COMP 6741 the Report of Project

LiaoXiaoyun

40102049 sharon.liaoxy@gmail.com

1. Introduction

The goal of this project is to construct a knowledge graph that can answer course-related question.

2. Vocabulary

1) The vocabularies I choose to reuse:
foaf:name: use to express course name.
DC:description: use to express course description.
DC:identifier: use to express course description.
DC:subject: use to express course subject.
owl:sameAs: use to express course page URL.
foaf:familyNanme: use to express student familyNanme.
foaf:givenName: use to express student givenName.
foaf:mbox: use to express student email.
rdfs:label: use to express class label and instance label.

2) Defining 5 classes and 6 properties for this knowledge graph:

- **University**

```
focu:University a rdfs:Class ;
  rdfs:label "University" ;
  rdfs:comment "The class of University" ;
  rdfs:subClassOf foaf:Organization .
```

- **Course**

```
focu:Course a rdfs:Class ;
  rdfs:label "Course" ;
  rdfs:comment "The class of course" .
```

- **Student**

```
focu:Student a rdfs:Class ;
  rdfs:label "Student" ;
  rdfs:comment "The class of student" ;
  rdfs:subClassOf foaf:Person .
```

- **Topic**

```
focu:Topic a rdfs:Class ;
  rdfs:label "Topic" ;
  rdfs:comment "The class of topic" .
```

- **The course has been completed**

```
focu:completedCourse a rdfs:Class ;
  rdfs:label "completed Course" ;
  rdfs:comment "completed Course" .
```

- **Relationship: ID of the completed Course**

```
focu:completedCourseID a rdfs:Class ;
  rdfs:label "ID of the completed Course" ;
  rdfs:comment "relationship" ;
  rdfs:domain focu:completedCourse ;
  rdfs:range focu:Course .
```

- **Relationship: The term of the completed Course**

```
focu:completedCourseTerm a rdfs:Class ;
  rdfs:label "term of the completed Course" ;
  rdfs:comment "relationship" ;
  rdfs:domain focu:completedCourse ;
  rdfs:range rdfs:Literal .
```

- **Relationship: The grade of the completed Course**

```
focu:completedCourseGrade a rdfs:Class ;
  rdfs:label "grade of the completed Course" ;
  rdfs:comment "relationship" ;
  rdfs:domain focu:completedCourse ;
  rdfs:range rdfs:Literal .
```

- **Relationship: has a completed Course**

```
focu:hasCompletedCourse a rdfs:Class ;
  rdfs:label "has a completed Course" ;
  rdfs:comment "relationship" ;
  rdfs:domain focu:Student ;
  rdfs:range focu:completedCourse .
```

- **Relationship: has a topic**

```
focu:hasTopic a rdfs:Class ;
  rdfs:label "has a topic" ;
  rdfs:comment "relationship" ;
  rdfs:domain focu:Course ;
  rdfs:range focu:Topic .
```

- **Relationship: offered at**

```
focu:offeredAt a rdfs:Class .
 rdfs:label "offered at" ;
  rdfs:comment "relationship" ;
  rdfs:domain focu:Course ;
  rdfs:range focu:University .
```

3) The examples of Instances of these classes

- **The instances of the university class**

```
focudata: cu a focu: University;
  rdfs: seeAlso <https://www.concordia.ca>;
  foaf: name "Concordia".
```

- **The instances of the course class**

```
focudata: COMP6741 a focu: Course;
  focu: hasTopic focudata: Intelligent_Systems,
            focudata: Knowledge_representation;
  focu: offeredAt focudata: cu;
  dc: description "Knowledge representation……...";
```

dc: identifier "6741";
dc: subject "COMP";
rdfs: seeAlso <https://www.concordia.ca/academics/graduate/calendar..... >;
foaf: name " Intelligent Systems (*) ".

- **The instance of the topic class**

focudata:Intelligent_Systems a focu:Topic ;
rdfs:label "Intelligent Systems" ;
owl:sameAs <http://dbpedia.org/resource/Artificial_intelligence> .

- **The instance of the student class**

focudata:0010 a focu:Student ;
focu:hasCompletedCourse focudata:0010_COMS842_Fall2020,
focudata:0010_COMS851_Fall2019;
foaf:familyNanme "Mason" ;
foaf:givenName "Cheryl" ;
foaf:mbox "Cheryl_Mason@gamil.com" .

- **The instance of the completed course**

focudata:0010_COMS842_Fall2020 a focu:completedCourse ;
focu:completedCourseGrade "A" ;
focu:completedCourseID focudata:COMS842 .

3. Knowledge Base Construction

1) The dataset

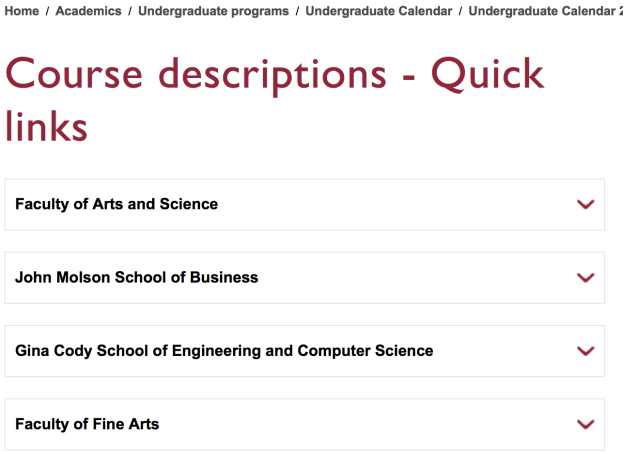
The dataset of this KB is web pages which contain course information on the Concordia website. I split the dataset into two parts, one is undergraduate courses and the other one is the graduate courses.

