and sch:resource linked to the resource.

5) For **topics**(sch:Topic), we had three members- name, link to course, link to lecture, link to lab

a) name- a new property called sch:labName was created with domain as sch:Topic and range as xsd:String.

b) link to course- a new property called sch:coveredInCourse was created with domain sch:Topic

and range as sch:Course.

c) link to lecture- a new property called coveredInLecture with domain sch:Topic and range as sch:lecture.

d) link to lab- a new property called coveredInLab with domain sch:Topic and range as sch:lab.

Knowledge Base Statistics

Total No of Triples	47052
No of distinct topics	876
No of distinct topics in COMP6741	554
No of distinct topics in SOEN6841	378
No of common topics between COMP6741 & SOEN6841	56

a) Changes Implemented from Iteration 1–

1) As suggested, we changed the schema domain from http://example.org to http://focu.io.

2) We changed the schema structure and removed the property from class definition as it was not required.

3) We created different URI for schema and data.

4) We now had bidirectional link with course and lecture & lab content.

5)We created topic subclass in lecture and lab to contain the topic and resource URI.

6)The knowledge base stats are added.

7)Extra vide and reading links are added in other material

3. KNOWLEDGE BASE CONSTRUCTION

Knowledge Base construction required knowledge about the dataset first-

a) Dataset Generation- The dataset collection can be divided into two main parts-

- collection of data for courses
- collection of lecture and lab contents

1. Collection of data for courses- to collect the data for courses, we required data from two different csv from the mentioned open data set-https://opendata.concordia.ca/datasets

which were- CU_SR_OPEN_DATA_CATALOG and CU_SR_OPEN_DATA_CATALOG_DESC.

The first csv had all the details for the course except course description which was present in the other csv. So, we fetched the data from both the csv and merged on 'Course ID' which was the common link and removed the fields which were not required and saved the result as 'courses.csv' in the Dataset folder with columns 'Course ID', 'Course Subject',' Course Number', 'Course Name', Course Description'.

For this functionality we developed fetch_courses.py

2. Collection of lecture and lab contents-

The resources for the data and other details are mentioned in Datset.docx.

We collected all the data for the two courses and renamed lecture slides as slides[number], worksheets as worksheet[no.],readings as readings[no] and other materials as others[no.].

Now we created a new folder called CoursesDataset in the Dataset folder and this CoursesDataset had folders for two courses named as COMP6741 and SOEN6481 for which we gathered the data.

COMP6741 has two folders lecture and lab.

In lecture, there are subfolders with naming [lecture no].[lecture name], for example lecture 1 had folder named "1.IntroductionToIntelligentSystems".

For labs, as the labs are supposed to be linked to a lecture, the naming convention for folders was [lab no.].[lab name]-[lecture no], for example if lab1 is linked to lecture 1- "1.labName-1"

Same was done for course SOEN6481

Now we were ready with our Dataset.

b) Process to populate Knowledge Base-

For this we need to run fetch_courses to prepare our courses.csv file containing the data for courses and then to populate knowledge base, we developed create_KB.py by using graph from rdflib.

So, to generate knowledge base simply execute create_KB after executing fetch_courses.

It performs six functionalities-

1. add_university()- Concordia_University is added to the graph according to the schema developed and to link to dbpedia, we used OWL.sameAs.

2. add_subjectsWithCourses()- It picks all the unique subjects from courses.csv and adds all the courses related to a subject, all the details required for sch:Course was collected in courses.csv.

3. add_lecture("course") – we used os.walk in directory /Dataset/CoursesDataset/"+course+"/lecture to iterate through each lecture folder and added all the slides, worksheets, readings, other materials to the graph and also linked lecture with course by getting the course URI as URIRef(schema+course)

This was done for both the courses.

4. add_labs("course")- we used os.walk in directory /Dataset/CoursesDataset/"+course+"/lab to iterate through each lab folder and added all the slides, worksheets, readings, other materials to the graph and also linked lab with lecture by getting the lecture URI as URIRef(schema+course+"l-"+lecture no)

This was done for the labs for both the courses.

5. add_topic()- a list was prepared with topic names and added to the graph along with link to dbpedia using OWL.sameAs.

6. save_graph()- It saves the graph as 'KnowledgeBase.nt' in the directory where the code is present

TOPIC POPULATION:

For populating the topics, we needed to find topics which are there in the dbpedia database, link them to our data and add related triples to the knowledge base.

