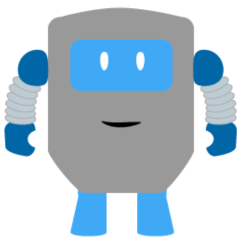
Concordia University Engineering and Computer Science

COMP 6741

Intelligent Systems

Project Report

Studybot



Submitted To: Prof. Dr. René Witte

Date: 23 April, 2021

Team **FL_G_02**

Indrayan Ittoo (40044893)

Raj Mistry (40119206)

Table of Contents

1. Abstract

Students often have a lot of questions about courses, course topics, course materials and so on. To answer these questions, they often have to navigate through several webpages, or try to ask friends who have completed the same course. This project proposed a new "friend" referred as Studybot, which is an intelligent agent that can answer university-related questions using a knowledge graph and natural language processing.
In the first part of the project, we are going to focus on building the knowledge graph. Therefore, we are going to explore the approach to build the knowledge graph, the vocabularies used or extended.
Then we are going to write a series of competency questions which will ascertain that the knowledge graph can support these queries.

2. Knowledge base

2.1 Vocabulary

For this project, we explored some of the public vocabularies, including:

- RDF
- FOAF
- OWL
- Dublin Core
- VIVO
- BIBO
- EVENT
- SKOS

Most of the vocabularies and ontologies does not completely fit the requirements and they have to be extended or should use other vocabularies.
The VIVO vocabulary [5][6] provides a good and complete option, but for the sake of exploring, we decided to go with Dublin Core [3].

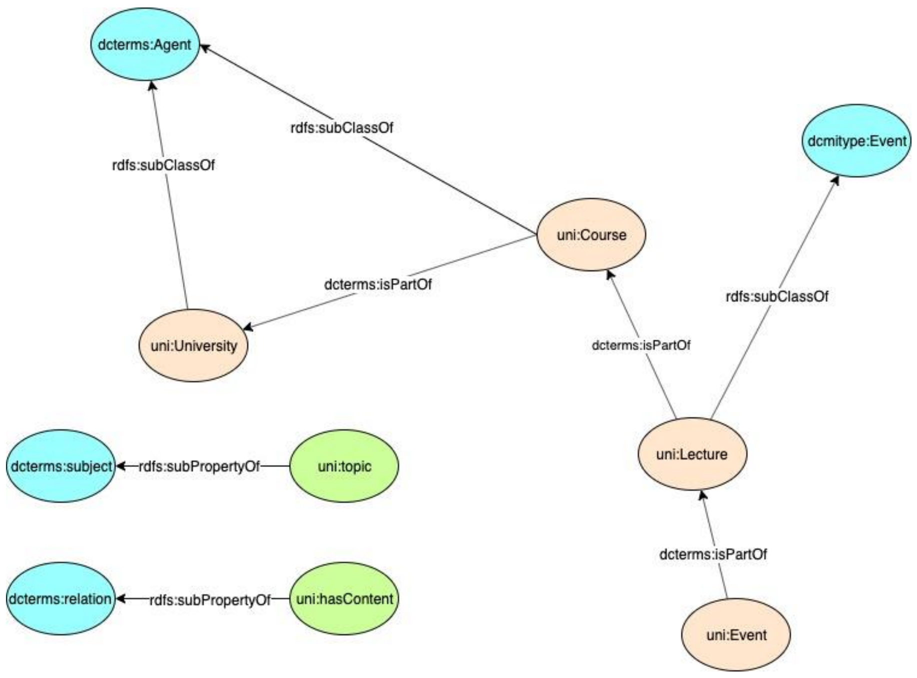
There are fifteen terms of the Dublin Core™ Metadata Element Set (also known as "the Dublin Core") plus several dozen properties, classes, datatypes, and vocabulary encoding schemes. The "Dublin Core" plus these extension vocabularies are collectively referred to as "DCMI metadata terms" ("Dublin Core terms" for short). These terms are intended to be used in combination with metadata terms from other, compatible vocabularies in the context of application profiles.

DCMI metadata terms are expressed in RDF vocabularies for use in Linked Data. Creators of non-RDF metadata can use the terms in contexts such as XML, JSON, UML, or relational databases by disregarding both the global identifier and the formal implications of the RDF-specific aspects of term definitions. Such users can take domain, range, sub property, and subclass relations as usage suggestions and focus on the natural-language text of definitions, usage notes, and examples.

The Dublin core provides a good basis for building out own vocabulary.

2.2 Schema

The diagram gives and overview of the main classes and properties extended from Dublin Core.



2.2.1 Prefixes

Prefix	URI	Comments
rdfs	<http://www.w3.org/2000/01/rdf-schema#>	
rdf	<http://www.w3.org/1999/02/22-rdf-syntax-ns#>	
xsd	<http://www.w3.org/2001/XMLSchema#>	
dbo	<http://www.dbpedia.org/ontology/>	
dbr	<http://www.dbpedia.org/resource/>	
dcterms	<http://purl.org/dc/terms/>	
dcmitype	<http://purl.org/dc/dcmitype/>	
dc	<http://purl.org/dc/elements/1.1/>	
uni	<http://uni.io/schema#>	
unidata	<http://uni.io/data#>	

2.2.2 Specifications

The following table provide the vocabulary that has been used in the schema

Vocabulary	Type of Term	Comments
rdf	rdf:type	Defines a type, used “a”
	rdf:Property	Defines a property
rdfs	rdfs:Class	Identifies a class
	rdfs:subClassOf	Identifies a sub class and the class from which it is sub classed
	rdfs:domain	Identifies the domain
	rdfs:range	Identifies the range
	rdfs:seeAlso	Provides a link to a related resource such as website
	rdfs:label	Provide a label
	rdfs:comment	Provides a description
dcterms	dcterms:Agent	A resource that acts, or has the power to act, in that case a University or Course
	dcterms:subject	A topic of the resource. Can use a URI or literal
	dcterms:relation	A related resource. Recommended to identify resource by a URI, but can use a string conforming formal identification
	dcterms:BibliographicResource	A bibliographic resource, for example a book, article or other documentary resource
dcmitype	dcmitype:Event	A class that is non-persistent, time based occurrence

2.2.3 Class Modeling

This section provides a description of the different terms used in the schema and for building the data components

University: Class

University is defined as a Class, and extends the Agent class from Dublin Core. The University is the primary agent that has the power to act

Schema	Data example

uni:University
  a rdfs:Class ;
  rdfs:subClassOf dcterms:Agent ;
  rdfs:label "University"@en ;
	unidata:Concordia_University
	  a uni:University ;
	  dcterms:title "Concordia University"@en ;
	  rdfs:seeAlso dbr:Concordia_University .

rdfs:comment "University information" .	

Course: Class
Course is defined as a Class, and extends the Agent class from Dublin Core. The class is an agent that has the power to act. The following terms are also used:

Schema	Data example
uni:Course a rdfs:Class ; rdfs:subClassOf dcterms:Agent ; rdfs:label "Course"@en ; rdfs:comment "Courses offered at University"@en .	unidata:COMP6741 a uni:Course ; dcterms:title "Intelligent Systems"@en ; dcmitype:subject "COMP" ; dcmitype:identifier "6741"; dcterms:description "Knowledge representation"@en ; rdfs:seeAlso <https://moodle.concordia.ca> ; uni:topic dbr:Intelligent_Systems ; dcterms:isPartOf unidata:Concordia_University ; uni:hasContent unidata:Doc1 .

