Project Report
Statistical Methods in AI
CS 471

Group Name: Classifiers
Team #: 1

Team Members:
Chittaranjan Rath (2018201007)
Prakash Nath Jha (2018201013)
Nitish Srivastava (2018201012)
Suraj Garg (2018202003)

Problem statement:
Attempt to guess programming problem tags for a problem
Performing Multi-label Classification for coding problems from different paradigm:

a) On Basis of problem Statement
b) On Basis of problem Solution

Approaches Considered:

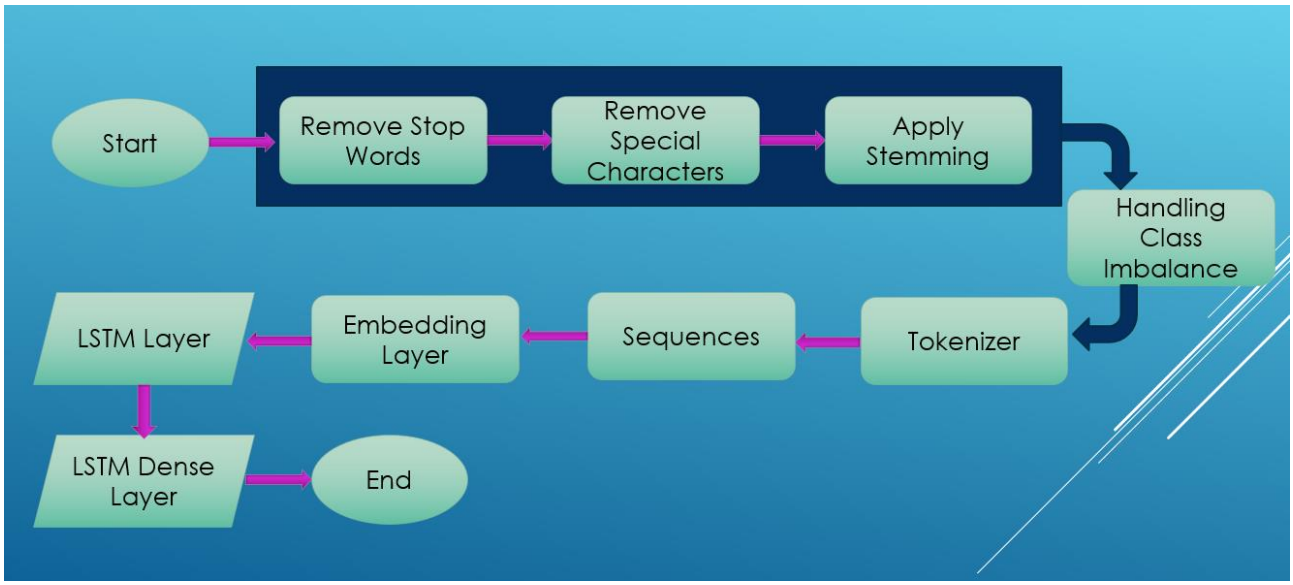
- Linear SVC
- LSTM

Motivation:

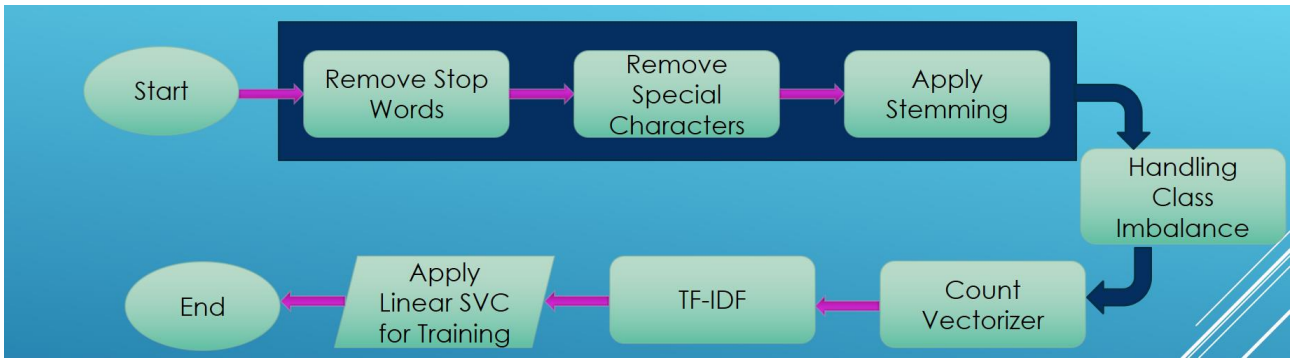
In order to choose the next problem to solve in CP we are dependent only on already tagged problems. Suppose if we get a problem where we don't know the tags for the problem and we may want to have prerequisite information for solving problem. It would be beneficial if we had a mechanism which could predict tags for a problem.
Our solution (model prediction) provides an assistance to user when he is not aware of approaches which might be taken for solving the problem.
Here this model will help us in providing some tags which will be helpful for us. So if our algorithm predicts some tags and actual tags are subset of those predicted then it would be helpful for us (i.e. having high recall is good for our scenario).

