Design Laboratory (CS69202)
Spring Semester 2023

Topic: SQL, ML, Socket

Date: April 5, 2024

Time: 2.00 PM - 5.00 PM

Instructions:

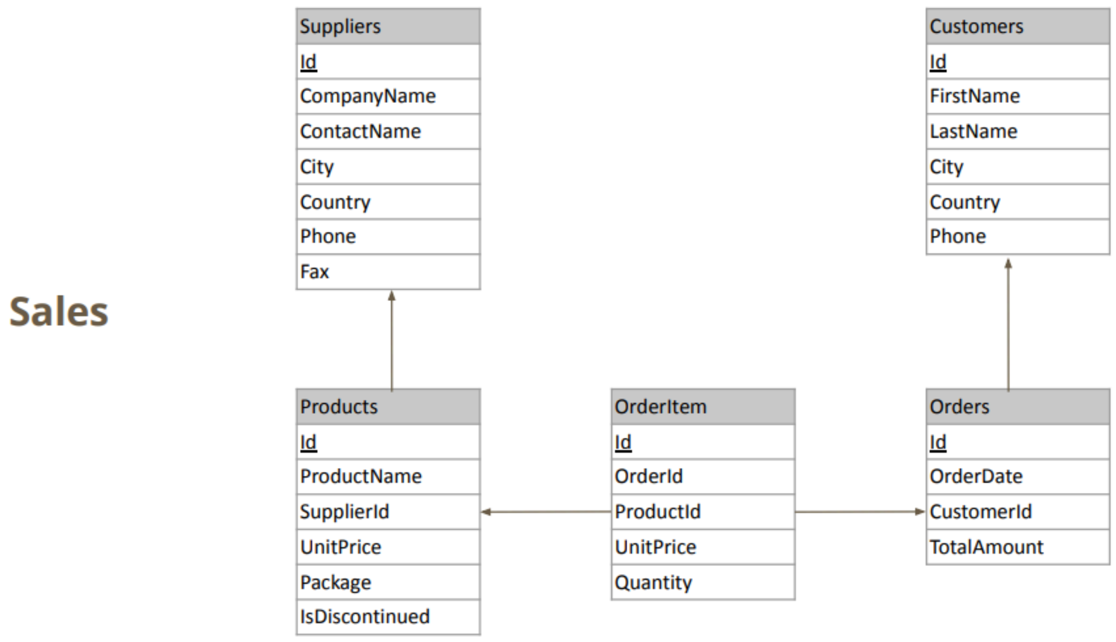
1. Attempt **all** the questions. The paper is divided into 3 parts : SQL, ML and Socket. There are 2 questions in SQL, 1 question in ML and 1 question in Socket.
2. There will be no internet access during the exam. Make sure to download any required materials, documentation, reference guides, and install coding prerequisites beforehand.
3. No communication is allowed between students once the exam has begun. Any form of communication will be considered a violation of exam rules.
4. Manage your time wisely. Be aware of the allocated time for the exam to ensure you have sufficient time to complete all questions.
5. For SQL, we will be using the same database as the one done in the Mid Sem test. Please make sure the tables and data are in place.
6. For ML, make sure you install all the prerequisites beforehand.
7. For Socket,
    a. Make sure you use C programming language or C++ programming language without STL.
    b. A proper error handling (e.g., when the syscall or the library calls fail) is expected.
    c. The inputs should be taken from the Terminal and results displayed in the Terminal, **if explicitly not mentioned**.
    d. The IP address and port number will be provided by Command Line Interface (if required).
8. You may provide a README file (not mandatory). If you don't, mention the steps as to how to run your program.