Terms associated with Course:

dcterms:title	Identifies the name of the course
dcmitype:subject	Identifies subject of course, e.g. COMP, SOEN
dcmitype:identifier	Identifies the number of the course, e.g 6741
dcterms:description	Provides a description of the course
rdfs:seeAlso	Link to webpage with the course information
uni:topic	Identifies topic of course, linked to DBpedia
dcterms:isPartOf	Identifies the University which course belongs to
uni:hasContent	Content, mainly the course outline

Lecture: Class
A lecture is an event that belongs to a course. Lecture extens the event class from Dublin Core as it is a non-persistent time base occurence

Schema	Data example
uni:Lecture a rdfs:Class ; rdfs:subClassOf dcmitype:Event ; rdfs:label "Lecture"@en ; rdfs:comment "Information about lecture"@en .	unidata:COMP6741L01 a uni:Lecture ; dcmitype:identifier "1"; dcterms:title "Introduction to Intelligent S."@en ; rdfs:seeAlso <https://moodle.concordia.ca> ; uni:topic dbr:Intelligent_Systems ; uni:hasContent unidata:Doc02 ; dcterms:isPartOf unidata:COMP6741 .

Terms associated with Lecture:

dcmitype:identifier	Identifies lectures number
dcterms:title	Identifies the lecture name
rdfs:seeAlso	Link to webpage with the lecture information
uni:topic	Identifies topic of a lecture, linked to DBpedia
dcterms:isPartOf	Identifies the course to which lecture belongs to
uni:hasContent	Content, any documents that is associated with lecture such as slides, worksheets, etc

Event: Class
An event extends a lecture as it is associated with the lecture and is also time based. It can be any event associated with a lecture, such as a tutorial or lab

Schema	Data example
uni:Event a rdfs:Class ; rdfs:subClassOf uni:Lecture ; rdfs:label "Lecture Event"@en ; rdfs:comment "Events associated with a Lecture such as Lab ror Tutorial"@en .	unidata:L674101 a uni:Event ; dcterms:type "LAB" ; uni:isPartOf unidata:COMP6741L01 ; uni:topic dbr:Python .
Terms associated with Event:	
dcterms:type	Identifies type of event, can be lab, tutorial or some other event
dcterms:isPartOf	Identifies the lecture to which event belongs to
uni:hasContent	Identifies content specific to that lab
uni:topic	Identifies topic of lab, linked to DBpedia

2.2.4 Property Modeling

topic: Property A topic is a sub property of subject, which identifies a link to DBpedia	
Schema	Data example
uni:topic a rdf:property ; rdfs:subPropertyOf dcterms:subject ; rdfs:label "Topic"@en ; rdfs:comment "URI"@en .	unidata:L674101 a uni:Event ; dcterms:type "LAB" ; uni:isPartOf unidata:COMP6741L01 ; uni:topic dbr:Python .

hasContent: Property Identifies the contents that are associated with a course, lecture or event	
Schema	Data example
uni:hasContent a rdf:property ; rdfs:subClassOf dcterms:relation ; rdfs:label "Content"@en ; rdfs:range dcterms:BibliographicResource rdfs:comment "Content associated with a course, lecture"@en .	unidata:COMP6741L01 a uni:Lecture ; dcmitype:identifier "1"; dcterms:title "Introduction to Intelligent Systems"@en ; rdfs:seeAlso <https://moodle.concordia.ca> ; uni:topic dbr:Intelligent_Systems ; uni:hasContent unidata:Doc02 ; dcterms:isPartOf unidata:COMP6741 .
The content itself is not described as part of the schema. The content is a BibliographicResource defined as follows: unidata:Doc06 a dcterms:BibliographicResource ; dcterms:type "LECTURE" ; dcterms:source /University/COMP6741/slides03.pdf> .	
dcterms:type	Identifies the type of content, e.g. OTHER, SLIDES, WORKSHEET
dcterms:source	Locates the resource

3. Data extraction

The data was extracted from the https://opendata.concordia.ca/datasets/. The following files were used to create the .csv files we require for the project:

- CU_SR_OPEN_DATA_CATALOG-37296852.csv (To get Course ID, Subject, Number, Title)

- ➢ CU_SR_OPEN_DATA_CATALOG_DESC.csv (To get Course Description)
- ➢ CATALOG.csv and Experiential Learning.csv (To get Course Website/seeAlso)

The required files for project were generated by doing some work manually like extracting required columns from, removing duplicates, adding seeAlso data, data for two courses and other work like merging 2 .csv files based on common column value is done using small python script: data.py.

```
import pandas as pd

data1 = pd.read_csv("Data.csv",encoding= 'latin1')
data2 = pd.read_csv("DES.csv",encoding= 'latin1')
#merge function by setting how='left'
output = pd.merge(data1, data2, on='Course ID', how='left')
output.to_csv("output.csv")

data3 = pd.read_csv("output.csv",encoding= 'latin1')
data4 = pd.read_csv("WEBSITE.csv",encoding= 'latin1')
output1 = pd.merge(data3, data4, on='Course ID', how='left')
output1.to_csv("course.csv")
```

data.py

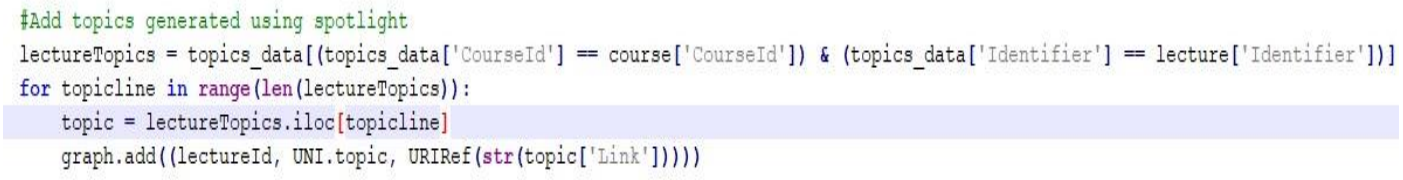
The following files are generated and used for creating Knowledge Graph:

1. course_data.csv: contains all the courses and descriptions extracted from opendata.concordia.
2. lecture_data.csv: contains the lectures information for 2 courses that were manually updated.
3. event_data.csv: contains the lab events information for 2 courses that were manually updated.
4. content_data.csv: contains the content information for 2 courses that were manually updated.
5. lab_content_data.csv: contains the lab content for 2 courses that were manually added.
6. topics.csv : contains topics for 2 courses for lectures generated from dbpedia spotlight sunig lecture materials(slides and Worksheets)
7. lab_topics.csv : contains topics for 2 courses for labs generated from dbpedia spotlight sunig lab material

