Recracking the Coding Interview: Midterm Report

Pranay Agrawal
Georgia Institute of Technology
pagrawal78@gatech.edu

Rohit Das
Georgia Institute of Technology
rohdas@gatech.edu

Vinnie Khanna
Georgia Institute of Technology
vkhanna32@gatech.edu

Raj Srivastava
Georgia Institute of Technology
rsrivastava60@gatech.edu

October 28, 2021

Abstract

We hope to take advantage of patterns often found in coding problems to gleam useful information such as BCR (Best Conceivable Runtime). Patterns include certain semantical context in problem descriptions, problem difficulty levels if available, and problem tags if available. Problem tags can doubly serve as a useful feature and a predictive class of problem category given a problem description, which may be very helpful in figuring out how to initially approach a problem. Thus far, we have scraped data from two websites, LeetCode and CodeForces, to compile a list of about 8500 coding problems, of which about 1500 are labeled with runtimes - those being scraped from LeetCode. After some initial data cleaning and analysis, we have achieved an accuracy of about .64 as a baseline for classifying runtimes from problem descriptions alone using TF-IDF embeddings and a Logistic Regression classifier. We hope to add more techniques to achieve a higher accuracy and F-1 score, including using BERT for generating word embeddings and an RNN to classify our runtimes. We also hope to expand further and attempt problem approach classification, normalize difficulty level between LeetCode and Codeforces and use it as a feature, and as a stretch goal attempt to create a generative Language Model to generate new problem titles and potentially new problem descriptions.

1. Goal

Coding interviews are a highly stressful experience for many software engineers. With so many different problem types, such as dynamic programming, string manipulation, graph algorithms, etc. it can be difficult to figure out what approach to take or what resulting runtime efficiency you are aiming for. Our ultimate goal is to provide some insights over which variables are most informative in determining the approach or optimal runtime for a problem.

Specifically, we aim to:
1) Create a classifier to identify optimal time complexity for solution given problem description, problem approach or topic, and normalized difficulty level
2) Create a classifier to identify difficulty or acceptance rate of problems given problem description, optimal runtime, and topic
3) Create a classifier to identify problem approach or topic given problem description, optimal runtime, and normalized difficulty
4) Cluster types of problems to identify trends
5) **If possible** automatically generate new coding problems - titles and descriptions - given difficulty level, target time complexity, and problem category

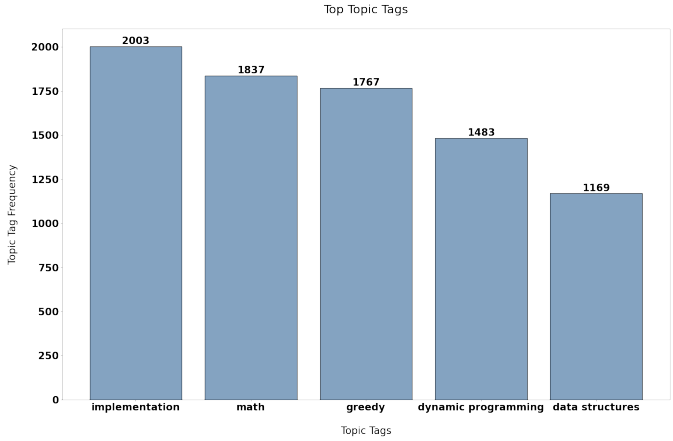
Our goals remains fairly consistent with our earlier proposal indications, as we have merely solidified our features and labels so we know which classes we can predict.

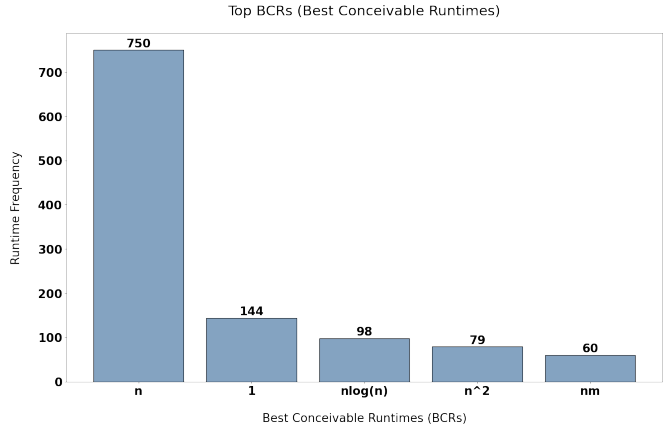
2. Progress

2.1. Data

Because no formal dataset existed of BCR-labeled coding problems, we had to construct it ourselves through webscraping using Selenium in Python for automated web-browsing. Initially, we only planned to webscrape from LeetCode, but our new member, Pranay, had the idea to augment our somewhat limited 1500 LeetCode problems with another 7000 or so from Codeforces. For both websites, we were able to scrape problem title, difficulty, description, and topic tags, while runtimes were limited to LeetCode. To scrape runtimes, we searched runtimes in order of upvotes from the LeetCode problem discussion tab.