Submission Format:

1. Create a folder with the name <Roll.No>_DesLab_EndSenTest. Inside the folder, create three folders SecA, SecB, SecC.
2. For each of the folders, do the following :
   a. SecA : Take the screenshot for output of queries in which SQL query is also visible if the screenshot does not fit in one page then take multiple screenshots and make a document including screenshots, 1st query is mandatory and from 2nd and 3rd you have to do one query.
   b. SecB : Please follow the deliverables as discussed in each task.
   c. SecC : Submit client.c and server.c files in the folder for the question in Section C.
3. Zip the master folder with the name <Roll.No>_DesLab_EndSenTest.zip and Submit it in Moodle.

SECTION - A
SQL

Consider the schema of the Sales Database as shown in the diagram below:



The tables mentioned in the above schema has the following attributes:

Suppliers:
Columns:
○ Id (Primary Key): Unique identifier for each supplier.
○ CompanyName: Name of the supplier company.
○ ContactName: Name of the contact person at the supplier.
○ City: City where the supplier is located.
○ Country: Country where the supplier is located.
○ Phone: Supplier's phone number.
○ Fax: Supplier's fax number (if applicable).

Products:
Columns:
○ Id (Primary Key): Unique identifier for each product.
○ ProductName: Name of the product.

○ SupplierId (**Foreign Key**): References the supplier who provides the product.
○ UnitPrice: Price per unit of the product.
○ Package: Packaging type for the product (e.g., box, bottle).
○ IsDiscontinued: Indicates whether the product is discontinued (yes/no).

OrderItem:
Columns:
○ Id (Primary Key): Unique identifier for each order item.
○ OrderId (**Foreign Key**): References the order to which the item belongs.
○ ProductId (**Foreign Key**): References the product included in the order item.
○ Quantity: Quantity of the product ordered.
○ UnitPrice: Price per unit of the product at the time of the order.

Orders:
Columns:
○ Id (Primary Key): Unique identifier for each order.
○ OrderDate: Date the order was placed.
○ CustomerId (**Foreign Key**): References the customer who placed the order.
○ TotalAmount: Total cost of the order.

Customers:

Columns:
○ Id (**Primary Key**): Unique identifier for each customer.
○ FirstName: Customer's first name.
○ LastName: Customer's last name.
○ City: City where the customer is located.
○ Country: Country where the customer is located.
○ Phone: Customer's phone number.

IMPORTANT NOTE:

1. In this problem Limit is allowed
2. The 1st query is mandatory and from 2nd and 3rd you have to do one query.

From the Sales database created, Write SQL queries to give output of the following queries :

1. Find the top 3 customers who have spent the most in total.

Do any ONE of 2 and 3 below.

2. Determine the customer who has made the highest number of orders in a single day.

3. Identify the customers who have placed orders on weekends.

SECTION - B
Machine Learning

In this task you are provided with the Quora Question Pair dataset. Each entry in the dataset consists of two questions (question1 and question2) and your task is to predict whether the two input questions are duplicate [is_duplicate: 1] or not [is_duplicate: 0].

We have divided the entire task into 5 sections. Each section is specific to subtasks like preprocessing, vectorization and so on. Specific to each section is a deliverable (a python file) and expected input and output format to be carried out to perform the subtask.

1. Preprocessing:

Please follow the following preprocessing steps for each of the questions:

1. convert everything to lowercase.
2. remove hyperlinks.
3. remove non-alphanumeric characters.
4. remove inflections with word lemmatizer from NLTK.
5. remove HTML tags.

Deliverable: Preprocessing.py
Input: the raw dataset in csv format
Output: preprocessed csv with name Preprocessed_input.csv

2. Getting Representations:

1. Shuffle and split the preprocessed dataset into the train and test part maintaining a split ratio of 80:20 (using train_test_split functionality of sklearn).
2. Train TfIdfVectorizer functionality of sklearn on all the questions in the train set.
3. For each entry in both train and test split, encode the question pair using trained tf-idf and store them in pickle format. The key-value pair for dictionary is as follows:
    a. Key: the index of entry
    b. Value: list of list: index 0 of the list contains representation for first question, index 1 contains representation of second question
4. Additionally, store the labels of train and test split in two separate pickle files.