So, it involved the following three steps-

Pre-Processing- Conversion of documents from other formats to plain text files. For this, we used tika library's parser.

Code-

```
parsed_pdf = parser.from_file(path)
data = parsed_pdf['content']
```

Entity Linking-

After converting the document to text files, we needed to run dbpedia spotlight over that document. To achieve this, we had to create a local dbpedia Spotligh server as the Web API cannot handle frequent requests.

Creating dbpedia spotlight server- we downloaded dbpedia-spotlight-1.0.0.jar and the English model and started the server using command-

java -Xmx16g -d64 -jar dbpedia-spotlight-1.0.0.jar en http://localhost:2222/rest

Once the server is started, we need to send our converted text file's data to the server which would return the linked entities. We used confidence value=0.7 and support=40.

Code-

```
response=(spotlight.annotate('http://localhost:2222/rest/annotate',data,confidence=0.7,support=40))
if response!=None:
        for res in response:
                source=str(res['surfaceForm'])
                uri=str(res['URI'])
                source=source.replace(' ', '_')
                topic=schema[str(source)]
                addTopic(topic,source,uri,course,lecture,None,s)
```

response returns all the enitites found in that document. We process each of them and add triples in the graph using 'surfaceFrom' which gives the original text from which the related entity is found and 'URI' which gives the dbpedia URL.

Triplification:

After extracting various topics, the related triples are added to the knowledge graph according to the schema structure.

The topic URI is created and linked to the dbpedia uri using OWL.sameAs, it is also linked to the course using property- coveredInCourse.

If the topic is found from lecture content, a subclass of lecture called topic is created which has properties hasTopic and resource, hasTopic contains the Topic URI and resource contains the URI of the file from where the content is extracted.

If the topic is found from lab content, a subclass of lab called topic is created which has properties hasTopic and resource, hasTopic contains the Topic URI and resource contains the URI of the file from where the content is extracted.

This way lecture and labs contains subclasses of the topics which has links to the Topic URI and resource URI.

4.CHATBOT DESIGN

The chatbot is designed using Rasa framework. To translate the questions into SPARQL queries, we defined few examples of intent related to the query in nlu.yml.

For example-

- intent: desc_from_course

  examples: |

    - What is course description for [COMP6741](course)?
    - What is course description for [SOEN6841](course)?
    - What is [SOEN6841](course) about?
    - What is [COMP6741](course) about?

The variable entitiy in these questions is [course], so we declared this is the slots field in domain.yml. Also, declared the intent desc_from_course in intents column.

After this, we added the story script in stories.yml for the intent.

- story: get description from course

  steps:

    - intent: greet
    - action: utter_greet
    - intent: desc_from_course
    - action: desc_from_course_info7

desc_from_course_info7 is the name of the function which will get executed for this intent.

Then we defined this function in actions.yml making a separate class for this intent which has function name returning the name of the function 'desc_from_course_info7' and run which is the function where we define the code.

To get the user's input value for course we used (tracker.slots['course']), after that we used this course value to generate the corresponding query as –

```
query="""
        prefix sch: <http://example.org/schema#/>
        SELECT ?description
        WHERE {
        sch:"""+c+""" sch:courseDesc ?description.
        }
        """
```

After generating this query, we send this to the fuseki server using SPARQLWrapper and get the related result of the query. This result is displayed using dispatcher.utter_message(text=result)

In this way, we convert queries for all questions and collect result from fuseki server.

Examples-

Q1) Which topics are covered in COMP6741?

Output-The topics are: 2_John ,3G ,A11 ,ACM ,AI ,AI_Winter ,AM ,API ,Activision ,Adjective ,Adverb ,Alexa ,Alfred ,Alison ,Alma ,Amazon ,Amazon_AWS ,Amazon_Alexa ,Amsterdam ,Anaconda ,Andrew

