

Project Title:

**PROBLEM CLASSIFICATION AND
RECOMMENDATION SYSTEM
FOR
COMPETITIVE PROGRAMMING**

Submitted by:

ABHISHEK VERMA (102097017)

SIDAK BHATIA (101916032)

AYUSH JINDAL (101916007)

PRINCE (101916056)

BE Third Year- CSE

CPG No. 1466

Under the Mentorship of

DR. SHREELEKHA PANDEY

Assistant Professor



Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

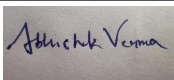
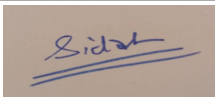

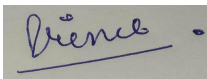
MARCH & 2022

TABLE OF CONTENTS

• Mentor Consent Form	3
• Project Overview	4
• Problem Statement	5
• Need Analysis	5
• Literature Survey	6
• Objectives	8
• Methodology	9
• Project Outcomes & Individual Roles	10
• Work Plan	10
• Course Subjects	10
• References	11

Mentor Consent Form

I hereby agree to be the mentor of the following Capstone Project Team

Project Title: Problem classification and Recommendation System For Competitive Programming		
Roll No	Name	Signatures
102097017	Abhishek Verma	
101916032	Sidak Bhatia	
101916007	Ayush Jindal	
101916056	Prince	

NAME of Mentor: Dr. Shreelekha Pandey

SIGNATURE of Mentor:

NAME of Co-Mentor(if any):

SIGNATURE of Co-Mentor:

1. Project Overview

Competitive Programming traces its roots to a competition held by ACM-ICPC at Texas A&M University in 1970. Competitive programming is a mind sport usually held over the Internet or a local network, involving participants trying to program according to provided specifications. Contestants are referred to as sports programmers. Competitive programming is recognized and supported by several multinational software and Internet companies, such as Google, Facebook, and IBM. There are several organizations that host programming competitions on a regular basis. A programming competition generally involves the host presenting a set of deterministic problems to the contestants (who can vary in number from tens to several thousand), and contestants are required to write computer programs capable of solving each problem. Judging is based mostly upon the number of problems solved and time spent for writing successful solutions, but may also include other factors (quality of output produced, execution time, program size, etc.). There are many competitive coding platforms namely CodeChef, Spoj, CodeForces, Hackerrank, and HackerEarth. The objective of these platforms is to provide a platform for practice, competition, and improvement for both students and professional software developers. Various types of programming contests are held on competitive programming platforms. Solving basic competitive programming problems requires two things.

1. Intermediate hold on any one programming language
2. Knowledge of basic algorithms in various domains like sorting, searching, number theory, greedy techniques, hashing, and dynamic programming. More importantly, you have to figure out what, when, and where to apply them.

It is thus very tricky for a beginner to get started and improve while doing competitive programming. The biggest difficulty a beginner faces is which problem to pick and which platform to solve problems on. A trap in which many beginners find themselves is that they pick problems that are too easy or too hard for them to solve. Such practice does not yield concrete results in programming contests and as a result, many beginners become demotivated and leave programming altogether. Another problem faced by most programmers is what type of algorithms and data structures are to be used in this particular problem. Our project aims to solve these dilemmas.

2. Problem Statement

Given a set of competitive programming problems with their description, including problem title, description, constraints, submission size, its explanation on not - classify the problems into predefined problem categories. Also given a set of competitive programmers and their profile information with submission history, including no. of problems solved, class of each problem, date of submission, and the difficulty of problems. We have to recommend a set of relevant problems which will improve competitive programmers' learning curve.

3. Need Analysis

An expertise in competitive programming is an asset to a programmer, a Computer Science fundamentalist, or a mathematician. It takes time and forbearance to enhance one's prowess in this discipline. Most of the time, novice users get perplexed and capitulated before acclimatizing themselves with different aspects of competitive coding. This often occurs due to the absence of proper recommendation systems to streamline the user's understanding. First, let us understand the Competitive Programmer's rationale behind practicing at Competitive Programming platforms:

1. To enjoy and have fun through the solving process
2. To get hired in product-based companies
3. To improve problem-solving skills

Some Common problems faced by Competitive Programmers are:

1. Identify which problems to solve: Selecting which problem to solve could be too tough given the vast amounts of problems that are available on sites like Codechef and Codeforces. For example, codeforces which was founded back in 2009 has more than 7000 problems in its repository. A new user might become intimidated by such a vast array of problems.
2. Identify the class of problem: Another Problem faced by the user is to select which class or tag of problem to solve. Sometimes a user might only solve a particular class of problems since those are the ones he finds interesting or easy. For example, a user may keep practicing problems on Dynamic programming or Graphs ignoring problems on topics he finds harder or less interesting like constructive algorithms. A good problem recommendation system must recommend a uniform array of problem types.

3. Identify the algorithm needed for a problem: This is among the most critical skills needed by a competitive programmer. Sometimes users may be able to get intuition or hint of which algorithm will be used for this problem depending on the keywords or the constraints given in the problem statement.