Deliverable: Representation.py
Input: Preprocessed_input.csv
Output: Train.csv, Test.csv, train_representation.pkl, test_representation.pkl, y_train.pkl, y_test.pkl .

3. Feature extraction

Please compute the following features for every data point in the csv from both train split [Train.csv] and test split [Test.csv] :

1. features ***q1_char_num, q2_char_num***: count of characters for both questions.
2. features ***q1_word_num, q2_word_num*** : count of words for both questions.
3. feature ***total_word_num*****:** sum of q1_word_num and q2_word_num.
4. feature ***differ_word_num*****:** absolute difference between q1_word_num and q2_word_num.
5. feature ***same_first_word*****:** 1 if both questions have the same first word otherwise 0.
6. feature ***same_last_word*****:** 1 if both questions have the same last word otherwise 0.
7. feature ***total_unique_word_num***: total number of unique words in both questions.
8. feature ***total_unique_word_withoutstopword_num*****:** total number of unique words in both questions without the stop words.
9. feature ***total_unique_word_num_ratio***: total_unique_word_num/ total_word_num.
10. feature ***common_word_num*****:** count of total common words in both questions.
11. feature ***common_word_ratio*****:** common_word_num/total_unique_word_num.
12. feature ***common_word_withoutstopword_num***: total common words in both questions excluding the stopwords.

Please compute the following features using the train_representation.pkl and test_representation.pkl:

13. feature ***cosine_similarity_tf_idf, euclidian_distance_tf_idf:*** measures similarity and distance between both pairs of questions with tf-idf representation.
14. feature ***rep_1***: tf-idf representation of question1.
15. feature ***rep_2:*** tf-idf representation of question2.

Your total number of features will be: 16 + |rep_1|+|rep_2|, where |x| is the size of vector x for each entry.

Kindly create a 2D numpy array of size num_of_data_points*total_num_of_features storing the features for each entry.

Deliverable: Features.py
Input: Train.csv, Test.csv, train_representation.pkl, test_representation.pkl.
Output: Features_train.npy, Features_Test.npy

4. Training Classification Model

Please train the following scikit-learn's classification model using the features obtained in Step 4.:

1. SVM
2. Neural Network

3. Logistic Regression

Dump the models into pickle format once trained.

Deliverable: Models.py
Input: Features_train.npy, y_train.pkl.
Output: svm.pkl, NeuralNetwork.pkl, and LogisticRegression.pkl

5. Evaluating Models

Please obtain predictions on the test split from each of the trained models and report the output of classifcation_report on the terminal for each of the trained models.

Deliverable: Evaluation.py
Input: Features_test.npy, y_test.pkl, svm.pkl, NeuralNetwork.pkl, and LogisticRegression.pkl
Output [On terminal]: Output of classification_report for each model.

6. Running Pipeline End-to-End

Now that you have all the components in place its time to finally run your ML pipelines end to end. Please provide a shell script which upon execution sequentially performs task 1 to 5.

Deliverable: runEndtoEnd.sh

SECTION - C
Socket Programming

CRC(Cyclic Redundancy Check) is a protocol which is used for error detection in socket/frames in the transport/data link layer of the TCP/IP protocol channel. Refer to the materials to know more.

Write a TCP Client and Server codes (Multi-Client Server) that does the following things :

