COMP4332/RMBI4310 (Spring 2018)
Project: Course Registration Data Analytics

Last Update: 15 Feb, 2018

Due Date for Phase 1: 21 Feb, 2018 1:30pm (via Canvas (softcopy))
Due Date for Phase 2: 9 March, 2018 1:30pm (in class (hardcopy) and via Canvas (softcopy))
Due Date for Phase 3: 28 March, 2018 1:30pm (via Canvas (softcopy))
Due Date for Phase 4: 11 April, 2018 1:30pm (in class (hardcopy) and via Canvas (softcopy))
Due Date for Phase 5: 25 April, 2018 1:30pm (in class (hardcopy))
Due Date for Phase 6: 4 May, 2018 1:30pm (in class (hardcopy) and via Canvas (softcopy))

1. Introduction

You are given a website which contains the information about course enrollment status in a period of a semester. There are the following objectives in this project.

1. The first objective is to find a list of courses which match some criteria.
2. The second objective is to predict the number of students in the waiting list of a lecture section of a course in a given time slot.

There are six phases in this project – Phase 1, Phase 2, Phase 3, Phase 4, Phase 5 and Phase 6. After we finish Phase 4, we complete tasks for the first objective. After we finish Phase 6, we complete tasks for the second objective (in addition to the first objective).

- In Phase 1, you are required to implement your system interface with *stubs*. (Details of stubs can be found in Section 6).
- In Phase 2, you are required to design the database format and start a database server according to the information given in the website. You are required to write a design report and write a database script. After the deadline of Phase 2, you will be provided a suggested answer of the database format.
- In Phase 3, you are required to continue implementing the system from Phase 1

for the feature of obtaining/crawling the information given in the website and storing this information into the database at the database server. After the deadline of Phase 3, you will be provided a suggested answer of the program used for Phase 3.

- In Phase 4, you are required to follow the design report in Phase 2 and write a design report for the first objective of this project and continue implementing your system from Phase 3 for the first objective.
- In Phase 5, you are required to write a design report for the second objective of this project.
- In Phase 6, you are required to follow the design report in Phase 5, continue implementing your system from Phase 4 for the second objective and write a final report.

Project should be completed in form of groups. For each group, only one copy is required.

In this project, you are required to use Python 3.6.3 (or above) for implementing the system, use MongoDB 3.4.10 (or above) for implementing the database, and use Keras 2.1.1 (or above) (built on TensorFlow 1.5.0rc0 (or above)) for implementing the data analytics part.

2. Milestones

1. Phase 1
   i. Implement your system interface with stubs with respect to the functionalities stated in Section 5. Please implement as many system components as possible so that your work in later phases can be reduced.
   ii. You are suggested to follow the interface designed by you in Phase 1 and continue implementing the system in later phases.
   iii. Hand in your system interface with stubs (i.e., program(s)) with respect to Section 9.
2. Phase 2
   i. You are given a website link containing links to webpages about the course enrollment status in different time slots of each of the days in a period of a semester.
   ii. Each webpage shows the course enrollment status and the course

information of different courses (offered by a single program/unit) in a time slot on a day. The detailed course information could be found in the webpage.

iii. Read some webpages and understand the structure of the webpages in your browser (e.g., the course enrollment status and the course information) (Note: There is no need to read the HTML codes/scripts of these webpages in this phase. Reading webpages in your browser is sufficient).

iv. Design a database format for these webpages

v. Set up a database server

vi. Write a NoSQL (insertion) script for illustrating how to insert 2 database documents/records each of which corresponds to the course enrollment status of a single course in a single time slot
(Note: This script just includes 2 examples of document insertions and thus the contents of 2 documents are hardcoded).

vii. Write a NoSQL (query) script for illustrating how to perform each of the operations for the first objective (i.e., each of the operations required in Section 5.3) with one query concrete example.
(Note: This script just includes 1 example of a document query (or a number of consecutive document queries) for each operation and thus the contents of queries are hardcoded).

viii. Write a report showing the database format, your NoSQL insertion script and your NoSQL query script with explanation.

ix. Hand in NoSQL Design Report and NoSQL Scripts with respect to Section 9.

x. After the deadline of Phase 2, you will be provided a suggested answer of the database format.

