MACHINE LEARNING BASED INTELLIGENT RECOMMENDATION SYSTEM DEVELOPMENT FOR STEPPING UP PROGRAMMING SKILL

BY

NESAR AHAMMED
ID: 163-15-8332
AND

FOYSAL AHAMMED
ID: 163-15-8331

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering

Supervised By

Md. Tarek Habib
Assistant Professor
Department of CSE
Daffodil International University

Co-Supervised By

Md. Riazur Rahman
Senior Lecturer
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

JULY 2020

APPROVAL

This Thesis titled **“Machine Learning Based Intelligent Recommendation System Development for Stepping up Programming Skill”**, submitted by Nesar Ahammed, ID No: 163-15-8332, Foysal Ahammed, ID No: 163-15-8331 to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 08 July, 2020.

BOARD OF EXAMINERS

[illegible]

Dr. Syed Akhter Hossain **Chairman**
Professor and Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

[illegible]

Dr. Md. Ismail Jabiullah **Internal Examiner**
Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Azmoon

Nazmun Nessa Moon **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

[illegible]

Dr. Mohammad Shorif Uddin **External Examiner**
Professor
Department of Computer Science and Engineering
Jahangirnagar University

DECLARATION

We hereby declare that, this project has been done by us under the supervision of **Md. Tarek Habib, Assistant Professor, Department of CSE and Md. Riazur Rahman, Senior Lecturer, Department of CSE** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:

[illegible]

Md. Tarek Habib
Assistant Professor
Department of CSE
Daffodil International University

Co-Supervised by:

[illegible]

Md. Riazur Rahman
Senior Lecturer
Department of CSE
Daffodil International University

Submitted by:

Nesar Ahammed

Nesar Ahammed
ID: 163-15-8332
Department of CSE
Daffodil International University

Foysal Ahammed

Foysal Ahammed
ID: 163-15-8331
Department of CSE
Daffodil International University

ACKNOWLEDGEMENT

First, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the final year project/internship successfully.

We really grateful and wish our profound our indebtedness to **Md Tarek Habib, Assistant Professor,** Department of CSE Daffodil International University and **Md. Riazur Rahman, Senior Lecturer, Department of CSE** Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisors in the field of "*Machine learning*" to carry out this project. Our supervisor's endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to **Dr. Syed Akhter Hossain**, **Professor and Head,** Department of CSE, for his kind help to finish our project and also to other faculty member and the staff of CSE department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

ABSTRACT

Programming online judges (POJs) are an emerging application scenario in e-learning recommendation areas. Specifically, they are e-learning tools usually used in programming practices. Usually, they contain a large collection of such problems, to be solved by students at their own personalized pace. These online judges play an outstanding role for any student who wants to practice programming. Basically, these online judge platforms provide the various type of programming problems made by many problem setters. Sometimes these problems are categorized by problem tag. The more problems in the online judges the harder the selection of the right problem to solve according to previous users performance, causing information overload and a widespread discouragement.

Many times, students try problems that are difficult than their skill level. Also, it is very troublesome for them to find-out suitable problems that match with his/her skill levels. These issues have not been addressed in any previous research works so we end up with a solution in that we use machine learning and recommendation algorithms.

This research works presents a non-personalized and content based collaborative filtering technique to mitigate this issue by suggesting programming problem.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Introduction

Competitive programming or contest programming are one kind-off mind sports arranged over the internet. During the programming contest, contestants were given a specified specification usually called problem statements. Contestants write programs according to the specification given in the problem statement. Programming Contests usually held on an online contest platform hosted by some educational institution supported by various tech companies worldwide. Over the last decades, programming contests popularity has increased to a great extent.

1.2 Motivation

The contest programming's acts as a backbone to create world class programmers. It involves a lot of programmers to willingly participate and learn a lot of things related to problem solving. There are currently thousands of programming problems available on numerous online judges over the internet. These problems are categorized by different kind-of topics such as Implementation, Mathematics, Number Theory, Data Structures, Dynamic Programming, Graph, Greedy and so on. Many online judges provided the difficulty levels of the problems. For example, Codeforces problems from the same topic can be of varying difficulty levels. For a programming contestant to be able to reach a good position at coding where he/she has enough critical thinking and skills to solve challenging problems takes years of practice and requires learning new algorithms continuously. Therefore, solving programming problems from various topics and slowly progressing to more difficult and challenging problems is a difficult task to do for a competitive programmer as programming problems in various automated online judges are not well organized at times. So, in order to address this problem of searching the right problems to solve next for a competitive, we have proposed a recommendation system for programming problems found in various online judges

1.3 Rationale of the Study

Newcomers face many difficulties when they start to learn programming. At the beginning they seem to be exited to do programming, but after a while they become disappointed. One of the reasons for this disappointment is not getting the right direction. They often try to solve a problem that is more difficult than their skill level. They get frustrated as most of the times they aren't able solve the problem even after trying numerous times

Sometimes it is seen that they are solving the same difficultly level problem more than they actually need, as a result of which, after spending a lot of time, they are not able to perform as expected.

This issue has not been addressed in any previous related works so we have come up with a non-classical approach to recommending programming problems to competitive programmers.

1.4 Research Questions

Every system is made up of many small simple parts. As such research problems usually becomes understandable once each individual section of the problems defined properly.

1.4.1 Recommender System

Recommender systems are the systems that help users to discover items that they might be interested in. Recommender systems are the key part of the modern ecommerce and consumer base online industries. Over the last decades, with the rise of Youtube, Amazon, Netflix, Google search and many other web services , recommender systems have been a regular things in our daily lives.

1.4.1.1 Non-Personalized Approach of Recommender System

Non personalized approach of recommender systems are the simplest type of recommender system. Non personalized recommender system gives recommendation to the users without thinking what user preferences is.

1.4.1.2 Personalized Approach of Recommender System

In personalized approach of recommendation, there are two types. content based recommendation system and collaborative filtering method.

In Collaborative filtering method, recommendation are based soley on past interactions recorded between users and items in order to give new recommendation. These interactions are stored in the so-called "user item-interaction matrix"

Unlike Collaborating filtering, that is depend on user and item interaction, Content based approaches uses additional information about user and item. The main approach of content-based recommendation system is building a model based on available "features" that observed user item interactions. A content-based recommendation system tries to recommend items (products or any other object) based on users profile. The user profile revolves around that user's preference and tastes.

1.5 Expected Outcome

In this research, we tried to build a system that will provide programming problems on the basis of user previous solve history. The outcome for this research project is to find out a solution for the problems we mentioned earlier by using recommendation approach.

1.6 Report layout

Chapter 1: Introduction

In this chapter we have discussed the introduction, motivation of the work, Rationale of the study and expected outcome of the research and the report layout.

Chapter 2: Literature Review

In this chapter, we have discussed the background of our research. We also provide the information of some related work, background, research summary, and scope of the problem and the challenges of this research.

Chapter 3: Research Methodology

In this chapter, we have discussed our working procedure. What's are in our proposed solution, how our proposed solution works. Our research subject and what was we used in our research. We also discussed sample data.

Chapter 4: Experimental Results and Discussion

In this chapter, we have discussed our experimental results and discussion about our results.

Chapter 5: Summary, Conclusion, and Implication for Future Research

In this chapter, we have discussed Summary of our whole research and some recommendations. We also include what needs for future research.

CHAPTER 2

Literature Review

2.1 Introduction

Our proposed solution is combination of cutting-edge modern technology’s that is easy to implement and possible to easily maintain in future.
There is not much research has been done using recommender systems before to make it easier to learn programming.
Most of the research of the recommendation system is industry centered, such as how an approach can be used in e-commerce, or how users can suggest good movies, music. However, in our research, we have taken ideas from some previous work.
To do that, we study literature as much as possible. We believe that there is no scope to develop new solution for this problem without study.

2.2 Related works

In previous, there are lots of works has been done with recommender systems but very few are related with competitive programming.

“NewsWeeder: Learning to filter Newnews-1995" [5]
NewsWeeder is a net news filtering systems, which let the users to rate his/her interest level for each article being read between 1 to 5, and then it learns the user profile based on these ratings. In this reasearch, NewsWeeder uses the words of their text as features. Then a content based recommendation systems learn the profile of a user’s interest based on the feature present on the basis of user has rated.