Work Performed:

LSTM Implementation:



Linear SVC Implementation:



Major Library Used:

- Beautiful Soup
- NLTK
- Keras for LSTM
- SKLearn
- Visualization Tools :
    - Matplotlib
    - Seaborn

Results:

Codeforces:

Linear SVC :

On Validation data

Hamming loss: 0.015843595543268828
Recall score: 0.8167536450207133
Precision score: 0.9323613049728309
F1 score: 0.8694578820595751
Roc-Auc Score: 0.9006383620039062

On Train data

Hamming loss: 0.004744824938038887
Recall score: 0.9450274058885807
Precision Score: 0.9816226383679485
F1 score: 0.962702771247846
Roc-Auc score: 0.9696318876067691

Output :

Actual Tags	Predicted Tags
('bitmasks', 'combinatorics', 'dp', 'math', 'number theory')	bitmasks, combinatorics, dp, math, number theory
('dp', 'dsu', 'graphs')	dp, dsu, graphs
('dfs and similar', 'number theory', 'trees')	dfs and similar, number theory, trees
('two pointers',)	greedy
('binary search', 'constructive algorithms', 'data structures')	binary search, constructive algorithms, data structures
('binary search', 'math')	geometry

('binary search', 'greedy', 'implementation', 'sortings')	binary search, greedy, implementation, sortings
('implementation',)	implementation
('graphs', 'greedy', 'shortest paths')	graphs, greedy, shortest paths
('greedy', 'sortings')	greedy, sortings
('binary search', 'data structures', 'greedy', 'two pointers')	binary search, data structures, greedy, two pointers
('brute force', 'greedy', 'sortings')	brute force, greedy, sortings
('math',)	math

LSTM :

On validation data:

Hamming loss: 0.08211164167434587
Recall score is 0.36719212352719643
Precision score is 0.4185405317110809
F1 score is 0.370921722328081
Roc – Auc score: 0.6372614361293356

Output :

Actual Tags	Predicted Tags
'binary search', 'math')	geometry, ternary search
('binary search', 'greedy', 'implementation', 'sortings')	binary search, data structures, dp, matrices
('implementation',)	math
('graphs', 'greedy', 'shortest paths')	dsu, geometry, ternary search

('greedy', 'sortings')	binary search, two pointers
('binary search', 'data structures', 'greedy', 'two pointers')	combinatorics, implementation, math, matrices
('brute force', 'greedy', 'sortings')	flows, graph matchings, greedy
('math',)	constructive algorithms
('binary search', 'data structures', 'sortings', 'two pointers')	greedy, implementation, math, ternary search
('bitmasks', 'dp', 'probabilities')	combinatorics, math, number theory
('dp',)	dp
('data structures',)	dp

Codechef:

Linear SVC :

On Train data

Hamming loss: : 0.00010737690524099963
Recall score: 0.9963988296196263
Precision score: 1.0
F1 score: 0.9981929128985215
Roc – Auc score:: 0.9981994148098132

On Validation data

Hamming loss: : 0.0035900320688860934
Recall score: 0.8169366364935684
Precision score: 0.9869656689791306
f1 score: 0.8877612732179232
Roc – Auc score:: 0.9081184551966692

Output:

Actual Tags	Predicted Tags
('simulation',)	simulation
('backtracking',)	
('digraph', 'graph')	digraph, graph
('simulation',)	simulation
('tree',)	
('sorting',)	sorting
('bruteforce',)	
('adhoc',)	adhoc
('adhoc',)	adhoc
('tree',)	tree
('combinatorics',)	combinatorics
('gcd',)	gcd, greedy
('strings',)	strings
('kruskal',)	kruskal
('simulation',)	simulation
('gcd',)	
('gcd',)	gcd
('bipartite', 'geometry', 'hard', 'maxflow', 'set')	bipartite, geometry, hard, maxflow, set
('greedy',)	greedy
('greedy',)	greedy

LSTM Codechef:

On validation Data:

Hamming loss: 0.032401229511968416
Recall score is 0.13628599316739873
Precision score is 0.034009075672246175
f1 score is 0.05092539350923068
Roc – Auc score: 0.5182189517291518

Output:

Actual Tag	Predicted Tag
('algebra',)	greedy
('bfs', 'dijkstra')	gcd, greedy
('bfs', 'dijkstra')	gcd, greedy
('gcd',)	greedy
('strings',)	simulation
('dynamic',)	greedy
('simulation',)	simulation
('game', 'theory')	dp, greedy
('expo', 'matrix', 'recurrence')	gcd, greedy, tree
('bfs', 'dijkstra')	gcd, greedy
('maths',)	greedy
('combinatorics',)	greedy
('binarysearch', 'fibonacci', 'hashing')	gcd, greedy, tree
('dp',)	greedy
('gcd',)	greedy
('segment', 'tree')	dp, greedy

('basic', 'pattern', 'stack')	gcd, greedy, tree
('simulation', 'sorting')	gcd, greedy

Discussion:

Data Collection:

In order to collect data about competitive programing we have visited many Websites like

- UVA
- A2OJ
- CodeForces
- CodeChef
- Peking Online Judge

Some of the above listed sites did not have the relevant data which was required like UVA , A2OJ did not have solutions to the problems.

