Competitive Coding Analysis

Vishal K.S V.R Chittaranjan Supreeth V.K
Varun Sapre
Dept of Computer Science, P.E.S University

November 20, 2016

1 Abstract

This project looks to bring out certain facts about the Competitive Coding domain that has picked up in the recent years. We look to determine if excellence in the field is based purely on practice, or if it depends on some amount of inherent skill of a person. In addition, we perform some basic exploratory analysis on the data to strengthen the confidence of some basic assumptions.

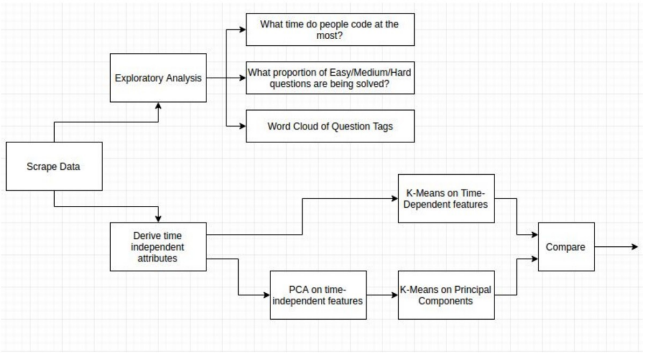
2 Introduction

In todays world, competitive coding has almost become a necessary skill. Many recruitment drives begin with a filtering session which is based on a coding round. It is necessary to evaluate the effectiveness of this process, as there is a possibility of good talent being lost due to the bias of the selection criteria. Furthermore, it is still unclear whether the ability to solve pure number theoretic or academic problems implies proficiency in a corporate scenario. In this project, we attempt to address these issues, by differentiating between features which are acquired over time and features which are inherent to a coder.

3 Problem Statement

There hasnt been any relevant work in this domain, so there was no data available for us to directly work on. We built a scraper to gather data from www.codeforces.com. This website consisted of almost 25,000 users, each having a profile which showcased the problems they have attempted along with their code for the submission. Due to lack of computing power and resources, we were unable to scrape all of the data, and had to resort to stratified samples across the sites leaderboard. We picked a random 50 users from each section of ranks of range 6000. In total, there were four sections, making it a total of 200 users. This data had to be cleaned due to the fact that some users had attempted far too few problems to be of any significance. From this data set, we look to extract two sets of features for each user, one which depicts the amount of effort, and another which represents intellectual ability. Using these features, we estimate which set is in better alignment with the Codeforces leaderboard.

4 Block Diagram



5 Exploratory Analysis

- Popular Coding time : On plotting the number of submissions that occur against the time of day, it was found that there was a significant peak towards 8:00 P.M.

- Proportion of problems : Upon computing question complexities based on the number of people that had solved them, pie-charts revealed that top ranked coders were solving equal amount of Easy, Medium, and Hard problems (approx. 33%) whereas more than 50% of the problems solved by the bottom ranked people were Easy.

- Question Tags Word Cloud : At the center of the cloud was the word Math surrounded by Data Structures and Brute Force. These seem to be the prominent skills required to tackle most problems.

6 Deriving Features from the Dataset

Once the data of the 200 odd users had been scraped, we extracted certain attributes from it. A few notable ones were :

- Code Complexity : Using the Python module lizard the cyclomatic complexity of each submission a user had made was computed. For lack of a better way, we took an average of all these complexities and assigned it as the average complexity of a user.

- Question Complexity : Similar to Code Complexity, this was also an average of the complexities of questions a user had solved over time.

- Accuracy : This was a measure of how many tries a user takes to solve a problem.

- Number of Languages : A count of the various languages a user had attempted to solve problems.

- Certain other attributes like average number of compilation/runtime/logic errors were also used. Each attribute was normalized to be a value between 0 and 1.

7 PCA

On deriving these attributes which supposedly represent a coders ability, we applied Principal Component Analysis on the 7 attributes, in order to reduce the data to 3 dimensions (for the sake of visualization)

8 Experiments

The unsupervised K-Means clustering algorithm (with k=3) was applied on each of the two datasets. One which had attributes that would only improve with time(number of submissions, number of problems etc), and another with the principal components of skill attributes mentioned previously. The sklearn module of python was used for K-Means, and the cluster-plot was made using matplotlib.

9 Results and Conclusions

The clusters in both cases were not very well defined, but were slightly better in the dataset which had the time-dependent attributes of a user. This goes to show that users who slog it out over the years, tend to get better than the others who arent as active. But at the same time, in the cluster-plot of the derived attributes, there were no distinct clusters at all implying that a user on top of the leaderboard may have very well been in the same league as the bottom ranked users in terms of ability. These experiments do not fully shed light on whether coding is a natural skill, or a developed one. The safest conclusion to make, would be to state that coding ability of a human is partly developed and partly acquired. At the same time, our approach towards the question at hand may have been flawed due to multiple reasons.

- Curse of averages : Since the averages of code and question complexities were taken, it might have resulted in a skew in the data leading to poor results at the clustering phase.

- K-Means clustering : In general, the clustering approach may not have been appropriate. Perhaps, a logistic regression model, to predict whether a user was good intellectually or not, may have yielded more interesting results. The drawback

was that, we did not have access to any reliable testing data, as we learnt that the leaderboards on the site do not fully reflect skill of a user.

- Size of sample : From a total of 25,000+ users, we scraped less than 1% of them. There is a possibility that the sample size may not have been significant enough to produce results.

10 References

1. www.lizard.ws : An extensible Cyclomatic Complexity Analyzer for many programming languages including C/C++ (doesn't require all the header files). We used the associated python module to analyze code complexities.

2. www.codeforces.com : One of the popular competitive coding websites, from where we scraped questions, and solutions by many users.