,Apache ,Apache_Software_Foundation ,Apple ,Apple_Siri ,Arity ,Artificial_Intelligence ,Aspergillus
,Automata_theory ,BRUTUS ,BRaunschweig ,BSD_License ,Bahar ,Berlin ,Bill_Gates ,Blender ,Boolean
,Boolean_operators ,Bracketing ,Budapest ,CAESAR ,CALPURNIA ,CIA ,CLEOPATRA
,CONCORDIA_UNIVERSITY ,CSV ,Caesar ,Cambridge_University_Press ,Canada ,Caribbean_Series
,Chabacano ,Chapter_11 ,Chicago ,Christian ,Clang ,Clark_Kent ,Cleopatra ,Clue ,Collaborative_Filtering
,Collaborative_tagging ,ConceptNet ,Concordia ,Concordia_University ,Coreference ,Creative_Commons
,Customer_service ,Cyc ,DBPedia ,DBpedia ,DBpedia_Spotlight ,DET ,DFA ,DL ,Daily_Planet
,Data_Integration ,Data_integration ,David_Wood ,Decision_Trees ,DeepDive ,Deep_Learning ,Denver
,Description_Logic ,Determiner ,Dewey_Decimal_Classification ,Dialog ,Dimensionality_reduction ,Disney
,Docker ,Doug ,Dublin_Core ,E3 ,EGFR ,ELIZA ,EOS ,Earth ,Eastern_Orthodox ,Elastic_Search
,Elasticsearch ,Enron ,Escherichia_coli ,Euclidean_distance ,Extrapolate ,Eötvös ,F-Measure ,F-measure
,FOAF ,Facebook ,Facebook_Messenger ,Facebook_bots ,False_negative ,False_positive
,Finite_State_Machine ,First_Nations ,Flickr ,Freebase ,Fuseki ,GCC ,Gartner ,Gefitinib ,Gene_Ontology
,GeoNames ,Geonames ,Git ,Google ,Google_Assistant ,Google_Search ,Governor ,Guantanamo_Bay ,HAL
,HBO ,HTML ,HTTP ,HTTP_GET ,Hamlet ,Hannes ,Hanover ,Harper_Collins ,Heidelberg
,Hidden_Markov_Models ,Hinrich ,Howard ,Hummel ,IBM ,IDE ,IDF ,IMDB ,ISO
,Information_Retrieval_(IR) ,Information_extraction ,Information_retrieval ,Intelligent_Systems ,Internet
,Inuit ,Italian_Greyhound ,JDK ,JSON ,James ,Java ,Jena ,Jeopardy! ,Jimmy_Olsen ,John_Berryman ,Json
,Judy_Hopps ,Julius ,KB ,Kal-El ,Kitchen_Aid ,Kitchenaid ,Knowledge_Graph ,Knowledge_Vault ,Kolkata
,Krypton ,Kyle ,L2_norm ,LGPL ,LOC ,Language_processing ,Last.fm ,Layer_Cake ,Lemmatization
,Lennon ,Leonardo_da_Vinci ,Leverage ,Lex_Luthor ,LinkedIn ,Linked_Data ,Linked_Open_Data ,Linux
,Liverpool ,Lois_Lane ,London ,Loyola_College ,Luanda ,Lucene ,Luke ,Léonard_de_Vinci ,MIT ,MR
,Macbeth ,Machine_Learning ,Machine_learning ,Manhattan_distance ,Martha_Kent ,Martin ,Mary ,Max
,McGill ,McGill_University ,Metadata ,Metafiction ,Metropolis ,Mexico ,Microsoft ,Milan ,Mitsuki
,Mona_Lisa ,Montreal ,Moodle ,MusicBrainz ,Métis ,NEs ,Named_Entity_Recognition
,National_Cancer_Institute ,Natural_Language_Processing ,Natural_Language_Processing_(NLP)
,Naïve_Bayes_Classifier ,Neptune ,Netflix ,Nexus ,Nile ,Noisy_Data ,NuBus ,NumPy ,Numpy
,O'Reilly_Media ,OEM ,OWL2 ,Ontology ,Open_standard ,Orange ,Orenda ,Orinoco ,Othello ,Overfitting
,PDF ,PHP ,PMID ,POS_Tags ,POS_tags ,POWER7 ,Pablo_Picasso ,Pacman ,Page_3 ,Paradise_Papers
,Paris ,Paris_Hilton ,Parry ,Pattern_matching ,Perceptrons ,Perry_White ,Personalization ,Peter_Norvig
,Physical_quantity ,Pokémon ,Prentice_Hall ,Preposition ,Pronoun ,PubMed ,Python ,Python_IDE ,Quebec
,Question_Answering ,Question_answering ,RDF ,RDF/XML ,RDFS ,RDF_Schema ,RDF_schema ,RDFa
,RT ,Rasa ,Reading ,Reinforcement_Learning ,Resident_Evil ,Resource_Description_Framework
,Resource_Description_Framework_(RDF) ,Rich ,Roman_Catholic_Church ,SHRDLU ,SIM ,SKOS
,SMAD3 ,SOAP ,SPARQL ,SQL ,SaaS ,Satori ,Scikit-learn ,Sebastian ,Semantic_Networks ,Semantic_Web
,Sheffield ,Sir_George_Williams_University ,Siri ,Skype ,Smallville ,Smart ,Snowball ,Socrates ,Solr ,Sony
,Split ,Springer ,Stanford ,Steve_Jobs ,Stuart_Russell ,Stubbs ,Superman ,Support_Vector_Machines ,TBL
,TF-IDF ,Technetium ,Tempest ,TensorFlow ,Test ,The_Artist ,The_Lego_Movie ,Tim_Berners-Lee
,Time_complexity ,Toy_Story_2 ,Turing_Test ,Twitter ,Type_II_error ,Type_I_error ,UC_Berkeley ,URI
,UTF-8 ,Unfriended ,UniProt ,Unicode ,Uniform_Resource_Locator ,United_States ,Unix ,Usenet ,VPN
,Vector_Space ,Vector_Space_Model ,Voltaire ,W3C ,W3C_Recommendation ,Waterloo ,Web_2.0
,Web_Ontology_Language ,Web_Ontology_Language_(OWL) ,Westworld ,Whirlpool ,Wikidata
,Wikileaks ,Windows ,Winograd ,WordNet ,Wordpress.com ,World_Wide_Web
,World_Wide_Web_Consortium ,World_Wide_Web_Consortium_(W3C) ,XHTML ,XML ,Yandex
,Yangtze ,YouTube ,Zoom ,Zootopia ,adverb ,affinity_chromatography ,agent-based ,ai ,algorithm
,application_layer ,archive ,artificial-intelligence ,auteur ,automation ,azure ,banana ,basic_english
,biological_pathway ,blackboard ,calpurnia ,cat ,chatbots ,chmod ,clause ,cloud ,computer_program
,computer_science ,confidentiality ,connectionist ,contingency_table ,cosine ,cross-validation
,current_version ,dam ,data_structure ,data_type ,deterministic_finite_automaton ,dog ,dot_product ,draft ,e-
commerce ,ecosystem ,emergency_medical_services ,entailment ,eval ,exon ,fantasy ,feature_extraction
,feature_vector ,finite-state_machine ,foaf ,food_web ,formal_language ,frbr ,frog ,functional_programming
,fuseki ,g4 ,gefitinib ,gene ,genus ,ghostwriter ,glutathione_S-transferase ,goalkeeper ,gold_standard