“Tapestry “ [6]
Tapestry is one of the old implementations of collaborative filter based recommendation systems. Tapestry was a manual collobaroative filtering systems. In Tapestry users help each other by recording through their reactions about a document which is referred to as annotation. In this systems you had to choose your own expert whose annotations you wanted to throw into the mix. Other users of the system can use this annotation or documents to filter their releivant documents.

“Bayesian networks” [8]**.**

Bayesian networks are a type of probabilities and graphical model consist of nodes and corresponding edges. The nodes hold decision trees and the edges represents user information’s. For example, a node can represent a variable such as someone’s weight, age. A node variable can be district such Gender or might be a continuous such as someone’s age. The model is created on the training dataset. This sort of model is not very large. This kind-off model are very fast, and as accurate as nearest neighbor method. Bayesian networks works better when then knowledge of user preferences updated slowly. They are not suitable for situations where users preferences updated very quickly.

“Horting Hatches an Egg: A New Graph-Theoretic approach to Collaborating filtering” [10]

In this research, a new and novel approach of rating based collaborative filtering is introduced. This technique is most suitable for e-commerce platforms for recommending homogeneous items such as cooker, washing machines, mixer. In this approach, nodes are representing the users and the edges between nodes represents the degree of the similarity between two users. The predications are made by traversing the graph through adjacent nodes at each step and then combining the results for the adjacent users.

“Recommender Systems in Ecommerce” [11].

In this research, a depth classification and example are founds for various recommender systems that are being used in E-commerce systems and how these recommendation systems give one-to-one personalized recommendations . Though, this kind-off recommendation systems are used successfully for quite some time now.

“A Recommender System for Programming Online Judges Using Fuzzy Information Modeling” [17]

This paper presents a recommendation framework to recommend problems to solve in online judges, through the use of fuzzy tools .

2.4 Scope of the problem

The necessity of a system that helps students by guiding and providing personalized programming problems is very much required.
Programming problems solving acts as a backbone to create world-class programmers. It involves a lot of programmers to willingly participate and learn a lot of knowledge regarding data structure and algorithms.
For being a world class programmer, one needs to have lots of patience and must go through a lot of hard works, frustrations.
For Bangladesh perspective, competitive programming is not as popular as China or Russia. That’s why only few people are doing competitive programming. As, less people does it, the chances of getting help are less. One may say that, there have online communities for helping programmers. But usually those communities have a environment that it only helps when a programmer become stuck on a problem. Online communities are not so helpful in order to get guidelines all the times.
With our systems, it is possible for students get suggestions and guideline all the time.

2.5 Challenges

The problem seemed a little difficult to us at the beginning because Recommendation system is complex by itself.
And all programming contest sites not provide Application Programming Interfaces (API) to access challenges. And Not all sites have programming problems tagged by categories.
We are mentioning few more challenges we faced during research.
1) Codeforces API that we used primarily on our implementation is relatively slow, we cannot made more than 4-5 request per minute to their server.
2) Many times, Codeforces server goes down and become inaccessible.
3) After fetching the data from the server, filtering it and storing valid data to the database is a little troublesome.

4) The default category of the problems always is not valid.

CHAPTER 3

Research Methodology

3.1 Introduction

The bellow diagram demonstrates the process of data collected for visualization. While the process is complete with 6 different segments. Where in initial stage, user (students) data will be collected from the input field (internal users) and a request is sent to our system. then our system will store the user Codeforces username to our database. The Codeforces username is required to retrieve the submission history of the user. Then a submission history request is sent to Codeforces system to retrieve a user submission history. In this research, the term submission history refers that, what problem user has been solved yet. The request has been done using Rest API provided by Codeforces. Then Codeforces will return the submission history for that user to our system. Then our system will analyze the data and generate recommendation by using content-based recommendation system approach.

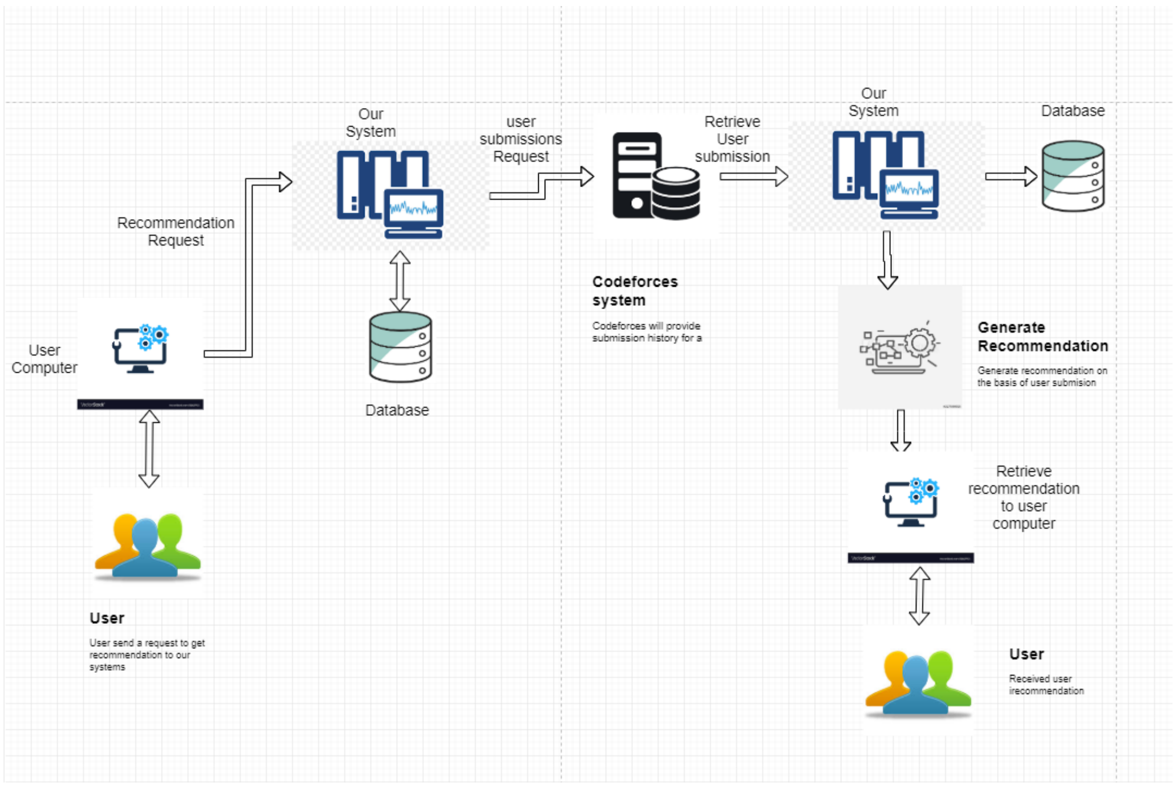

Recommendation Request
User Computer
User
User send a request to get recommendation to our systems
Our System
Database
user submissions Request
Codeforces system
Codeforces will provide submission history for a
Retrieve User submission
Our System
Database
Generate Recommendation
Generate recommendation on the basis of user submision
Retrieve recommendation to user computer
User
Received user irecommendation

Fig 3.1: Working Procedure for the Proposed Model

Then user will view all the recommendation our system made in his/her computer.

3.2 Objective

There have many recommendation systems approaches that are being used in many real world scenarios. But all the approaches are not be helpful in recommending programming problems.. Now we have come to several criteria that our systems should fulfill. These are :

1. Recommend Programming problems that has not been solved yet by the user
2. Recommend Programming problems that are most likely to be solvable by the user, I.e , recommended problems must be in a difficulty range based on the user's skills level.
3. Recommend programming problems in a way that will ensure the growth of a user in almost every topic rather than the partial growth of certain or few topics.
4. Recommend programming problems in such a way that will ensure the growth of the user as a good problem solver.

3.3 Working Procedure with Flow Chart

Basic Algorithm of Proposed Model.