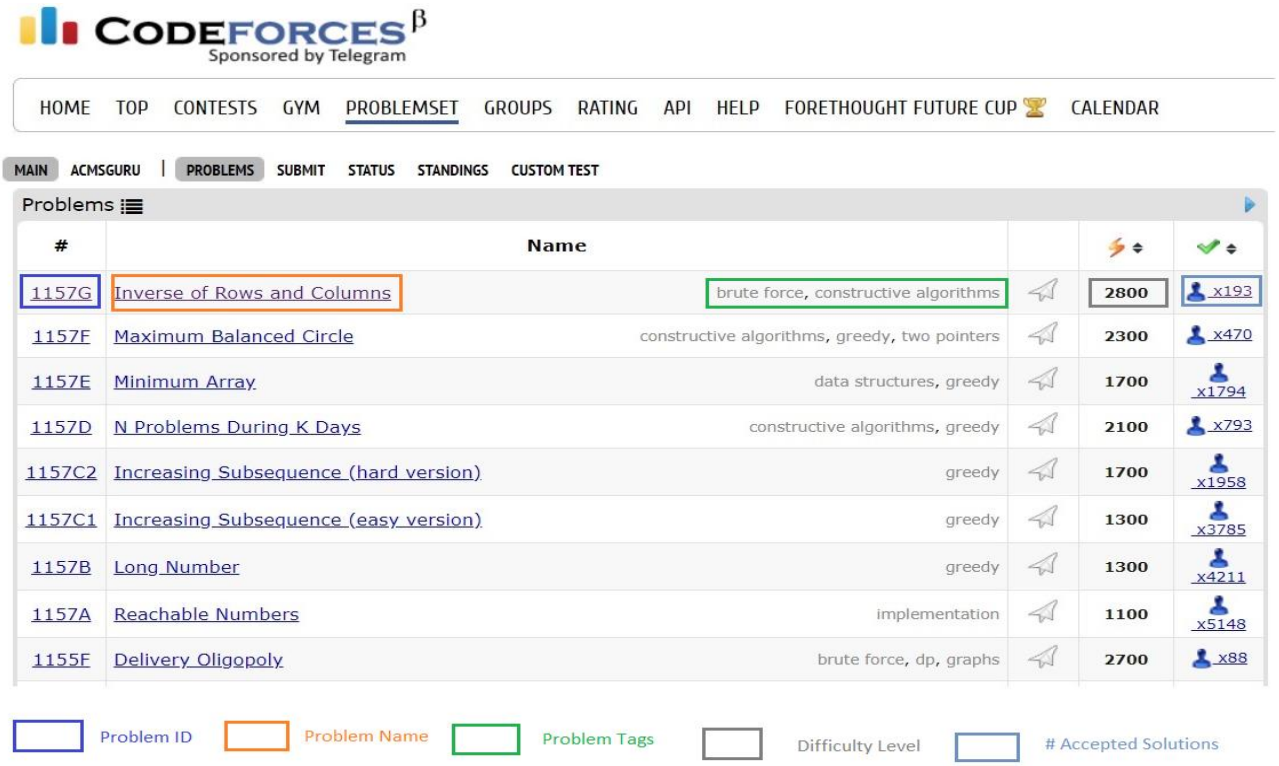
Hence we collected data from two major sources:

- Code Forces
- Code Chef

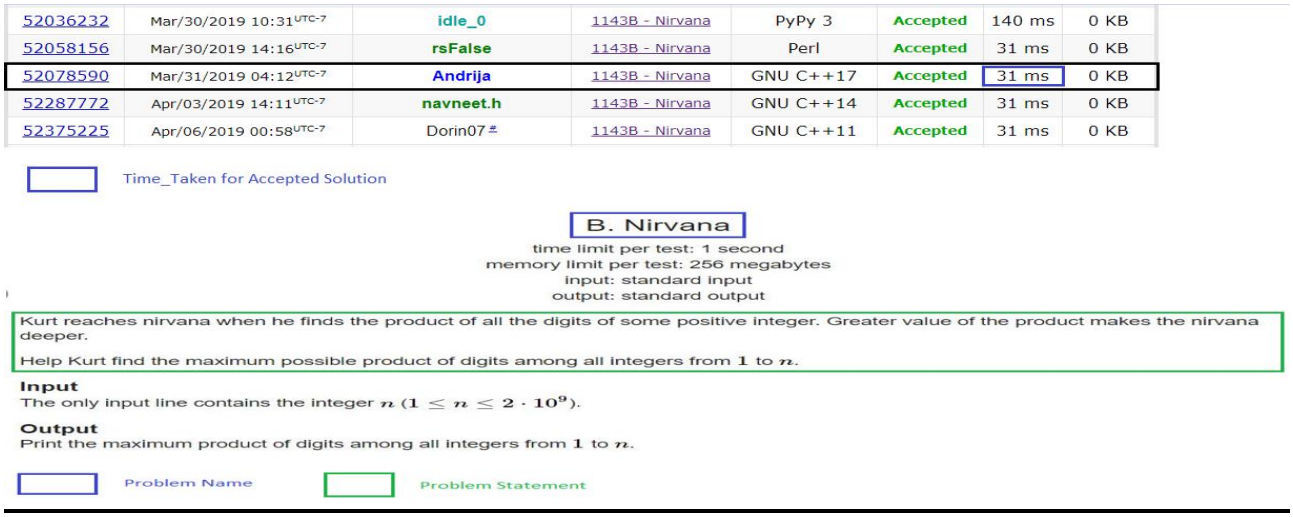
Code Forces Data Collection:

Home Page of CodeForces look like diagram below:

Below is observation highlighted in figure:



This Page contains Links to solution of problems highlighted above. It also shows the problem statement that we have captured.
We are ignoring Input and Output Section of the problem Statement.



Our Sample Data Set looks like something below:

- It shows sample tuple from our dataset and the captured solution from Codeforces
- We have ignored many irrelevant information from website.