,grammar ,grapefruit ,graph ,ground_truth ,gzip ,harmonic ,hash_table ,historical_fiction ,idf ,incidence_matrix ,instance_variable ,integral ,intelligent_agent ,international ,iq ,irreflexive ,iso ,iter ,k-means ,kernel ,knowledge_representation ,language_model ,lemma ,lemmatization ,library ,list_comprehension ,log2 ,logic ,machine_learning ,machine_learning_algorithm ,mad_scientist ,mathematical_model ,mathematical_notation ,mathematician ,matrix ,mature_technology ,memory ,metadata ,metamorphosis ,microdata ,miroirs ,mutant ,mutation ,nash ,natural_language ,newsgroup ,object-oriented_language ,object-oriented_programming ,ontologies ,ontology ,open_access ,open_data ,open_source ,open_standard ,overfit ,overfitting ,parallel_distributed_processing ,parse_tree ,parser ,part-of-speech ,pear ,personalized_medicine ,pineapple ,planet ,point_mutation ,polygon ,pop ,predicate ,programming_language ,protein ,protip ,pseudocode ,psychotherapist ,public_knowledge ,qi ,quantifier ,query_language ,query_string ,rdf ,rdfs ,recommender_system ,regression ,regular_expression ,reification ,restful ,rhombus ,rote_learning ,scikit-learn ,search_engine ,semantic ,silo ,skolemization ,social_networking ,source_code ,spa ,specification ,spruce ,static_variable ,statistical_model ,string_literal ,strings ,supervised_learning ,text_editor ,text_mining ,tf-idf ,thesauri ,tomato_sauce ,triangle ,tuple ,uri ,validation_set ,vector ,vector_space ,vector_space_model ,vigilante ,w3.org ,wan ,webcam ,wgs84 ,wine

Q2) Which topics are covered in lecture 1 for SOEN6841?

