[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.

[CS209A-25Fall] 最终项目 (100 分)

背景

在软件开发过程中，会出现许多问题。开发者可能会求助于问答网站来
发布问题并寻求答案。

Stack Overflow 是一个面向程序员的问答网站，它属于 Stack Exchange 网络。Stack Overflow 为用户提供了一个提问和回答问题的平台，通过会员身份和积极参与，用户可以对问题进行上下投票，并以类似的方式编辑问题和答案。

到一个维基。Stack Overflow 的用户可以获得声望点和“徽章”；例如，一个人在问题或回答问题上获得“赞”的投票时会获得 10 声望点，并且可以根据他们的宝贵贡献获得徽章。用户随着声望的增加解锁新的权限，比如投票、评论甚至编辑其他人的帖子。

在这个最终项目中，我们将使用 Spring Boot 开发一个网络应用程序，用于存储、分析和可视化与 Java 编程相关的 Stack Overflow 问答数据，目的是了解与 Java 编程相关的常见问题、答案和解决活动。

数据收集（10 分）

在 Stack Overflow 上，与 Java 编程相关的问题通常会被标记为 java。你可以使用这个 java
用于识别与 java 相关的问题的标签。一个问题及其所有答案和评论统称为一个线程。

对于 Stack Overflow 上与 Java 相关的线程，我们希望回答以下一系列问题。您应首先从 Stack Overflow 收集适当的
数据来回答这些问题。请查阅
官方 Stack Overflow REST API 文档，了解收集不同类型数据的 REST API。

- 你可能需要创建一个 Stack Overflow 账户才能使用其完整的 REST API 服务。
- API 请求受速率限制。请仔细设计和执行你的请求，否则你可能会很快达到每日配额。
- 与 Stack Overflow REST 服务的连接有时可能不稳定。因此，请开始数据
  尽快收集！

Stack Overflow 上超过 100 万条线程被标记为 java。您不必收集它们全部。然而，
你应该收集至少 1000 个线程的数据，才能从数据分析中获得有意义的见解。

重要提示：

数据收集是离线的，这意味着您需要先收集并持久化数据。建议
你使用数据库（例如 PostgreSQL、MySQL 等）来存储数据。但是，将数据存储在普通文件中也是可以的。换句话说，
当用户与你的应用程序交互时，服务器应该从你的本地数据库（或本地文件）获取数据，而不是实时向 Stack Overflow
发送 REST 请求。

因此，对于以下问题的数据分析应在您收集的数据集上进行。也就是说，
我们首先收集 Stack Overflow 数据的一个子集（例如，标记为 java 的 1000 个线程），然后使用这个子集回答以下问题。

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

第一部分：数据分析（60 分）

对于本部分中的每个问题，你应该：

- 确定回答问题所需的数据 设计并实现后端数据分析 在前端使用适当的图表可视化结果。
-
-

换句话说，当用户通过浏览器与你的 Web 应用程序交互时，用户可以选择感兴趣的分析，这会向服务器发送请求；服务器执行相应的数据分析并将结果返回到前端，前端在网页上可视化这些结果。

你的工作将由以下方面进行评估：

- 数据分析是否有意义且相关，也就是说，它确实能够通过在适当的数据上进行适当的分析来回答问题。回答一个问题可能有多种方法。要富有创造力！可视化是否有效地传达了想法，也就是说，用户能够获得他们想要的信息
-

  通过查看可视化图表，可以立即了解情况。查看数据可视化目录以获取灵感。
- 你是否能从分析结果中得出有意义的见解，也就是说，你应该在演示中对你对分析/可视化结果提供的适当解释和讨论。

