Multi-label classification for Tag predictions in Programming Challenges

1st Deepshika Reddy
Department of Mathematics
Stevens Institute of Technology
Hoboken, United States
dreddyag@stevens.edu

2nd Amit Singh
Department of Mathematics
Stevens Institute of Technology
Hoboken, United States
asingh70@stevens.edu

3rd Shivani Mogili
Department of Mathematics
Stevens Institute of Technology
Hoboken, United States
smogili@stevens.edu

***Abstract*—The purpose of the study and implementation of this methodology is to learn and understand how 'Multi-Label Classification' works. In order to address this Code force' platform is appropriate. Code Force is a Programming language Competitive platform where Questions are posted, usually its quite challenging to come up with an approach for solving looking at the problem description. So, tagging for questions is essential for a better user experience. When the problem is posted, in most cases, participants, based on their experience, can narrow down the approaches to solve the problem. The goal is to narrow down the scope of problem-solving by tagging the problem with the methodologies that can be used to solve them. For a beginner developer, some problem statements might appear too challenging, and there is a high requirement of suggesting the solution methodology. We know that , it's quite hard to tag the problems for problem creators, and thus we focus on automating tagging using Machine Learning and Deep Learning Algorithms. The tags predicted can be either generic, such as' Math', or more specific, such as' Dynamic Programming' or' Brute Force.' Each and every problem can have multiple tags associated with it, thus making it a Multi-Label Classification.**

I. Introduction

It is observed that from interview preparation to selecting the candidates in the technology (Computer Science) domain, coding challenges are important criteria. The Companies use the coding challenging to filter the candidates and also access their knowledge and problem-solving skills. Due to lengthy questions or lack of time for a beginner, it's quite hard to come up with a solution quickly. In order to aid the candidates, many coding platforms have difficulty levels' and 'problem tagging' for every problem. The methodology of auto-tagging is what is implemented using Machine Learning and Deep Learning algorithms, and we compare their accuracy score through weighted hamming loss, Precision, Recall, F1-score, and hyper-parameter tuning. The system proposed will predict either generic tags like 'statistics' or specific tags like 'Brute Force.'The challenge encountered here could be every data point will generally belong to one class, but as every data point here has multiple tags, each class may belong to multiple data points. Another challenge is if we know the source code of the solution, it is easy to predict the tags, but predicting based on textual content of the problem statement is quite hard. Thus we gathered the data set from Kaggle and represented it in different architecture, and tried to solve it using ML and DL algorithms. In Section 3, we describe our solution, Section 4, we compare the performance of various models used, Section 5, we suggest future work, and in Section 6, we conclude the analysis. The dataset is formed based on scrapping all the problem statements from the Code Force' Platform. We are using the dataset from the Kaggle, which is in the form of a .csv file. It consists of 4 columns ,namely 'contest' ,'problem-name','problem-statement','problems-tags'.'contest' is a numerical quantity and the rest of the attributes are 'categorical' in nature. There are a total of 8343 data points among them 'problems-tags' . It utilized a memory of '260.0'KB of RAM size when loaded on Jupyter Notebook. Based on the problem statement, we used both Machine Learning and Deep Learning Models. We are using onevsrestclassifier(Logistic regression). The loss is measured by varying hyper-parameters. Random Forest /Random Forest Classifier is another robust way to perform Multi-label classification along with concepts of NLP like term matrix-document matrix. Some concepts of NLP like Word2Vec and one hot encoding are also used. Finally, we are using Deep Learning models like LSTM to solve the problem and perform analysis on loss. Experimental Results: The Existing solutions focus on Logistic Regression and other Machine Learning models to solve the problem. But it was identified that with the concept of NLP, deep learning models are performing better with good 'Weight-hamming-score', 'Avg-Precision', 'Avg-Recall', 'Avg-F1score', and other hyper-parameters.