Output- AI-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

Artificial_Intelligence-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

Edge-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

Great_Wall-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

Manhattan_Project-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

PowerPoint-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

PricewaterhouseCoopers-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

Project_Management_Institute-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

Project_management-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

Software_Projects-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

Software_projects-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/sli

des01.pdf

Switzerland-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

driverless_car-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

project_management-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

smartphone-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

waterfall-
file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/lecture/1.Intro_to_Software_Project_Management/slides01.pdf

Q3) What are contents of lecture 1 for COMP6741?

Output- The contents are:
file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/lecture/1.Intelligent_Systems_Introduction/worksheet01.pdf,
file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/lecture/1.Intelligent_Systems_Introduction/slides01.pdf,

Q4) What are contents of lab 2 for COMP6741?

Output- The contents are: file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

Q5) What is the web link for COMP6741?

Output- The link is: https://moodle.concordia.ca/moodle/course/view.php?id=132738

Q6) What is the name of course with 6721 of COMP?

Output- The course is: APPLIED ARTIFICIAL INTELLIGENCE

Q7) What is MECH6741 about?

Output- Introduction to mechatronics; basic elements of mechatronic systems. Measurement systems: including principles of measurement systems; sensors and transducers; signal conditioning processes and circuits; filters and data acquisition. Actuation systems: mechanical actuation systems and electrical actuation systems. Controllers: control modes; PID controller; performance measures; introduction to digital controllers and robust control. Modeling and analysis of mechatronic systems; performance measures; frequency response; transient response analysis; stability analysis. A project.

Q8) What is the course outline for COMP6741?

Output- file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf

Q9) Give slides for lecture 1 for COMP6741?

Output-

file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/lecture/1.Intelligent_Systems_Introduction/slides01.pdf

Q10) Give worksheets for 2 for COMP6741?

Output-
file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/lecture/2.Knowledge_Graphs/worksheet02.pdf

Q11) Give lab Content for 2 for COMP6741?

Output- Readings-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

Readings-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/1.Intro_to_python-2/readings1.txt

Q12) Which topics are covered in lab 2 for COMP6741?

Output- Apache-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

Concordia-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

Java-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

Jena-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

RDF-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

RDF/XML-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

RDFS-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

RDF_Schema-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

SPARQL-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

intelligent_agent-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

open_source-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

rdfs-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

source_code-file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/labs/2.Intro_to_RDF-2/readings2.txt

Q13) Which courses cover Alexa?

Output- INTELLIGENT SYSTEMS-COMP6741

5. QUERIES

1. Which topics are covered in [course]?

->We have a property sch:coveredInCourse in sch:Topic which would help to get the answer.For example, query to get topics for course COMP6741-

```
prefix sch: <http://focu.io/schema#/>

prefix schd: <http://focu.io/data#/>

SELECT ?topic

WHERE {

 ?topic sch:coveredInCourse schd:COMP6741.

}ORDER BY(?topic)
```

Output-

```
"topic" ,

"http://example.org/schema#/Intelligent_agent" ,

"http://example.org/schema#/Intelligent_system" ,

"http://example.org/schema#/Knowledge_base" ,

"http://example.org/schema#/Knowledge_graph" ,

"http://example.org/schema#/Linked_data" ,

"http://example.org/schema#/Machine_learning" ,

"http://example.org/schema#/Ontology",

"http://example.org/schema#/Recommender_system",

"http://example.org/schema#/Text_mining",
```

2. Which topics are covered in [lecture_no] for [course]?

->We have a property sch:coveredInLecture in sch:Topic which would help to get the answer. But for that we'll need to get the lecture from lecture no and course.For example, query to get topics for lecture no.1 of course COMP6741-

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

prefix sch: <http://focu.io/schema#/>
```

```
prefix schd: <http://focu.io/data#/>


SELECT ?topic
WHERE {
 ?lecture sch:lectNumber "1"^^xsd:int.
 ?lecture sch:forCourse schd:COMP6741.
 ?topic sch:coveredInLecture ?lecture.
}ORDER BY(?topic)
```

Output-