Sample From Data Set

id	name	problem statement	tags	difficulty	solution	time_taken	author
1143C	Queen	You are given a rooted tr	dfs and similar		using namespace std; int main(){	46 ms	munjalvaibh

1143B	Nirvana	Kurt reaches nirvana whe	brute force,math,number theory		#include <bits/stdc++.h>
					using namespace std;
					int f(int n){
					if(n<10)return max(1,n);
						31 ms	Andrija

1143A	The Doors	Three years have passes a	implementation		#include <bits/stdc++.h>	155 ms	179000
1142D	Foreigner	Traveling around the wor	dp		no code found	NA	NA

Solution Captured

By Andrija, contest: Codeforces Round #549 (Div. 2), problem: (B) Nirvana, **Accepted**, #

```
#include <bits/stdc++.h>
using namespace std;
int f(int n){
if(n<10)return max(1,n);
return max((n%10)*f(n/10),9*f(n/10-1));}
int main(){
int n;cin>>n;
cout<<f(n);}
```

›Judgement Protocol

Test: #1, time: 15 ms., memory: 0 KB, exit code: 0, checker exit code: 0, verdict: OK

Input
```
396
```
Output
```
216
```
Answer
```
216
```
Checker Log
```
ok answer is '216'
```

Similar steps are followed for CodeChef also. Below are snapshot for CodeForces website

Problem Statement

Problem Title

Problem Code

Coin Flip | Problem Code: **CONFLIP**

Tweet Like Share 25 people like this. Sign Up to see what your friends like.

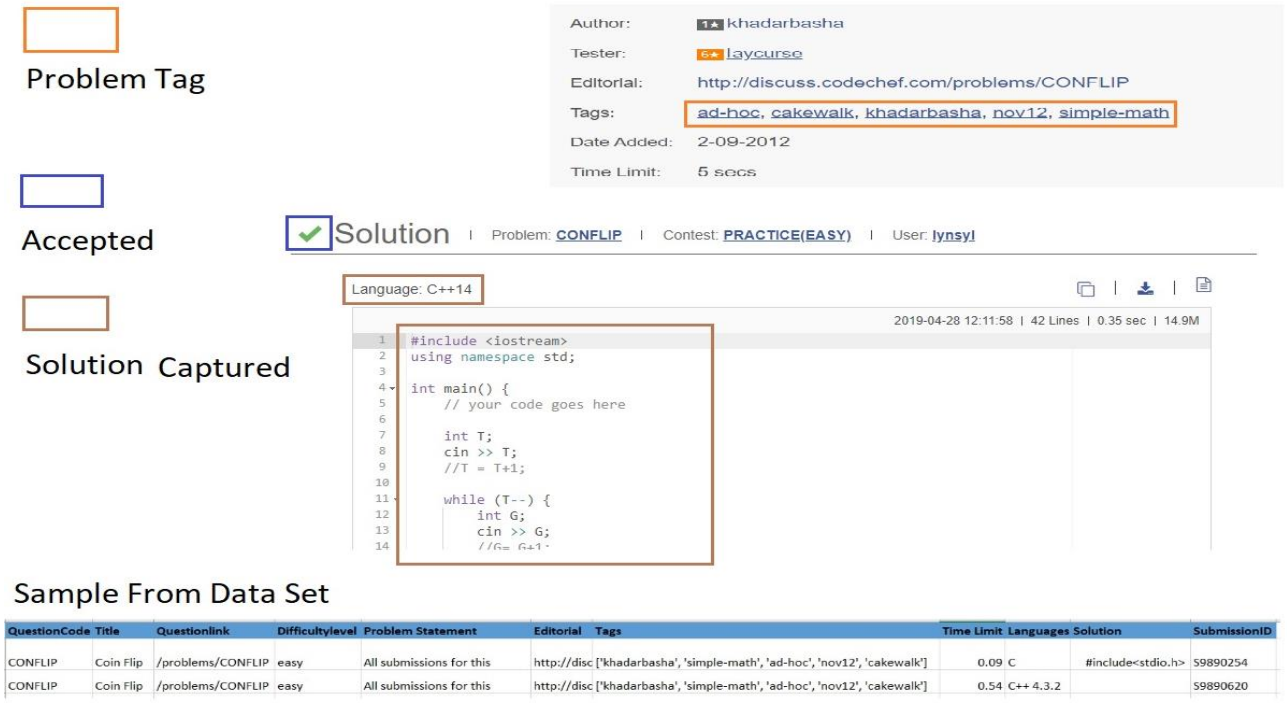
All submissions for this problem are available.

Little Elephant was fond of inventing new games. After a lot of research, Little Elephant came to know that most of the animals in the forest were showing less interest to play the multi-player games.Little Elephant had started to invent single player games, and succeeded in inventing the new single player game named **COIN FLIP**.

In this game the player will use **N** coins numbered from **1** to **N**, and all the coins will be facing in "Same direction" (Either **Head** or **Tail**),which will be decided by the player before starting of the game.

The player needs to play **N** rounds.In the **k**-th round the player will flip the face of the all coins whose number is less than or equal to **k**. That is, the face of coin **i** will be reversed, from **Head** to **Tail**, or, from **Tail** to **Head**, for **i ≤ k**.

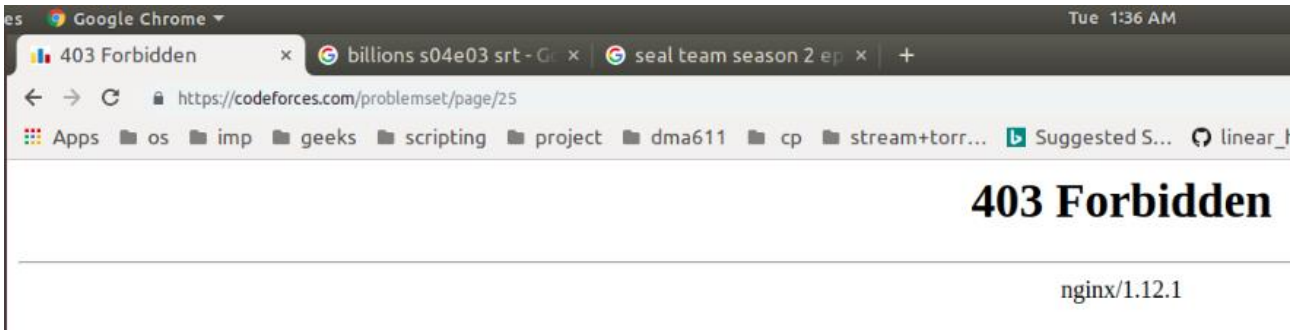
Elephant needs to guess the total number of coins showing a particular face after playing **N** rounds. Elephant really becomes quite fond of this game **COIN FLIP**, so Elephant plays **G** times. Please help the Elephant to find out the answer.



