COMP474/6741 INTELLIGENT SYSTEMS

INTELLIGENT AGENT (PART 1 & 2)

PROJECT REPORT

Submitted By,

Aditi Bhayana(Student Id: 40083419)

Nikitha Jayant Bangera(Student id: 40088393)

Introduction

The purpose of this project is to build an intelligent agent that can answer the university-related questions by building a knowledge base graph and obtaining answers with the help of SPARQL queries.

Vocabulary

While finalising the Vocabulary list for our Knowledge Graph, we reused some of the existing vocabularies and defined some new vocabularies to define our concepts.

Reused Vocabularies

Below is the list of vocabularies which were reused while defining our knowledge graph:

- **FOAF.name->** This vocabulary is reused for defining the name of the university, as foaf.name defines the name of “something” in a simple textual string. This vocabulary was chosen as our university can be considered as a thing which has to be named.
- **RDFS.seeAlso->** This vocabulary is reused for defining the link between the courses and the web page on which the course was found. We chose this vocabulary as it indicates that the web page resource might provide additional information about the subject resource which is our Course Class.
- **DC.identifier->** This vocabulary is reused to define course number in the course class as it helps in providing the reference to the subject resource within a given context.
- **DC.title->** This vocabulary is reused to describe the course name as it is used to give a name to the subject resource.
- **DC.Subject->** This vocabulary is reused to identify the subject to which the course belongs, as it identifies the type of the subject resource.
- **DC.description->** This vocabulary is reused to identify the course description in the course class. It is reused as it helps in keeping account of a resource.
- **FOAF.primaryTopicOf->** This vocabulary is reused to identify the topic which is part of the course class, as this property relates something to a document that is mainly about it.
- **FOAF.givenName->** This vocabulary is reused to define the first name of students, as the givenName property is used to describe the first/given name of a person.
- **FOAF.familyName->** This vocabulary is reused to define the last name of students, as the familyName property is used to describe the last/family name of a person..
- **FOAF.mbox->** This vocabulary is reused to define the email address of a student, as the mbox property defines the relationship between the owner of a mailbox and a mailbox.
- **DBpedia Term Period->** This vocabulary is reused to define the semester property for Student Transcript which has rdfs:label as term period and rdfs:range as TimePeriod. As this property was related to a time period, we reused it to define our semester property.

User-Defined Vocabularies:

Some of the vocabularies which are developed in the schema model of our knowledge base graph are as follows:

- **focu:Student ->** This vocabulary is used to represent the Student class. We have created this Class on our own instead of reusing the existing one. This class is a subclass of Person Class. For this reference, we have chosen the existing Person class and made use of rdfs:subClassOf property so that our newly created Student class can refer to all the properties of foaf:Person Class. To identify this class, we have used an rdfs:label to give a label as “Student” defined in English language.
- **focu:University ->** This vocabulary is defined to represent the University class. We have created this Class using rdfs:Class, with a rdfs:label named as “University” which is defined in English language.
- **focu:Courses ->** This vocabulary is defined to represent the Courses class. We have created this Class using rdfs:Class, along with a rdfs:label named as “Courses” which is defined in English language.
- **focu:Transcript ->** This vocabulary is defined to represent the Transcript class. We have created this Class using rdfs:Class, along with a rdfs:label named as “Transcript” which is defined in English language.
- **focu:StudentId ->** This vocabulary is defined to represent the StudentId property using rdf:Property, along with rdfs:label named as “studentId” defined in English language. The focu:studentId property also has an identifier comment using rdfs:comment as “University id of a student” with the language tag as English.
- **focu:isEnrolledAt ->** This vocabulary is defined to represent the isEnrolledAt property which is created to identify the relation between the student and university, using rdf:Property along with rdfs:label named as “isEnrolledAt” defined by English language tag. There is also an identifier comment using rdfs:comment given as “Enrolled in University” with the language tag as English. The domain and range of this property is defined as Student and University respectively, as the students can be enrolled in university.
- **focu:isofferedBy ->** This vocabulary is defined to represent the “is offered by” property which is created to identify the relation between the Courses and university using rdf:Property, along with rdfs:label named as ”isofferedBy” defined in English language. Also included are an identifier comment using rdfs:comment as

"University offers courses" with language tag as English. The domain and range of this property are defined as Courses and University respectively, as the courses are offered by the university.