4. Identify Problem's difficulty level: Classifying the problem based on difficulty is not an easy task since the difficulty is a subjective trait that depends on the user's skill level. However, it is an important attribute nonetheless.

The Problem recommendation system for competitive programming perfectly solves the above common problems. There is no platform currently available on the internet which solves all of the above problems.

4. Literature Survey

This section includes a survey of current research in text classification and recommendations. Problem classification based on description is the first part of this thesis, and recommending problems that will improve users' problem-solving skills is the second part.

4.1 Classification

Identifying tags or keywords from text has been a significant application of classification on text data. Research has been done on this topic to use the technique in various application areas, like tagging questions and answers on Stack Overflow, Quora, etc. In the case of Question And Answer, sites such as Stack Overflow or Quora tagging allow users to explore more related content, build and showcase expertise in a given area and in general get more visibility to the question at hand. In this thesis, we try to apply this technique to competitive programming platforms such as Codechef, Spoj, and Codeforces.

There are various ways in which we can apply machine learning in the competitive programming domain. Currently, problems are tagged manually, and there are multiple issues with this. Tags are high level. They tell if a problem has math involved or binary search involved but does not give us any information about what in math or how complex it is. There are also a lot of problems without tags. In this proposal, one of our objectives is to address this particular issue. There are several approaches to text categorization, such as decision rule-based, knowledge base-based, text similarity-based, and so on. In this thesis, we focus on text categorization using text similarity, in that too

many emphases are given to Keyword extraction. Keyword extraction is a topic that has been generating increasing interest both in industry and academia. Two approaches were identified for tag prediction for the questions; one was to build a global multiclass classifier and then use methods such as One vs. All to select the final class. The second approach was to build a discriminant classifier for each of the tags and then predict the final tags choosing the most likely tags. Since we need to predict more than one tag for each of the questions, it was decided to use the second approach. To identify features from the problem description, mainly two approaches were identified. First, using Frequent Pattern Mining and second, using Term Frequency Inverse Document Frequency (TF-IDF). Term frequency is the simplest measure to weigh each term in a text. In this method, each term is assumed to have importance proportional to the number of times it occurs in a text.

While term frequency concerns term occurrence within a text, inverse document frequency (IDF) concerns term occurrence across a collection of texts. The intuitive meaning of IDF is that terms which rarely occur over a collection of texts are valuable. The importance of each term is assumed to be inversely proportional to the number of texts that contain the term. In the Frequent Pattern Mining approach, we find the occurrence frequency of each word, then based on the percentage occurrence of words, we decide which words to take as features.

4.2 Recommendation

Recommendation algorithm actually is a method to connect users and items, which is at first calculate the user's property model, and then calculate the item's property model, and then the item is recommended to the user by different methods. Collaborative filtering recommendation and content-based recommendation are the two most widely used recommendation algorithms for research and application. The idea of a content-based recommendation algorithm is based on the item that the user has chosen, and then choose the item with similar characteristics from the candidate recommendation item as the recommendation results.

However, collaborative filtering gets the items on the explicit or implicit information via the collection of user's past behavior, according to the user's preference for items found the correlation of items by itself, and then based on these correlations to recommend, collaborative filtering focuses on the user's preferences or the ratings of items. This thesis requirement was that recommendations should be such problems, which will improve the problem-solving skills of a competitive programmer. For that reason, an alternate recommendation strategy using Word2Vec is used. First, recommendations are generated from Word2Vec, and then problems are filtered so that

only those problems are recommended to the user, which will improve his problem-solving skills. According to thomaset al, word2vec has distributed representations of words in a vector space that help to learn algorithms to achieve better performance in natural language processing tasks by grouping similar words. Recently, Mikolov introduced the Skip-gram model, an efficient method for learning high-quality vector representations of words from large amounts of unstructured text data.

Unlike most of the previously used neural network architectures for learning word vectors, training of the Skip-gram model does not involve dense matrix multiplications. This makes the training extremely efficient. In this thesis, word2vec with the skip-gram model is used for recommendation. Here the successful submissions of users are considered as the input sentences, and the model is trained on these sentences. Word vectors are generated for each of the problems in the sentence. Later when a new user arrives and asks for problems to solve, at that time, problem recommendation is made using this trained model and previous submissions of that user.