1. The client and server establish a connection. fork() or select() system call is used at the server end to handle multiple client requests (**At most 10 clients** can be handled at one point of time). An acknowledgement message is sent to the particular client's end for successful connection.
2. In the client end, the client enters an arithmetic expression in the form of a string (without spaces). Example : 2+3*4 (this is passed as an input to the server by the client). Only the four operations +,-,*,/ are allowed with *,/ having higher priority over +,- . All are left to right associative. Single digit numbers are to be used. Output can be more than one digit. (eg. 2+3*4 = 2+(3*4)= 2+12 = 14). No braces are required.
3. Each character of the arithmetic expression is converted to an 8-bit representation of its ASCII value. For the character '2' in the given expression 2+3*4 , its ASCII is 50. Convert it into an 8 bit binary string (00110010) and store it in any datatype you want to store (you are free to use any data structure but not STL library calls).
4. Append a '1' to the converted binary string Therefore the resultant is a 9-bit string (of the form **1 xxxx xxxx** where **xxxx xxxx** is the **8-bit ASCII representation of a character** in an expression. Example, for '2' the representation is **1 0011 0010**).
5. The 9-bit binary string (**1 0011 0010**) is to be processed through a method which converts it into a 12-bit **CRC output generated string**. The generator function for the **CRC** is always **1101**. Example : The CRC for 9-bit string of '2' will have input string **1 0011 0010 000** (3 zeros appended as per CRC definition) and the output generated from **CRC** with **1101** as the generating polynomial **(1101)** is **1 0011 0010 011.** Here **011** is the remainder obtained from the CRC operation. Refer to class lecture.
6. The expression string is then sent to the server in 12-bit format. For the expression "2+3*4", the string sent is 5*12 = 60-bit string where each of the 5 12-bit strings represents the 12-bit string explained before.

7. The string is processed in the server by checking if the data is valid by passing through a CRC checker function. If the data is invalid, an error message is sent to the respective client. Else, the expression is **brought back** t**o its original string** and the **expression is evaluated** using **stack implementation of expression evaluation** and the output generated from the server is sent to the client.
8. This process goes on between server and client until the client gives the '\exit' command which requests the server to terminate the connection.

Implement the above code by doing the following things in the client and server side:

<u>Client Side (client.c):</u>

1. Connect to Server.
2. Create a CRC polynomial converter function so that on receiving the string, it converts each character into 12-bit strings and returns the same to a variable. **Generator function is 1101**.
3. Take input from the user and process it in a CRC polynomial converter and pass it to the server.
4. Wait till the server processes your requests and sends the output.
5. Do it till the user wants to exit the server.

<u>Server Side (server.c):</u>

1. Create a socket, and listen for client connections.
2. After the successful connection, use fork() or select() (any 1 will suffice as per your design decisions) to handle multiple clients. Make sure at most 10 clients can be serviced at one time.
3. After receiving input from a particular client, verify the input using CRC polynomial checker to check each 12-bit string. **Generator function is always 1101**.
4. If the checker detects an error, it gives the error signal and sends an error message to the client.
5. If the checker finds the input valid, it evaluates the arithmetic expression (after converting the string into the original arithmetic expression) and returns the output to the client.
6. This communication between server and client continues until either of them stops/exits the connection

Sample Example 1:

Client-Server connection established.

The terminal will look as below.

```
>Connected to Server
>2+3*4
>12
```

Explanation :

The client sends the expression “2+3*4” of the 12-bit format :

```
1 0011 0010 011 (‘2’)
1 0010 1011 001 (‘+’)
1 0011 0011 110 (‘3’)
1 0010 1010 100 (‘*’)
1 0011 0100 111 (‘4’)
```

The string sent by the client to the server is

```
100110010011100101011001100110011110100101010100100110100111
```

The server receives the above string from the client.

Every 12-bit chunk 100110010011, 100101011001, 100110011110, 100101010100 and 100110100111 are checked for error and if valid, the string is reverted to the original expression 2+3*4 and solves the expression and gives the output.

Sample Example 2:

Client-Server connection established.

The terminal will look as below.

```
>Connected to Server
>2+3
>5
```

Explanation :

The client sends the expression “3+4” of the 12-bit format :

1 0011 0011 110 (‘3’)
1 0010 1011 001 (‘+’)
1 0011 0100 111 (‘4’)

The string sent by the client to the server is

100110011110100101011001100110100111

The server receives the above string from the client.

Every 12-bit chunk 100110011110, 100101011001 and 100110100111 are checked for error and if valid, the string is reverted to the original expression 3+4 and solves the expression and gives the output. Else outputs errors on the client side.