- **focu:takesCourse ->** This vocabulary is defined to represent the "takescourse" property which is created to identify the relation between the Courses and Student using rdf:Property, along with a rdfs:label named as "takesCourse" defined in English language, also including an identifier comment using rdfs:comment as "Students takes courses" with language tag as English. The domain and range of this property is defined as Student and Courses respectively, as this property is modelled on students taking the courses.
- **focu:isAwarded ->** This vocabulary is defined to represent the "isAwarded" property which is created to identify the relation between the Student and Grade classes using rdf:Property, along with a rdfs:label named as "isAwared" defined in English language, also includes an identifier comment rdfs:comment as "Student is awarded the grade" with language tag as English.
- **focu:hasTranscipt->** This vocabulary is defined to represent the "hasTranscipt" property which is created to identify the relation between the Student and Transcript classes using rdf:Property, along with rdfs:label named "hasTranscript" defined in English language, also included with an identifier comment rdfs:comment as "Student has transcript" with language tag as English. The domain of this property is defined as Students.

Knowledge Base Construction

This section provides explanation of the different sections in the knowledge base, which are listed below:

- **Data Set Description**

  Our Data Set describes all the courses along with their description, offered by Concordia university and the student information of those studying at Concordia University and taking those courses. To accumulate these information, we have created three different files that are StudentRecord.csv, Courses.csv and Topics.csv.

- **Courses.csv:** To fetch the details about the Courses, we did web scraping by writing a script in Python, using BeautifulSoup library which is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML. Using this library we created a csv file which contains information about the courses such as the course name, description, course number, course link and course subject.
- **StudentRecord.csv:** This dataset was created manually by entering the records of different random students with their student_id, first_name, last_name, email_address and student_transcript record. The student_transcript record contains information about the courses that the student has taken along with their grades and the semester in which the course was taken by the student.
- **Topics.csv:** To get all the topic names and their DBpedia links associated with a particular course, a python script has been written using pySpotlight Library which returns all the annotated URis from the DBpedia Spotlight for a given text by passing support and confidence level. The text contains the name and description of the course.

- **Tools involved in populating the Data Set into the Knowledge Base:**
  - universityTripleGenerator
  - courseTripleGenerator
  - topicsTripleGenerator
  - studentTripleGenerator
  - transcriptTripleGenerator
- **Process involved in populating the Knowledge Base with the Data Set:**

  Our Knowledge base is populated by running all the above mentioned scripts which are present in the KnowledgeBaseAndQuery.py Python file. The description of each script is given below:

  - The **universityTripleGenerator** function populates the triple of the University instance. The properties which are part of the university subject are rdf:type to describe the University class, foaf:name to describe the name

of the University and rdfs:seeAlso to get more information about the university present in DBpedia..

- The **courseTripleGenerator** function is used to populate each Course instance. The rdf:type property is used to describe the Course class. The name of the course subject is mentioned using the dc:title property. The course subject intance's subject is described by the dc:subject property. The dc:description property is used to attach the description of the course to each course subject instance. We make use of focu:isofferedBy property to link the university information which offers the course. Finally, rdfs:seeAlso property is used to provide an external link in order to get more information about the course.
- The **topicsTripleGenerator** function populates each Topic instance by making use of relevant properties. The first property is the rdf:type property to mention the Topic class of the topic instance. The dc:title property describes the name of the topic subject instance. The third property is the owl:sameAs which points to the DBpedia link of the topic instance. The last property, foaf:primaryTopicOf is used to link the topic to the course instance, to which the topic instance is part of.
- The **studentTripleGenerator** function populates the student instances using some of the properties. The rdf:type property is used to describe the Student class. The second property is focu:studentId which describes the id of each student. The foaf:givenName and foaf:FamilyName properties are used to describe the first name and last name of each student instance respectively. The foaf:mbox property provides the information about a student's email id.
- The **transcriptTripleGenerator** function is used to populate the transcript instances for each student. The first property rdf:type is used to describe the transcript class. The second property dc:identifier is used to describe that the transcript instance is of a particular student using its student id. The focu:takesCourse property shows the link between the Transcript class and the Course class. The focu:isAwarded property is used to describe the grade that the student is awarded for a particular subject. The property dbOntology:termPeriod describes the semester to which the student was enrolled.