II. Related Work

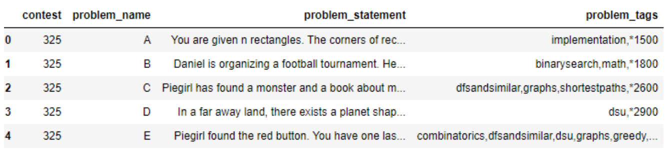
The problem that is to be solved is traditionally solved by Machine Learning algorithms only. Logistic Regression with varied hyperparameter tuning has been performed, and the best with high accuracy is selected. Moreover, the text classification problem is approached, which consists of assigning a text document into one or more topic categories. The methodology employed a Bayesian approach into multi-class, multi-label document classification in which the multiple classes from a document were represented by a mixture model. Additionally, the problem of automated tag advice as a multi-label text classification task in the context of the "Tag Recommendation in Social Bookmark Systems." The suggested method was built using Binary Relevance (BR) and a naive Bayes classifier as the base learner, which were then evaluated utilizing the F-measure. In the Proposed solution, we use Deep Learning models like LSTM with hamming loss, a micro-f1 score that has better accuracy.

III. Our Solution

This section elaborates your solution to the problem.

A. *Description of Dataset*

The dataset was retrieved from the Kaggle repository[1] ; it is consists of the online programming challenges questions and related tags from the Codeforce platform. The data has a four columns contest, problem name, problem statement, problem tags. The dataset has two primary columns: problem statement and problem tags. Problem Statement is the full text from the problem page, and Problem Tags are comma-separated tagged classes for problem statement. Data consist of a total of 8343 rows.



	contest	problem_name	problem_statement	problem_tags
0	325	A	You are given n rectangles. The corners of rec...	implementation,*1500
1	325	B	Daniel is organizing a football tournament. He...	binarysearch,math,*1800
2	325	C	Piegirl has found a monster and a book about m...	dfsandsimilar,graphs,shortestpaths,*2600
3	325	D	In a far away land, there exists a planet shap...	dsu,*2900
4	325	E	Piegirl found the red button. You have one las...	combinatorics,dfsandsimilar,dsu,graphs,greedy,...

Fig. Dataset

Data preprocessing

Given data contain lots of words and character required to remove as that can cause a negative impact on the training process. Data consist of many HTML codes and Latex symbols for mathematical definitions that do not contribute toward the problem statement. We removed these HTML codes and the latex character for data cleaning and pre-processing. We identified data also consisting of lot stop words, newline characters, non-ASCII characters, digits, punctuation, and Unicode. They do not provide any additional information and only increase the problem's dimensionality, so we removed all these characters. Furthermore, we convert text to lowercase.

We performed Lemmatization and Stemming on the problem statement. Lemmatization considers the context and converts the word to its meaningful base form, Lemma. Sometimes, the same word can have multiple different Lemmas. We should identify the Part of Speech (POS) tag for the term in that specific context. Stemming is the process of producing morphological variants of a root word. Stemming and Lemmatization help us to achieve the root forms of a derived term.

We also dropped some null rows we found in the tags column. We separated program difficulty text from tags and saved them to 2 new columns problem-difficulty and problem-tags-values. No duplicate data was found during pre-processing of data.

Statistics Description

We worked on some descriptive statistics after preprocessing. Calculated number unique tag, the average tags per problem, the maximum number of tags on the problem set, minimum number on the problem set. We got an average of 2 to 3 tags on an average per problem statement.

Observation	
Maximum number of tags per problem statement	11
Minimum number of tags per problem statement	1
Avg. number of tags per problem statement	2.5

We performed uni-gram and Multi-gram Analysis for Problem Tag. Unigram tag contains 9 percent of data, i.e., tag column containing only one tag. Multi-gram has 91 percent of data, i.e., tags column with more than two or more than two tags.