Step 1 – Start: - User logged in our system

Step 2 − User view’s current recommendation made by our system for him/her

Step 3 − User send a new recommendation request to our system

Step 4 − Our system will request Codeforces web server to provide the submission history for the user

Step 5 − Codeforces provide the submission history for the user to our system

Step 6 − Our system analyzes the submission history and extract necessary data from that history.

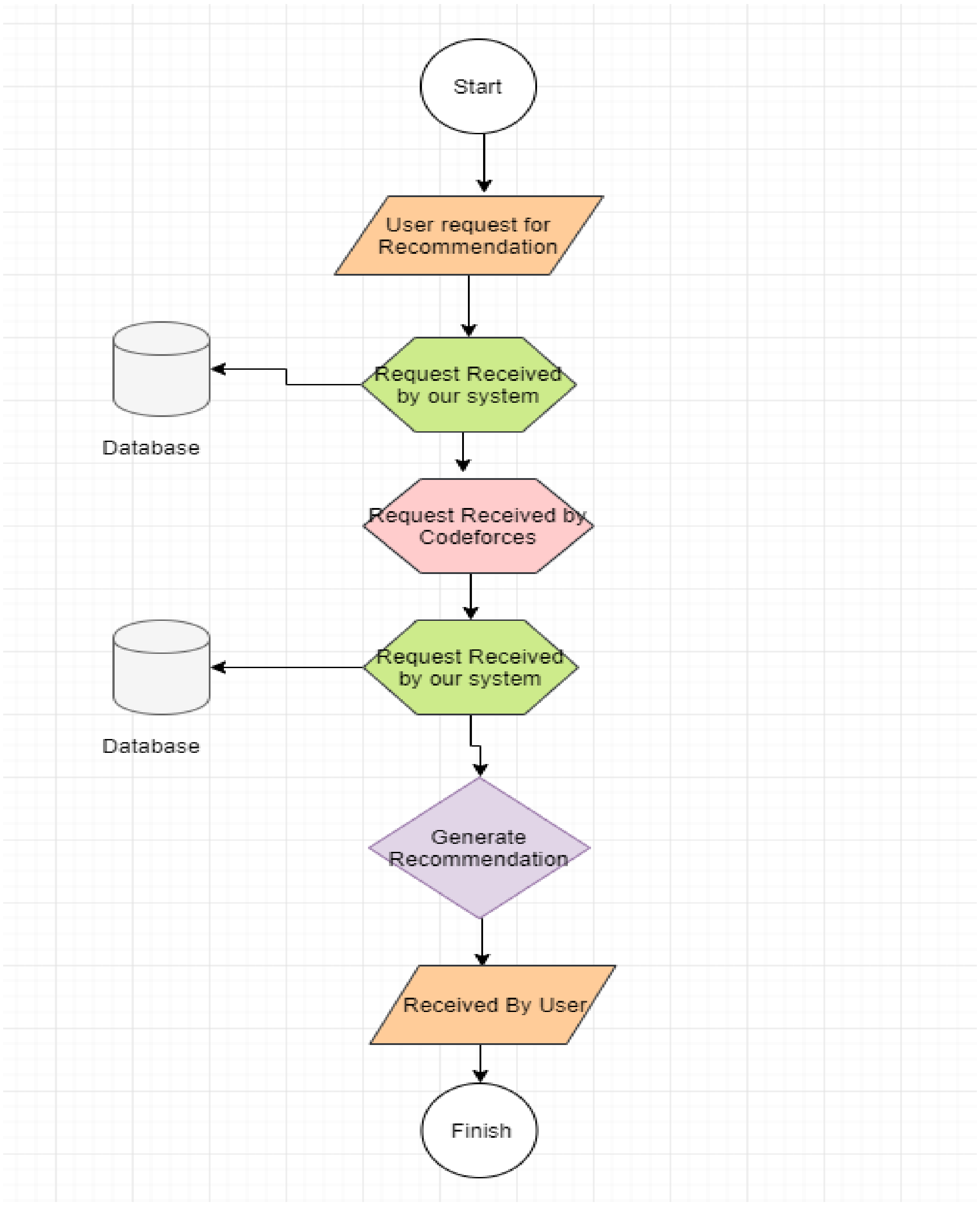
Step 7 − Then our system generate recommendation on the basis of user submission history.

Step 8 − System response the user by sending the generated recommendation.

Step 9 – System response the user by sending the generated recommendation.

Step 10 – User receives the recommendation made by our system.

Step 11 – End



Start
User request for Recommendation
Database
Request Received by our system
Request Received by Codeforces
Database
Request Received by our system
Generate Recommendation
Received By User
Finish

Fig 3.2: Flow Chart of proposed model

3.4 Research Subject and Instrumentation

Research Subject:

Research subject: We have proposed a solution by using content-based recommendation technic, that will recommend programming problems to a user on the basis of user previous problem-solving history.

Instrumentation:

Design:

Draw.io: This is an open-source online tool for designing diagram, such that flow charts, process diagrams, org charts, UML diagrams, ER models, network diagrams, and much more.

Report Writing:

Google Docs: This is an online-based document application. That is used in online at any computer with internet. And access user can view, edit

Microsoft word: Microsoft word is a mostly used word processing application for creating any types of documents like letters, quizzes, and student assignment

Grammarly: This is a platform that checks grammar. It’s basically a writing app, that help to write clear and effective English.

Development:

Phpstorm: Phpstorm is an IDE that is designed to handle large php projects. It is developed by Jetbrains. Phpstorm supports all the type of related tools also testing and debugging tools. We used student version of Phpstorm. It has all the features that a proffesional verson have .

Project Management:

MeisterTask: MeisterTask is an online tools for project management actually, It is similar to Trello.

Communication:

Skype: Skype is a calling platform that helps us communicate with each other for group discussion.

Gmail: We use Gmail for communication and file transfer with each other.

Data Store and Share:

Google Drive: Google drive is a online-based cloud storage service to store any kind of data.

We store our all document in Google drive.

Google Site: We publish a web page that contain our research related information on Google sites.

3.5 Data collection

We collected data from different sources. Our primary data sources was Codeforces. We collected around 7000 programming problems from codeforces. We stored those data in our system's database. then our secondary data source was kaggle.

Codeforces data:

Kaggle data:

3.6 Structures of Online Judges

Currently there are many active online coding judges which provide various programming problems of wide ranges of categories. Among these online judges there are some judges which don't contain good quality problems. A serious problem solver usually solves problems from online judges that contain a wide range of quality problems. Codeforces, Uva, ATCoder , Sphere online judges (SPOJ) are among the most popular online judges.

1) Codeforces [1]

Codeforces is a Russian based online judges dedicated to competitive programming. Codeforces was created and currently maintained by a group of notable competitive programmers from Saratov State University led by Mikhail Mirzayanov (named as Mike Mirzayanov) . Codeforces arranged different types of programming contests. Div-

1, Div-2 or Div-3 held about 2-3 times in a week, Also codeforces educational contests (2-2.5 hours), held 2-3 times per month. In codeforces it is possible to challenge or hack other contestants' solutions. CF also has social-networking through the use of internal public blogs and Personal messaging.

2) Sphere Online Judge (SPOJ) [2]

Sphere Online Judge or SPOJ called in short is an online judge with over 500,000 registered users. Its problem archive consists of over 40,000 problems. All the problems in the problem archive were prepared by its community of problem setters or are taken from programming contests that are already held. SPOJ allows users to organize contests under their own rules and also includes a forum where problem solvers can discuss a problem.

3) ATCoder [3]

AtCoder is an Online judges website based in Japan.

Basically there are three types of contests at AtCoder:

- AtCoder Grand Contest (AGC). This is the best contest at Atcoder. The problems have high originality and require interesting observations to solve.
- AtCoder Regular Contest (ARC). The problems may be a bit typical compared to AGC problems, but still problem solvers can enjoy them and these are good for practice.
- AtCoder Beginner Contest (ABC). These types of contests are mainly targeted for newcomers at competitive programming. The problem sets are easy and educational.

For the purpose of our proposed solution we would like to use the API provided by Codeforces. By using codeforces API we will fetch problems with categories such as Implementation, Graphs, Mathematics , String Processing , Computational Geometry and so on. It also provides the difficulty level of each problem in a problem category and statistics of a problem. We can also retrieve submission history for a certain user who has a codeforces account.