QuestionCode	Title	Questionlink	Difficultylevel	Problem Statement	Editorial	Tags	Time Limit	Languages	Solution	SubmissionID
CONFLIP	Coin Flip	/problems/CONFLIP	easy	All submissions for this	http://disc	['khadarbasha', 'simple-math', 'ad-hoc', 'nov12', 'cakewalk']	0.09	C	#include<stdio.h>	S9890254
CONFLIP	Coin Flip	/problems/CONFLIP	easy	All submissions for this	http://disc	['khadarbasha', 'simple-math', 'ad-hoc', 'nov12', 'cakewalk']	0.54	C++ 4.3.2		S9890620

Problem faced During Data Collection:

- At Multiple places we were not able to both problem statement along with their solution like at UVA, A2OJ etc.
- While web Scrapping Codeforces was blocking requests randomly at many instances.
- Unstructured solutions of problem (like we have solution from tester but not for setter)

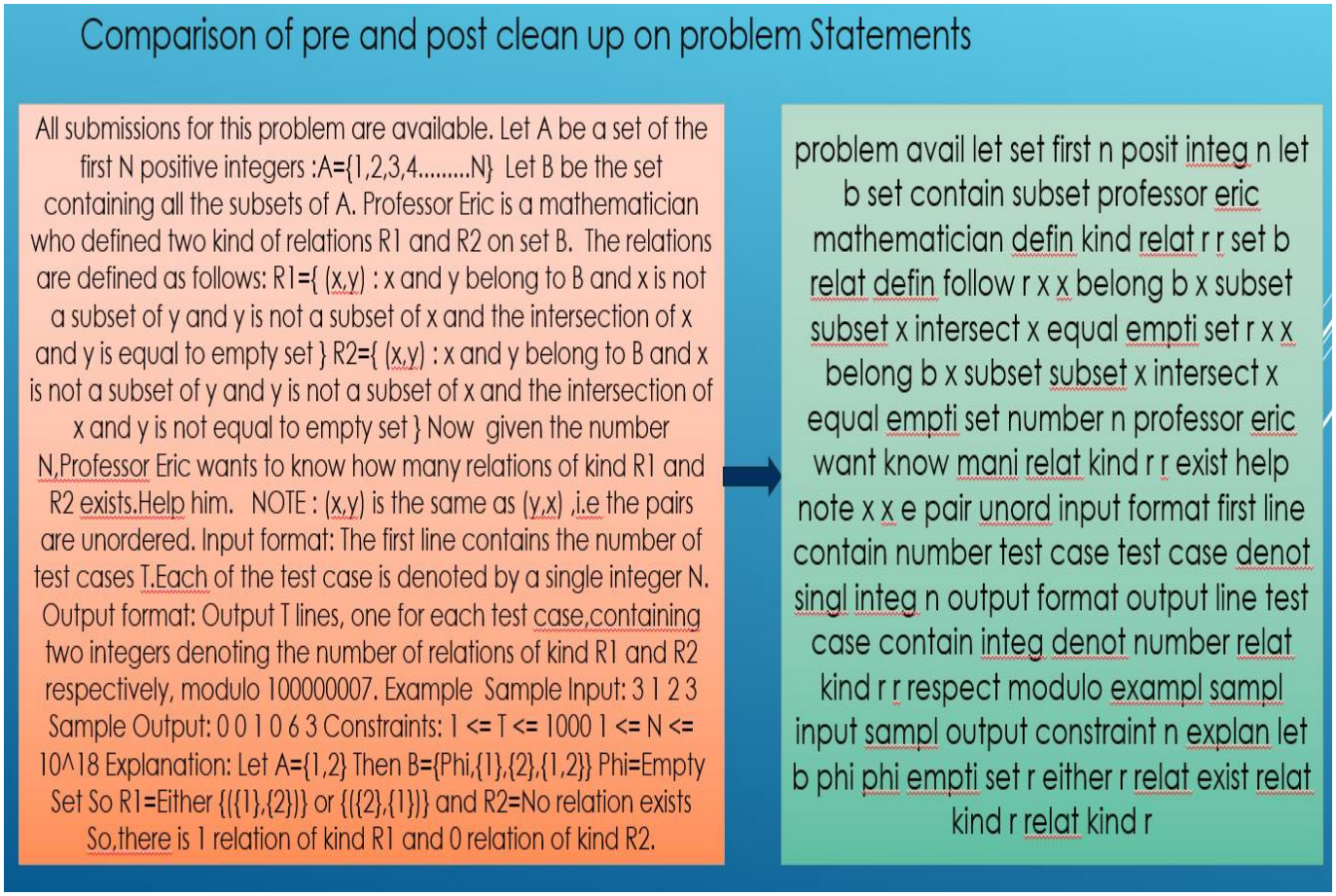


Model Creation:

Since our dataset had problem statement along with solutions (multiple solution of same problem was also available in our dataset) we planned two approaches :