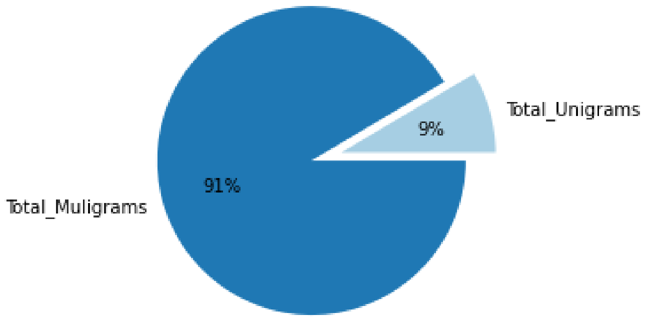


Fig. Uni-gram and Multi-gram Analysis for Problem Tags

The below figure shows frequently used tags in problem statement.

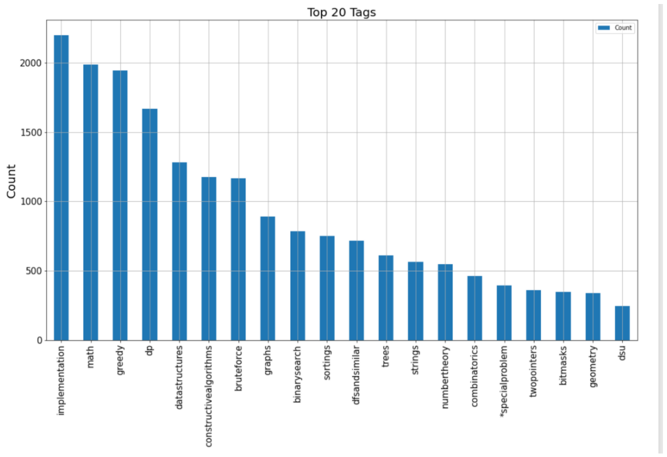


Fig. Top 20 frequently used tags

1. Implementation
2. Math
3. greedy
4. dp
5. datastructures
6. constructivealgorithms
7. bruteforce
8. graphs
9. binarysearch
10. sortings
11. dfsandsimilar
12. trees
13. strings
14. numbertheory
15. combinatorics

16. specialproblem
17. twopointers
18. bitmasks
19. geometry
20. dsu

The below figure shows number of words in the Tagst.

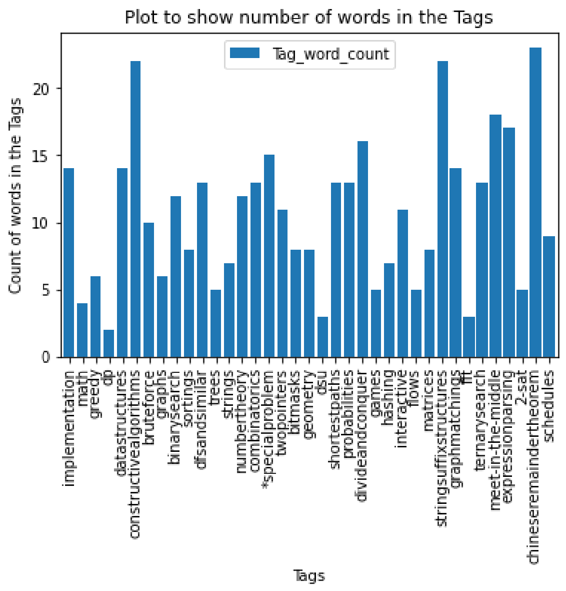


Fig. Plot to show number of words in the Tags

The below figure shows a histogram distribution of tags per problem statement.