3.7 Recommendation phase:

Before processing a recommendation from a user statistic for a user, we have to build a structure that holds information for problems in each category such as the problem's category, the number of accepted submissions, the number of total submissions and the difficulty level. As for the accepted number of submissions and total submissions we were able to acquire them using the Codeforces API. Using this information, we have to build a problem profile. Some examples of problem profiles are:

Problem Statistics of a 1000 difficulty level Problem of Mathematics Category

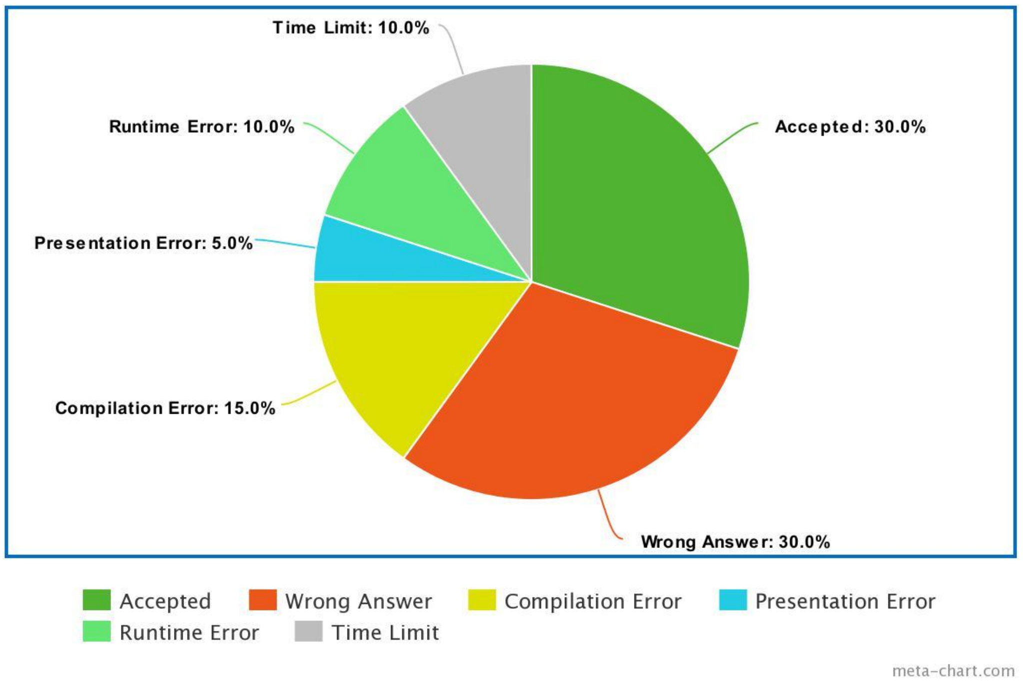


Fig 3.3: Statistics of a 1000 difficulty Problem of Mathematics Category

Problem Statistics of a 1100 difficulty Problem of Ad Hoc Category

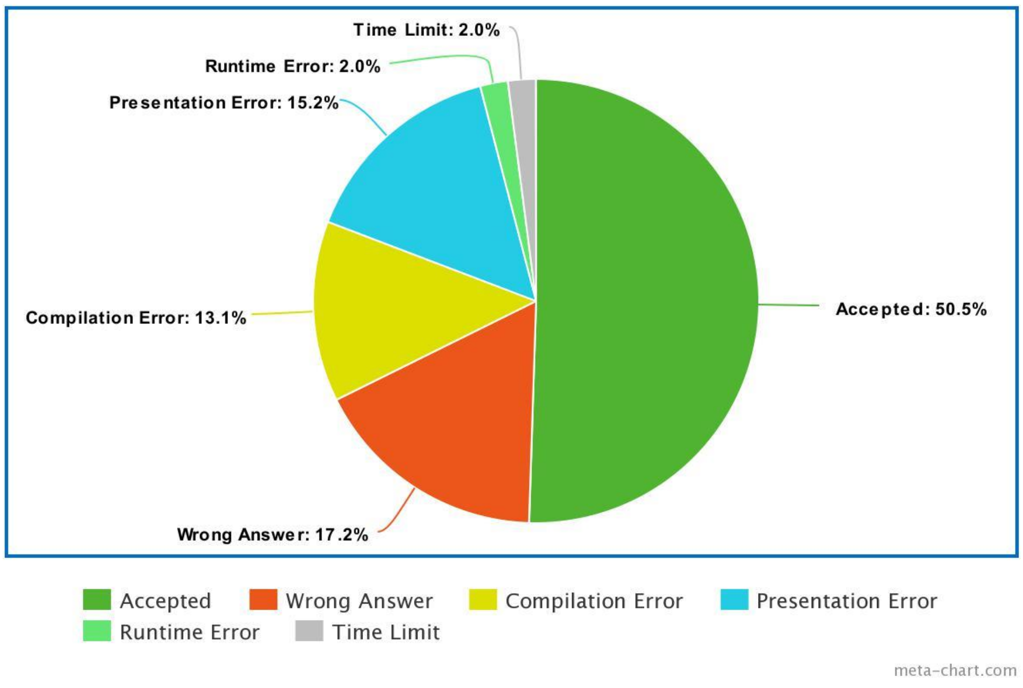


Fig 3.4: Statistics of a 1100 Problem of Ad Hoc Category

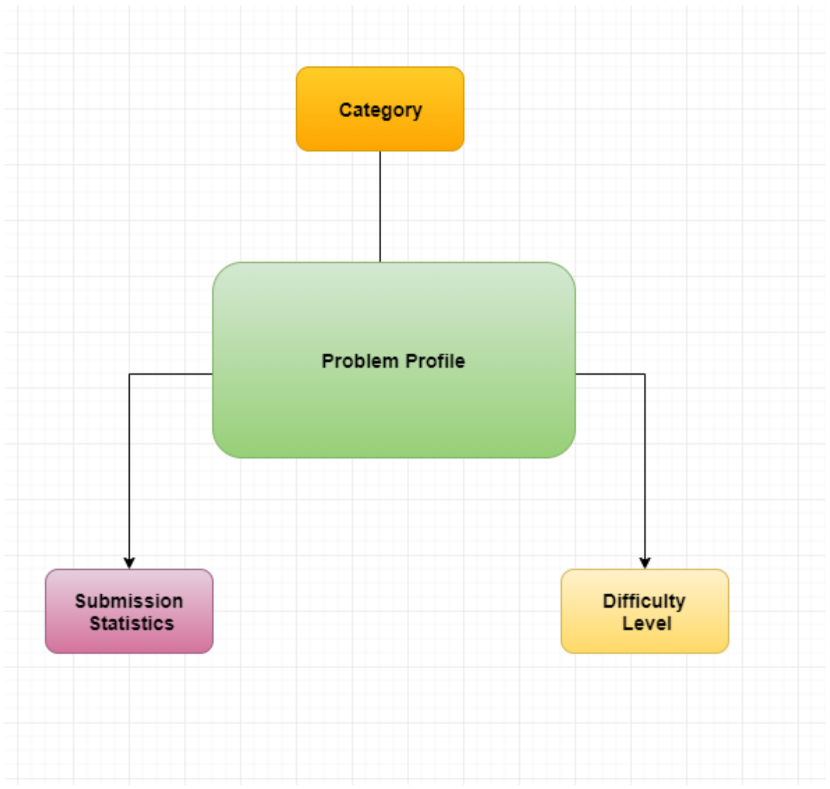


Fig 3.5: Profile Building for a user

There are two types of users in an online judge.

1. New Users
2. Existing Users

New Users:

For new users, we don’t have any statistics or history for which we can analyze or evaluate the user’s skill level as a competitive programmer . So in this case , we will use non - personalized recommendation techniques. We will choose K Problems from each problem category. These K problems will have the lowest difficulty (easiest problems) in those categories. In Codeforces, difficulty level ranges from 500 to 3800 with 3800 being the most challenging level. In Certain categories (Probabilities, FFT) with difficulty below 1500 may not exist. In that case the next lowest level will be considered. The k problems suggested from each category will be randomly picked. One question may arise, why not suggest only easy Ad-hoc/introduction problems rather than picking a certain number of problems from each category ? As our objective was to ensure the growth of a problem solver in the all rounder basis so we exposed the new user to problems from other categories as well, but those problems are the easiest from their respective categories.

