CSE210 Advanced Object Oriented Programming Coursework 2019

Release date: 11th, Mar, 2019
Deadline: 12:00PM, 23rd, Apr, 2019

1. Description

The objective of the coursework is to develop a practical application for data processing, analysis and content search based on the object-oriented principles you learn from the CSE210 module.

1. 1. Dataset

The coursework uses a subset of the dataset (file name: twitterDataset.xlsx) published at http://followthehashtag.com/. It contains 204,820 short messages (tweets) collected from Twitter (https://twitter.com), during the period of 14-16, April, 2016, from various locations in the United States. The dataset contains many different topics, e.g., weather, leisure, sports, etc.

The dataset is stored in an EXCEL file and needs to be processed from your Java program. Each record (row) contains information about a tweet. The columns are explains as follows.

- Tweet Id – the ID of a tweet
- Date – the date on which a tweet is published
- Hour – the time when a tweet is published
- User Name – name of a user
- Nickname – nickname of a user
- Tweet content – the actual message
- Favs – number of users who like the tweet
- RTs – number of users who re-tweet the tweet (republish)
- Latitude – latitude of the location where a tweet is published
- Longitude – longitude of the location where a tweet is published
- Followers – number of followers of a user (the values are the same for the same user)

1. 2. Data Processing and Analysis

To process data stored in an EXCEL file, you can use the Apache POI (the Java API for Microsoft Documents) library. The API is available at:
https://poi.apache.org/download.html#POI-3.17

You need to download it and put it into your classpath in order to use it. A tutorial on how to use the library is at:
http://poi.apache.org/spreadsheet/quick-guide.html#CellContents

You need to write code to extract data from the dataset and then perform some simple data analysis, e.g., find who the top users are. The detailed requirements are explained in Section 2.

1. 3. Content Search

Effective tweet searching is an important functionality for users, especially when there are millions of tweets published in each single day. The search process takes a user's input keyword as a query and then retrieves relevant tweets. Two simple searching strategies are considered.

1. Text matching – The idea is to find tweets based on matching the user's keyword (only one word) with all the tweets. A tweet is regarded as relevant if the keyword appears in it. This is a basic feature of the application.
2. Full text search – This is considered as an advanced feature of the application. To simplify the problem, you do NOT need to study the techniques in the field of Information Retrieval. According to the research reported in [1], the full text search functionality on Twitter is based on a modified version of the Apache Lucene API. However, the actual full text search on Twitter is extremely complicated. This coursework only attempts to build a much simpler full text search engine based on the original Lucene API, only for limited amount of Twitter data. Lucene can be downloaded at: https://lucene.apache.org/core/. A tutorial is available here: https://lucene.apache.org/core/7_7_1/demo/overview-summary.html.

1.4. Objects in the Application

It is important to follow good object-oriented programming principles and practice, e.g., object modelling, functionality decomposition, code reuse. At the same time, you need to ensure correctness and take robustness and efficiency into account.

You need to design a number of information objects to represent the objects together with the methods for this application. You need to design a number of helper classes in order to realise the functionalities. A separate class for testing purposes is also needed. Below is an example about the objects for this coursework; however, you can have your own design following good programming practice and style.

- Tweet – represents a tweet message;
- User – represents a user of the Twitter.
- LinkedList – represents a list of (sorted) objects.
- Utility and helper classes – as needed for data processing, analysis, and content search.
- Test class – for testing the implemented functionalities.

2. Requirements and Tasks

You need to complete the following tasks.

- **T1**: develop information objects and helper classes (with their methods) needed for this application with good coding style, and design an easy-to-use interface for testing (command window is enough).

- **T2**: efficiency and robustness (assessed during program testing).
- **T3**: complete the following tasks which will be checked and marked during the demonstration sessions.
  Note: FAILING TO PRESENT AT THE DEMONSTRATION SESSION WILL RESULT IN ZERO MARK FOR THE DEMONSTRATION PART (SEE MARKING SCHEME).
  - *T3-1*: *load data in the program.*
  - *T3-2*: *print top 10 tweets.* Tweets should be ranked based on the sum of: (1) number of users who like the tweet (Favs), and (2) number of users who re-tweet the tweet (RTs). Relevant information, such as when, and by whom it is published, should also be shown.
  - *T3-3*: *print top 10 users.* Users should be ranked based on the number of followers that they have.
  - *T3-4*: *tweet search based on string matching*.
  - *T3-5*: *tweet search based on full text searching using Lucene*.
- **T4**: use Javadoc comments to document your codes and generate HTML javadoc.
- **T5**: demonstrate your work in the same order as specified in the lab group file (note that the time for each demonstration varies and you probably need to wait a bit longer than expected). You might be asked to modify your code during the demonstration (to detect plagiarism). The ability to modify code is measured in percentage and marks for demonstration will be scaled. Being able to modify code according to requirements will get 100%, not able to modify anything (will report for potential plagiarism) will get 0%. Other cases will be considered accordingly. Feedback will be provided during the demonstration if possible.

3. Deliverables

You should deliver **ONE FILE** (via ICE) according to the following description.

- All source codes, compiled classes and any other supplementary files regarded necessary.
- HTML documentation. This can be generated by using either the javadoc utility or the plug-in in your IDE.
- A separate **README** file explaining how to run your code, e.g., path for the dataset, how to install external libraries, and how to test all the functionalities.
- **DO NOT** include the external APIs and the original dataset in your submission as the size will exceed the limit on ICE.
- *Submit all the source codes, compiled classes, documentation and any supplementary files regarded necessary, in **ONE ZIPPED FILE**.*

After demonstrating your application, you will be asked to submit a test sheet printed on papers.

Note: the formal procedure for submitting coursework at XJTLU is strictly followed.

4. References

[1] Busch, M., Gade, K., Larson, B., Lok, P., Luckenbill, S., & Lin, J. (2012, April). Earlybird: Real-time search at twitter. In 2012 ieee 28th international conference on data engineering (pp. 1360-1369). IEEE.