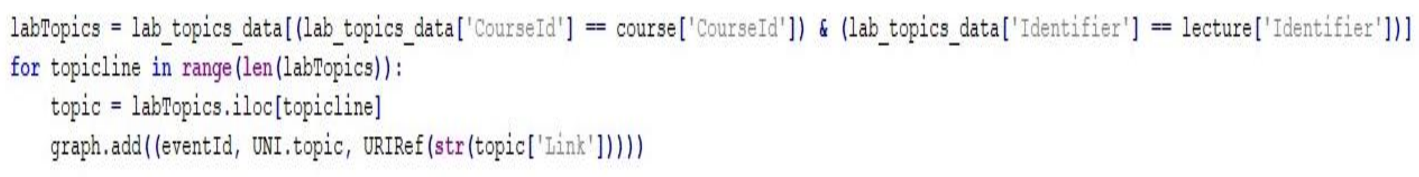
3.1 Topic Generation

⇨ In order to generate topics for the two courses we followed the following steps:

- ➢ First we collected all the lecture materials for the 2 courses i.e. all the lecture slides, worksheets and lab pdfs (for COMP 6741 we saved content from moodle into pdf)
- ➢ After that we converted these pdfs into separate text file using **'pdftotxt.py'** file. While creating this text file we removed duplicate lines in text file for particular material (slide01 , worksheet02,etc) in order to minimize unnecessary duplicate requests on dbpedia spotlight.
- ➢ Than we used this text files in **'topics_generation.py'** file which is reading this files line by line and making request on local dbpedia spotlight server to get topics from it. During this for lecture we removed the duplicate topics for particular lecture and lab.
- ➢ This topics are saved in topics.csv and lab_topics.csv file.
- ➢ After this we updated **'kbuilder.py'** which is used to generate knowledge graph.
- ➢ We read the 2 topics csv files and iterate through its line to add topics for 2 courses.



```
#Add topics generated using spotlight
lectureTopics = topics_data[(topics_data['CourseId'] == course['CourseId']) & (topics_data['Identifier'] == lecture['Identifier'])]
for topicline in range(len(lectureTopics)):
    topic = lectureTopics.iloc[topicline]
    graph.add((lectureId, UNI.topic, URIRef(str(topic['Link']))))
```



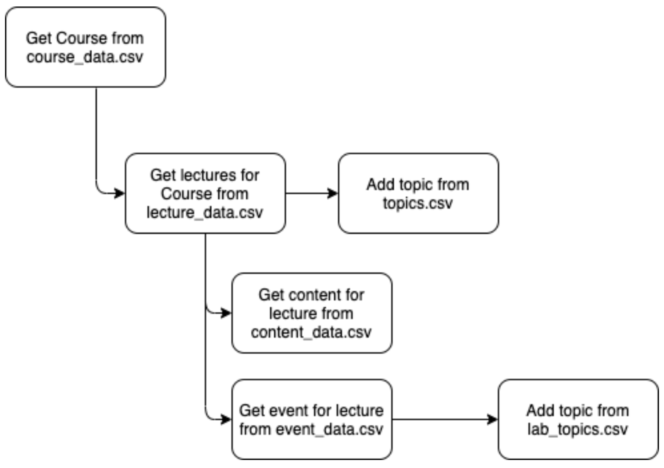
```
labTopics = lab_topics_data[(lab_topics_data['CourseId'] == course['CourseId']) & (lab_topics_data['Identifier'] == lecture['Identifier'])]
for topicline in range(len(labTopics)):
    topic = labTopics.iloc[topicline]
    graph.add((eventId, UNI.topic, URIRef(str(topic['Link']))))
```

4. Knowledge base

The building of the knowledge base uses python programming language and the following librairies:

1. pandas – for reading the .csv data files
2. uuid – for generating unique identifies for the resources
3. rdflib –
    a. Graph – for creating knowledge graph
    b. Namespace – to manage the namespace use and creation
    c. RDFS, RDF DC, DCTERMS – provided namespace in rdflib
    d. URIRef – for creating URI
    e. Literal – for adding literal vales, not URI

The following diagram give a general flow of the code structure.


Get Course from course_data.csv
Get lectures for Course from lecture_data.csv
Add topic from topics.csv
Get content for lecture from content_data.csv
Get event for lecture from event_data.csv
Add topic from lab_topics.csv

- The university is added through the code to the graph

```
#Add Univeristy to graph
graph.add( (URIRef(UNIDATA.Concordia_University), RDF.type , UNI.University))
graph.add( (URIRef(UNIDATA.Concordia_University), DCTERMS.title , Literal("Concordia University", lang="en")))
graph.add( (URIRef(UNIDATA.Concordia_University), RDFS.seeAlso, DBR.Concordia_University))
```

- For each line in the course file, the information for the course is read, and inserted into the graph. The ID of the course is obtained from the course_data.csv file

```
for courseline in range(len(course_data)):
  course = course_data.iloc[courseline]

  courseId = UNIDATA[str(course['CourseId'])]
  graph.add((courseId, RDF.type, UNI.Course))
….
```

- For each course, the lecture for the course and information of lectures are inserted in graph The ID of the lecture is a unique Id generated. Given temporal existence of lecture.

```
#For each course, add lectures
  courseLectures = lecture_data[lecture_data['CourseId'] == course['CourseId']]
  for lectureline in range(len(courseLectures)):
    lecture = courseLectures.iloc[lectureline]
```

```
                lectureId = UNIDATA["l" + str(uuid4())]
                graph.add((lectureId, RDF.type, UNI.Lecture))
           ...
```

- For each lecture, the events (lab or tutorial) and the contents of the lecture are inserted in the graph

```
                #Content associated withe a lecture
           lectureContents = content_data[(content_data['CourseId'] == course['CourseId']) & (content_data['Identifier'] ==
           lecture['Identifier'])]
                for contentline in range(len(lectureContents)):
                  content = lectureContents.iloc[contentline]
           ....
      #Events (labs/tutorials) associated with a lecture
       lectureEvents = event_data[(event_data['CourseId'] == course['CourseId']) & (event_data['Identifier'] ==
lecture['Identifier'])]
       for eventline in range(len(lectureEvents)):
         event = lectureEvents.iloc[eventline]
         eventId = UNIDATA["e" + str(uuid4())]
```

- Content creation. A unique ID is created for each content
  Local content should be place in the fuseki server in the /webapp folder. The namespace LOCAL was used in the code

```
           def add_documents(location, type):
         contenId = UNIDATA["c" + str(uuid4())]
             graph.add((contenId, RDF.type, DCTERMS.BibliographicResource))
             graph.add((contenId, DCTERMS.type, Literal(type)))
             graph.add((contenId, DCTERMS.source, URIRef(location)))
             return contenId
```