Pseudocode:

```
for each category from categories:
   count = 0
   for each level starting from lowest level from problem category :
       count ++
       suggest this problem
       if count == k:
           break
```

Existing User:

For existing users we will retrieve all the required information (user statistics , history , solved problems) from codeforces using codeforces username. Then the fetched data will contain the list of problems solved by the user in Codeforces. Afterwards, we will build a user profiles for the user. The user profile will contain the problems solved by that user in each difficulty level for each and every category of problems. A sample user profiles represented using bar chart is given below:

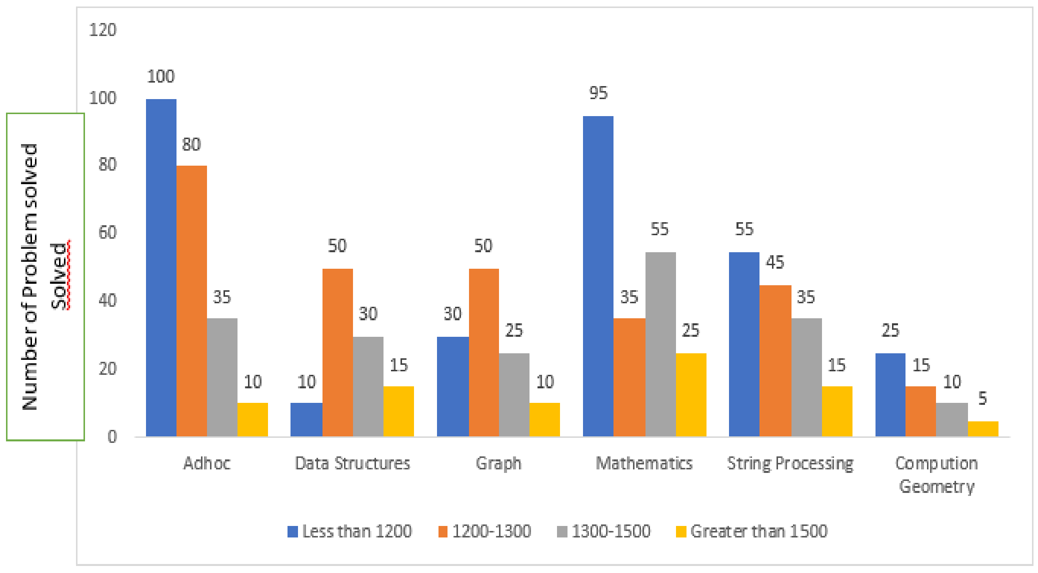


Fig 3.6: Statistics of a number of problems solved by a user from each category

Next we will iterate each category one by one and we will try to determine the user's skill level in each category. It also may happen that the user is good at some category while not as good at some other categories. The reason for this is often programmers team up in a group of three when they participate in many onsite contests for which an individual programmer does not work on all the problems in the problem set. This process builds biases towards some categories or topics for that competitive programmer. But to perform better in programming contests, all the team members should need to be well adept in almost all topics. In any onsite programming contests if one programmer gets stuck on a problem due to some corner cases or any errors, having another team member who has significant expertise on the same topic will help a lot for identifying the error. So our goal is to ensure growth of a competitive programmer in all categories by providing the suitable problems.

In order to recommend problems that match a user's level we will filter out all the problems solved by that user in a specific category. Then, we will filter the problems that are solved based on difficulty level. A chart showing problems solved by a user named 'zim2520 in Ad Hoc category are given below.

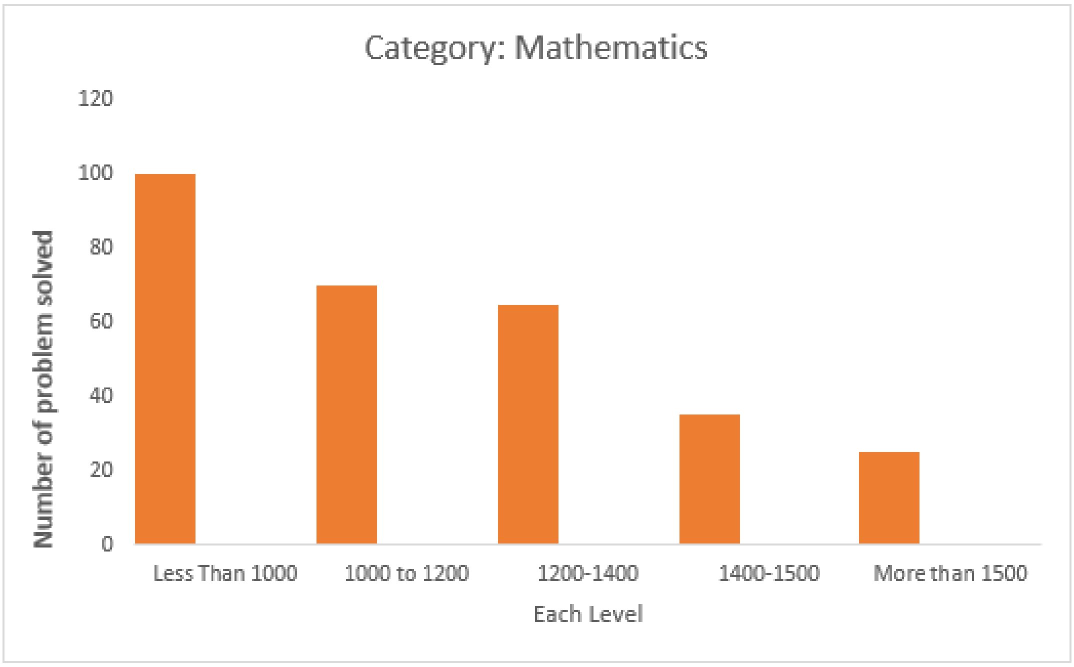


Fig 3.7: Statistics of a number of problem solved by a user from Mathematics category (in integer)

The problem with the above method is that it does not hold accurate information regarding the level of the user currently in. It is possible that one may simply assume that the user is capable enough to solve a level with rating 1400-1500 problems so recommended problems should be of level with difficulty rating 1400-1500 only. But that is not true. Often a programmer is able to solve some higher-level problems but not capable enough to solve other problems in that difficulty level, It basically means it requires to solve a significant amount of programming problems in a lower level in order to gain consistency. But now a question may arise; how many problems need to be solved in certain categories to ensure the mastery of that level? The number of problems in each category in Codeforces is not large. Also the number of problems in each difficulty level are not equal. So the number of problems is not a good way to evaluate the masteriness of a level. Thus, we came up with the idea of considering the ratio of problems solved to the total number of problems in that specific level in a specific category.

Now we are getting data in the below format:

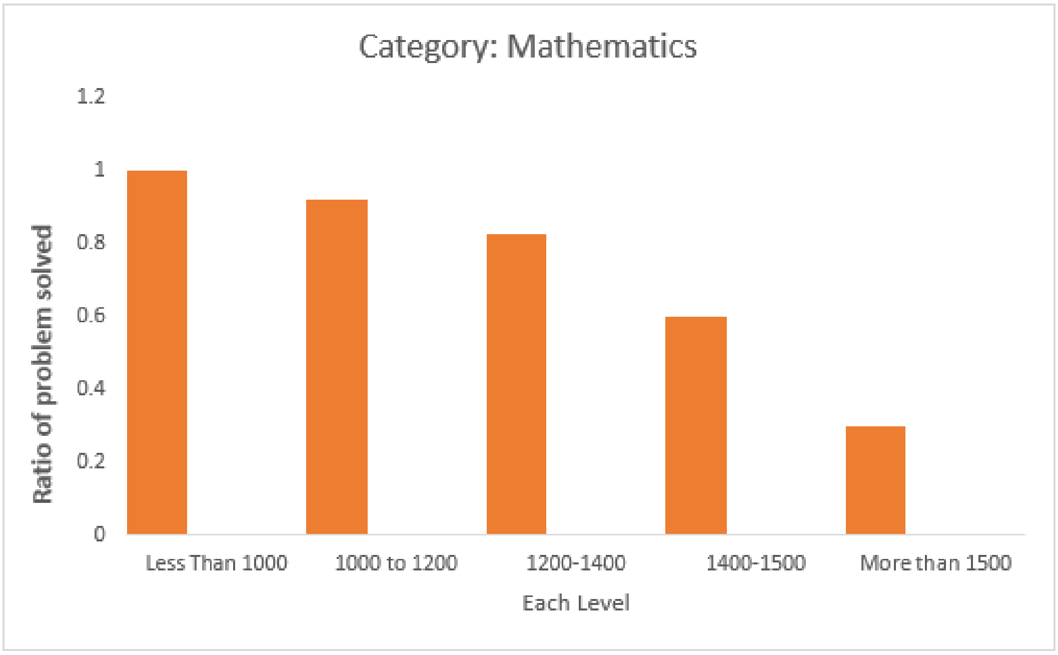


Fig 3.8: Statistics of a number of problem solved by a user from Mathematics category (in ratios)