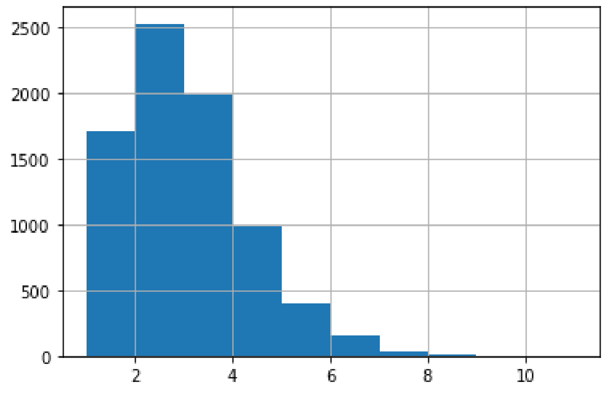
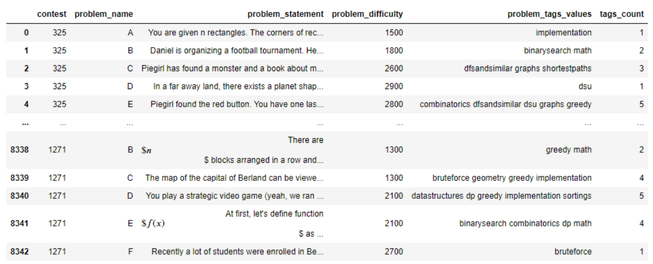


Fig. Distribution of tags per problem statement

After data cleaning and prepossessing, the final dataset got for training and testing.



	contest	problem_name	problem_statement	problem_difficulty	problem_tags_values	tags_count
0	325	A	You are given n rectangles. The corners of rec...	1500	implementation	1
1	325	B	Daniel is organizing a football tournament. He...	1800	binarysearch math	2
2	325	C	Piegirl has found a monster and a book about m...	2600	dfsandsimilar graphs shortestpaths	3
3	325	D	In a far away land, there exists a planet shap...	2900	dsu	1
4	325	E	Piegirl found the red button. You have one las...	2800	combinatorics dfsandsimilar dsu graphs greedy	5
...	...	...	...	...	...	...
8338	1271	B	There are $n $ blocks arranged in a row and...	1300	greedy math	2
8339	1271	C	The map of the capital of Berland can be viewe...	1300	bruteforce geometry greedy implementation	4
8340	1271	D	You play a strategic video game (yeah, we ran ...	2100	datastructures dp greedy implementation sortings	5
8341	1271	E	At first, let's define function $f(x) $ as ...	2100	binarysearch combinatorics dp math	4
8342	1271	F	Recently a lot of students were enrolled in Be...	2700	bruteforce	1

Fig. Final dataset

B. Machine Learning Algorithms

We used machine learning and deep learning techniques to classify the programming problem statements into the target tags.

1. Logistic Regression using OnevsRest Classifier

We have used Logistic Regression using Onevs-Rest Classifier. Logistic Regression is a supervised classification algorithm that only takes discrete values as input. We Weare using Multinomial Logistic Regression where the target variable can have three or more possible types which have no quantitative significance. One-vs-rest is a heuristic method for using binary classification algorithms for multi-class classification. It includes splitting the multi-class dataset into various binary classification problems. A binary classifier is then trained on all binary classification problems, making predictions using the most confident model. The calculated parameters are the F1 score which can be interpreted as the harmonic mean of the precision and recall. In a multi-label case, the average F1 score of each class with weighting depends on the average parameter.

2. RandomForestClassifier using OnevsRest Classifier

Random forests are supervised learning algorithms. It can be used for and regression and classification. Random forests create decision trees on randomly selected data samples, get a prediction from each tree, and select the best solution by means of voting. It also provides a pretty good indicator of the feature importance.Random forests have a variety of applications, including classification and feature selection.

3. Bi-LSTM (Bi-directional long short term memory)

The deep learning model that we use is a Bi Long-Short Term Memory (LSTM) network, which improves the traditional Recurrent Neural Networks (RNNs). Bi-LSTM is making of any neural network where sequence information flow in both directions backward (future to past) or forward(past to future). This makes a bi-LSTM different from the regular LSTM. With the regular LSTM, we can only make input flow in one direction, and it can be backward or forward. LSTM addresses the short-term memory challenge that RNNs have. LSTMs can preserve long-term memory, which is critical since we deal with a text classification task. When generating the labels, it is essential to safeguard the long-term dependencies in the problem statements. To obtain the final predicted labels, we use a relu activation function. We implement the LSTM architectures using PyTorch13, a Python-based open-source deep-learning library.