5. Queries

⇨ The following queries have been used to test the graph. The queries can be found in the accompanying queries.sparql file. In the report, we will limit to 10 queries, but more are defined in the queries.sparql file

```
# Question 1
# What is the course [title] and [description] of [subject] [number]?
# What is the course title and description of COMP 6741?
SELECT ?title ?descr
WHERE {
   ?course dcmitype:subject "COMP" .
   ?course dcmitype:identifier "6741" .
   ?course dcterms:title ?title .
   ?course dcterms:description ?descr
}

# Question 2
# Which topics are covered in [subject] [number] lectures?
# Which topics are covered in [COMP] [6741] lectures?
SELECT ?course ?lecture ?topic
WHERE {
   ?course1 dcmitype:subject "COMP" .
   ?course1 dcmitype:identifier "6741" .
   ?course1 dcterms:title ?course .
   ?lecture1 dcterms:isPartOf ?course1 .
   ?lecture1 dcmitype:identifier ?lecture .
   ?lecture1 uni:topic ?topic
} ORDER BY ?lecture

# Question 3
# Which lecture of [subject][number] covers [topic]?
# Which lecture of [COMP][6741] covers [Knowledge_grap]?
SELECT ?number
WHERE {
    ?course dcmitype:subject "COMP" .
    ?course dcmitype:identifier "6741" .
    ?lecture dcterms:isPartOf ?course .
    ?lecture uni:topic dbr:Knowledge_Graph .
    ?lecture dcmitype:identifier ?number
}

# Question 4
# How many courses are offered at Concordia University?
SELECT (count(?courseId) as ?CourseCount)
WHERE{
    ?courseId rdf:type uni:Course
}

# Question 5
# What are the recommended reading materials for [subject][number] by lecture?
# What are the recommended reading materials for [COMP][6741] by lecture?
SELECT ?number ?source
WHERE {
    ?course dcmitype:subject "COMP" .
    ?course dcmitype:identifier "6741" .
    ?lecture uni:hasContent ?content .
    ?lecture dcterms:isPartOf ?course .
    ?lecture dcmitype:identifier ?number .
    ?content dcterms:type "READING" .
    ?content dcterms:source ?source .
} order by ?number


# Question 6
# What are the contents in [subject][number] for each lecture?
# What are all the contents for COMP 6741 for each lecture?
SELECT ?number ?content_type ?source
WHERE {
    ?course dcmitype:subject "COMP" .
    ?course dcmitype:identifier "6741" .
    ?lecture uni:hasContent ?content .
    ?lecture dcterms:isPartOf ?course .
    ?lecture dcmitype:identifier ?number .
    ?content dcterms:type ?content_type .
    ?content dcterms:source ?source .
} order by ?number

# Question 7
# List the courses for subject [Subject]?
# List the courses for subject COMP?

SELECT (concat(?subject, " ", ?number) AS ?courseName) ?title
WHERE {
    ?course dcmitype:subject "COMP" .
    ?course dcmitype:subject ?subject .
    ?course dcmitype:identifier ?number .
    ?course dcterms:title ?title .
} ORDER BY ?number
```

```
#Question 8
# What is the outline for [Subject] [number] ?
# What is the outline for COMP 6741 ?
SELECT  ?outline
WHERE {
   ?course rdf:type uni:Course .
   ?course dcmitype:subject "COMP" .
   ?course dcmitype:identifier "6741" .
   ?course uni:hasContent ?content .
   ?content dcterms:type "OUTLINE" .
   ?content dcterms:source ?outline .
}

# Question 9
# Which courses cover [topic]?
# Which courses cover [Machine_learning]?
SELECT (concat(?subject, " ", ?number) AS ?courseName)
WHERE {
   ?course dcmitype:subject ?subject .
   ?course dcmitype:identifier ?number .
   ?lecture dcterms:isPartOf ?course .
   ?lecture uni:topic dbr:Machine_learning .
}


# Question 10
# Does [subject][number] has [event]?
# Does [COMP][6841] has [LAB]?
ASK{
   ?course dcmitype:subject "COMP" .
   ?course dcmitype:identifier "6741" .
   ?lecture dcterms:isPartOf ?course .
   ?event dcterms:isPartOf ?lecture .
   ?event dcterms:type "LAB".
}
```

⇨ Following are the 3 queries asked in Project 2:

# Query 1
# For a course c, list all covered topics t, printing out their English labels and their DBpedia URI, together with the course event URI (e.g., 'lab3') and resource URI (e.g., 'slides10') where they appeared.

- For this Query we have two separate queries Q1_1 and Q1_2 for lecture and lab topics.
- Q1_1

```
SELECT ?topic ?lectureID ?resource
WHERE {
  ?course dcmitype:subject "COMP".
  ?course dcmitype:identifier "6741".
  ?lecture dcterms:isPartOf ?course .
  ?lecture uni:topic ?topic.
  ?lecture dcmitype:identifier ?lectureID.
  ?lecture uni:hasContent ?content .
  ?content dcterms:type "SLIDES" .
  ?content dcterms:source ?resource .
} ORDER BY ?lectureID
```

- Q1_2

```
SELECT ?topic ?labID ?resource
WHERE {
  ?course dcmitype:subject "COMP".
  ?course dcmitype:identifier "6741".
  ?lecture dcterms:isPartOf ?course .
  ?event dcterms:isPartOf ?lecture .
  ?event dcterms:type "LAB".
  ?event uni:topic ?topic .
  ?event dcmitype:identifier ?labID.
  ?event uni:hasContent ?content.
  ?content dcterms:source ?resource.
} ORDER BY ?labID
```

# Query 2
# For a given topic t (DBpedia URI), list all courses where they appear, together with a count, sorted by frequencyQ2

- Q2

```
SELECT ?courseName (count (?courseName) as ?count)
WHERE {

?course rdf:type uni:Course.
?lecture dcterms:isPartOf ?course .
?event dcterms:isPartOf ?lecture .
?event dcterms:type "LAB".
{?lecture uni:topic dbr:Machine_learning}  UNION {?event uni:topic dbr:Machine_learning }.
?course dcterms:title ?courseName.

}  GROUP BY ?courseName ORDER BY DESC(?count)
```

# Query 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 → Lab Notes)

- For this Query we have two separate queries Q3_1 and Q3_2 for lecture and lab topics.
- Q3_1

```
SELECT ?courseName ?lectureID ?resource
WHERE {
  ?course rdf:type uni:Course.
  ?lecture dcterms:isPartOf ?course .
  ?course dcterms:title ?courseName.
  ?lecture uni:topic dbr:Machine_learning.
  ?lecture dcmitype:identifier ?lectureID.
  ?lecture uni:hasContent ?content.
  ?content dcterms:type "SLIDES".
  ?content dcterms:source ?resource.
} ORDER BY ?lectureID
```

- Q3_2

```
SELECT ?courseName ?labID ?resource
WHERE {
  ?course rdf:type uni:Course.
```

```
   ?lecture dcterms:isPartOf ?course .
   ?course dcterms:title ?courseName.
   ?event dcterms:isPartOf ?lecture .
   ?event dcterms:type "LAB".
   ?event uni:topic dbr:Machine_learning.
   ?event dcmitype:identifier ?labID.
   ?event uni:hasContent ?content.
   ?content dcterms:source ?resource.
 } ORDER BY ?labID
```

6. Chatbot using Rasa

Rasa is an open source machine learning framework for automated text and voice-based conversations. Understand messages, hold conversations, and connect to messaging channels and APIs.

For this project, the chatbot will be implemented with Rasa. New intents will be created, and each intent will be linked to a query. New entities will also be added to identify the values the chatbot can query on.