3. Phase 3

i. Read the HTML file of one (or some) of the webpages at the website.

ii. Understand the code of the HTML file.

iii. Continue implementing the system from Phase 1 for the feature of the data crawling part which obtains the information of each webpage and inserts this information into the database of the database server set up in Phase 2

iv. Hand in your system interface (i.e., program(s)) supporting the data crawling and DB insertion features with respect to Section 9.

v. After the deadline of Phase 3, you will be provided a suggested answer of the program used for Phase 3.

4. Phase 4
   i. In Phase 4, you are required to follow the design report in Phase 2 and write a design report for the first objective of this project.
   ii. Continue implementing the system from Phase 3 for the first objective.
   iii. Hand in NoSQL Design Report and your system interface (i.e., program(s)) supporting NoSQL queries with respect to Section 9.
5. Phase 5
   i. In Phase 5, you are required to write a design report for the second objective of this project.
   ii. You should list 5 possible models you want to try
   iii. Note that a possible model can be a "neural network" with a set of input parameter values and another possible data mining model can be a "neural network" with another set of input parameter values. Obviously, one of the possible models can be a "recurrent neural network" with a set of input parameter values. One possible (complicated) model could be a set of a "neural network" (with a set of input parameter values) used in a particular period and another "neural network" (with another set of input parameter values) used in the remaining period.
   iv. Hand in the design report for data analytics with respect to Section 9.
6. Phase 6
   i. In Phase 6, you are required to follow the design report in Phase 5.
   ii. Continue implementing the system from Phase 4.
   iii. The final report should include
      1. your design for each of the 2 objectives (i.e., the design for the first objective used in Phase 4 and the design for the second objective used in Phase 5).
      2. the answer of each of the following "specific" questions. If possible, please elaborate the answers of these questions together with the design.

         (Note that the following "specific" questions are used for the report. Your program to be submitted/written should accept any input with the format stated in Section 5).

         - Specific Question 1a:
           to find a list of courses which course titles or course descriptions match text "Visualizing Data".
         - Specific Question 1b:
           to find a list of courses each of which has a lecture section

(e.g., “L1” and “L2”) in a time slot between 10am on 28 Jan (inclusively) and 3pm on 28 Jan (inclusively) where the number of students in the waiting list of this lecture section is greater than or equal to twice the number of students enrolled in this lecture section in that time slot.

- Specific Question 2:
  to predict the number of students in the waiting list of the course with the course code “COMP1942” (Lecture No. 1) at 2:30pm on 26 Jan.

3. an additional section called "Contribution". For each member in the group, please write a sub-section with about 500 words stating your contributions to the project (e.g., what you have done for this project)

iv. Test your system with different possible inputs.

v. Check that you can fulfill all requirements.

vi. Finally, hand in the final report and program code with respect to Section 9 and be ready for the demonstration.

3. Website Details

You are given a website link containing links to webpages about the course enrollment status in different time slots of each of the days in a given period of a given semester. In each of the days, there are the following time slots

- 00:00am
- 00:30am
- …
- 11:30pm

except the following time slots (since they have the same enrollment status as 7:30am)

- 8:00am
- 8:30am
- 9:00am

When we write “9:30am” as a time slot, what we really mean is the time slot between 9:30am and 10am. For simplicity, in the project, you just simply write it as “9:30am” as a time slot.

It is worth mentioning that in real life applications, we may encounter “missing”

data. In the link, we also removed some time slots of some days in the given period to simulate this scenario.

Some course information found in each webpage is shown as follows.

1. Course Code
2. Course Title
3. No. of Units/Credits
4. Pre-requisite
5. Exclusion
6. Course Description
7. A list of sections of the course each with
    a. Section (the lecture/tutorial/lab (or others) section number)
    b. Date & Time (the data and the time of a lesson of the course)
    c. Room (the room/venue of the course)
    d. Instructor (the instructor of this course)
    e. Quota (the maximum number of students to be enrolled in this course)
    f. Enrol (the number of students enrolled in this course)
    g. Avail (the number of available seats in this course)
    h. Wait (the number of students in the waiting list of this course)
    i. Remarks (some remarks related to the course)

4. Format and Sample Program

4.1 Format

1. "Course Code" is represented in the format of "CCCCXXXX" or "CCCCXXXXC" where "C" denotes a capitalized letter and "X" denotes a digit. One example of the course code is "COMP4332" and another example is "RMBI4310". Another example of the course code is "COMP1022P".
2. "Time Slot" is represented in the format of "YYYY-MM-DD HH:mm" where "YYYY" denotes the year, "MM" denotes the month and "DD" denotes the day of the month, "HH" denotes the hour and "mm" denotes the minute. One example of the time slot is "2018-02-01 15:30".