- **Entity Linking Process to DBpedia:**

  In this process , there were only course topic names which were to be linked with their corresponding DBpedia links. For this, we made use of owl:sameAs Web Ontology which indicates that the two URI references actually refer to the same thing: the individuals have the same "identity".

Queries

The following points describe the translation of the queries which are used to retrieve information from our knowledge base graph:

1) **Total number of triples in the graph**:

   The first query is about displaying the count of all the triples that are present in our knowledge base. The graph pattern in the WHERE clause involves the variables to match the subject, predicate and object pattern present in the knowledge base graph. The SELECT clause returns the count of all the triples using the COUNT function.

   Example:

```
Hello, I am your smart university agent. Please choose one of the options mentioned below

1. Query 1
2. Query 2
3. Query 3
4. Query 4
5. Query 5
6. Query 6
7. Customize Query
8. Exit
1
Total number of Triples:27008
```

2) **Total number of students, courses, and topics:**

This query is about displaying the count of all the students, courses and topics that are present in our knowledge base.The main SELECT clause returns the count of the students, courses, and topics respectively using the COUNT function. In order to extract all the student, course and topic triples, the main SELECT clause contains three inner SELECT queries. In each of these inner queries, within the graph pattern in the WHERE clause, the first triple pattern uses RDF:type property to bind the subject variable to the respective class. The second triple pattern tries to match all the property instances and their property values to the subject variable.

Example:

```
1. Query 1
2. Query 2
3. Query 3
4. Query 4
5. Query 5
6. Query 6
7. Customize Query
8. Exit
2
Total number of students:15, total number of courses:2184 and total number of topics:2421
```

3) **For a course c, list all covered topics using their (English) labels and their link to DBpedia:**

This query displays all the topics along with their DBpedia links for the given course.The graph pattern in the WHERE consists of three triple patterns. The first triple pattern binds the topic subject variable to the given course name using the property foaf:primaryTopicOf. The second triple pattern tries to match the dc:title property and its value to the topic subject variable. The third triple pattern tries to match the owl:sameAs property and its value to the topic subject variable. Finally, the SELECT clause stores the topic title and its respective URI in the query result set which is displayed to the user.

Example:

```
1. Query 1
2. Query 2
3. Query 3
4. Query 4
5. Query 5
6. Query 6
7. Customize Query
8. Exit
3
Enter the course name:Parallel Programming
The following topics are part of the course Parallel Programming:
Topic title:Neumann and Topic URI:http://dbpedia.org/resource/Neumann_boundary_condition
Topic title:Neumann and Topic URI:http://dbpedia.org/resource/Balthasar_Neumann
Topic title:memory and Topic URI:http://dbpedia.org/resource/Computer_memory
Topic title:memory and Topic URI:http://dbpedia.org/resource/Random-access_memory
Topic title:Parallel programming and Topic URI:http://dbpedia.org/resource/Parallel_computing
Topic title:scalability and Topic URI:http://dbpedia.org/resource/Scalability
Topic title:massively parallel and Topic URI:http://dbpedia.org/resource/MIMD
Topic title:commodity and Topic URI:http://dbpedia.org/resource/Commodity_computing
Topic title:heterogeneous computing and Topic URI:http://dbpedia.org/resource/Heterogeneous_computing
Topic title:message passing and Topic URI:http://dbpedia.org/resource/Message_passing
Topic title:parallel programming and Topic URI:http://dbpedia.org/resource/Parallel_computing
Topic title:message-passing and Topic URI:http://dbpedia.org/resource/Message_passing
Topic title:multi-core and Topic URI:http://dbpedia.org/resource/Multi-core_processor
Topic title:parallel processing and Topic URI:http://dbpedia.org/resource/Parallel_computing
Topic title:load balancing and Topic URI:http://dbpedia.org/resource/Weight_distribution
```

4) **For a given student, list all courses this student completed, together with the grade:**