6.1 Intents

The nlu.yml file has to be modified to include the intents. For each intents, a question will be asked in with different ways of asking the same questions, so that the model can be trained to identify the question intent.
The domain.yml file has to be updated to identify the intents

```
- intent: about_course
- intent: about_outline
- intent: about_course_number
- intent: about_course_reading
- intent: about_subject
- intent: about_course_content
- intent: about_lecture_topics
- intent: about_event
- intent: about_lecture_by_topic
- intent: about_course_by_topic
```

6.2 Entities

The domain.yml file has to be modified to identify the different entities to be extracted from the questions, namely (course) which is the course, (event) which the event type e.g. LAB, and (topic)

```
entities:
  - course
  - event
  - topic
```

The slots section is also modify to identify the type and default value for the entities

```
slots:
  course:
    type: text
    initial_value: "NA"
  event:
    type: text
    initial_value: "NA"
  topic:
    type: text
    initial_value: "NA"
```

6.3 Stories

A single story will be used to basically ask for the questions. The storie.yml file has to be modified to process the different possible intents

```
- story: get course info
  steps:
  - intent: greet
  - action: utter_iamunibot
  - or:
    - intent: about_course
    - intent: about_outline
    - intent: about_course_number
    - intent: about_course_reading
    - intent: about_subject
    - intent: about_course_content
    - intent: about_lecture_topics
    - intent: about_event
    - intent: about_lecture_by_topic
    - intent: about_course_by_topic
  - action: action_course_query
```

6.4 Actions

The domain.yml has to be modified to include the action. Finally, the actions.py file has to be modified to process the intent.

6.4.1 Program structure

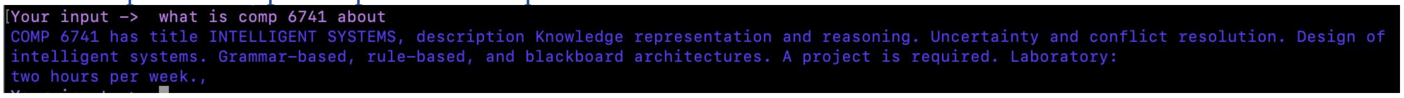
1) Create a new class *ActionCourseQuery* that is attached to the action *action_course_query*
2) Get the information for the *course, topic, event and intent* from the tracker slots
3) Call the *buildquery()* function by passing *course*, *topic*, *event and intent* as parameters
4) In *buildquery()* verify the intent and call the appropriate sparql query through a request. Process the response and return the output message
    a. The topic is identify by annotating the topic through dbpedia spotlight

    ```
    spotligh_url = 'http://api.dbpedia-spotlight.org/en/annotate?text={text}'
    ```

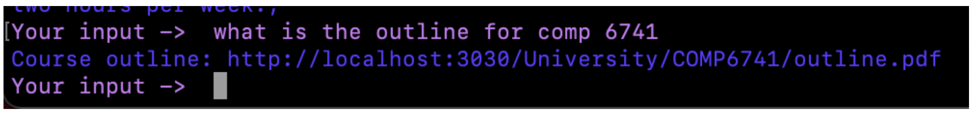
5) Send the message back to console through dispatcher

The queries have been defined at the beginning of the file.
The fuseki server to query is defined on localhost:3030

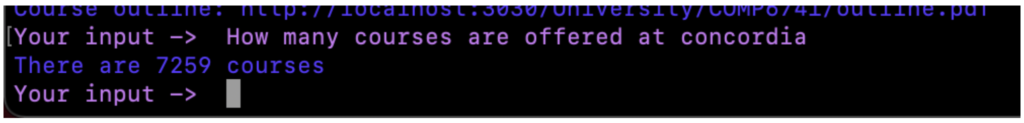
6.4.2 Examples – Sample output for each questions



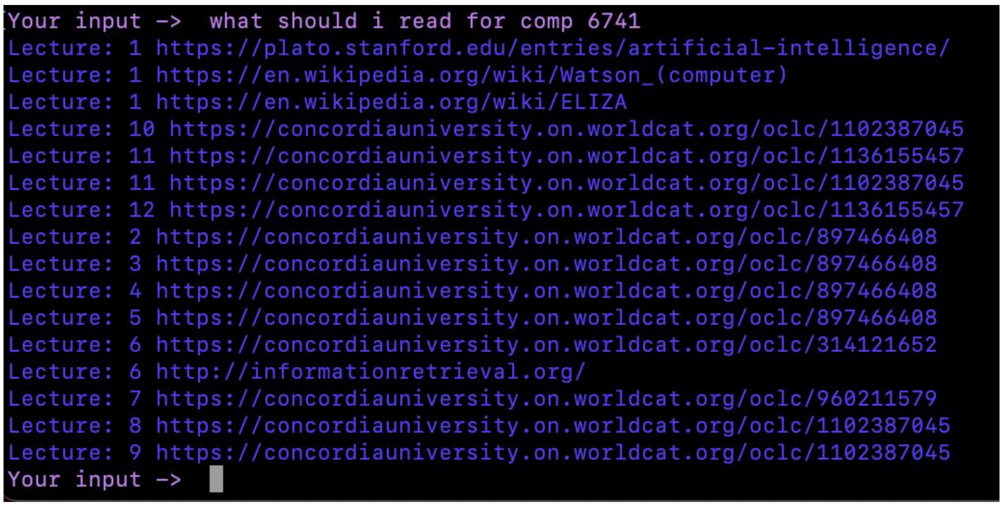
```
[Your input ->  what is comp 6741 about
COMP 6741 has title INTELLIGENT SYSTEMS, description Knowledge representation and reasoning. Uncertainty and conflict resolution. Design of
intelligent systems. Grammar-based, rule-based, and blackboard architectures. A project is required. Laboratory:
two hours per week.,
```



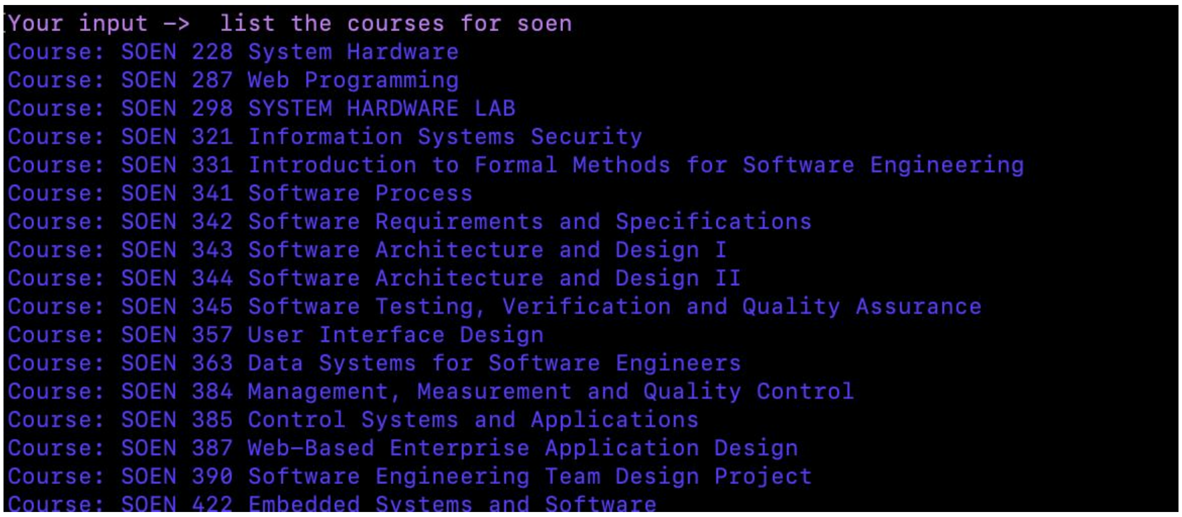
```
[Your input ->  what is the outline for comp 6741
Course outline: http://localhost:3030/University/COMP6741/outline.pdf
Your input ->
```