4.2 Sample Format and Sample Program

4.2.1 Database Format

The database format will be given after the deadline of Phase 2. The details will be posted in the course homepage at that time.

4.2.2 Data Crawling Program with the Database Insertion Feature

The data crawling program with the database insertion feature will be given after the deadline of Phase 3. The details will be posted in the course homepage at that time.

5. Program Requirement

The program should be implemented in text (console) mode.

In the program, you have to let the user *choose* and *change* one of 5 features – "Collection Dropping and Empty Collection Creating", "Data Crawling", "Course Search", "Waiting List Size Prediction" and "Waiting List Size Training".

The following shows the 5 features.

5.1 Collection Dropping and Empty Collection Creating
(This feature will be used for the demonstration purpose.
The detailed implementation of this feature will be completed by you in Phase 3.)

Input: none
Output: Display a message "Collection dropping and empty collection creating are successful" (after the collection(s) is(are) removed and the new empty collection(s) is(are) created).

5.2 Data Crawling
(The detailed implementation of this feature will be completed by you in Phase 3.)

Input: a URL (e.g., "http://course.cse.ust.hk/comp4332/index.html")
or
a special keyword (i.e., "default")
Output: If the input is "default", display a message "Data Crawling is successful and all data are inserted into the database" (after all data are crawled from the default URL given in the project webpage and are inserted into the database). Otherwise, do the same prompt operation but the URL used is the URL typed in the input.

5.3 Course Search
(The detailed implementation of this feature will be completed by you in Phase 4. But, the design of this feature will be completed in Phase 2.)

We have the following two operations for a course search.

1. Course Search by Keyword

2. Course Search by Waiting List Size

Note: Although there are some missing data in this project (which may require "prediction"), in this part/feature, you just perform these operations for a course search only based on the data given to you. There is no need to perform any "prediction" in this part.

5.3.1 Course Search by Keyword

Input: a text (k) where this text is called "keyword(s)"
Output: A list of courses which course titles, course description or course remarks match the given text k.

In the output, for each course, please show "Course Code", "Course Title", "No. of Units/Credits", a list of sections of the course each with "Section", "Date & Time", "Quota", "Enrol", "Avail" and "Wait".
Please sort the list of courses in ascending order of "Course Code".
(Within a single course, please sort in ascending order of "Sections")

We say that a phrase P matches text k if at least one of the words in phrase P is equal to one of words in k. For example, if P = "Big Data Mining and Management" and k = "Mining", then P matches k. If P = "Big Data Mining and Management" and k = "Risk Mining", then P matches k too. If P = "Big Data Mining and Management" and k = "Mining Management", then P matches k.

5.3.2 Course Search by Waiting List Size

Input: A non-negative real number f
Starting Time Slot (start_ts)
Ending Time Slot (end_ts)
Output: A list of courses each of which has a lecture section (e.g., "L1" and "L2") in a time slot, says match_ts, between start_ts (inclusively) and end_ts (inclusively) where the number of students in the waiting list of this lecture section is greater than or equal to f multiplied by the number of students enrolled in this lecture section in that time slot.

In the output, for each "distinct" course, please show "Course Code", "Course Title", "No. of Units/Credits", "Matched Time Slot", a list of sections (including both lecture

sections and non-lecture sections) of the course each with "Section", "Date & Time", "Quota", "Enrol", "Avail", "Wait" and "Satisfied" (all shown with the content/values recorded in the time slot match_ts).
Note that "Matched Time Slot" is a new attribute in this query and it is equal to match_ts.
If a single course satisfies the required condition in *multiple* time slots (no matter which lecture section of this course satisfies the required condition), we just show the latest time slot among all these time slots in which this course satisfies the required condition.
Thus, each course should appear at most once in the output.

Note that "Satisfied" is another new attribute in this query. It is equal to "Yes" if the number of students in the waiting list of this section is greater than or equal to f multiplied by the number of students enrolled in this section in that time slot. It is equal to "No" otherwise. Attribute "Satisfied" is not needed to be considered in Phase 2.

Please sort the list of courses in ascending order of "Course Code".
(Within a single course, please sort in ascending order of "Sections")

5.4 Waiting List Size Prediction
(The detailed implementation of this feature will be completed by you in Phase 6. But, the design of this feature will be completed in Phase 5.)