4. SGDClassifier using OnevsRest Classifier

We have also used SGDClassifier using OnevsRest Classifier. The class SGDClassifier implements a simple stochastic gradient descent learning method that supports different loss functions and penalties for classification. Decision boundary of an SGDClassifier trained with the hinge loss, equivalent to a linear SVM. SGDClassifier supports multi-class classification by connecting multiple binary classifiers in a “one versus all” (OVA) scheme. A binary classifier is learned for each of the K classes that distinguish between that and all other K1 classes.

C. Implementation Details

Training and Testing dataset

We have 7842 rows after pre-processing in our dataset. We split that data into training and testing data. We perform an 80-20 split, creating a training set of 6273 rows and 1569 rows of the testing test.

Train data dimensions are : (6273, 55516) Y : (6273, 37)
Test data dimensions are : (1569, 55516) Y: (1569, 37)

Metrics Metric used for evaluating models.

Precision
Precision is a measure of how many of the positive predictions made are correct (true positives). The formula for it is:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall
Recall is a measure of how many of the positive cases the classifier correctly predicted, over all the positive cases in the data.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1-Score
F1-Score is a measure combining both precision and recall, and it is generally described as the harmonic mean of the two.

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

Hamming loss
The fraction of the wrong labels to the total number of labels.For the binary case (imbalanced or not), HL=1-Accuracy. When we consider the multi-label use case, we should decide how to extend accuracy to this case. The method chosen in the hamming loss was to give each label equal weigh

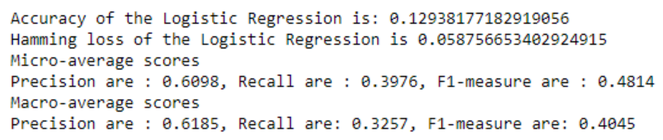
Accuracy
Accuracy is a ratio of the correctly predicted classifications (both True Positives + True Negatives) to the total test dataset.

Multilabel Confusion Matrix
In multilabel confusion matrix , the count of true negatives is , false negatives is , true positives is and false positives is.

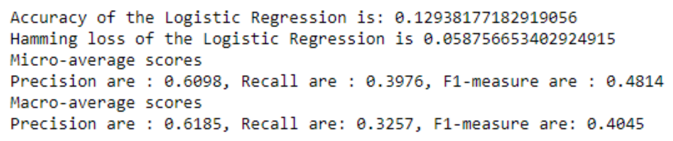
Multiclass data will be treated as if binarized under a one-vs-rest transformation. Returned confusion matrices will be in the order of sorted unique labels in the union of (y-true, y-pred). This has been employed for both Random Forest Classifier ,SVM and Logistic Regression. We can view the TP,FP,TN,FN for every label for the Machine Learning algorithms.

1. Logistic Regression with OnevsRestClassifier Model

After training with Logistic Regression using OnevsRest Classifier with a solver as ’sag’. We got micro a f1 score of 0.48.

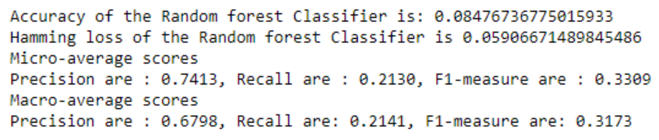

```
Accuracy of the Logistic Regression is: 0.12938177182919056
Hamming loss of the Logistic Regression is 0.058756653402924915
Micro-average scores
Precision are : 0.6098, Recall are : 0.3976, F1-measure are : 0.4814
Macro-average scores
Precision are : 0.6185, Recall are: 0.3257, F1-measure are: 0.4045
```