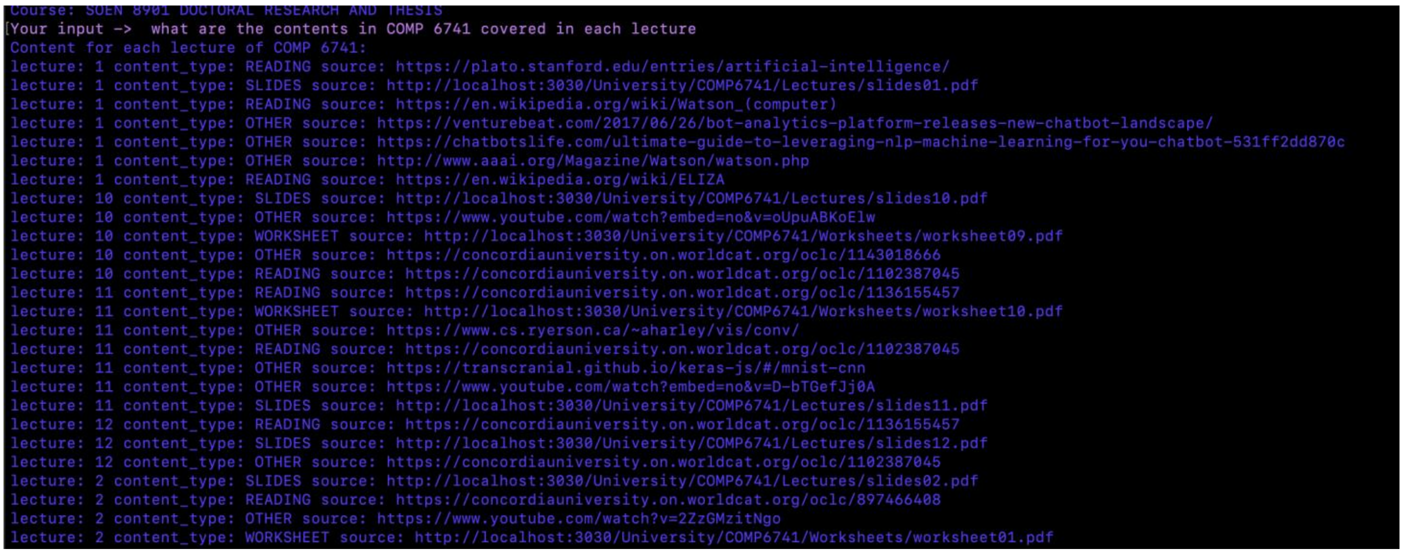
```
[Your input ->  How many courses are offered at concordia
There are 7259 courses
Your input ->
```



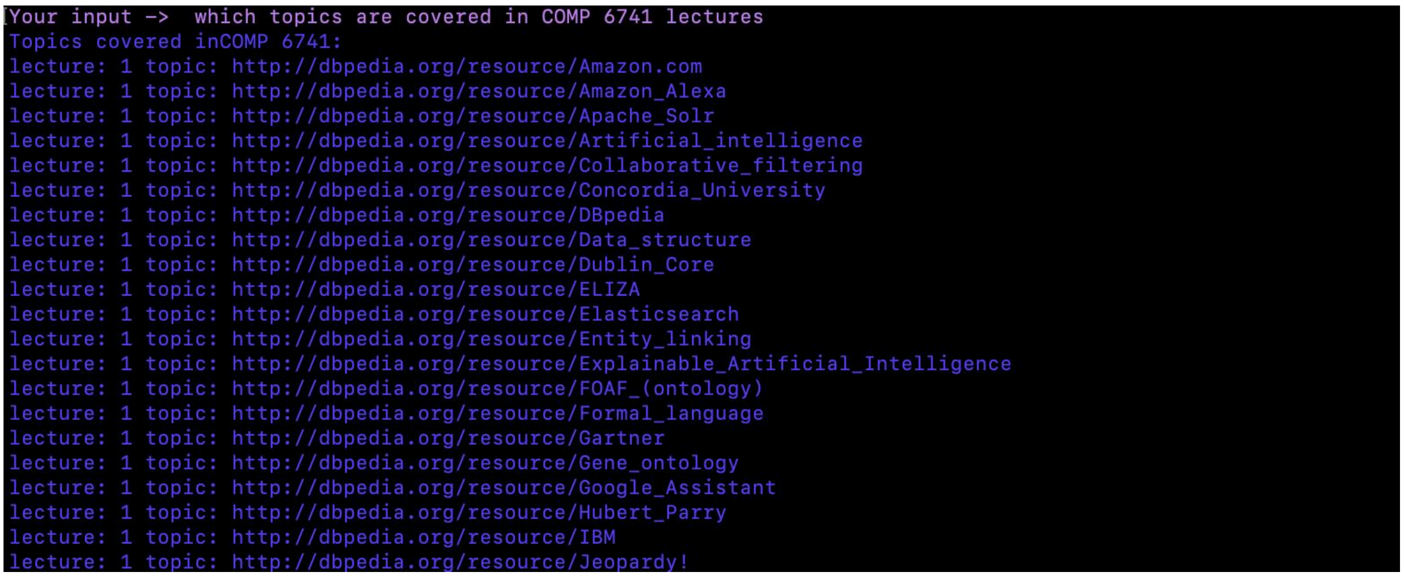
```
Your input ->  what should i read for comp 6741
Lecture: 1 https://plato.stanford.edu/entries/artificial-intelligence/
Lecture: 1 https://en.wikipedia.org/wiki/Watson_(computer)
Lecture: 1 https://en.wikipedia.org/wiki/ELIZA
Lecture: 10 https://concordiauniversity.on.worldcat.org/oclc/1102387045
Lecture: 11 https://concordiauniversity.on.worldcat.org/oclc/1136155457
Lecture: 11 https://concordiauniversity.on.worldcat.org/oclc/1102387045
Lecture: 12 https://concordiauniversity.on.worldcat.org/oclc/1136155457
Lecture: 2 https://concordiauniversity.on.worldcat.org/oclc/897466408
Lecture: 3 https://concordiauniversity.on.worldcat.org/oclc/897466408
Lecture: 4 https://concordiauniversity.on.worldcat.org/oclc/897466408
Lecture: 5 https://concordiauniversity.on.worldcat.org/oclc/897466408
Lecture: 6 https://concordiauniversity.on.worldcat.org/oclc/314121652
Lecture: 6 http://informationretrieval.org/
Lecture: 7 https://concordiauniversity.on.worldcat.org/oclc/960211579
Lecture: 8 https://concordiauniversity.on.worldcat.org/oclc/1102387045
Lecture: 9 https://concordiauniversity.on.worldcat.org/oclc/1102387045
Your input ->
```