Input: Course Code (cc)
Lecture number (ln) (e.g., the input should be "1" denoting "L1")
Time Slot (ts)

Output: "N_1, N_2, N_3, N_4, N_5"
where N_i denotes the number of students in the waiting list of the lecture number (ln) (if any) of the course cc in the given time slot (ts) predicted by Model i for each i in [1, 5] (Note that these 5 numbers are integers.)

Note: Since we know that training a model may take some time, in general, "cc" could be any course code. However, in our demonstration, we will test with the course code "cc" starting from "COMP1942", "COMP42", "COMP43" or "RMBI" only (i.e., (1) the COMP course ("COMP1942"), (2) any COMP course with starting course digits equal to "42" or "43" and (3) any RMBI course). Thus, you just need to train your model with the data from the course with the course code "cc" starting from

these course code prefixes described above **before** our demonstration. When we use this feature in our demonstration, you just need to **load** the trained model and perform the prediction of this feature based on the trained model.

If there is no lecture section of the course (cc) specified in the input or if the lecture number entered (ln) is not offered for the course (cc) specified in the input, we just need to show "There is no lecture section and thus there is no prediction result."

Although it is possible that we could use 5 models for one course and we could also use 5 "different" models for another course, for the sake of simplicity, please use the same 5 models for any course needed. Of course, even if the 5 models are the same (with the same set of "input" parameter values) for any two courses, we know that each of the 5 models could be trained with different enrollment data from different courses, resulting in different "model" parameter values (e.g., the weight values between neurons in a neural network which should be found from the data).

5.5 Waiting List Size Training
(This feature will be used for your "own" training purpose before we have the real feature from Section 5.4.
The detailed implementation of this feature will be completed by you in Phase 6. But, the design of this feature will be completed in Phase 5.)

Input: none
Output: Display a message "Waiting list size training is successful" (after the training process on the waiting list size finishes).

6. Stubs

Stub is one of the major components of the top-down design in the software engineering cycle. A program with stubs is a program that

- can be executed
- does something to tell you that it can be executed successfully
- doesn't (yet) do the details of what you really want it to do
- has the structure of how you want it done

In the later processes of the software engineering cycle, the program with stubs needs to be refined so that it can achieve what you really want.

You are required to implement the system interface with stubs in Phase 1. You can implement the system with menus and allow the users to input some keys to enter some function/action pages. For each function/action, you can implement with stubs. For instance, if we enter a text in "Course Search by Keyword", no matter how the input is, the information of a predefined (or hard-coded) list of courses should be displayed. Of course, you have to modify the hard-coded components in later phases for the *real* implementation. This task requires you to write the flow of the program without filling in the details of each function.

There is no special requirement for the system interface with stubs. You are urged to finish implementing the system components as much as possible (e.g. comment, menu displaying, error checking and the result displaying) in Phase 1 so that you can be familiarized with the programming language in Phase 1 and can lessen the workload in later phases. Note that you are not required to implement any code related to the NoSQL script and any code related to the data crawling part in Phase 1. What we check with your program is whether it can be executed with the features required in Section 5.

7. Grading Policy

Phase 1

- **System Interface with Stubs (i.e., program(s)) (softcopy) (10%)**

Phase 2

- **NoSQL Design Report (hardcopy) (5%)**
- **NoSQL Scripts (softcopy) (5%)**

Phase 3

- **Program Supporting Data Crawling and DB Insertion (softcopy) (10%)**

Phase 4

- **NoSQL Design Report (hardcopy) (5%)**
- **Program Supporting NoSQL queries (softcopy) (10%)**

Phase 5

- **Design Report for Data Analytics (hardcopy) (5%)**

Phase 6

- **Final Report (hardcopy) (20%)**
- **Final program (softcopy) (30%)**

Mark Deduction

- **Phase 2 and Phase 3**

  No late submission is allowed for Phases 2 and 3 since the schedule of our project is quite tight. All submissions after the deadline of each of these phases will *not* be accepted.

- **Phase 1, Phase 4, Phase 5 and Phase 6**

Number of Days Late	Deduction (out of 100 marks)
1	10
2	30
3	70
4 or above	100

8. Demonstration

You need to sign up for a demonstration. No group is allowed to demonstrate twice. All members of each group should be present during the demonstration. The duration for each group will be about 20 minutes. You should bring your laptop for demonstration. You could run your database server and your system at your laptop during the demonstration. Before you come to the demonstration, make sure that your database server stores all information collected from the website given (i.e., the data crawling part from the default website has been done before the demonstration).