We used hyperparameters tuning for OneVsRestClassifie. We used learning rate 1.2, penalty as l2, and solver as liblinear for Logistic Regression. As this is a multilabel classification problem, we need to look into the micro f1 score, which was 0.48. After hyper-parameter tuning for logistic regression, there was not much significant difference.


```
Accuracy of the Logistic Regression is: 0.12938177182919056
Hamming loss of the Logistic Regression is 0.058756653402924915
Micro-average scores
Precision are : 0.6098, Recall are : 0.3976, F1-measure are : 0.4814
Macro-average scores
Precision are : 0.6185, Recall are: 0.3257, F1-measure are: 0.4045
```

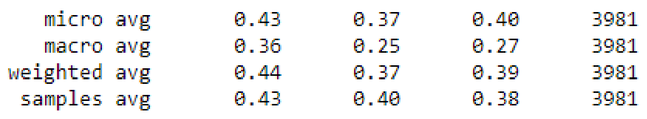
2. RandomForestClassifier using OnevsRest Classifier

After training with RandomForestClassifier with OneVsRestClassifie and we got 0.33 as an f1 micro score. RandomForestClassifier does not perform well with our data. RandomForestClassifier is lower than SGDClassifier and LogisticRegression for our data.

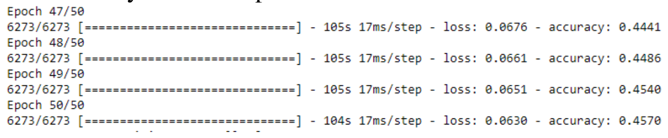

```
Accuracy of the Random forest Classifier is: 0.08476736775015933
Hamming loss of the Random forest Classifier is 0.05906671489845486
Micro-average scores
Precision are : 0.7413, Recall are : 0.2130, F1-measure are : 0.3309
Macro-average scores
Precision are : 0.6798, Recall are: 0.2141, F1-measure are: 0.3173
```

3. Bi-LSTM (Bi-directional long short term memory)

After training with Bi LSTM model with Dense layer of 20 with activation as relu, Dense layer of 37 with activation as sigmoid, learning rate as 0.0001, and loss as binary crossentropy, we got an accuracy of 0.45 and f1 score of 0.37.

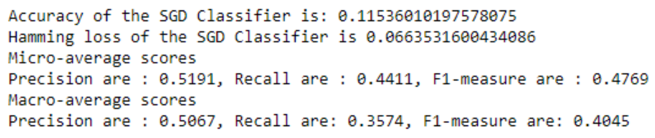

```
   micro avg       0.43      0.37      0.40      3981
   macro avg       0.36      0.25      0.27      3981
weighted avg       0.44      0.37      0.39      3981
 samples avg       0.43      0.40      0.38      3981
```

Accuracy after 50 epoch

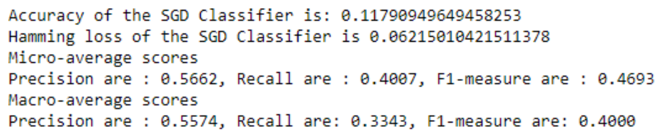

```
Epoch 47/50
6273/6273 [==============================] - 105s 17ms/step - loss: 0.0676 - accuracy: 0.4441
Epoch 48/50
6273/6273 [==============================] - 105s 17ms/step - loss: 0.0661 - accuracy: 0.4486
Epoch 49/50
6273/6273 [==============================] - 105s 17ms/step - loss: 0.0651 - accuracy: 0.4540
Epoch 50/50
6273/6273 [==============================] - 104s 17ms/step - loss: 0.0630 - accuracy: 0.4570
```

4. SGDClassifier using OnevsRest Classifier Model

After training with SGDClassifier using OnevsRest Classifier with a loss as 'hinge', alpha as 0.00001, and penalty as 'l1'. we got 0.47 as f1 micro score.