The entrance of undergraduate courses:
https://www.concordia.ca/academics/undergraduate/calendar/current/courses-quick-links.html

The entrance of the graduate courses:
https://www.concordia.ca/academics/graduate/calendar/current.html

Through these entrance pages, I can extract all courses information over all the course pages.

For example, how to extract undergraduate courses:
Step 1: Get the URL of faculties from the entrance page



Step 2: Get the URL of departments

Faculty of Arts and Science

Applied Human Sciences AHSC	Interdisciplinary Studies INTE
Biology BIOL	Interdisciplinary Studies in Sexuality SSDB
Chemistry and Biochemistry CHEM	Irish Studies IRST

Step 3: Get course information from the course pages of department

Courses

AHSC 220 ***Lifespan Growth and Development for Practitioners*** (3 credits)
This survey course provides an interdisciplinary overview of biopsychosocial patterns of development over the lifespan, from conception to death. Students learn about theories of human development, with an emphasis on typical normative development, and on application of theory to practice. The course material covers key issues in development, major milestones of development, and major life events. In addition, students are given opportunities to think critically and to become better able to interpret and assess research within the field.
NOTE: Students who have received credit for PSYC 230 may not take this course for credit.
NOTE: Students registered in a Psychology program may not take this course for credit.

AHSC 223 ***Relationships Across the Lifespan*** (3 credits)
This course is designed to provide a theoretical overview of how relationships are formed, sustained, and developed/changed in each stage of human life. A variety of theories and perspectives are explored.
NOTE: AHSC students may not take this course for credit.
NOTE: Students who have received credit for AHSC 220 or for this topic under an AHSC 298 number may not take this course for credit.

2) Technical Detail

- Python 3.7.
- BeautifulSoup: reading the web page through a given URL.
- Re: extracting course information by regex expression from web page content.
- Urllib: constructing RDFS and knowledge base.
- pandas: export course information to excel files.
- Pyspotlight: a useful tool for annotation.

3) The process of constructing the knowledge base.

a) Define the schema for the knowledge base.

```
def creat_a_rdf():
    g = Graph()
    g.bind('foaf', u'http://xmlns.com/foaf/0.1/')
    g.bind('dc', u'http://purl.org/dc/elements/1.1/')
    g.bind('focu', u'http://focu.io/schema#')
    g.bind('dbpedia', u'http://dbpedia.org/')
    g.bind('focudata', u'http://focu.io/data#')
    g.bind('owl', u'http://www.w3.org/2002/07/owl#')

    g.add((focu.Student, RDF.type, RDFS.Class))
    g.add((focu.Student, RDFS.subClassOf, FOAF.Person))
    g.add((focu.Student, RDFS.label, Literal('Student')))
    g.add((focu.Student, RDFS.comment, Literal('The class of student')))
```

b) Extrac the course information from web pages.

From the entrance pages accessing all the course pages, then loading it, extracting the course information into a course list.

- Get undergraduate course list:
  undergraduate_course = get_all_undergraduate_courses('https://www.concordia.ca/academics/undergraduate/calendar/current/courses-quick-links.html')

- Get graduate course list:
  fasc_course = get_faculty_course_url('https://www.concordia.ca/academics/graduate/calendar/current/fasc.html')
  fofa_course = get_faculty_course_url('https://www.concordia.ca/academics/graduate/calendar/current/fofa.html')
  jmsb_course = get_faculty_course_url('https://www.concordia.ca/academics/graduate/calendar/current/jmsb.html')
  cs_course = get_all_cs_courses()

- How to extract the course information from web pages:
  I use regex expressions to match the course information, but the formats of the course information in all web pages are not unique, there are slight differences in some courses, so I would miss these courses. I try my best to match more courses, I use several regex expressions to math undergraduate courses and graduate courses.

c) Construct knowledge graph
Merging undergraduate courses and graduate courses into a list, then using this list to annotate topics and to construct course triples.

- Construc courses triples

```
def add_course(g,course_info_list):
    for course_info in course_info_list:
        subject = course_info[0]
        id = course_info[1]
        name = course_info[2]
        description = course_info[3]
        url = course_info[4]

        course_code = subject+id
        course = URIRef(u'http://focu.io/data#'+course_code)

        g.add((course, RDF.type, focu.Course))
        g.add((course, FOAF.name, Literal(name)))
        g.add((course, RDFS.seeAlso, URIRef(url)))
        g.add((course, DC.subject, Literal(subject)))
        g.add((course, DC.identifier, Literal(id)))
        g.add((course, focu.offeredAt, focudata.cu))
        g.add((course, DC.description, Literal(description)))
```

- Get annotation of topic
  topics_info_list = get_annotate(course_info_list)