This query displays all the courses that the given student has completed along with the grade and the semester in which the student has taken the course. In the graph pattern of the WHERE clause, the first triple pattern binds the transcript class to the transcript subject variable using the RDF:type property.The second triple pattern tries to match the studentId with the transcript subject variable using dc:identifier property. For a given student name, we make use of an inner query to get the studentId which matches with the given student name.The third triple pattern tries to match the transcript subject to the course name using the focu:takesCourse property. The fourth triple pattern tries to match the transcript subject through the focu:isAwarded property with the grade. Finally, the fifth triple pattern tries to match the transcript subject with the semester variable using the property dbOntology:termPeriod.

Example:

```
1. Query 1
2. Query 2
3. Query 3
4. Query 4
5. Query 5
6. Query 6
7. Customize Query
8. Exit
4
Enter the name of the student:Jill William
Jill William has completed the Course Soil Testing and Properties with the Grade:C in the term Winter2019
```

5) **For a given topic, list all students that are familiar with the topic (i.e., took, and did not fail, course that covered the topic):**

This query displays all the students with their student ids, first name and given name for a given topic name associated with the courses that the student has taken and not failed. In the graph pattern of the WHERE clause, the first triple pattern binds the transcript class to the course name using the focu:takesCourse property. In order to extract the course name, an inner query has been written in which the triple pattern inside the graph pattern tries to match the topic subject through the dc:title property with the topic name. The next triple pattern inside the graph pattern tries to match the transcript subject through the foaf:primaryTopicOf property with the course name. Finally, the FILTER function is applied to the resultset of the SELECT query to fetch only those students who have not received any “F” grade in their transcript for the courses taken by them.

Example:

```
1. Query 1
2. Query 2
3. Query 3
4. Query 4
5. Query 5
6. Query 6
7. Customize Query
8. Exit
5
Enter the topic:recursion
Below is the list students familiar with the topic recursion:
Student id:10089382 and the Student Name:Jack Myers
```

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):**

This query returns the list of topics that the student is familiar with, for the courses he/she had taken up and cleared the course better than an "F" grade. The first triple pattern in the WHERE clause is bound to the topic subject class through the rdf:type property. The second triple pattern tries to match the course name to the topic subject with the help of the foaf:primaryTopicOf property. Using the dc:title property, the third triple pattern tries to match the topic subject with the name of the topic.

In order to extract the course name, an inner query is written. The graph pattern in the WHERE clause of this inner query is bound to the transcript subject variable through the rdf:type property. The second triple pattern tries to match the transcript subject with the studentId using dc:identifier. The third triple pattern tries to match the transcript subject to the course name with the help of focu:takesCourse property. A FILTER NOT EXISTS function is used to filter out the students who did not score an F grade in the courses that they took.

To fetch the studentId, another inner query is made use of. In the graph pattern of this inner query, the triple patterns try to match the studentId with the studentId provided by the user.

Example:

```
1. Query 1
2. Query 2
3. Query 3
4. Query 4
5. Query 5
6. Query 6
7. Customize Query
8. Exit
6
Enter the student id:10089357
The student with the student id 10089357 is familiar with the following topics:
Topic Name:data acquisition
Topic Name:pattern recognition
Topic Name:Pattern recognition
Topic Name:principal component analysis
Topic Name:neural networks
Topic Name:expert systems
Topic Name:Pattern Recognition
Topic Name:feature extraction
```

7) Customized Query:

This option allows the user to type in a customised query of their choice, whose results will be fetched from the knowledge base graph and displayed to the user.

Example:

Link Analysis:

We analysed the performance of the linking approach of topics to their relevant DBpedia links by randomly picking 100 topics along with their links.

While doing the analysis on the randomly picked 100 topic links, we found out that there are 5 links which were incorrect out of 100. As 95 out of 100 links were correct, the Accuracy score for our topic dataset is 95%.