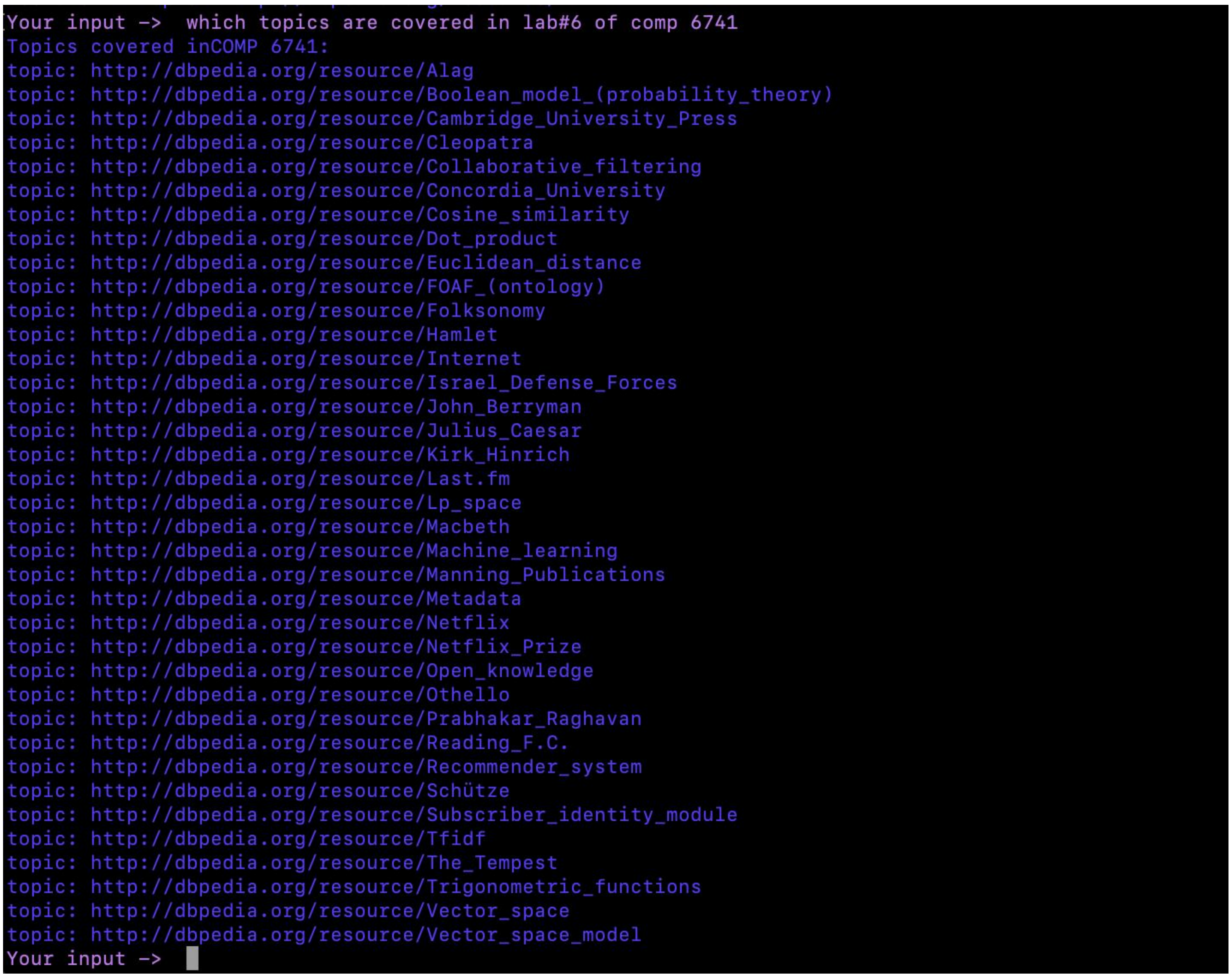
```
Your input ->  list the courses for soen
Course: SOEN 228 System Hardware
Course: SOEN 287 Web Programming
Course: SOEN 298 SYSTEM HARDWARE LAB
Course: SOEN 321 Information Systems Security
Course: SOEN 331 Introduction to Formal Methods for Software Engineering
Course: SOEN 341 Software Process
Course: SOEN 342 Software Requirements and Specifications
Course: SOEN 343 Software Architecture and Design I
Course: SOEN 344 Software Architecture and Design II
Course: SOEN 345 Software Testing, Verification and Quality Assurance
Course: SOEN 357 User Interface Design
Course: SOEN 363 Data Systems for Software Engineers
Course: SOEN 384 Management, Measurement and Quality Control
Course: SOEN 385 Control Systems and Applications
Course: SOEN 387 Web-Based Enterprise Application Design
Course: SOEN 390 Software Engineering Team Design Project
Course: SOEN 422 Embedded Systems and Software
```