- Construct topic triples:

```
def add_topics(g,topics_info_list):
    existed_topic = []
    for topic in topics_info_list:
        course_code = topic[0]
        topic_label = topic[1]
        topic_uri = topic[2]
        topic = URIRef(u'http://focu.io/data#' + topic_label.replace(" ", "_"))
        if topic_label not in existed_topic:
        # add a new topic entity
            g.add((topic, RDF.type, focu.Topic))
            g.add((topic, RDFS.label, Literal(topic_label)))
            g.add((topic, OWL.sameAs, URIRef(topic_uri)))
            existed_topic.append(topic_label)
        course = URIRef(u'http://focu.io/data#'+course_code)
        g.add((course, focu.hasTopic, topic))
```

- Construct student triples
  Loading student information from a text file.

```
student = URIRef(u'http://focu.io/data#'+student_id)
# focudata.student  # = rdflib.term.URIRef(uri)
g.add((student, RDF.type, focu.Student))
g.add((student, FOAF.givenName, Literal(first_name)))
g.add((student, FOAF.familyNanme, Literal(last_name)))
g.add((student, FOAF.mbox, Literal(email)))

if len(info_list)>3: # get completed course information
    completed_courses_list = info_list[3].split(',')
    for completed_course in completed_courses_list:
        completed_course_info = completed_course.split(' ')
        course_code = completed_course_info[0]
        term = completed_course_info[1]
        grade = completed_course_info[2]

        completed_code = student_id+"_"+course_code+"_"+term
        completed_course = URIRef(u'http://focu.io/data#' + completed_code)
        course = URIRef(u'http://focu.io/data#' + course_code)

        g.add((completed_course, RDF.type, focu.completedCourse))
        g.add((completed_course, focu.completedCourseID, course))
        g.add((completed_course, focu.completedCourseGrade, Literal(grade)))
        g.add((student, focu.hasCompletedCourse, completed_course))
```

4. Link Analysis

I use dbpedia spotlight server to automatically annotate the topics in the course description. Because the spotlight web server limits the frequency of annotation, so I build a local host on my computer. And I also use a useful tool Pyspotlight to send the annotation request, it makes code clearer.

1) Annotation function

```
def get_annotate(course_list):
    host = 'http://localhost:2222/rest/annotate'
    confidence = 0.8
    support = 20
    count = 0
    topics_list = []
    f = open("topics_uri.txt", "a+")
    for course in course_list:
        description = course[2] + ": " + course[3]
        count = count + 1
        if description != '':
            try:
                course_id = course[0]+course[1]
                annotations = annotate(host, description, confidence, support)
```

2) Annotation parameter
host = 'http://localhost:2222/rest/annotate'
confidence = 0.8
support = 20
Text = course name + description

3) Analyzing data
- Annotate 20 courses description.
- Expect to get 115 links.
- Actually getting 101 links, 1 link of them is wrong, others are correct, and missing 14 links.
- Accuracy = 100/115 = 0.87
- Precise = 100/101= 0.99
- Recall = 100/114= 0.88

Precise rate is higher than recall rate, because I set confidence = 0.8, it means I want to make sure the annotation is correct rather than get more uncertain annotations.

	ID	courese	Description	Expected	Generated	Result	Link
0				Compiler		missing	
1				optimization	optimization	correct	http://dbpedia.org/resource/Program_optimization
2	COMP6421	Compiler Design (*)	Compiler organization and implementation: lexical analysis and parsing, syntax-directed translation, code optimization. Run-time systems. A project is required.	lexical analysis	lexical analysis	correct	http://dbpedia.org/resource/Lexical_analysis
4				Computer Networks		missing	
5				Packet switching		correct	http://dbpedia.org/resource/Packet_switching
6				Internet Protocol	Internet Protocol	correct	http://dbpedia.org/resource/Internet_Protocol
7				IPv6	IPv6	correct	http://dbpedia.org/resource/IPv6
8				UDP	UDP	correct	http://dbpedia.org/resource/User_Datagram_Protocol
9				TCP	TCP	correct	http://dbpedia.org/resource/Transmission_Control_Protocol
10	COMP6461	Computer Networks and Protocols	Direct link networks: encoding, framing, error detection, flow control, example networks. Packet switching and forwarding: bridges, switches. Internetworking: Internet Protocol, routing, addressing, IPv6, multicasting, mobile IP. End-to-end protocols: UDP, TCP. Network security concepts. Application-level protocols.	Network security	Network security	correct	http://dbpedia.org/resource/Network_security
11				relational databases		missing	
12				optimization	optimization	correct	http://dbpedia.org/resource/Query_optimization
13				Information integration	Information integration	correct	http://dbpedia.org/resource/Information_integration
14				Data warehouse	Data warehouse	correct	http://dbpedia.org/resource/Data_warehouse
15				Data mining	Data mining	correct	http://dbpedia.org/resource/Data_mining
16				OLAP	OLAP	correct	http://dbpedia.org/resource/Online_analytical_processing
17	COMP6521	Advanced Database Technology and Applications	Review of standard relational databases, query languages. Query processing and optimization. Parallel and distributed databases. Information integration. Data warehouse systems. Data mining and OLAP. Web databases and XML Active and logical databases, spatial and multimedia data management. Laboratory: Two hours per week.	XML	XML	correct	http://dbpedia.org/resource/XML
18				Semantic Web	Semantic Web	correct	http://dbpedia.org/resource/Semantic_Web
19				World Wide Web Consortium	World Wide Web Consortium	correct	http://dbpedia.org/resource/World_Wide_Web_Consortium
20				Extensible Markup Language	Extensible Markup Language	correct	http://dbpedia.org/resource/XML
21				Resource Description Framew	Resource Description Framew	correct	http://dbpedia.org/resource/Resource_Description_Framewo
22				Semantic Web	Semantic Web	correct	http://dbpedia.org/resource/Semantic_Web
23				ontology	ontology	correct	http://dbpedia.org/resource/Ontology_engineering
24	COMP6531	Foundations of the Semantic Web	Web markup languages, World Wide Web Consortium (W3C) standards, Extensible Markup Language (XML) Resource Description Framework (RDF), schema for markup languages, Semantic Web, ontology development, markup languages for ontologies, Web Ontology Language (OWL), logical foundations of ontologies, description logics, reasoning with ontologies. A project is required.	Web Ontology Language (OW	Web Ontology Language (OW	correct	http://dbpedia.org/resource/Web_Ontology_Language