5. Objectives

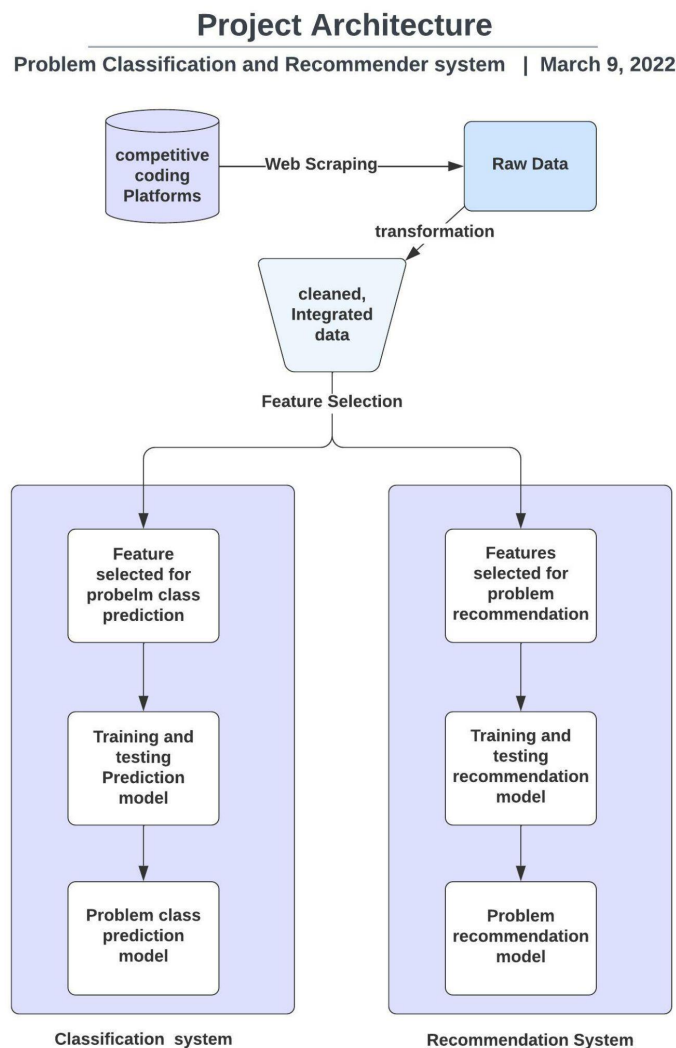
- To develop a platform for Recommending coding problems to users, based on their current rating as well as by analyzing the topic tags and difficulties of the problems they have already solved.
- Enable the user to improve his problem-solving abilities as well as help him/her to increase his contest rating on platforms like Codechef and Codeforces.
- Develop a system that is able to predict the class/topic tags of a programming problem through various Machine Learning algorithms.
- Identify the algorithm needed to solve a particular problem as well as problem difficulty.
- Develop a simple, fast, and user-friendly web application that provides all these features.

6. Methodology

Every project works well when it is planned and executed well. In this chapter, we chart out high-level design (architecture) for the Competitive Problem Classification and Recommendation System. We plan the tasks that need to be carried out to develop the architecture and design a timeline for it.

6.1 Project Architecture

The architecture of the project is a set of components that are needed to establish the Problem Class Prediction Model and Problem Recommendation Model from the data at Disparate Coding Platforms as shown in the figure below.



7. Work Plan

Project Timeline
Problem Classification and Recommender System | March 13 , 2022

Sr. No	Activity	MONTH	Jan	Feb	Mar	April	May	Aug	Sept	Oct	Nov	Dec																														
		WEEK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1.	Project identification	Plan																																								
		Actual																																								
2.	Analysis and study of existing technologies	Plan																																								
		Actual																																								
3.	ETL initiative, planning and completion of ETL activities	Plan																																								
		Actual																																								
4.	Define data extraciton tools and Identifying features	Plan																																								
		Actual																																								
5.	Identifying various application models and algorithms	Plan																																								
		Actual																																								
6.	Training models and comparing accuracies	Plan																																								
		Actual																																								
7.	Design of interfaces for the end user application	Plan																																								
		Actual																																								
8.	Implementation of interfaces and unit testing	Plan																																								
		Actual																																								
9.	Integration testing	Plan																																								
		Actual																																								
10.	deployment and user feedback	Plan																																								
		Actual																																								
11.	Result Evaluation	Plan																																								
		Actual																																								
12.	Final Report	Plan																																								
		Actual																																								

8. Project Outcomes & Individual Roles

Outcome: Extension for coding websites that can classify and predict problems.

Individual Roles:

- **Sidak Bhatia:** Analysis and study of existing technologies; planning and completion of ETL activities; Define data extraction tools and identifying features; Identifying various application models and algorithms
- **Abhishek Verma:** Identifying various application models and algorithms; Training models and comparing accuracies; Design of interfaces for the end user application
- **Ayush Jindal:** Implementation of interfaces and unit testing; integration testing; deployment and user feedback
- **Prince:** User Feedback; Result Evaluation; Documentation

9. Course Subjects

- Machine Learning
- Artificial Intelligence
- Data Structures and Algorithms
- Web Development
- Software Engineering

10. References

[1]<https://analance.ducenit.com/kb/identify-keywords-and-tags-from-millions-of-text-questions/>

[2] <https://github.com/bubuntux/competitive-programming>

[3] <https://raiderhacks.com/competitiveprogramming>

[4]<https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94dae32a87b>

[5]<https://www.c-sharpcorner.com/article/what-is-competitive-programming-and-why-it-is-importantcontests/>

[6] <https://www.hackerearth.com/getstarted-competitive-programming/>

[7] <https://www.sciencedirect.com/topics/computer-science/inverse-document-frequency>

[8] RECOMMENDATION ENGINE FOR COMPETITIVE CODING QUESTIONS USING
RESTRICTED BOLTZMANN MACHINES, A HYBRID APPROACH
(<http://ijarcs.info/index.php/ijarcs/article/download/5321/4512>)