Below screenshot shows the analysis of all the 100 links generated through DBpedia Spotlight for randomly picked 100 topics:

Neumann	http://dbpedia.org/resource/Balthasar_Neumann	Parallel Programming	Correct	
parallel processing	http://dbpedia.org/resource/Parallel_computing	Parallel Programming	Correct	
message-passing	http://dbpedia.org/resource/Message_passing	Parallel Programming	Correct	
massively parallel	http://dbpedia.org/resource/MIMD	Parallel Programming	Correct	
commodity	http://dbpedia.org/resource/Commodity_computing	Parallel Programming	Correct	
multi-core	http://dbpedia.org/resource/Multi-core_processor	Parallel Programming	Correct	
heterogeneous computing	http://dbpedia.org/resource/Heterogeneous_computing	Parallel Programming	Correct	
memory	http://dbpedia.org/resource/Computer_memory	Parallel Programming	Correct	
load balancing	http://dbpedia.org/resource/Weight_distribution	Parallel Programming	Incorrect	http://dbpedia.
message passing	http://dbpedia.org/resource/Message_passing	Parallel Programming	Correct	
scalability	http://dbpedia.org/resource/Scalability	Parallel Programming	Correct	
Parallel programming	http://dbpedia.org/resource/Parallel_computing	Parallel Programming	Correct	
Machine Learning	http://dbpedia.org/resource/Machine_learning	Machine Learning	Correct	
polynomial regression	http://dbpedia.org/resource/Polynomial_regression	Machine Learning	Correct	
overfitting	http://dbpedia.org/resource/Overfitting	Machine Learning	Correct	
model selection	http://dbpedia.org/resource/Model_selection	Machine Learning	Correct	
logistic regression	http://dbpedia.org/resource/Logistic_regression	Machine Learning	Correct	
decision trees	http://dbpedia.org/resource/Decision_tree_learning	Machine Learning	Correct	
neural networks	http://dbpedia.org/resource/Artificial_neural_network	Machine Learning	Correct	
Support vector machines	http://dbpedia.org/resource/Support_vector_machine	Machine Learning	Correct	
Computational learning theory	http://dbpedia.org/resource/Computational_learning_theo	Machine Learning	Correct	
deep belief networks	http://dbpedia.org/resource/Deep_belief_network	Machine Learning	Correct	
Unsupervised learning	http://dbpedia.org/resource/Unsupervised_learning	Machine Learning	Correct	
k-means	http://dbpedia.org/resource/K-means_clustering	Machine Learning	Correct	
expectation maximization	http://dbpedia.org/resource/Expectationâ€“maximization_	Machine Learning	Correct	
principle component analysis	http://dbpedia.org/resource/Principal_component_analysis	Machine Learning	Correct	
dimensionality reduction	http://dbpedia.org/resource/Dimensionality_reduction	Machine Learning	Correct	
dynamical systems	http://dbpedia.org/resource/Dynamical_system	Machine Learning	Correct	
Markov	http://dbpedia.org/resource/Hidden_Markov_model	Machine Learning	Correct	
Reinforcement learning	http://dbpedia.org/resource/Reinforcement_learning	Machine Learning	Correct	
computer	http://dbpedia.org/resource/Computer_science	Advanced Game Development	Incorrect	https://en.wiki
Game engine	http://dbpedia.org/resource/Game_engine	Advanced Game Development	Correct	
Artificial Intelligence	http://dbpedia.org/resource/Artificial_intelligence	Advanced Game Development	Correct	
AI	http://dbpedia.org/resource/Artificial_intelligence	Advanced Game Development	Correct	
pathfinding	http://dbpedia.org/resource/Pathfinding	Advanced Game Development	Correct	
collision detection	http://dbpedia.org/resource/Collision_detection	Advanced Game Development	Correct	
mobile gaming	http://dbpedia.org/resource/Mobile_game	Advanced Game Development	Correct	
realism	http://dbpedia.org/resource/Philosophical_realism	Advanced Game Development	Incorrect	https://en.wiki
3D	http://dbpedia.org/resource/3D_computer_graphics	Advanced Game Development	Incorrect	https://en.wiki
Computer Vision	http://dbpedia.org/resource/Computer_vision	Computer Vision	Correct	
perceptual organization	http://dbpedia.org/resource/Perception	Computer Vision	Correct	
OpenCV	http://dbpedia.org/resource/OpenCV	Computer Vision	Correct	
numerical methods	http://dbpedia.org/resource/Numerical_analysis	Topics in Scientific Computation	Correct	
scientific computation	http://dbpedia.org/resource/Computational_science	Topics in Scientific Computation	Correct	
nonlinear	http://dbpedia.org/resource/Nonlinear_system	Topics in Scientific Computation	Correct	
ordinary differential equations	http://dbpedia.org/resource/Ordinary_differential_equatio	Topics in Scientific Computation	Correct	
finite difference method	http://dbpedia.org/resource/Finite_difference_method	Topics in Scientific Computation	Correct	
numerical stability	http://dbpedia.org/resource/Numerical_stability	Topics in Scientific Computation	Correct	
collocation	http://dbpedia.org/resource/Collocation_method	Topics in Scientific Computation	Correct	