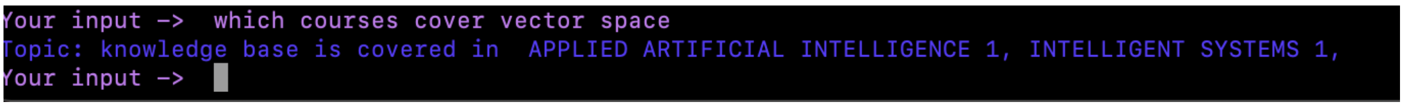
```
Course: SOEN 8901 DOCTORAL RESEARCH AND THESIS
Your input ->  what are the contents in COMP 6741 covered in each lecture
Content for each lecture of COMP 6741:
lecture: 1 content_type: READING source: https://plato.stanford.edu/entries/artificial-intelligence/
lecture: 1 content_type: SLIDES source: http://localhost:3030/University/COMP6741/Lectures/slides01.pdf
lecture: 1 content_type: READING source: https://en.wikipedia.org/wiki/Watson_(computer)
lecture: 1 content_type: OTHER source: https://venturebeat.com/2017/06/26/bot-analytics-platform-releases-new-chatbot-landscape/
lecture: 1 content_type: OTHER source: https://chatbotslife.com/ultimate-guide-to-leveraging-nlp-machine-learning-for-you-chatbot-531ff2dd870c
lecture: 1 content_type: OTHER source: http://www.aaai.org/Magazine/Watson/watson.php
lecture: 1 content_type: READING source: https://en.wikipedia.org/wiki/ELIZA
lecture: 10 content_type: SLIDES source: http://localhost:3030/University/COMP6741/Lectures/slides10.pdf
lecture: 10 content_type: OTHER source: https://www.youtube.com/watch?embed=no&v=oUpuABKoElw
lecture: 10 content_type: WORKSHEET source: http://localhost:3030/University/COMP6741/Worksheets/worksheet09.pdf
lecture: 10 content_type: OTHER source: https://concordiauniversity.on.worldcat.org/oclc/1143018666
lecture: 10 content_type: READING source: https://concordiauniversity.on.worldcat.org/oclc/1102387045
lecture: 11 content_type: READING source: https://concordiauniversity.on.worldcat.org/oclc/1136155457
lecture: 11 content_type: WORKSHEET source: http://localhost:3030/University/COMP6741/Worksheets/worksheet10.pdf
lecture: 11 content_type: OTHER source: https://www.cs.ryerson.ca/~aharley/vis/conv/
lecture: 11 content_type: READING source: https://concordiauniversity.on.worldcat.org/oclc/1102387045
lecture: 11 content_type: OTHER source: https://transcranial.github.io/keras-js/#/mnist-cnn
lecture: 11 content_type: OTHER source: https://www.youtube.com/watch?embed=no&v=D-bTGefJj0A
lecture: 11 content_type: SLIDES source: http://localhost:3030/University/COMP6741/Lectures/slides11.pdf
lecture: 12 content_type: READING source: https://concordiauniversity.on.worldcat.org/oclc/1136155457
lecture: 12 content_type: SLIDES source: http://localhost:3030/University/COMP6741/Lectures/slides12.pdf
lecture: 12 content_type: OTHER source: https://concordiauniversity.on.worldcat.org/oclc/1102387045
lecture: 2 content_type: SLIDES source: http://localhost:3030/University/COMP6741/Lectures/slides02.pdf
lecture: 2 content_type: READING source: https://concordiauniversity.on.worldcat.org/oclc/897466408
lecture: 2 content_type: OTHER source: https://www.youtube.com/watch?v=2ZzGMzitNgo
lecture: 2 content_type: WORKSHEET source: http://localhost:3030/University/COMP6741/Worksheets/worksheet01.pdf
```



```
Your input ->  which topics are covered in COMP 6741 lectures
Topics covered inCOMP 6741:
lecture: 1 topic: http://dbpedia.org/resource/Amazon.com
lecture: 1 topic: http://dbpedia.org/resource/Amazon_Alexa
lecture: 1 topic: http://dbpedia.org/resource/Apache_Solr
lecture: 1 topic: http://dbpedia.org/resource/Artificial_intelligence
lecture: 1 topic: http://dbpedia.org/resource/Collaborative_filtering
lecture: 1 topic: http://dbpedia.org/resource/Concordia_University
lecture: 1 topic: http://dbpedia.org/resource/DBpedia
lecture: 1 topic: http://dbpedia.org/resource/Data_structure
lecture: 1 topic: http://dbpedia.org/resource/Dublin_Core
lecture: 1 topic: http://dbpedia.org/resource/ELIZA
lecture: 1 topic: http://dbpedia.org/resource/Elasticsearch
lecture: 1 topic: http://dbpedia.org/resource/Entity_linking
lecture: 1 topic: http://dbpedia.org/resource/Explainable_Artificial_Intelligence
lecture: 1 topic: http://dbpedia.org/resource/FOAF_(ontology)
lecture: 1 topic: http://dbpedia.org/resource/Formal_language
lecture: 1 topic: http://dbpedia.org/resource/Gartner
lecture: 1 topic: http://dbpedia.org/resource/Gene_ontology
lecture: 1 topic: http://dbpedia.org/resource/Google_Assistant
lecture: 1 topic: http://dbpedia.org/resource/Hubert_Parry
lecture: 1 topic: http://dbpedia.org/resource/IBM
lecture: 1 topic: http://dbpedia.org/resource/Jeopardy!
```



```
Your input ->  which topics are covered in lab#6 of comp 6741
Topics covered inCOMP 6741:
topic: http://dbpedia.org/resource/Alag
topic: http://dbpedia.org/resource/Boolean_model_(probability_theory)
topic: http://dbpedia.org/resource/Cambridge_University_Press
topic: http://dbpedia.org/resource/Cleopatra
topic: http://dbpedia.org/resource/Collaborative_filtering
topic: http://dbpedia.org/resource/Concordia_University
topic: http://dbpedia.org/resource/Cosine_similarity
topic: http://dbpedia.org/resource/Dot_product
topic: http://dbpedia.org/resource/Euclidean_distance
topic: http://dbpedia.org/resource/FOAF_(ontology)
topic: http://dbpedia.org/resource/Folksonomy
topic: http://dbpedia.org/resource/Hamlet
topic: http://dbpedia.org/resource/Internet
topic: http://dbpedia.org/resource/Israel_Defense_Forces
topic: http://dbpedia.org/resource/John_Berryman
topic: http://dbpedia.org/resource/Julius_Caesar
topic: http://dbpedia.org/resource/Kirk_Hinrich
topic: http://dbpedia.org/resource/Last.fm
topic: http://dbpedia.org/resource/Lp_space
topic: http://dbpedia.org/resource/Macbeth
topic: http://dbpedia.org/resource/Machine_learning
topic: http://dbpedia.org/resource/Manning_Publications
topic: http://dbpedia.org/resource/Metadata
topic: http://dbpedia.org/resource/Netflix
topic: http://dbpedia.org/resource/Netflix_Prize
topic: http://dbpedia.org/resource/Open_knowledge
topic: http://dbpedia.org/resource/Othello
topic: http://dbpedia.org/resource/Prabhakar_Raghavan
topic: http://dbpedia.org/resource/Reading_F.C.
topic: http://dbpedia.org/resource/Recommender_system
topic: http://dbpedia.org/resource/Schütze
topic: http://dbpedia.org/resource/Subscriber_identity_module
topic: http://dbpedia.org/resource/Tfidf
topic: http://dbpedia.org/resource/The_Tempest
topic: http://dbpedia.org/resource/Trigonometric_functions
topic: http://dbpedia.org/resource/Vector_space
topic: http://dbpedia.org/resource/Vector_space_model
Your input ->
```



```
Your input ->  which courses cover vector space
Topic: knowledge base is covered in  APPLIED ARTIFICIAL INTELLIGENCE 1, INTELLIGENT SYSTEMS 1,
Your input ->
```

7. Statistics

Statistics	**Count**
Number of triples	53519
Number of courses	7259
Number of lectures	24
Number of events	24
Number of contents	173
Number of unique topics (COMP 6741 combining lab and lecture)	803
Number of unique topics (COMP 6721 combining lab and lecture)	722
Number of topic instance for COMP 6741	865
Number of topic instance for COMP 6721	799

8. Running program

1) Ensure rdflib, and pandas are installed in python environment
2) Place the university folder on the Fuseki server /webapp folder
3) Run the kbuilder.py
4) Run the Fuseki server and create a new dataset uni and upload Knowdlegde_base.nt
5) From the Queries.sparql file, copy with PREFIX section, copy the query that should be executed

9. References

[1] RDF Schema 1.1
https://www.w3.org/TR/rdf-schema/

[2] FOAF Vocabulary
http://xmlns.com/foaf/spec/

[3] Dublin Core Metadata Initiative
https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/dcmitype/Event

[4] Vivo Core Ontology
https://lov.linkeddata.es/dataset/lov/vocabs/vivo

[5] Vivo Tutorial by Shanshan Chen, Yuyin Sun, Ying Ding
https://info.sice.indiana.edu/~dingying/Teaching/S604/VIVO-tutorial-v1.pdf

[6] Merge 2 CSV files.
https://www.geeksforgeeks.org/how-to-merge-two-csv-files-by-specific-column-using-pandas-in-python/

[7] Write SPARQL query in Fuseki-Server
https://www.youtube.com/watch?v=5-UfFV5XmTI

[8] Remove blanks from text file
https://www.kite.com/python/answers/how-to-remove-empty-lines-from-a-string-in-python