25			Review of first-order logic, relational algebra, and relational calculus. Fundamentals of logic programming. Logic for knowledge representation. Architecture of a knowledge-base system. Fundamentals of deductive databases. Top-down and bottom-up query processing. Some important query processing strategies and their comparison.	deductive databases		missing	
26				first-order logic	first-order logic	correct	http://dbpedia.org/resource/First-order_logic
27				relational algebra	relational algebra	correct	http://dbpedia.org/resource/Relational_algebra
28				relational calculus	relational calculus	correct	http://dbpedia.org/resource/Relational_calculus
29				logic programming	logic programming	correct	http://dbpedia.org/resource/Logic_programming
30	COMP6591	Introduction to Knowledge-Base Systems	Project or term paper on current research topics.	knowledge representation	knowledge representation	correct	http://dbpedia.org/resource/Knowledge_representation_and
31			Introduction to the methods and proof techniques of Paul Erdös that are particularly applicable to Computer Science. Proof of Bertrand's postulate. The Erdös-Szekeres and the de Bruijn-Erdös theorems. Ramsey's theorem and Ramsey numbers. Van der Waerden's theorem and Van der Waerden numbers. Delta-systems and a proof of the Erdös-Lovász conjecture. The Erdös-Ko-Rado theorem. Extremal graph theory. Random graphs and graph colouring. The probabilistic method and its applications in theoretical Computer Science. A project is required.	Computer Science		missing	
32				Discrete Mathematics	Discrete Mathematics	correct	http://dbpedia.org/resource/Discrete_mathematics
33				Paul Erdös	Paul Erdös	correct	http://dbpedia.org/resource/Paul_Erdős
34				Ramsey	Ramsey	correct	http://dbpedia.org/resource/Ramsey_Abbey
35				theorem	theorem	correct	http://dbpedia.org/resource/Theorem
36				Ramsey	Ramsey	correct	http://dbpedia.org/resource/Ramsey_Abbey
37				theorem	theorem	correct	http://dbpedia.org/resource/Theorem
38					Van	Wrong	http://dbpedia.org/resource/Van
39				Extremal graph theory	Extremal graph theory	correct	http://dbpedia.org/resource/Extremal_graph_theory
40				graph	graph	correct	http://dbpedia.org/resource/Graph_theory
41	COMP6621	Discrete Mathematics of Paul Erdös		probabilistic method	probabilistic method	correct	http://dbpedia.org/resource/Probabilistic_method
42			General properties of algorithmic computations. Turing machines, universal Turing machines. Turing computable functions as a standard family of algorithms. Primitive recursive functions. Church's thesis, recursive sets. Recursively enumerable sets and their properties. Rice's theorem. Time and space complexity measures. Hierarchy of complexity measures. Advanced topics in complexity theory. A project is required.	algorithm		missing	
43				Turing computable	Turing computable	correct	http://dbpedia.org/resource/Computable_function
44				recursive functions	recursive functions	correct	http://dbpedia.org/resource/Computable_function
45				theorem	theorem	correct	http://dbpedia.org/resource/Theorem
46				Time	Time	correct	http://dbpedia.org/resource/Time
47	COMP6641	Theory of Computation		complexity theory	complexity theory	correct	http://dbpedia.org/resource/Computational_complexity_the
48			Mathematical preliminaries; Empirical and theoretical measures of algorithm efficiencies; Optimization and combinatorial techniques and algorithms including greedy algorithms, dynamic programming, branch-and-bound techniques and graph network algorithms; Amortized complexity analysis; String matching algorithms; NP-complete problems and approximate solutions; Probabilistic algorithms. A project is required.	Amortized complexity analysis		missing	
49				algorithm	algorithm	correct	http://dbpedia.org/resource/Algorithm
50				dynamic programming	dynamic programming	correct	http://dbpedia.org/resource/Dynamic_programming
51				graph	graph	correct	http://dbpedia.org/resource/Graph_theory
52				String matching	String matching	correct	http://dbpedia.org/resource/String_searching_algorithm
53	COMP6651	Algorithm Design Techniques		NP-complete	NP-complete	correct	http://dbpedia.org/resource/NP-completeness
54			Representation and generation of combinatorial objects; search techniques; counting and estimation. Projects on selected applications from combinatorics and graph theory.	combinatorics	combinatorics	correct	http://dbpedia.org/resource/Combinatorics
55	COMP6661	Combinatorial Algorithms		graph theory	graph theory	correct	http://dbpedia.org/resource/Graph_theory