Renaissance	http://dbpedia.org/page/Renaissance	Studies in Renaissance Literature	Correct	
E-Learning	http://dbpedia.org/resource/Educational_technology	Special Issues in Educational Technology	Correct	
Organization Theory	http://dbpedia.org/page/Organizational_studies	Organization Theory	Correct	
Economics	http://dbpedia.org/page/Economics	Economic Policy After Keynes	Correct	
Keynesian economics	http://dbpedia.org/page/Keynesian_economics	Economic Policy After Keynes	Correct	
Public Policy	http://dbpedia.org/page/Public_policy	Economic Policy After Keynes	Correct	
Canada	http://dbpedia.org/page/Canada	Ageing and Public Policy	Correct	
POLI	http://dbpedia.org/page/POLI	Ageing and Public Policy	Correct	
Canadian	http://dbpedia.org/page/Canada	Ageing and Public Policy	Incorrect	https://en.wiki
Imagination	http://dbpedia.org/page/Imagination	Ageing and Public Policy	Correct	
Quebec	http://dbpedia.org/page/Quebec	Ageing and Public Policy	Correct	
Nationalist	http://dbpedia.org/page/Nationalism	Policy Making and the National Purpose in Ca	Correct	
American	http://dbpedia.org/page/United_States	Policy Making and the National Purpose in Ca	Correct	
E-Learning	http://dbpedia.org/resource/Educational_technology	Studies in English Literature	Correct	
English	http://dbpedia.org/resource/English_language	Studies in English Literature	Correct	
Early English	http://dbpedia.org/resource/English_Gothic_architecture	Studies in English Literature	Correct	
Web design	http://dbpedia.org/page/Web_design	Digital Innovation in Journalism	Correct	
Building information modeling	http://dbpedia.org/page/Building_information_modeling	Building information modeling	Correct	
(BIM)	http://dbpedia.org/page/Building_information_modeling	Building information modeling	Correct	
Building information modeling(B	http://dbpedia.org/page/Building_information_modeling	Building information modeling	Correct	
MEng	http://dbpedia.org/page/Master_of_Engineering	Graduate Seminar in Building and Civil Engin	Correct	
white paper	http://dbpedia.org/page/White_paper	Graduate Seminar in Building and Civil Engin	Correct	
Maturin Murray Ballou	http://dbpedia.org/page/Maturin_Murray_Ballou	Graduate Seminar in Building and Civil Engin	Correct	
Houghton Mifflin	http://dbpedia.org/page/Houghton_Mifflin_Harcourt	Graduate Seminar in Building and Civil Engin	Correct	
shear stress	http://dbpedia.org/page/Shear_stress	Structural Systems for Buildings	Correct	
Membane	http://dbpedia.org/page/Membrane	Structural Systems for Buildings	Correct	
Electrical wiring	http://dbpedia.org/page/Electrical_wiring	Structural Systems for Buildings	Correct	
Wind engineering	http://dbpedia.org/page/Wind_engineering	Wind Engineering and Building Aerodynamic	Correct	
Aerodynamics	http://dbpedia.org/page/Aerodynamics	Wind Engineering and Building Aerodynamic	Correct	
planetary boundary layer	http://dbpedia.org/page/Planetary_boundary_layer	Wind Engineering and Building Aerodynamic	Correct	
Wind	http://dbpedia.org/page/Wind	Wind Engineering and Building Aerodynamic	Correct	
Turbulence	http://dbpedia.org/page/Turbulence	Wind Engineering and Building Aerodynamic	Correct	
Snow	http://dbpedia.org/page/Snow	Wind Engineering and Building Aerodynamic	Correct	
Aerodynamics	http://dbpedia.org/page/Aerodynamics	Wind Engineering and Building Aerodynamic	Correct	
Atmospheric dispersion modelin	http://dbpedia.org/page/Atmospheric_dispersion_modelin	Wind Engineering and Building Aerodynamic	Correct	
Energy management	http://dbpedia.org/page/Energy_management	Computer-Aided Building Operation	Correct	
Computer Simulation	http://dbpedia.org/page/Computer_simulation	Computer-Aided Building Operation	Correct	
HVAC	http://dbpedia.org/page/HVAC	Computer-Aided Building Operation	Correct	
Systems theory	http://dbpedia.org/page/Systems_theory	Fundamentals of Facility Management	Correct	
mathematical optimization	http://dbpedia.org/page/Mathematical_optimization	Fundamentals of Facility Management	Correct	
Energy	http://dbpedia.org/page/Energy	Energy Management in Buildings	Correct	
HVAC	http://dbpedia.org/page/HVAC	Energy Management in Buildings	Correct	
Life-cycle assessment	http://dbpedia.org/page/Life-cycle_assessment	Energy Management in Buildings	Correct	
Stucco	http://dbpedia.org/page/Stucco	Durability of Building Materials	Correct	
Thermal insulation	http://dbpedia.org/page/Thermal_insulation	Durability of Building Materials	Correct	
Wood	http://dbpedia.org/page/Wood	Durability of Building Materials	Correct	
Waterproofing	http://dbpedia.org/page/Waterproofing	Durability of Building Materials	Correct	
AirBarrier	http://dbpedia.org/page/Air_barrier	Durability of Building Materials	Correct	
Structural loads	http://dbpedia.org/page/Structural_load	Durability of Building Materials	Correct	
Air Pollution	http://dbpedia.org/page/Air_pollution	Durability of Building Materials	Correct	