1. 主题趋势（15 分）

我们在这门课程中涵盖了各种主题，例如泛型、集合、I/O、Lambda、多线程、套接字、反射、Spring Boot 等。

了解这些主题在指定时间段（例如过去 3 年）内的 Stack Overflow 趋势（即活动）很有趣？你可以根据你的兴趣决定时间段。

你可以通过问题、答案或评论的发布来衡量 Stack Overflow 的活跃度，或者通过指标来衡量
用户参与度，例如点赞、点踩、接受的答案或编辑。

2. 主题共现（15 分）

主题共现可能揭示开发者经常面临跨越多个技术领域的交叉挑战的区域，例如 spring-boot 和安全性、并发和测试。

在 Stack Overflow 上，哪些 Java 主题组合出现的频率最高？你可以根据你的兴趣选择 N（当然，N 不应该太小或太大）。

3. 多线程常见陷阱（15 分）

多线程通常被认为是编程中最具挑战性的主题之一。许多开发人员难以应对并发错误、竞态条件、死锁以及可能微妙且难以察觉的性能问题。

难以重现。

让我们使用 Stack Overflow 数据来识别开发者在使用 Java 多线程时遇到的前 N 个常见问题、错误或痛点。
对于这个问题，你不能仅使用标签信息，因为标签信息可能过于笼统而缺乏信息。你需要进一步分析线程内容（例如，
问题文本、答案文本、代码片段、异常类型、错误信息等）。

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.

消息等）来识别多线程相关的问题，可能使用正则表达式匹配或自然语言处理等高级技术。

4. 可解问题与难解问题（15 分）

让我们比较那些能获得及时、高质量回答的 Java 问题与那些难以解决的问题（例如，没有接受答案的问题、没有高赞答案的问题，或长时间后仍没有答案的问题）。

时间）。哪些独特的特征区分了这两个问题组？

你可以探索诸如问题清晰度、问题时间、主题复杂度、用户声誉或代码片段存在等因素，以识别有助于问题可解性的模式。当然，你也可以根据你的兴趣探索其他因素。至少应确定 3 个因素。

第二部分：RESTful API（10 分）

前端应尽可能通过 RESTful API 与后端进行通信。例如，

对于第一部分中主题共现分析，前端可以向后端发送 REST 请求以获取分析结果（例如，前 N 对共现主题及其频率），结果以 json 格式返回，然后在网页上可视化这些结果。

在演示过程中，我们可能会要求你展示 REST API，例如通过访问类似

在浏览器中输入 http://localhost:8080/api/cooccurrence?topN=10，它应该以 JSON 格式返回出现频率最高的 10 对共现主题。至少应有 2 个 REST 端点可用。

第三部分：可视化和洞察（20 分）

在项目展示中，你应该展示你的网络应用程序，特别是可视化部分。你还应该解释你从数据分析中获得的洞察，例如：

- 为什么某些 Java 主题在特定时间段内在 Stack Overflow 上显示出增加/减少的趋势
  周期？这些趋势背后可能的原因是什么？
- 为什么某些主题对在 Stack Overflow 上经常同时出现？
- 你认为你识别出的 Java 多线程的常见陷阱确实是 Java 开发者面临的常见且重要的问题吗？为什么？

- 哪些因素导致一个问题在 Stack Overflow 上是可解的或难以解决的？你认为这些因素合理吗？为什么？

需求

Web 框架

你应该只使用 Spring Boot 作为 Web 框架。

后端/数据分析

你应该自己实现数据分析，使用 Java 特性，如 Collections、Lambda 和 Stream（如果你熟悉数据库操作，使用 SQL 也可以）。你应该能够解释

数据分析逻辑。

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.

你不可以输入数据给 AI，让 AI 进行数据分析，并使用 AI 的响应作为你的数据分析结果。

例如，AI 可能能够告诉你 Java 多线程中的前 N 个常见陷阱，但你不会

能够解释 AI 如何得到结果。如果你将数据分析委托给 AI，该问题将不得分。

数据分析结果应由服务器在每次客户端发送请求时动态生成。你不应该预先计算结果并将其存储为静态内容，然后简单地显示预先计算的结果

静态内容在前端。如果你这样做，将扣除 20 分。

前端

前端功能，如数据可视化和交互控制，可以实现在任何

编程语言（例如 JavaScript、HTML、CSS 等），以及任何第三方库或框架。