56			Efficient algorithms and data structures to solve geometric problems. Problems discussed include convex hulls, line intersections, polygon triangulation, point location, range searching, Voronoi diagrams, Delaunay triangulations, interval trees and segment trees, arrangements, robot motion planning, binary space partitions, quadtrees, and visitility. Algorithmic methods include plane sweep, incremental insertion, randomization, divide and conquer. Emphasis will be given to computation and complexity, with applications in computer graphics, computer aided design, geographic information systems, networks, mesh generation, databases, and robot motion planning. A project is required.	algorithms		missing	
57				polygon triangulation	polygon triangulation	correct	http://dbpedia.org/resource/Polygon_triangulation
58				range searching	range searching	correct	http://dbpedia.org/resource/Range_searching
59				Voronoi	Voronoi	correct	http://dbpedia.org/resource/Voronoi_diagram
60				Delaunay	Delaunay	correct	http://dbpedia.org/resource/Robert_Delaunay
61				robot	robot	correct	http://dbpedia.org/resource/Robotics
62				motion planning	motion planning	correct	http://dbpedia.org/resource/Motion_planning
63				computer graphics	computer graphics	correct	http://dbpedia.org/resource/Computer_graphics
64				computer aided design	computer aided design	correct	http://dbpedia.org/resource/Computer-aided_design
65				mesh generation	mesh generation	correct	http://dbpedia.org/resource/Mesh_generation
66	COMP6711	Computational Geometry		motion planning	motion planning	correct	http://dbpedia.org/resource/Motion_planning
67			The course covers heuristic and adversarial searches for concrete applications. It then discusses automated reasoning, advanced knowledge representation and dealing with uncertainly for Artificial Intelligence applications. Finally, it introduces autoencoders, recurrent neural networks and sequence to sequence models. A project is required. Laboratory: two hours per week.	Artificial Intelligence		missing	
68				heuristic	heuristic	correct	http://dbpedia.org/resource/Heuristic
69				automated reasoning	automated reasoning	correct	http://dbpedia.org/resource/Automated_reasoning
70				knowledge representation	knowledge representation	correct	http://dbpedia.org/resource/Knowledge_representation_and
71	COMP6721	Applied Artificial Intelligence (*)		recurrent neural networks	recurrent neural networks	correct	http://dbpedia.org/resource/Recurrent_neural_network
72			Pre-processing. Feature extraction and selection. Similarity between patterns and distance measurements. Syntactic and statistical approaches. Clustering analysis. Bayesian decision theory and discriminant functions. Clustering and classification techniques. Applications. A project is required. Laboratory: two hours per week.	Feature extraction	Feature extraction	correct	http://dbpedia.org/resource/Feature_extraction
73				Bayesian decision theory	Bayesian decision theory	correct	http://dbpedia.org/resource/Bayes_estimator
74	COMP6731	Pattern Recognition (*)		discriminant	discriminant	correct	http://dbpedia.org/resource/Discriminant
75			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.	Intelligent Systems	Intelligent Systems	correct	http://dbpedia.org/resource/Artificial_intelligence
76	COMP6741	Intelligent Systems (*)		Knowledge representation	Knowledge representation	correct	http://dbpedia.org/resource/Knowledge_representation_and
77			Introduction to natural language processing. Structure of English. Grammars and parsing. Lexical and compositional semantics. Pragmatic issues. Applications in text mining and information extraction. A project is required.	natural language processing	natural language processing	correct	http://dbpedia.org/resource/Natural-language_processing
78				compositional semantics	compositional semantics	correct	http://dbpedia.org/resource/Principle_of_compositionality
79	COMP6751	Natural Language Analysis		text mining	text mining	correct	http://dbpedia.org/resource/Text_mining
80			Fundamental algorithms, techniques, and software engineering principles for 3D graphics. Introduction to real-time graphics application architecture; review of basic 3D concepts of modelling, viewing, and rendering. 3D graphics functions, pipeline, and performance. Hierarchical 3D graphics. Algorithms for occlusion culling, collision detection, photorealism, shadows, and textures. Current trends and state-of-the-art graphics and physics algorithms. Laboratory: Two hours per week.	3D graphics		missing	
81				software engineering	software engineering	correct	http://dbpedia.org/resource/Software_engineering
82				occlusion culling	occlusion culling	correct	http://dbpedia.org/resource/Hidden_surface_determination
83				photorealism	photorealism	correct	http://dbpedia.org/resource/Photorealism
84	COMP6761	Advanced 3D Graphics for Game Programming		physics	physics	correct	http://dbpedia.org/resource/Game_physics