1) On the Basis of Problem Statement:

- In our model we have taken as input the problem statement and we have performed various cleaning activities like Counter Vectorizer, TF IDF handling Class Imbalance.
- After that the model was fed with a pre-processed input.
- Below diagram shows the results of before and after stemming on problem statement :



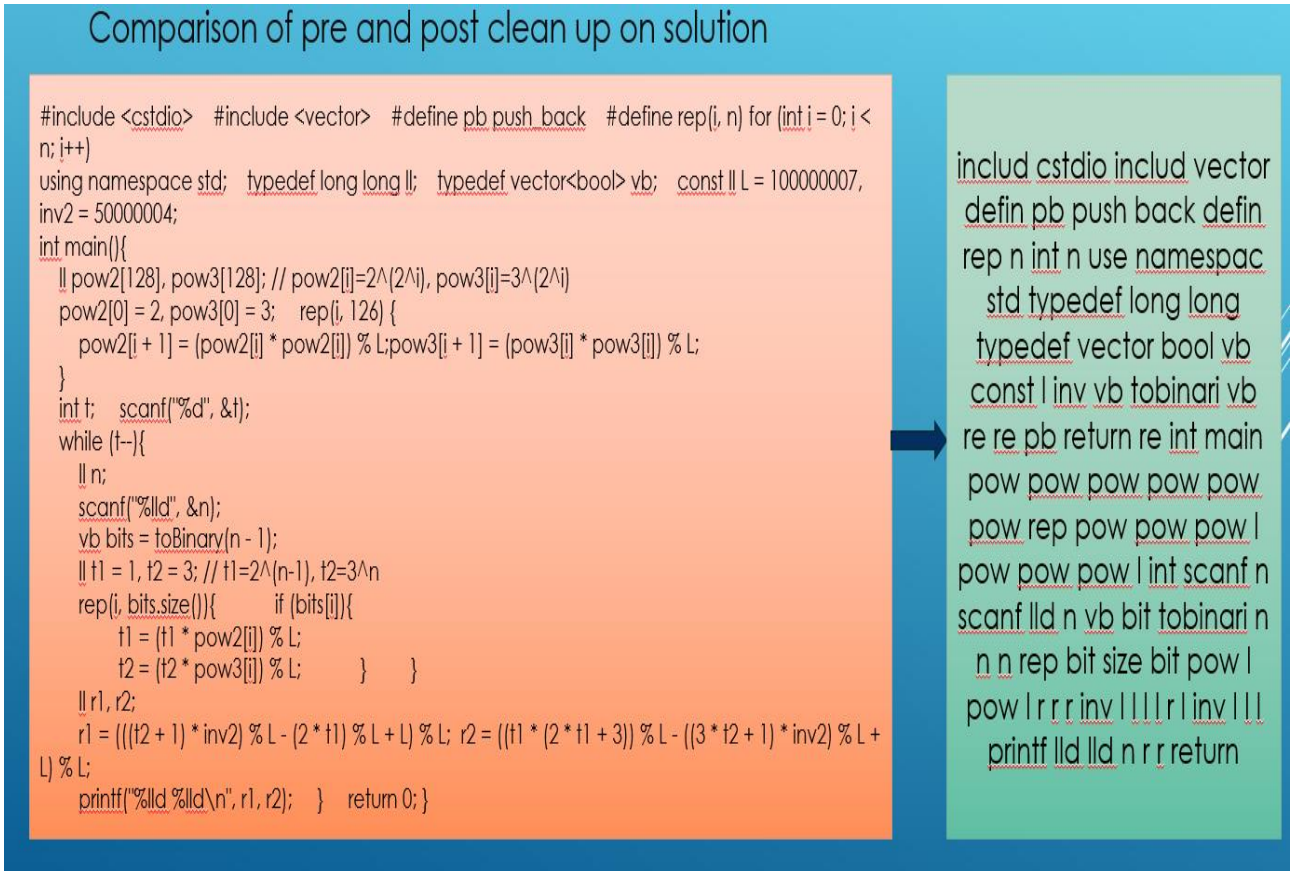
- As per above figure there is we do get useful information for processing if stemming is applied. Hence we are performing this operation.

- We proposed two type of model:
  - LSTM
  - Linear SVC
- We found out post analysing our results that linear SVC and LSTM perform not up to mark.
- Based on problem statement approaches applied, model is underperforming because it's quiet difficult to manually interpret problem statement.
- We do not get much information about the problem tags from problem Statements.

- As Linear SVC, the data is high dimensional we are achieving separable hyperplane. Hence results are better.

2) On the Basis of Solution :

- In our model we have taken as input the solution and we have not performed any kind of data pre processing like clearing stopwords, applying stemming.
- Below diagram shows the results of before and after stemming on solution:



- As per above figure there is we do not get any useful information for processing if stemming is applied. Hence we are not preforming this operation.
- We found out post analysing our results that linear SVC outperform LSTM.
- We have observed the above result because of LSTM does not considers programming language semantic.

- As Linear SVC , the data is high dimensional we are achieving separable hyperplane. Hence results are better.