From the above list of given ratios of all the difficulty levels we will choose the highest level that has exceeded a certain ratio. If the ratio exceeds 0.5 then we do not recommend problems from that level but recommend problems from the next level. if the ratio exceeds 0.4 but is less than 0.7 then we recommend problems from that level and from the next level. But if the ratio is below 0.4 then only problems from the current level get recommended.

In this way, our recommender system will choose first k problems from each category and make a recommendation to the user.

Pseudocode :

For a user, getting recommended user skill level for each category

```
UserLevel = get the lowest level available in that category
Ratio = 0
Get List of solved problems for the user
Map with key as category and value as recommended level
for each category from all category
   Make a list of number of solved problems for each difficulty level
   Make a list of ratio of solved problems and total problems for each difficulty
level
   for each level from level ratio list
       if level ratio > RecommendedLevelRatio
           UserLevel = UserLevel+1
           recommendedLevelRatio = minimum of (level ratio + 1, 0.5)
        add category and UserLevel + 1 to map
        if reommendedLevelRatio > 0.4 and RecommendedLevelRatio < 0.5
           add category and userLevel+1 to the Map
return Map
```

For a given user, making recommendation

```
Get list of all available problems
Get list of user solved problems
For each category
   Get UserRecommendedLevel
   Filter a list of unsolved problems for the given category and user Recommended
Level
   Choose K random problems from filtered list and make recommendation
```

3.8 Implementation Requirement

The details system information is given below:

Hardware Requirement (minimum):

Processor:

Over 3.0 Ghz and 3 core CPU and multithreading enabled.

Memory:

At least 8GB of physical RAM.

Storage:

At least 20 GB of HDD space.

Software Requirement:

Operating System

- Linux - Ubuntu 18.04
- Windows - Windows 10 (professional)

Required Environments

- Xampp
- Php V7
- Mysql

Packages

- Laravel >= 6
- Gazzle Http

CHAPTER 4

Experimental Results and Discussion

4.1 Introduction

We have tested our recommender system by taking 10 users of Codeforces. From these 10 users we have picked 3 different types of users (one who is expert, one who is in intermediate level and another one is beginner) for analyzing the results.

4.2 Experimental Results and Analysis

The results and the analysis of the results are given below.

4.2.1 Analysis for the Expert User

Our Recommendation for the user:

Table 4.1: Recommendation for the Expert User

Problem ID	**Problem Name**	**Category**	**Difficulty Rating**
363B	Fence	Dynamic Programming	1200
1108D	Diverse Garland	Dynamic Programming	1500
489C	Given Length and Sum of Digits..	Greedy	1400
1338A	Powered Addition	Mathematics	1100
1335B	Construct the String	String Processing	1200
1084C	The Fair Nut and String	Implementation	1500
1327B	Princesses and Princes	Graph Theory	1300
1331E	Jordan Smiley	Computational Geometry	1600

Analysis:

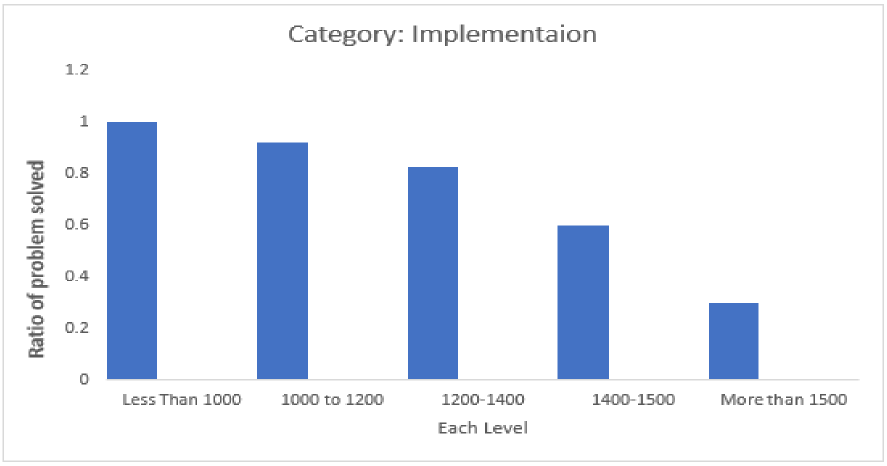


Fig 4.1: Analysis for the Expert User

From the above chart we can see that in "implementation" categories our expert user has solved more than 50% (1400-1500) difficulty problems but has solved less than 40% problems in level with difficulty more than 1500. Therefore the user needs to solve a bit more in level with difficulty 1200-1400 in order to level up.

4.2.2 Analysis for the Mid-Level User

Our Recommendation for the user:

Table 4.2: Recommendation for the Mid-Level User

Problem ID	**Problem Name**	**Category**	**Difficulty Rating**
706B	Interesting drink	Dynamic Programming	1200
1348D	Phoenix and Science	Binary Search	1400
489C	Phoenix and Beauty	Data structure	1400
1343B	Balanced Array	Mathematics	800

1342B	Binary Period	String Processing	1000
1336B	Xenia and Colorful Gems	Two pointer	1100
1020B	Badge	Graph Theory	1300

Analysis:

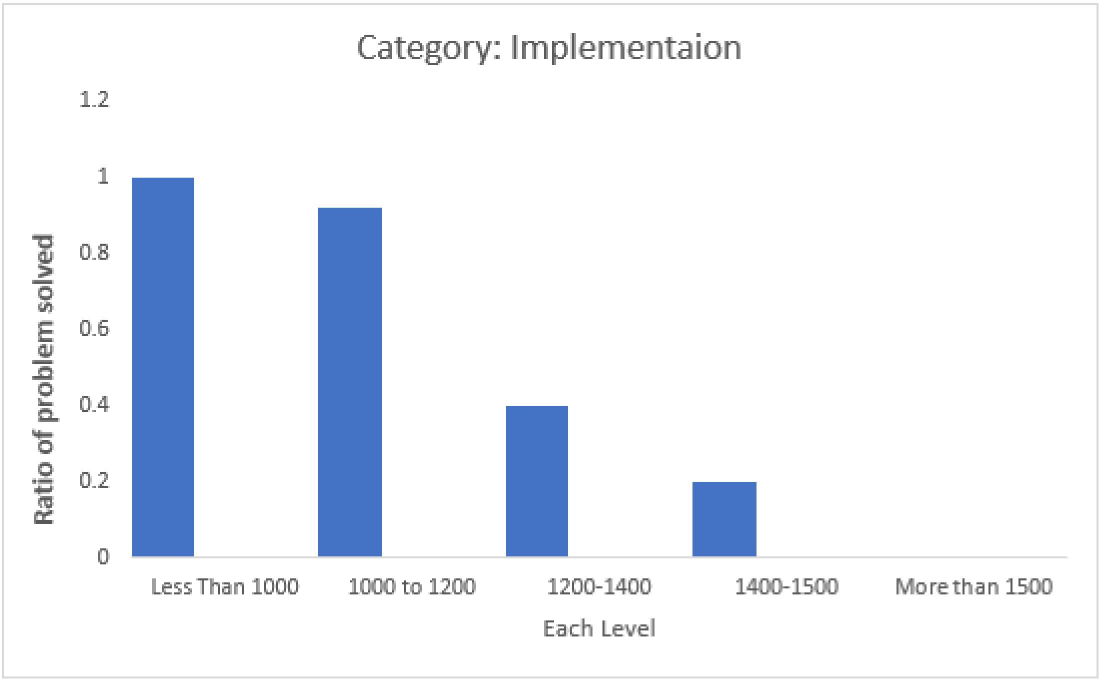


Fig 4.2: Analysis for the Mid-Level User

In the “implementation” category, our intermediate users have solved more than 80% of the level with 1000 to 1200 problems. So this level of problems are not necessary to be solved by the user. Our intermediate user needs to solve the problems from 1400 - 1500 difficulty rating as only 20% of problems have been solved from that category. So, the recommended problems are both from difficulty rating 1200-1400 and 1400 - 1500 which fits the current skill level of our intermediate user.