85		Image Processing (*)	Digital image fundamentals; image enhancement: histogram processing, filtering in the spatial domain, filtering in the frequency domain; image restoration and reconstruction; image segmentation: line detection, Hough transform, edge detection and linking, thresholding, region splitting and merging; image compression; introduction to wavelet transform and multi-resolution processing. A project is required. Laboratory: two hours per week.	histogram	histogram	correct	http://dbpedia.org/resource/Histogram
86				image segmentation	image segmentation	correct	http://dbpedia.org/resource/Image_segmenta
87				Hough transform	Hough transform	correct	http://dbpedia.org/resource/Hough_transforr
88				edge detection	edge detection	correct	http://dbpedia.org/resource/Edge_detection
89	COMP6771			wavelet transform	wavelet transform	correct	http://dbpedia.org/resource/Wavelet_transfo
90		Statistical Natural Language Processing	The course covers robust methods to natural language processing (NLP) and their applications to manipulate large text collections. Topics covered in this course include: Zipf's law, information retrieval, statistical machine translation, N-gram language models and smoothing techniques, word sense disambiguation, part-of-speech tagging and probabilistic grammars and parsing. A project is required.	information retrieval		missing	
91				natural language processing (	natural language processing (	correct	http://dbpedia.org/resource/Natural-languag
92				statistical machine translatior	statistical machine translatio	correct	http://dbpedia.org/resource/Statistical_mach
93				N-gram	N-gram	correct	http://dbpedia.org/resource/N-gram
94				word sense disambiguation	word sense disambiguation	correct	http://dbpedia.org/resource/Word-sense_disa
95	COMP6781			part-of-speech tagging	part-of-speech tagging	correct	http://dbpedia.org/resource/Part-of-speech_t
96		Information Retrieval and Web Search (*)	Basics of information retrieval (IR): Boolean, vector space and probabilistic models. Tokenization and creation of inverted files. Weighting schemes. Evaluation of IR systems: precision, recall, E-measure. Relevance feedback and query expansion. Application of IR to Web search engines: XML, link analysis, PageRank algorithm. Text categorization and clustering techniques as used in spam filtering. A project is required. Laboratory: two hours per week.	information retrieval (IR)		missing	
97					Boolean	correct	http://dbpedia.org/resource/Boolean_algebra
98				vector space	vector space	correct	http://dbpedia.org/resource/Vector_space
99				Relevance feedback	Relevance feedback	correct	http://dbpedia.org/resource/Relevance_feedb
100				query expansion	query expansion	correct	http://dbpedia.org/resource/Query_expansio
101				XML	XML	correct	http://dbpedia.org/resource/XML
102				PageRank	PageRank	correct	http://dbpedia.org/resource/PageRank
103	COMP6791			algorithm	algorithm	correct	http://dbpedia.org/resource/Algorithm
104		Bioinformatics Algorithms	The principal objectives of the course are to cover the major algorithms used in bioinformatics; sequence alignment, multiple sequence alignment, phylogeny; classifying patterns in sequences; secondary structure prediction; 3D structure prediction; analysis of gene expression data. This includes dynamic programming, machine learning, simulated annealing, and clustering algorithms. Algorithmic principles will be emphasized. A project is required.	clustering algorithms		missing	
105				Bioinformatics	Bioinformatics	correct	http://dbpedia.org/resource/Bioinformatics
106				multiple sequence alignment	multiple sequence alignment	correct	http://dbpedia.org/resource/Multiple_sequence_alignment
107				phylogeny	phylogeny	correct	http://dbpedia.org/resource/Phylogenetic_tree
108				secondary structure	secondary structure	correct	http://dbpedia.org/resource/Biomolecular_structure
109				gene	gene	correct	http://dbpedia.org/resource/Gene
110				dynamic programming	dynamic programming	correct	http://dbpedia.org/resource/Dynamic_programming
111				machine learning	machine learning	correct	http://dbpedia.org/resource/Machine_learning
112	COMP6811			simulated annealing	simulated annealing	correct	http://dbpedia.org/resource/Simulated_annealing
113		Bioinformatics Databases and Systems	The principal objectives of the course are to survey the needs of bioinformatics for data management, knowledge management, and computational support; to provide in-depth description of an example of each kind of database and system; and to introduce advanced database technology and software technology relevant to the needs of bioinformatics. A project is required.	database		missing	
114				Bioinformatics	Bioinformatics	correct	http://dbpedia.org/resource/Bioinformatics
115				knowledge management	knowledge management	correct	http://dbpedia.org/resource/Knowledge_management
116	COMP6821			bioinformatics	bioinformatics	correct	http://dbpedia.org/resource/Bioinformatics