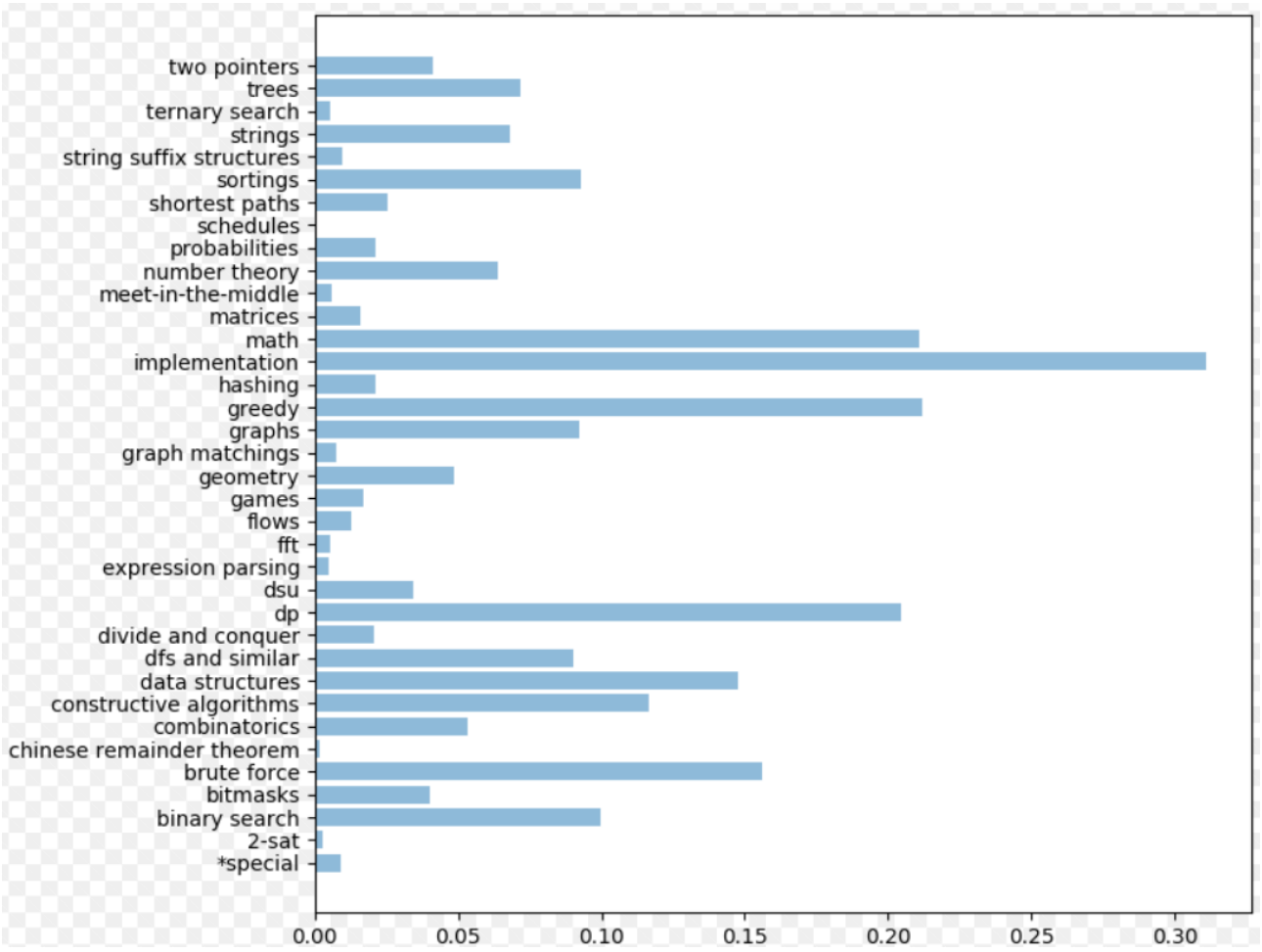
Github Link:

https://github.com/gargsuraj12/SMAI_FinalProject.git

Data Set Observation :

Code Forces dataset :

The data set collected from Codeforces contain multiple solution for same problem and the sample collected was along with the tags also. But there were some tags of very high frequency i.e. those tags were present in most data sample.
Along with this, there were some tag found which were found very rare i.e. they were sparse as to data set collected.
Hence post observation we found that dataset is suffering from class imbalance problem.
Below figure shows scenario of problem:

