[CS209A-25Fall] Final Project (100 points)

Background

In the process of software development, many questions will arise. Developers may resort to Q&A website to post questions and seek answers.

Stack Overflow is such a Q&A website for programmers, and it belongs to the Stack Exchange Network. Stack Overflow serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki. Users of Stack Overflow can earn reputation points and "badges"; for example, a person is awarded 10 reputation points for receiving an "up" vote on a question or an answer to a question, and can receive badges for their valued contributions. Users unlock new privileges with an increase in reputation, like the ability to vote, comment, and even edit other people's posts.

In this final project, we'll use Spring Boot to develop a web application that stores, analyzes, and visualizes Stack Overflow Q&A data w.r.t. `java programming`, with the purpose of understanding the common questions, answers, and resolution activities associated with Java programming.

Data Collection (10 points)

On Stack Overflow, questions related to Java programming are typically tagged `java`. You could use this `java` tag to identify java-related questions. A question and all of its answers and comments are together referred to as a `thread`.

For **java-related threads** on Stack Overflow, we are interested in answering a list of questions as described below. You should first collect proper data from Stack Overflow to answer these questions. Please check the official Stack Overflow REST API documentation to learn the REST APIs for collecting different types of data.

- You may need to create a Stack Overflow account in order to use its full REST API service.
- API requests are subject to rate limits. **Please carefully design and execute your requests, otherwise you may reach your daily quota quickly**.
- Connections to Stack Overflow REST service maybe unstable sometimes. So, **please start the data collection ASAP!**

There are over 1 million threads tagged with `java` on Stack Overflow. You DON'T have to collect them all. Yet, you should collect data for **at least 1000 threads** in order to get meaningful insights from the data analysis.

Important:

Data collection is ***offline***, meaning that you need to collect and persist the data first. It is recommended that you use a database (e.g., PostgreSQL, MySQL, etc.) to store the data. However, it is also fine if you store the data in plain files. In other words, when users interact with your application, **the server should get the data from your local database** (or local files), instead of sending REST requests to Stack Overflow on the fly.

Hence, the data analysis for the below questions should be performed on the dataset you collected. That is, we first collect a subset of Stack Overflow data (e.g., 1000 threads tagged `java`) and then answer the following questions using this subset.

Part I: Data Analysis (60 points)

For each question from this part, you should:

- Figure out which data is needed to answer the question
- Design and implement the data analysis on the backend
- Visualize the results on the frontend using proper charts.

In other words, when interacting with your web application from the browser, users could select interested analysis, which sends requests to the server; the server performs corresponding data analysis and returns the results back to the frontend, which visualizes the results on the webpages.

Your work will be evaluated by:

- whether the data analysis is meaningful and relevant, i.e., it can indeed answer the question with proper analysis on proper data. There could be multiple ways to answer a question. Be creative!
- whether the visualizations effectively convey the idea, i.e., users can get the information they want instantly by looking at the visualization. Take a look at the data visualization catalogue for inspirations.
- whether you could draw meaningful insights from the analysis results, i.e., you should provide proper explanations and discussions on the analysis/visualization results in your presentation.

1. Topic Trends (15 points)

We have covered various topics in this course, e.g., generics, collections, I/O, lambda, multithreading, socket, reflection, spring boot, etc.

It's interesting to know, what are the Stack Overflow trends (i.e., activities) of these topics over a specified time period (e.g., the past 3 years)? You may decide the time period by your interest.

You may measure Stack Overflow activity by the post of questions, answers, or comments, or by indicators of user engagement such as upvotes, downvotes, accepted answers, or edits.

2. Co-occurrence of Topics (15 points)

Topic co-occurrences may reveal areas where developers often face cross-cutting challenges that span multiple technical domains, e.g., `spring-boot` and `security`, `concurrency` and `testing`.

What are the top N pairs of Java topics that most frequently appear together on Stack Overflow? You may choose N by your interest (of course, N should not be too small or too large).

3. Common Pitfalls in Multithreading (15 points)

Multithreading is often regarded as one of the most challenging topics in programming. Many developers struggle with concurrency bugs, race conditions, deadlocks, and performance issues that can be subtle and hard to reproduce.

Let's use Stack Overflow data to identify the top N recurring problems, errors, or pain points that developers encounter when working with Java multithreading.

For this question, you cannot only use tag information, which could be too general to be informative. You need to further analyze thread content (e.g., question text, answer text, code snippets, exception types, error

messages, etc.) to identify multithreading-related problems, probably using advanced techniques such as regular expression matching or NLP.

4. Solvable vs. Hard-to-Solve Questions (15 points)

Let's compare Java questions that receive timely, high-quality answers with those that remain hard to solve (e.g., questions without accepted answers, without high-upvote answers, or without any answer after a long time). What distinctive characteristics differentiate these two groups of questions?

You may explore factors such as question clarity, question timing, topic complexity, user reputation, or the presence of code snippets to identify patterns that contribute to a question's solvability. Of course, you could also explore other factors by your interest. **At least 3 factors** should be identified.

Part II: RESTful APIs (10 points)

The frontend should communicate with the backend through RESTful APIs whenever possible. For example, for the co-occurrence of topics analysis in Part I, the frontend could send a REST request to the backend to GET the analysis results (e.g., top N pairs of co-occurred topics and their frequencies) in `json` format, and then visualize the results on the webpage.

We may ask you to demonstrate the REST APIs during your presentation, e.g., by accessing something like `http://localhost:8080/api/cooccurrence?topN=10` in the browser, which should return the top 10 pairs of co-occurred topics in `json` format. **At least 2** REST endpoints should be available.

Part III: Visualization and Insights (20 points)

During the project presentation, you should demonstrate your web application, specifically the visualization parts. You should also explain the insights you obtained from the data analysis, for example:

- Why certain Java topics show increasing/decreasing trends on Stack Overflow over a certain time period? What are possible reasons behind these trends?
- Why certain pairs of topics frequently co-occur on Stack Overflow?
- Do you think the common pitfalls in Java multithreading you identified are indeed common and significant problems faced by Java developers? Why?
- What factors contribute to a question being solvable or hard-to-solve on Stack Overflow? Do you think these factors are reasonable? Why?

Requirements

Web Framework

You should only use `Spring Boot` as the web framework.

Backend/Data Analysis

You should implement the data analysis by yourself, using Java features such as Collections, Lambda, and Stream (SQL is also fine if you are comfortable with database operations). You should be able to explain the data analysis logic.

You **CANNOT** feed the data to AI, ask AI to do the analysis, and use AI responses as your data analysis results. For example, AI may be able to tell you the top N common pitfalls in Java multithreading, but you won't be able to explain how AI gets the results. **You will get 0 point for the question if you delegate the data analysis to AI.**

Data analysis results should be **dynamically generated** by the server everytime clients send a request. You **SHOULD NOT** precompute the results and stored it as a static content then simply display the precomputed static content on the frontend. **20 points will be deducted if you do so.**

Frontend

Frontend functionalities, such as data visualization and interactive controls, could be implemented in any programming language (e.g., JavaScript, HTML, CSS, etc.) with any 3rd-party libraries or framework.