```
"topic" ,
 "http://example.org/schema#/Intelligent_Systems_Introduction" ,
```

3. What lecture content was covered in [lecture no.] for [course]?

->We have sch:hasContent in sch:lecture to get the answer but to get the lecture we can use lecture no and course. For example- query to get contents for lecture no.1 of course COMP6741-

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix sch: <http://focu.io/schema#/>
prefix schd: <http://focu.io/data#/>


SELECT ?content ?property ?value
WHERE {
 ?lecture sch:lectNumber "1"^^xsd:int.
 ?lecture sch:forCourse schd:COMP6741.
 ?lecture sch:hasContent ?content.
 ?content ?property ?value.
 FILTER(?property!=rdf:type)
}
```

Output-

"content" , "property" , "value" ,

"http://example.org/schema#/COMP6741content-lecture1" , "http://example.org/schema#/hasWorksheets" , "file:///C:/Users/Kshitij/PycharmProjects/Assignment1/src/Dataset/CoursesDataset/COMP6741/lecture/1. Intelligent_Systems_Introduction/worksheet01.pdf" ,

"http://example.org/schema#/COMP6741content-lecture1" , "http://example.org/schema#/hasSlides" , "file:///C:/Users/Kshitij/PycharmProjects/Assignment1/src/Dataset/CoursesDataset/COMP6741/lecture/1. Intelligent_Systems_Introduction/slides01.pdf" ,

4. What lab content was covered in [lab no.] for [course]?

->We have sch:Content in sch:lab to get the answer but to get the lab, we can use lab no and course. There's no direct link between course and lab.So, we have to go through lecture as lecture and lab are connected. We first find all lectures for the course,then find all the labs for the lectures, then find the labs with given lab no. and finally through this lab we get the content. For example- query to get contents for lab no.1 of course COMP6741-

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

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

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

prefix sch: <http://focu.io/schema#/>

prefix schd: <http://focu.io/data#/>

SELECT ?content ?property ?value

WHERE {

 ?lecture sch:forCourse schd:COMP6741.

 ?lab rdfs:subClassOf ?lecture.

 ?lab sch:labNumber "1"^^xsd:int.

 ?lab sch:Content ?content.

 ?content ?property ?value.

 FILTER(?property!=rdf:type).

}
```

Output-

"content" , "property" , "value" ,

"http://example.org/schema#/COMP6741content-lab1" , "http://example.org/schema#/Readings" , "file:///C:/Users/Kshitij/PycharmProjects/Assignment1/src/Dataset/CoursesDataset/COMP6741/labs/1.Intro_to_python-2/readings1.txt" ,

5. What is the web link for a [course]?

->We can use refs:seeAlso on the course to find the answer.For example, web link to the course COMP6741-

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

prefix sch: <http://focu.io/schema#/>

prefix schd: <http://focu.io/data#/>

SELECT ?link

WHERE {

 schd:COMP6741 rdfs:seeAlso ?link.

}
```

Output-

"link" ,

"https://moodle.concordia.ca/moodle/course/view.php?id=132738" ,

6. What is the name of course with [course number] of [subject]?

->We can use sch:number and sch:subject to get the course and then sch:courseName to get the answer. For example, name of the course with course number 6741 and subject COMP-

```
prefix sch: <http://focu.io/schema#/>

SELECT ?name

WHERE {

 ?course sch:number "6741".

 ?course sch:subject "COMP".

 ?course sch:courseName ?name.

}
```

Output-

"name" ,

"INTELLIGENT SYSTEMS" ,

7. What is course description for [course]?

->We can use sch:courseDesc to find the answer. For example, description for course MECH6741-

prefix sch: <http://focu.io/schema#/>

prefix schd: <http://focu.io/data#/>

SELECT ?description

WHERE {

schd:MECH6741 sch:courseDesc ?description.

}

Output-

"description" ,

"Introduction to mechatronics; basic elements of mechatronic systems. Measurement systems: including principles of measurement systems; sensors and transducers; signal conditioning processes and circuits; filters and data acquisition. Actuation systems: mechanical actuation systems and electrical actuation systems. Controllers: control modes; PID controller; performance measures; introduction to digital controllers and robust control. Modeling and analysis of mechatronic systems; performance measures; frequency response; transient response analysis; stability analysis. A project." ,

8. What is the course outline for [course]?

->We can use sch:outline to find the answer. For example, outline for course COMP6741

prefix sch: <http://focu.io/schema#/>

prefix schd: <http://focu.io/data#/>

SELECT ?outline

WHERE {

schd:COMP6741 sch:outline ?outline.