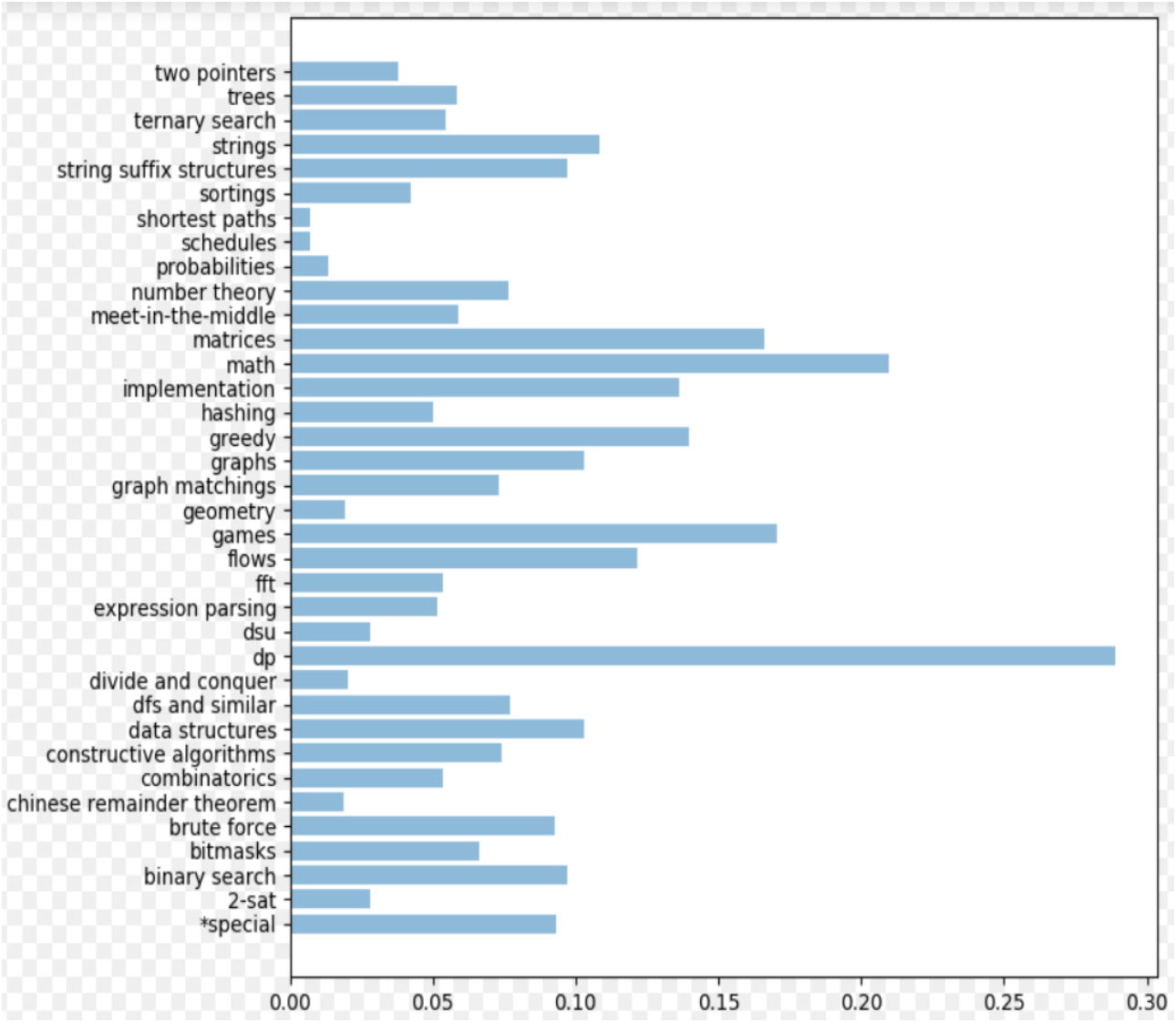


Observation :

Here we see “implementation”: tag is more frequent as compared to FFT tags or *“expression parsing” tags*. In order to resolve this issue, we have undergone below steps:

- In order to reduce high frequency data, we have done under sampling
- After that we have observed some data samples were remain in rarest state as it was before hence, we have performed over sampling on the entire data set again.

Post both process below diagram shows the results :



Observations:

- Initially we saw FFT tags were very rare in dataset. but after performing bot operations its frequency has increased considerably

- We had seen “implementation” tags were most frequent, but now its frequency is relatively decreased.

Code Chef dataset :

Again, the data set collected from CodeChef contain multiple solution for same problem and the sample collected was along with the tags also. But there were some tags of very high frequency i.e. those tags were present in most data sample.
Along with this, there were some tag found which were found very rare i.e. they were sparse as to data set collected.
Hence post observation we found that dataset is suffering from class imbalance problem.

Observation:

Our original dataset was highly skewed i.e. there were many tags which were very rare in the dataset and there were few tags which were highly frequent in the dataset.
Since we are performing multilabel classification the highly frequent tags and the relatively low frequent tags were bot assigned to many different samples i.e. A sample had label as [‘greedy’,’fft’] , another had label as [‘probability’,’greedy’] and there were many sample which contains greedy thus in order to minimise the effect of highly frequent tag (greedy) , it was set – off i.e. we made fft tags (one of rare tag) was given more importance than the highly frequent tags.
After doing this we have performed both :

- Under Sampling
- Over Sampling

And found the figure for same dataset shown above as “regex” was highly infrequent in original dataset and in final dataset it has become considerably more frequent.

DataSet For CodeForces:

DataSet :
https://www.dropbox.com/sh/qqesaj9aoiwwygo/AABrw_ueAZubETK4UiDUJKbma?dl=0

DataSet For CodeChef:

first.csv : https://1drv.ms/u/s!Ag1YzU8OUDQmgb9diSm3PVcdop1x-w
third.csv : https://1drv.ms/u/s!Ag1YzU8OUDQmgb9bD1-XfOuo4_ebJQ
Question.csv :
https://1drv.ms/u/s!Ag1YzU8OUDQmgb9mdL_4GgwvyDDUyQ
Solutions : https://1drv.ms/u/s!Ag1YzU8OUDQmgb9Z9_941QP0a93g5Q

Future Directions:

Making LSTM learn programming language semantics using Code2Vec embedding for better predication.
Scope of Code Generation using problem Statement.
Generalizing the approaches for more number of programming languages.

Task Assignments:

Tasks (Subtasks)	Member(s)
Web Scrapping	
Analysis of Websites	Suraj , Nitish and Chittaranjan
CodeForces Data Gathering	Chittaranjan
CodeChef Data Gathering	Nitish , Prakash
Data Pre-Processing	
Removing Stop Words and Special Characters	Suraj
Implement TF-IDF	Chittaranjan
Implement Stemming	Nitish

Exploratory Data Analyses	
Analyses problem in data set	Suraj , Chittaranjan
Class Imbalance Handle Strategies	Prakash, Nitish
Implementation of Handling Class Imbalance	Chittaranjan, Nitish
Analysed Approaches	Team
LSTM Implementation	Suraj
Linear SVC Implementation	Prakash
Analysis of Evaluation Metric	Nitish ,
Visualization	
Implement of Word Cloud	Nitish
Performance of Model	Chittaranjan
Class Imbalance	Prakash