Classification of Rating & Tags for Codeforces Competitive Programming Problems

Nilay Tripathi Nicholas Belov Sriraam Ramakrishnan

October 20, 2024

1 Background & Introduction

Codeforces[1] is a website centered around competitive programming, where problems are designed to test ingenuity in mathematics and coding knowledge. The problems are parameterized puzzles and competitors must write programs that solve the puzzles for all possible valid inputs, subject to time and memory constraints.

Codeforces hosts competitive programming contests roughly once a week which have on average 30000 participants worldwide. After each contest, problems are released free for anyone to attempt on the Codeforces platform. When released each problem is assigned a numeric rating which correlates with the problem's general level of difficulty. Additionally, each problem is assigned specific tags, which give the mathematical and/or programming topics that are deemed most relevant to solving the problem.

For our project, we would like to construct a model that predicts either the numeric difficulty rating or the tags (or both, if time and resources permit).

2 Example Problem & Tags

Consider the problem G. Perfect Matching[2] from a recent Codeforces contest. This is a particularly challenging problem with a rating of 3400. The problem's tags include *bitmasks*, *bruteforce*, *hashing*, and *meet-in-the-middle*, which suggest that the solution requires a combination of each of these techniques.

A decent competitive programmer would be able to immediately infer that this problem is brute force given it's extremely tight bound of $n \leq 20$. An even better competitive programmer should be able to tell that meet-in-the-middle and bitmasks might be useful which is commonly true in problems that use MEX.

What is much harder to work out is whether or not the problem is hard. Normally in contest, a competitor can check the total number of people that solved a problem to get an estimate of the problem's difficulty. We will not be providing such information to the model to see if it can infer the difficulty organically.

3 Obtaining Data

We think this project neatly fits into the "novel dataset" category. Obtaining data will likely be achieved through scraping the Codeforces website with a Python script to retrieve the problem text, tags, and ratings for a subset of the problems on the website.

4 Methods

The most appropriate model for this task seems to be a recurrent neural network (RNN), but we may also explore more complicated models such as transformers to extract features and patterns from the source text.

[1] https://codeforces.com/
[2] https://codeforces.com/contest/2002/problem/G