}

Output-

"outline" ,

"file:///C:/Users/Kshitij/PycharmProjects/Assignment1/src/Dataset/CoursesDataset/COMP6741/outline.pdf" ,

9. Give slides for [lecture no] for [course]?

->We first find the lecture using sch:lectNumber and sch:Course, then sch:hasContent to find all the contents of that lecture and finally use sch:hasSlides to get the slides. For example, to get the slides of lecture 1 for course COMP6741-

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

prefix sch: <http://focu.io/schema#/>

prefix schd: <http://focu.io/data#/>

SELECT ?slides

WHERE {

?lecture sch:lectNumber "1"^^xsd:int.

?lecture sch:forCourse schd:COMP6741.

?lecture sch:hasContent ?content.

?content sch:hasSlides ?slides.

}

Output-

"slides" ,

"file:///C:/Users/Kshitij/PycharmProjects/Assignment1/src/Dataset/CoursesDataset/COMP6741/lecture/1. Intelligent_Systems_Introduction/slides01.pdf" ,

10. Give worksheets for [lecture no] for [course]?

->We first find the lecture using sch:lectNumber and sch:Course, then sch:hasContent to find all the contents of that lecture and finally use sch:hasWorksheets to get the slides. For example, to get the worksheets of lecture 1 for course COMP6741-

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

prefix sch: <http://focu.io/schema#/>

prefix schd: <http://focu.io/data#/>

SELECT ?worksheet WHERE {

?lecture sch:lectNumber "1"^^xsd:int.

```
 ?lecture sch:forCourse schd:COMP6741.
 ?lecture sch:hasContent ?content.
 ?content sch:hasWorksheets ?worksheet.
}
```

Output-

"worksheet" ,

"file:///C:/Users/Kshitij/PycharmProjects/Assignment1/src/Dataset/CoursesDataset/COMP6741/lecture/1. Intelligent_Systems_Introduction/worksheet01.pdf" ,

11. Give lab Content for [lecture no] for [course]?

->We first find the lecture using sch:lectNumber and sch:Course, then find all the labs related to this lecture and finally use sch:Content to get content of those labs. For example, lab contents related to lecture no.3 for course COMP6741.

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

SELECT ?content ?property ?value
WHERE {
 ?lecture sch:lectNumber "3"^^xsd:int.
 ?lecture sch:forCourse schd:COMP6741.
 ?lab rdfs:subClassOf ?lecture.
 ?lab sch:Content ?content.
 ?content ?property ?value.
 FILTER(?property!=rdf:type)
}
```

Output-

"content" , "property" , "value" ,

"http://example.org/schema#/COMP6741content-lab3" , "http://example.org/schema#/Readings" , "file:///C:/Users/Kshitij/PycharmProjects/Assignment1/src/Dataset/CoursesDataset/COMP6741/labs/3.First_Intelligent_Agent-3/readings3.txt" ,

12. What topics are covered in [course]?

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
prefix sch: <http://focu.io/schema#/>
prefix schd: <http://focu.io/data#/>
SELECT ?topic_label ?lecture ?lab ?fromoutline ?fromlecture ?fromlab
WHERE {
  {
  ?topic sch:coveredInCourse schd:COMP6741.
  ?topic rdfs:label ?topic_label.
  ?topic sch:resource ?fromoutline
  }
  UNION{
  ?lecture sch:forCourse schd:COMP6741.
  ?s rdfs:subClassOf ?lecture.
  ?s sch:hasTopic ?topic.
  ?s sch:resource ?fromlecture.
  }
  UNION{
  ?lecture sch:forCourse schd:COMP6741.
  ?lab rdfs:subClassOf ?lecture.
  ?d rdfs:subClassOf ?lab.
  ?d sch:hasTopic ?topic.
  ?d sch:resource ?fromlab.
  }
}
```

Output:

(rdflib.term.Literal('Orenda', lang='en'), None, None, rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)

(rdflib.term.Literal('Orenda', lang='en'), None, None, rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)