4.2.3 Analysis for the Beginner Level User

Our Recommendation for the user:

Table 4.3: Recommendation for the Beginner Level User

Problem ID	**Problem Name**	**Category**	**Difficulty Rating**
785A	Anton and Polyhedrons	Implementation	600
141A	Amusing Joke	Sorting	800
1165B	Polycarp Training	Data structure	1200
999A	Mishka and Contest	Brute-Force	700
723A	The New Year: Meeting Friends	Mathematics	1000
734B	Anton and Digits	Greedy	700
750A	New Year and Hurry	Binary Search	800

Analysis:

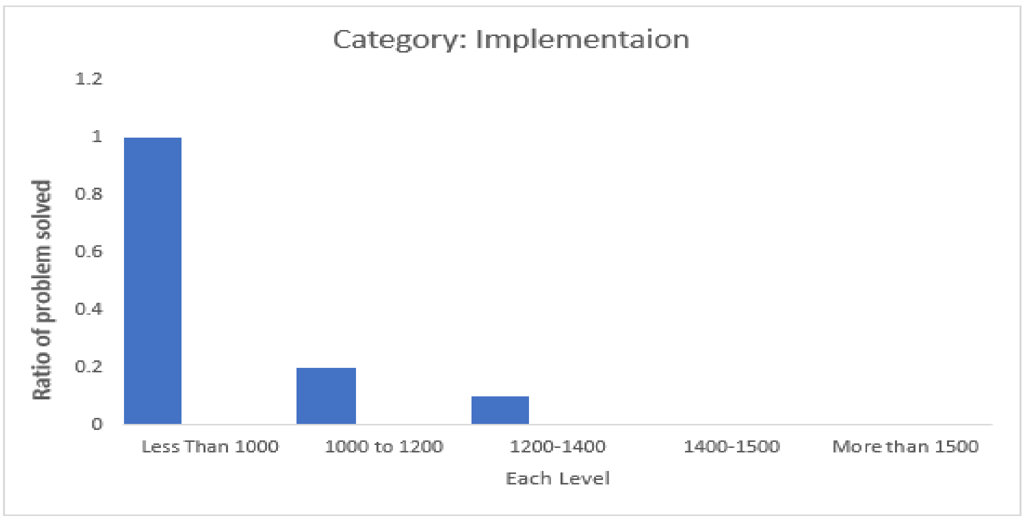


Fig 4.3: Analysis for the Beginner Level User

In the "Implementation" category, our beginner user has solved more than 80% problem with difficulty less than 1000 . And our user solved less than 20% problems from the level with difficulty 1000 to 1200 and solved less than 10% difficulty level 1200-1400 problems. So, it makes sense to recommend him only difficulty level 1000-1200 problems for now.

4.3 Evaluation

1) Online evaluation

This evaluation is done by allowing few user to use our recommendation system. Every user used our system for last 30 days to practice programming. Our commendation system generated a recommendation list for every user for daily basis. Basically to evaluate our system we used following steps:

1. The recommendation engine created a recommendation list on the basis of user submission history
    a. If user is new to the system then it will use non-personalized recommendation technic to recommend
    b. If user is already in our system then recommendation engine will create a user profiles based on user submission history
2. User will attempt those problems.

3. When a user solved a problem, we will collect the following data for evaluation
    a. Number of attempts make by user to solve this problem
    b. How many problem user has been solved for this category?
    c. User rating for this problem recommendation (1 to 10)

Then we calculated our evaluation matrix from collected data. A small snapshot of the matrix is given below:

Table 4.6: Snapshot of evaluation (online user)

	1st Recommendation	2nd Recommendation	3rd Recommendation	4th Recommendation	4th Recommendation
User-1	0.9024	0.6924	0.7290	0.9589	0.6482
User-2	0.5065	0.7912	0.7223	0.7378	0.7788
User-3	0.9087	0.7920	0.7224	0.9322	0.7787
User-4	0.7501	0.9686	0.8598	0.8838	0.7501

4.4 Discussion

From the above results we can ensure that our recommender systems are able to recommend suitable problems to the user that will be most likely to be solvable and also ensure the proper growth of the user.

CHAPTER 5

Summary, Conclusion and Implication for Future Research

5.1 Summary of the Study

Though the whole study, we tried to build a system that will support competitive programmers like a coach. Our works are motivated from the contests like ACM International Collegiate Programming Contests (ICPC), IOI (Informatics Olympiad). We hope that our system will help programmers to perform better in these types of contest. Our system may help in the preparation for a student for coding interview. In coding interview, challenges given to a candidate is much similar to competitive programming problem. If any one, wants to practice for interview then he/she can use our systems.

5.2 Conclusions

In conclusion, Competitive programming have many and many benefits. By competitive programming one may become a desirable candidate to major companies. It is an opportunity to be seen by large companies like google, facebook, Amazon and many other authorities in the IT industry. Competitive programming made a student to become more disciplined , faster and focused programmer. It also helps student to do team works . In short, the benefit of competitive programming is not be concluded. We tried our best to find a solution to help competitive programmers to do programming in cheerful environment. We tried to provide a system that will help competitive programmers to come out of frustration.

5.3 Implication for Further Study

Not all the system are not 100% perfect. Perfection is always a work in progress, There our system is only at its early stages. Though our proposed solutions fulfill all the requirement we have discussed for a recommender system that will recommend programming problems to programmers. However, Our proposed solution took advantages of codeforces API. In our recommendation systems, we uses the feature for codeforces category and rating system. But all the online judges doesn’t provide category and rating. There are new challenges added on contest site regularly. The main sources of these problems is the contest arranged by those sites. There are few more things that our proposed solution not cover. Few of are:-

1. Un-Categorized Problems: New problems are added on online judges at time passes. Most of these problems comes from contest held on those sites. To categorize a new problem machine learning might be able to use in this scenario. We can train a model to categorize new problems by using existing dataset
2. Dealing with problems that falls multiple category
3. More advanced system that helps faster growth for a competitive programmer.

REFERENCES

[1] “Codeforces online judge “, Available at <<https://codeforces.com>> [Last accessed: 07 June, 2020]

[2] “Spehre online judge(SPOJ) “, Available at <<https://www.spoj.com/>> [Last accessed: 07 June, 2020]

[3] “AtCoder Online Judge “, Available at <<https://atcoder.jp/>> [Last accessed: 07 June, 2020]

[4] “Xampp local server” , Available at <<https://www.apachefriends.org/index.html>> [Last accessed: 01 April, 2020]

[5] Ken Lang. (1995), NewsWeeder : Learning to Filter Netnews (To appear in ML 95)

[6] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM. December.

[7] Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., and Riedl, J. (1997). GroupLens: Applying Collaborative Filtering to Usenet News. Communications of the ACM, 40(3), pp. 77-87.

[8] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In Proceedings of CSCW ’94, Chapel Hill, NC.

[9] Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and Evaluating Choices in a Virtual Community of Use. In Proceedings of CHI ’95.

[10] Aggarwal, C. C., Wolf, J. L., Wu K., and Yu, P. S. (1999). Horting Hatches an Egg: A New Graph-theoretic Approach to Collaborative Filtering. In Proceedings of the ACM KDD’99 Conference. San Diego, CA. pp. 201-212.