Below is the screenshot showing Incorrect Links which are not relevant to their respective courses:

topic_name	topic_uri	Course	Result	Correct_Link
load balancing	http://dbpedia.org/resource/Weight_distribution	Parallel Programming	Incorrect	http://dbpedia.org/page/Load_balancing_(computing)
computer	http://dbpedia.org/resource/Computer_science	Advanced Game Development	Incorrect	https://en.wikipedia.org/wiki/Computer
realism	http://dbpedia.org/resource/Philosophical_realism	Advanced Game Development	Incorrect	https://en.wikipedia.org/wiki/Realism_(arts)
3D	http://dbpedia.org/resource/3D_computer_graphics	Advanced Game Development	Incorrect	https://en.wikipedia.org/wiki/3D
Canadian	http://dbpedia.org/page/Canada	Ageing and Public Policy	Incorrect	https://en.wikipedia.org/wiki/Canadians

Chatbot Method:

Our University Agent chatbot makes use of Regular Expressions .which are made available through the “re” module in Python in order to translate the questions asked by the user to the chatbot into SPARQL queries. A Regular Expression(RegEx), is a sequence of characters that forms a search pattern. RegEx can be used to check if a string contains the specified search pattern.

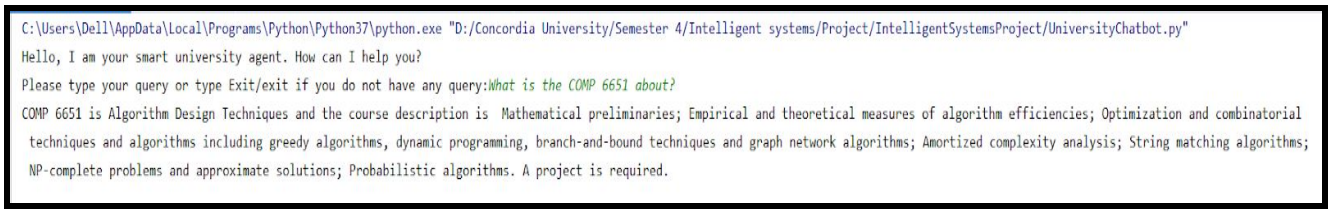
```
result = re.search(r'''[W|w]hat is the (?P<courseName>.*\bw*\b)\?$''', question, flags=re.IGNORECASE)
```

The above screenshot shows the Regex Pattern which we have used to search and match the input question. The re.search() method in Python’s “re” module takes a regular expression pattern and searches for that pattern within the given input question. If the pattern is present in the input question then it returns a matched object. Similar approach is followed for all the other input questions.