5. Queries

In each query, I use SPARQL construct to generate a new graph, the query output can be ttl format.

1) Total number of triples in the KB

In this question, I use a blank node to record the query result.

```
CONSTRUCT {
_:v  rdfs: label ?Triples.
_:v  rdfs: comment 'Total number of triples in the KB'.
 }
Where {
   select (COUNT (*) as ?Triples)
    WHERE
    {
       ?s ?p ?o
      }
}
```

The output is:

```
[] rdfs: label 52616;
        rdfs: comment "Total number of triples in the KB".
```

2) Total number of students, courses, and topics
In this question, using union keyword to union three kinds of triples, and also using a blank node to record the total number.

```
CONSTRUCT {
_:v  rdfs: label ?Triples.
_:v  rdfs: comment 'Total number of students, courses, and topics'.
 }
Where {
 SELECT (COUNT (*) as ?Triples)
  WHERE {
   {?student rdf: type focu: Student.}
   union
   {?course rdf: type focu: Course.}
   union
   {? topic rdf: type focu: Topic.}
   }
   }
```

The output is:
[] rdfs: label 8470;
  rdfs: comment "Total number of students, courses, and topics".

3) For a course c, list all covered topics using their (English) labels and their link to DBpedia
In this question, using course name to search the course entity.

```
 construct {
  ? topic owl: sameAs ?link.
  ? topic rdfs: label ?label.
 }
 Where {
  ? course focu: hasTopic ?topic.
  ? topic owl: sameAs ?link.
  ? topic rdfs: label ?label.
  ? course foaf: name '%s'.
 }
```

The output is:
focudata: automated_reasoning rdfs:label "automated reasoning" ;
   ns1:sameAs <http://dbpedia.org/resource/Automated_reasoning> .

4) For a given student, list all courses this student completed, together with the grade
  In this question, search the student by his first name and last name.

```
 Construct {
  ?completedCourse foaf: name ?courseName.
  ?completedCourse focu: completedCourseGrade ?grade.
 }
 where{
  ?student foaf: givenName '%s'.
  ?student foaf:familyNanme '%s'.
  ?student focu: hasCompletedCourse ?completedCourse.
  ?completedCourse focu: completedCourseID ?course.
  ?course foaf: name ?courseName.
  ?completedCourse focu: completedCourseGrade ?grade.
```

The output is:

focudata: 0008_ENCS6041_Fall2019 ns2:completedCourseGrade "A" ;
ns1:name " Creativity, Innovation, and Critical Thinking " .

5) For a given student, list all courses this student completed, together with the grade

```
construct{
  ?student foaf:givenName ?givenName.
  ?student foaf:familyNanme ?familyNanme.
 }
 where{
  ?course focu:hasTopic ?topic.
  ?topic rdfs:label '%s'.
  ?student foaf:givenName ?givenName.
  ?student foaf:familyNanme ?familyNanme.
  ?student focu:hasCompletedCourse ?completedCourse.
  ?completedCourse focu:completedCourseID ?course.
  ?completedCourse focu:completedCourseGrade ?grade.
  Filter (?grade != 'F')
```

The output is :
focudata: 0011 ns1:familyNanme "Kennedy" ;
ns1:givenName "Joan" .

6) For a student, list all topics (no duplicates) that this student is familiar with (based on the completed courses for this student that are better than an "F" grade)

```
construct {
?student foaf:konws ?topicLabel
}
where{
 select
   distinct ?topicLabel ?student
 where{
  ?student foaf:givenName '%s'.
  ?student foaf:familyNanme '%s'.
  ?student focu:hasCompletedCourse ?completedCourse.
  ?completedCourse focu:completedCourseID ?course.
  ?completedCourse focu:completedCourseGrade ?grade.
  ?course focu:hasTopic ?topic.
  ?topic rdfs:label ?topicLabel.
  Filter (?grade != 'F')
}
```

The output is :
focudata:0010 foaf:konws "ethnography" .

6. Chatbot

In this part, I use the spacy library to translate the input questions into SPARQL queries.

1) Tokenization and Lemmatization

```
def nlp_question(question):
    lemma_list = []
    doc = nlp(question)

    for token in doc:
        lemma_list.append(token.lemma_)
```

For example, question2 "Which courses did Lucas take?" after tokenization and lemmatization, I will get a token.lemma list ['which', 'course', 'do', 'Lucas', 'take', '?'].
Lemmatization of token can fuzzy match more similar sentences that represent the same meaning.

2) Named Entity

We can get person name and topic name from question's named entities, but it sometimes fail to name topic entities. I extract student name and topic name by checking tokens' tag and dependent.

```
entity = ""
last_one_compound = False
for token in doc:
    if token.dep_ == 'compound':
        entity = entity + " " + token.text
        last_one_compound = True
    elif token.dep_ != 'compound' and last_one_compound == True:
        entity = entity + " " + token.text
        last_one_compound = False
    else:
        last_one_compound = False

if entity == "":
    for i in range(0, len(doc)):
        if doc[i].tag_ == 'NN':
            entity = doc[i].text
            if i - 1 >= 0 and doc[i - 1].tag_ == 'JJ':
                entity = doc[i - 1].text + " " + entity

if entity == "":
    for i in range(0, len(doc)):
        if doc[i].tag_ == 'NNP':
            entity = doc[i].text
```

3) Get course id

Define a new matcher to match course id.

```
nlp = spacy.load('en_core_web_sm')
matcher = Matcher(nlp.vocab)
pattern_course_id = [{"TEXT": {"REGEX": "[A-Z]{4}"}},
                     {"TEXT": {"REGEX": "[0-9]{1,4}"}}]
matcher.add("course_id", None, pattern_course_id)

def get_course_id(doc):
    course_id = ''
    matchers = matcher(doc)
    for match_id, start, end in matchers:
        course_id = doc[start:end].text
    return course_id
```