[11] Schafer, J. B., Konstan, J., and Riedl, J. (1999). Recommender Systems in ECommerce. In Proceedings of ACM E-Commerce 1999 conference.

[12] Anil Poriya, Tanvi Bhagat, Neev Patel and Rekha Sharma. Article: Non Personalized Recommender Systems and User-based Collaborative Recommender Systems. International Journal of Applied Information Systems6(9):22-27, March 2014.

[13] “Application Programming Interfaces (API) “ Available at : <<https://www.mulesoft.com/resources/api/what-is-an-api>> . [Last accessed: 10 Jan, 2020 at 11:05 PM]

[14] “Codeforces API “, Available at <<https://codeforces.com/apiHelp>> [Last accessed: 05 March, 2020]

[15] “Guzzle HTTP “, Available at <<http://docs.guzzlephp.org/en/stable/>> . [Last accessed: 05 March, 2020]

[16] “Heroku“, Available at <<https://heroku.com/>> . [Last accessed: 08 May, 2020]

[17] R Yera Toledo , Yailé Caballero Mota and Luis Martínez , Article : A Recommender System for Programming Online Judges Using Fuzzy Information Modeling, March - 2018

APPENDICES

Appendix A: Application Programming Interfaces (API)

We used API in our application in multiple purses. API is a acronym of Application Programming Interfaces. API Allows two application two communicate with each other [13]. In our Application we used Codeforces API [14] and Guzzle HTTP [15] Library to fetch Problems from Codeforces. Guzzle HTTP [15] also uses to retrieve a submission of user.

Appendix B: Costly System

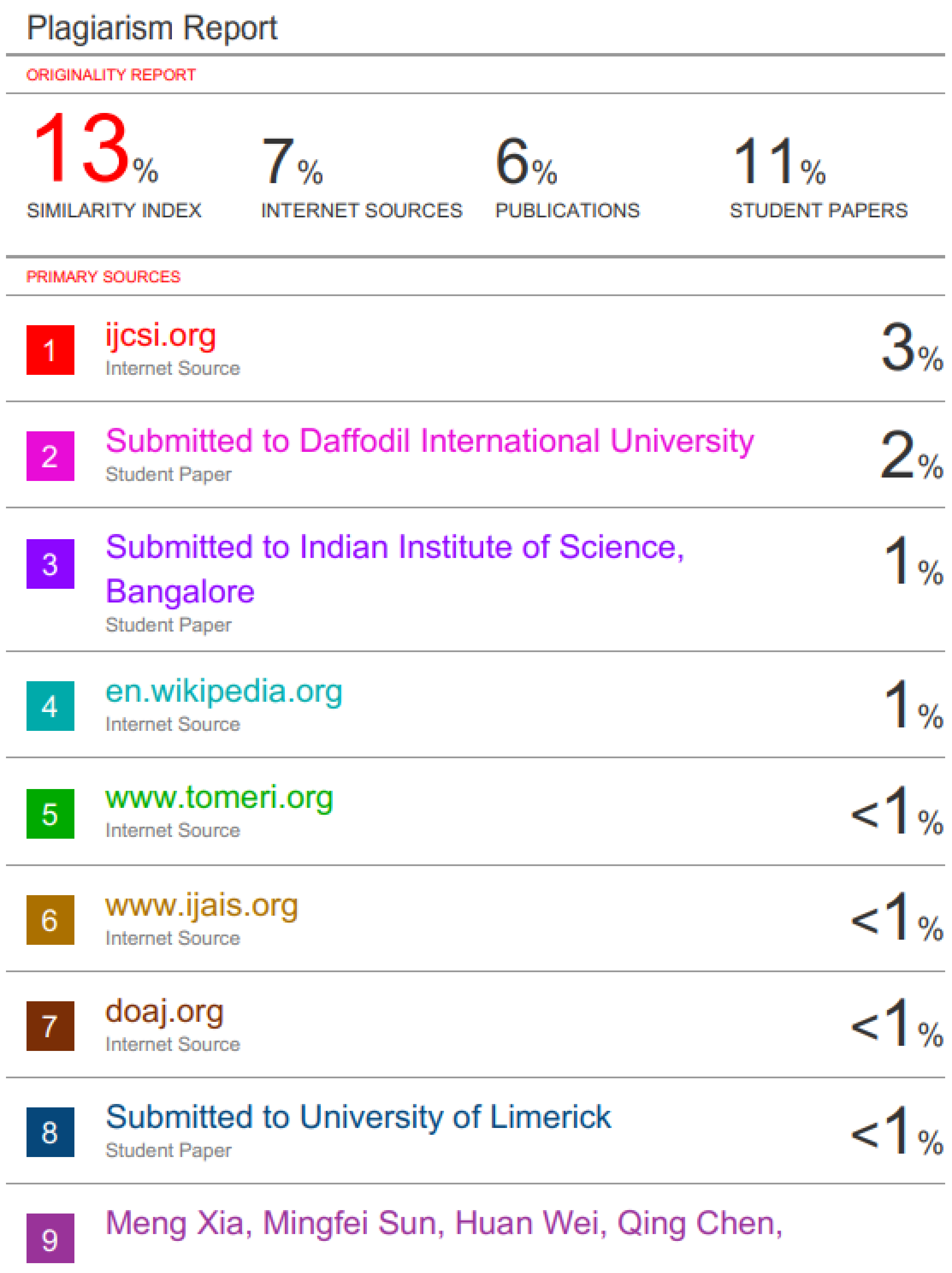
Our System is hosted on Heroku[16]. Heroku is a Paas (Platform as a service) platform provides container to host web application. Heroku gives subscription based services and the subscription is costly. And our system also uses databases to store the programming's problems initially, The subscription for database is also costly.

Appendix C: Web based interface

Now a day's cloud computing is much reliable and scalable than locally running system, It makes the system to accessible our system to almost all type of users from any geolocations.

Appendix D: Open Source Repository

Works for this repository is not yet completed. If any one wants to contribute this project, then he/she can fork this repository and able to contribute.
Link: <<https://github.com/nesarjony/cfassistant-mysql>>



Plagiarism Report

ORIGINALITY REPORT

13%	7%	6%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ijcsi.org Internet Source	3%
2	Submitted to Daffodil International University Student Paper	2%
3	Submitted to Indian Institute of Science, Bangalore Student Paper	1%
4	en.wikipedia.org Internet Source	1%
5	www.tomeri.org Internet Source	<1%
6	www.ijais.org Internet Source	<1%
7	doaj.org Internet Source	<1%
8	Submitted to University of Limerick Student Paper	<1%
9	Meng Xia, Mingfei Sun, Huan Wei, Qing Chen,	

	Yong Wang, Lei Shi, Huamin Qu, Xiaojuan Ma. "PeerLens", Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19, 2019 Publication	<1%
10	"Advanced Data Mining and Applications", Springer Science and Business Media LLC, 2014 Publication	<1%
11	Submitted to University of Warwick Student Paper	<1%
12	Submitted to University of Edinburgh Student Paper	<1%
13	www.sushitakeaway.es Internet Source	<1%
14	Submitted to Manchester Metropolitan University Student Paper	<1%
15	graduate.gradsch.uga.edu Internet Source	<1%
16	citeseerx.ist.psu.edu Internet Source	<1%
17	files.grouplens.org Internet Source	<1%
18	Wai Lam. "HIGH-DIMENSIONAL LEARNING	

	FRAMEWORK FOR ADAPTIVE DOCUMENT FILTERING*", Computational Intelligence, 02/2003 Publication	<1%
19	research.ijais.org Internet Source	<1%
20	Submitted to RMIT University Student Paper	<1%
21	Submitted to University of Surrey Roehampton Student Paper	<1%
22	kunz-pc.sce.carleton.ca Internet Source	<1%
23	Submitted to Universiti Teknikal Malaysia Melaka Student Paper	<1%
24	Submitted to University of Abertay Dundee Student Paper	<1%
25	www.ijcaonline.org Internet Source	<1%
26	Submitted to University of South Australia Student Paper	<1%
27	Ken Lang. "NewsWeeder: Learning to Filter Netnews", Elsevier BV, 1995 Publication	<1%
	Submitted to VIT University	