Question 1:

Question 1 asks to fetch the description of a particular course which the user wishes to know. Our chatbot tries to look for the course subject and number using regex pattern, for example COMP 6651 in the question asked by the user and returns the course description for that particular course by executing a relevant query in our knowledge base graph.

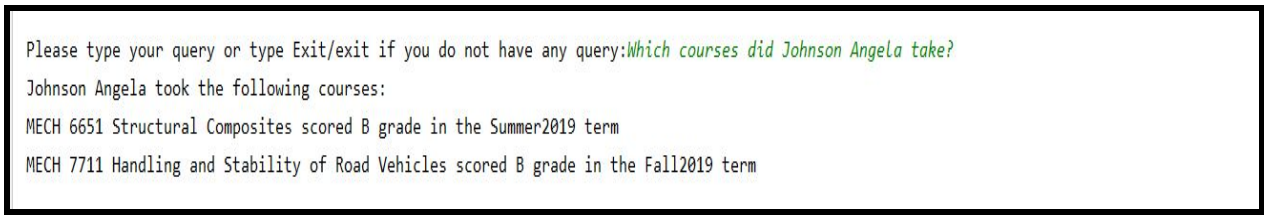
Below is the screenshot for Question 1 along with the answer given by our chatbot:



Question 2:

Question 2 asks to give the information about the courses that a particular user took. Our chatbot with the help of regex pattern tries to look for the course subject, number, grade received and the term in which the student completed those courses, for example here the user asks the chatbot about the courses Johnson Angela took and our chatbot replies with all the above details by translating the input question into SPARQL query and fetching the relevant data of Angela Johnson from our knowledge graph.

Below is the screenshot for the above Question along with the answer given by our chatbot:



Question 3:

Question 3 is about displaying the courses which cover a particular topic that a user wishes to know. Our chatbot with the help of regex pattern searches for the topic name that the user is trying to ask in his/her question and once it matches, the chatbot transforms the question into a query which looks up for the course lists that cover the particular topic present in the user's question and displays the course list to the user.

Below is the screenshot for the above Question along with the answer given by our chatbot:

```
Please type your query or type Exit/exit if you do not have any query:Which courses cover expert systems?
The following courses cover expert systems :
Advanced Pattern Recognition
```

Question 4:

Question 4 asks to display all the students who are familiar with a particular topic. Our chatbot with the help of regex pattern tries to look for the students acquainted with those topics, for example here the user asks the chatbot about the students who are familiar with the topic “anisotropic” and our chatbot replies by giving the names and ids of those students familiar with the topic, by translating input question into relevant SPARQL query and fetching the data from our knowledge graph.

Below is the screenshot for the above Question along with the answer given by our chatbot:

```
Please type your query or type Exit/exit if you do not have any query:Who is familiar with anisotropic?
Below is the list students familiar with the topic anisotropic:
Student id:10089377 and the Student Name:Rahul Agrawal
Student id:10089359 and the Student Name:Johnson Angela
```

Question 5:

Question 5 asks to display all the topics a user is familiar with. Our chatbot with the assistance of the regex pattern tries to look for the topics that the student is familiar with, for example here the user wants to know about the topics that Jack Myers is familiar with. Our chatbot lists those topic names which Jack Myers is familiar with by fetching the data from our knowledge graph with the help of the relevant SPARQL query.

Below is the screenshot for the above Question along with the answer given by our chatbot:

```
Please type your query or type Exit/exit if you do not have any query:What does Jack Myers know?
The student Jack Myers is familiar with the following topics:
Topic Name:recursion
Topic Name:least-squares
Topic Name:normalized
Topic Name:convergence
Topic Name:lattice
Topic Name:LMS
Topic Name:adaptive filter
```