Below is our top 5 most frequent topic tags and runtimes after data cleaning





Our data is separated into 3 main csv files, cleaned-leetcode, cleaned-codeforces, and cleaned-combined. The following is a dataframe excerpt from cleaned-leetcode and cleaned-codeforces:

```
pd.set_option('max_colwidth', 25)
print(df_lc.iloc[0], "\n")
print(df_lc.iloc[3], "\n")
```

```
id                                1
title                       Two Sum
title_slug                  two-sum
difficulty                        1
description    Given an array of int...
topics              Array, Hash Table
runtime                           n
Name: 0, dtype: object

id                                4
title          Median of Two Sorted ...
title_slug     median-of-two-sorted-...
difficulty                        3
description    Given two sorted arra...
topics              Array, Binary Search
runtime                   log(min(m,n))
Name: 3, dtype: object
```

```
pd.set_option('max_colwidth', 25)
print(df_cf.iloc[0], "\n")
print(df_cf.iloc[-2], "\n")
```

```
Unnamed: 0                        0
id                                1
title                     Watermelon
tags                brute force, math
difficulty                      800
description    One hot summer day Pe...
Name: 0, dtype: object

Unnamed: 0                     6926
id                             6928
title                Game of Chance
tags              math, probabilities
difficulty                     3500
description    The King wants to mar...
Name: 6926, dtype: object
```

As you can see above, the LeetCode examples, Two Sum and Median of Two Sorted Arrays, have labeled runtimes of n and log(min(m,n)) respectively, while the Codeforces examples Watermelon and Game of Chance have no such runtime label. Another key difference is that LeetCode problem difficulties are rated from 1 to 3 corresponding to easy, medium, and hard respectively, while Codeforces difficulties are an integer in the range of 800 to 3500.

As mentioned above, data cleaning and reorganization was necessary, particularly for cleaning up webscraped content like html tags, and of course to combine datasets originating from separate websites.

2.2. Methodology

2.3. Preliminary Results

The first attempt of classifying optimal time complexity given problem description used TF-IDF and logistic regression on the Leetcode problem data. First, the sklearn TF-IDF vectorizer was used on the problem descriptions to create a 2D document-term matrix. Then, this was used as the input features for the logistic regression model. The metrics used were accuracy and weighted F_1 score. A weighted F_1 score is essentially a macro F_1 score where the F_1 score of each label is multiplied by the number of true instances of that label, and then these weighted F_1 scores are averaged out. In other words, it is a weighted average of the F_1 score of each label where the weights are determined by how frequently the label shows up in the data. We decided to use this over macro F_1 because the the Leetcode problems dataset is very imbalanced.

The main hyperparameters tuned were the different combinations of n-grams for the TF-IDF vectorizer and the regularization strength. While many combinations of n-grams were tested (including unigrams, bigrams, trigrams, unigrams and bigrams, unigrams and bigrams and trigrams, etc.), a combination of unigrams, bigrams, and trigrams performed best. Increasing regularization strength did little to improve results, but decreasing regularization strength significantly worsened the results. Lastly, different solvers (optimization algorithms) for the logistic regression model had little to no effect on performance.

In exploring the data, we noticed that there were a total of 157 different runtime labels among 1406 data points with non-null runtimes, where 1131 data points had one of the top 5 most common labels. Since so many labels very few data points, we limited our model to the top 10 most common runtimes, leaving us with 1222 total data points.

The results of the model were a 67.9% accuracy and weighted F_1 score of 0.571. These results are not too satisfactory overall because the most common label 'n' shows up 750 times, so it makes up $750/1222 = 61.4\%$ of the dataset, meaning it did not perform very well overall. Therefore, the results of the model differed from what we expected. We were looking for an accuracy that was significantly higher than the proportion that the most common label has in the dataset (we expected something an accuracy around 75% to 85%). The main reasons for the poor performance are likely the small dataset as well as the imbalance in the dataset.

Category	LeetCode	Codeforce	Total
Easy	392	1520	1912
Medium	826	1387	2213
Hard	350	4021	4371
Total	1568	6928	**8496**

Table 1. Data Makeup

2.4. Work Division

Raj worked on some of the scraping part of the Leetcode dataset, some of the data exploration, and created the logistic regression model that used TF-IDF.
Pranay worked on scraping the run-time part of the LeetCode dataset. Furthermore, he worked on scraping the entire Codeforces dataset. Lastly, he helped with testing some

3. Plan to Complete Project

3.1. Future Tasks

In the short-term, we hope to improve the TF-IDF and logistic regression model if possible. Furthermore, we hope to use a Python module like ELI5 in order to view the weights associated with each term for each label. This can allow us to view which words are most associated with certain runtimes, providing useful information for people to use when determining the best conceivable runtime of a software engineering interview problem.
In the long-term, we have many tasks that were listed in our goals that fall under one of three categories: classification, clustering, and building a language model. For classification, we will try ensemble methods such as random forest and SVM from sklearn alongside TF-IDF. Models from sklearn are valuable because the ELI5 module can be used on them to see which words are more positively or negatively weighted for certain labels in order to see which words best predict runtime. Later, we will use more sophisticated techniques such as LSTM, BERT, and Ro

3.2. Project Changes

Since our group has had the addition of a new person since the original proposal, our group is taking on additional tasks. One major addition is including a new dataset of problems from Codeforces. In doing so, we can compare our results between LeetCode and Codeforces. Furthermore, adding a new dataset allows us to compare problems between LeetCode and Codeforces. Lastly, we are more keen on attempting the ambitious task of problem generation (titles and descriptions). This would require a good

4. Code Repository

Our team code can be viewed at the following link: https://github.com/VinnieKhanna/RCTCI.