(rdflib.term.Literal('Linked_Open_Data', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Hummel', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('Hummel', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('AI', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('agent-based', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Kyle', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Kyle', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('iso', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('iso', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('Artificial_Intelligence', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Page_3', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('Page_3', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('First_Nations', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('First_Nations', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Concordia', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Concordia', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('Inuit', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('Inuit', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('RDF', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('programming_language', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Intelligent_Systems', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('RDFS', lang='en'), None, None,

rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('webcam', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('webcam', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('ISO', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('Java', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('dam', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('dam', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Métis', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('Métis', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Python', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('CONCORDIA_UNIVERSITY', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('CONCORDIA_UNIVERSITY', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Natural_Language_Processing_(NLP)', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Information_Retrieval_(IR)', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Machine_Learning', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Moodle', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('Moodle', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('emergency_medical_services', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('emergency_medical_services', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('SPARQL', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('blackboard', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Personalization', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)

(rdflib.term.Literal('IDE', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Concordia University', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Concordia_University', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/outline.pdf'), None, None)
(rdflib.term.Literal('Concordia University', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)
(rdflib.term.Literal('Concordia_University', lang='en'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/SOEN6841/outline.pdf'), None, None)

13.For a given topic t (DBpedia URI), list all [courses] where they appear.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
prefix sch: <http://focu.io/schema#/>
prefix schd: <http://focu.io/data#/>

SELECT ?course (sum(?topicCount) as ?count)
WHERE {
 {
 SELECT ?course (count(*) as ?topicCount)
 WHERE
 { ?topic owl:sameAs <http://dbpedia.org/resource/Artificial_intelligence> .
      ?topic sch:coveredInCourse ?course.
      ?lecture sch:forCourse ?course.
      ?s rdfs:subClassOf ?lecture.
      ?s sch:hasTopic ?topic.

 }GROUP BY ?course
 }UNION
 {
 SELECT ?course (count(*) as ?topicCount)
 WHERE
 {
 ?topic owl:sameAs <http://dbpedia.org/resource/Artificial_intelligence> .
 ?topic sch:coveredInCourse ?course.
 ?lecture sch:forCourse ?course.
 ?lab rdfs:subClassOf ?lecture.
```

```
  ?d rdfs:subClassOf ?lab.
  ?d sch:hasTopic ?topic.
  }GROUP BY ?course
  }
}GROUP BY ?course
ORDER BY ?topicCount
```

Output:

```
(rdflib.term.URIRef('http://example.org/schema#/SOEN6841'), rdflib.term.Literal('2',
datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#integer')))
(rdflib.term.URIRef('http://example.org/schema#/COMP6741'), rdflib.term.Literal('20',
datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#integer')))
```

14.For a given topic t , list the precise course URI, course event URI and corresponding resource URI

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
prefix sch: <http://focu.io/schema#/>
prefix schd: <http://focu.io/data#/>

SELECT ?course ?lecture ?lab ?fromoutline ?fromlecture ?fromlab
WHERE {
  {
  schd:Knowledge_Graph sch:coveredInCourse ?course.
  schd:Knowledge_Graph rdfs:label ?topic_label.
  schd:Knowledge_Graph sch:resource ?fromoutline
  }
  UNION{
  ?lecture sch:forCourse ?course.
  ?s rdfs:subClassOf ?lecture.
  ?s sch:hasTopic schd:Knowledge_Graph.
  ?s sch:resource ?fromlecture.
  }
  UNION{
  ?lecture sch:forCourse ?course.
  ?lab rdfs:subClassOf ?lecture.
  ?d rdfs:subClassOf ?lab.
  ?d sch:hasTopic schd:Knowledge_Graph.
```

```
  ?d sch:resource ?fromlab.
  }
}
```

Output:

```
(rdflib.term.URIRef('http://example.org/schema#/COMP6741'),
rdflib.term.URIRef('http://example.org/schema#/COMP6741-l2'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/lecture/2.Knowledge_Graphs/slides02.pdf'), None)
(rdflib.term.URIRef('http://example.org/schema#/COMP6741'),
rdflib.term.URIRef('http://example.org/schema#/COMP6741-l5'), None, None,
rdflib.term.URIRef('file:///D:/Chatbot/Dataset/CoursesDataset/COMP6741/lecture/5.Linked_Open_Data_and_Applications/slides05.pdf'),None)
```