Note that although your database server stores all information collected from the default website given before the demonstration, at the end of our demonstration, we will execute the feature in Section 5.1 and then the feature in Section 5.2 (with a non-default URL) so that we could test your data crawling operation and your database operation. After that, we will execute the feature in Section 5.3 (not Section 5.4 and Section 5.5 since training a model (related to Section 5.4 and Section 5.5) takes time).

In this project, you can use at most 2 coupons for the "final program" part of Phase 6. In this project, you are required to implement every component required in this project even though you have coupons. The following shows the instructions.

1. Each group can use at most two coupons regardless of the total number of members in the group.
2. Each coupon can be used to add 20% of the total score of the final program part
3. If the total score computed together with the scores from coupons for the final program part is greater than 100%, the total score will be truncated to 100%.
4. Each coupon is used once only.
5. After each coupon is used, it could not be used again.
6. Each coupon is non-transferrable. That is, the coupon with a unique ID can be used only by the student in the group who obtained it in class.
7. Please bring the coupon(s) to the demonstration.

9. Turn-in

1. **Phase 1**
   a. **System Interface with Stubs (i.e. program(s)) (softcopy)**
   Please zip the following files into a single file and submit as a ZIP file. The file name is "XXX.zip" where "XXX" is your group ID. If your group ID is 12, then your file name is "12.zip".
      i. A soft copy of your Python program (e.g., "main.py")
      ii. readme file (readme.txt) which contains
         1. group information
         2. file list
         3. file description
         4. method of execution (e.g., "python main.py")
         5. known bugs of your system

2. **Phase 2**
   a. **NoSQL Design Report (hardcopy)**
      i. the database format
      ii. the NoSQL insertion script (with explanation)
      iii. the NoSQL query script (with explanation)
   b. **NoSQL Scripts (softcopy)**
   Please zip the following files into a single file and submit as a ZIP file. The file name is "XXX.zip" where "XXX" is your group ID. If your group ID is 12, then your file name is "12.zip".
      i. A soft copy of your insertion script (e.g., "insertion.js")
      ii. A soft copy of your query script (e.g., "query.js")
      iii. readme file (readme.txt) which contains
         1. group information
         2. file list
         3. file description
         4. known bugs of your system
3. **Phase 3**
   a. **Program Supporting Data Crawling and DB Insertion (softcopy)**
   Please zip the following files into a single file and submit as a ZIP file. The file name is "XXX.zip" where "XXX" is your group ID. If your group ID is 12, then your file name is "12.zip".

i. A soft copy of your Python program (e.g., "main.py")
ii. readme file (readme.txt) which contains
    1. group information
    2. file list
    3. file description
    4. method of execution (e.g., "python main.py")
    5. known bugs of your system

4. **Phase 4**
    a. **NoSQL Design Report (hardcopy)**
        i. The requirement is the same as the NoSQL Design Report in Phase 2 except that in Phase 4, we should consider the new attribute "Satisfied" mentioned in Section 5.3.2
    b. **Program Supporting NoSQL queries (softcopy)**
        Please zip the following files into a single file and submit as a ZIP file. The file name is "XXX.zip" where "XXX" is your group ID. If your group ID is 12, then your file name is "12.zip".
        i. A soft copy of your Python program (e.g., "main.py")
        ii. readme file (readme.txt) which contains
            1. group information
            2. file list
            3. file description
            4. method of execution (e.g., "python main.py")
            5. known bugs of your system

5. **Phase 5**
    a. **Design Report for Data Analytics (hardcopy)**
        A hard-copy design report includes the following:
        i. The specification of the 5 models
6. **Phase 6**
    a. Your Final Report (hardcopy)
        i. The requirement could be found in Section 2
    b. **Your Final Program (softcopy)**
        Please zip the following files into a single file and submit as a ZIP file. The file name is "XXX.zip" where "XXX" is your group ID. If your group ID is 12, then your file name is "12.zip".
        i. A soft copy of your Python program (e.g., "main.py")
        ii. readme file (readme.txt) which contains

1. group information
2. file list
3. file description
4. method of execution (e.g., “python main.py”)
5. known bugs of your system