4) Classify the question type

Classify the questions by checking its lemma tokens and person entity and topic entity.

```
course_about_vocab_list = ['what', 'about']
course_take_vocab_list = ['which', 'course', 'take']
course_cover_vocab_list = ['which', 'course', 'cover']
topic_familiar_vocab_list = ['who', 'familiar']
student_know_vocab_list = ['what', 'know']

course_id = get_course_id(doc)
if(all(x in lemma_list for x in course_about_vocab_list) and course_id != ''):
    question1(course_id,g)
elif(all(x in lemma_list for x in course_take_vocab_list) and entity != ''):
    question2(entity, g)
elif(all(x in lemma_list for x in course_cover_vocab_list) and entity != ''):
    question3(entity, g)
elif(all(x in lemma_list for x in topic_familiar_vocab_list) and entity != ''):
    question4(entity, g)
elif (all(x in lemma_list for x in student_know_vocab_list) and entity != ''):
    question5(entity, g)
else:
    print("Sorry, I have no idea.")
```

5) SPARQL-query

a) question1 = "What is COMP 474 about?"

```
# question1 = "What is COMP 474 about?"
def question1(course_id,g):
    if len(course_id.split(' '))<2:
        print("No this course.")
        return

    subject = course_id.split(' ')[0]
    identifier = course_id.split(' ')[1]
    qres = g.query(
    """
       select
          ?description
       where{
            ?course dc:description ?description.
            ?course dc:subject '%s'.
            ?course dc:identifier '%s'.
      }
    """%(subject,identifier)
    )
    for row in qres:
        print(row.description)

    if len(qres) == 0 :
        print("No course %s"%course_id)
```

b) question2 = "Which courses did Joan Kennedy take?"

```
def question2(student_name, g):
    if len(student_name.split(' '))<2:
        print("No this student.")
        return

    givenName = student_name.split(' ')[0]
    familyNanme = student_name.split(' ')[1]

    qres = g.query(
        """
           select
              ?course_name
           where{
                ?student foaf:givenName '%s'.
                ?student foaf:familyNanme '%s'.
                ?student focu:hasCompletedCourse ?hasCompletedCourse.
                ?hasCompletedCourse focu:completedCourseID ?course.
                ?course foaf:name ?course_name
          }
        """ % (givenName, familyNanme)
    )
    for row in qres:
        print(row.course_name)

    if len(qres) == 0 :
        print("%s did not take any courses."%student_name)
```

c) question3 = "Which courses cover Expert Systems?"

```
def question3(topic, g):
    if len(topic)==0:
        print("No this topic.")
        return

    qres = g.query(
        """
          select
             ?course_name
          where{
              ?course foaf:name ?course_name.
              ?course focu:hasTopic ?topic.
              ?topic rdfs:label '%s'
          }
        """ % (topic.lower())
    )
    for row in qres:
        print(row.course_name)

    if len(qres) == 0 :
        print("No course covers it")
```

d) question4 = "Who is familiar with Natural Language Processing?"

```
def question4(topic, g):
    if len(topic)==0:
        print("No this topic.")
        return

    qres = g.query(
        """
        select
            ?givenName ?familyNanme
          where{
            ?course focu:hasTopic ?topic.
            ?topic rdfs:label "%s".
            ?student foaf:givenName ?givenName.
            ?student foaf:familyNanme ?familyNanme.
            ?student focu:hasCompletedCourse ?completedCourse.
            ?completedCourse focu:completedCourseID ?course.
            ?completedCourse focu:completedCourseGrade ?grade.
            Filter (?grade != 'F')
          }
        """ %(topic.lower())
    )
    for row in qres:
        print(row.givenName+" "+row.familyNanme)

    if len(qres) == 0 :
        print("No one familiars %s."%topic)
```

e) question5 = "What does Joan Kennedy know?"

```
def question5(student_name, g):
    if len(student_name.split(' '))<2:
        print("No this student.")
        return

    givenName = student_name.split(' ')[0]
    familyNanme = student_name.split(' ')[1]
    qres = g.query(
        """
          select
            distinct ?topicLabel
          where{
           ?student foaf:givenName '%s'.
           ?student foaf:familyNanme '%s'.
           ?student focu:hasCompletedCourse ?completedCourse.
           ?completedCourse focu:completedCourseID ?course.
           ?completedCourse focu:completedCourseGrade ?grade.
           ?course focu:hasTopic ?topic.
           ?topic rdfs:label ?topicLabel.
           Filter (?grade != 'F')
          }
        """ %(givenName,familyNanme)
    )
    for row in qres:
        print(row.topicLabel)

    if len(qres) == 0 :
        print("%s knows nothing."%student_name)
```

6) Chatbot demo

```
Please enter your Question, or enter Q to quit.
>Hi,What is COMP 474 about?
Prerequisite: COMP 352 or COEN 352. Rule-based expert systems, blackboard architecture, and agent-based. Knowledge acquisition and
 representation. Uncertainty and conflict resolution. Reasoning and explanation. Design of intelligent systems. Project. Lectures:
 three hours per week. Laboratory: two hours per week.
>"Which courses did Lucas Wang take?"
 Distributed System Design
 Natural Language Analysis
 Intelligent Systems (*)
>Which courses cover Expert Systems?
 Intelligent Systems (*)
Intelligent Systems
>Who was familiar with Natural Language Processing?
Lucas Wang
Joe Lee
>What does Joan Kennedy know
knowledge representation
automated reasoning
recurrent neural networks
heuristic
distributed computing
distributed systems
concurrency control
Client-server
interprocess communication
CORBA
remote procedure call
concurrency
>Q
```