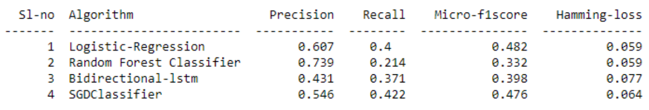

```
Accuracy of the SGD Classifier is: 0.11536010197578075
Hamming loss of the SGD Classifier is 0.0663531600434086
Micro-average scores
Precision are : 0.5191, Recall are : 0.4411, F1-measure are : 0.4769
Macro-average scores
Precision are : 0.5067, Recall are: 0.3574, F1-measure are: 0.4045
```

Then We used hyperparameters tuning for OneVsRestClassifie. Then we used loss as 'hinge', alpha as 0.0001, penalty as 'l2', for SGDClassifier. We got an f1 micro score of 0.46. After hyper-parameter tuning for SGDClassifier, there was not much significant difference.


```
Accuracy of the SGD Classifier is: 0.11790949649458253
Hamming loss of the SGD Classifier is 0.06215010421511378
Micro-average scores
Precision are : 0.5662, Recall are : 0.4007, F1-measure are : 0.4693
Macro-average scores
Precision are : 0.5574, Recall are: 0.3343, F1-measure are: 0.4000
```

IV. Comparison

The below figure shows performance metrics of the different algorithms



Sl-no	Algorithm	Precision	Recall	Micro-f1score	Hamming-loss
1	Logistic-Regression	0.607	0.4	0.482	0.059
2	Random Forest Classifier	0.739	0.214	0.332	0.059
3	Bidirectional-lstm	0.431	0.371	0.398	0.077
4	SGDClassifier	0.546	0.422	0.476	0.064

Table: Model results on test set.

From the above table we can see that Logistic Regression has the best f1-score compared to Random forest Regressor and Bilstm . This may be due to small dataset size and also over fitting. Techniques like oversampling or under sampling can be applied to make the dataset with various tags more balanced. As the dataset is small , machine learning model is performing better than deep learning model.

V. Future Directions

This research work focuses on predicting tags based on the textual description of programming problem statements. For improving model accuracy, we can add more data to the dataset. We can perform more data cleaning to remove irrelevant words and characters so that the model can be trained more efficiently. We can perform hyper tuning with more parameters and different values to improve the model's accuracy. Even though in our result we got better result with logistic regression. We can observe that the deep learning models achieve significantly higher performance by experimenting with regularization techniques such as drop-out and weight decay changes and changing parameters. The deep learning model performs better on all the considered metrics. We can use other different deep learning models like LSTM with Word2Vec.

VI. Conclusion

In this work we investigated how machine learning models and deep learning techniques perform in a novel multi-class multi-label text classification problem. Our findings show that deep learning approaches significantly outperform traditional widely accepted IR techniques like tf-idf. Moreover, we experimented with different combinations of text representations and neural network architectures, finding Tokenizer + LSTM as the option that yields the best performance. Finally, we were able to experience first hand the challenges and issues arising when having a limited amount of data, an issue that is common in the deep learning literature. Regarding possible future research directions, the need to collect more data for this task is of crucial importance. This could be done by either gathering data from more programming challenges websites or slightly different domains that can provide similar data samples. Another way to get more training data is by artificially augmenting the dataset either by the use of synonyms or adversarial networks. The availability of more training data will then allow researchers to use more advanced text embeddings such as BERT and XLNet which are the current state-of-the-art in the NLP Field. Last but not least, different lines of work. we consider as baselines the tf-idf with Decision Tree, since it achieves better performance than Random Forest in terms of the Weighted Hamming Score and the Random Classifier. We can observe that the deep learning models achieve significantly higher performance compared to our baselines, performing better on all the considered metrics.

References

[1] Online programming challenges questions and related tags from the Codeforce platform. "https://www.kaggle.com/immortal3/codeforces-dataset." 2021

[2] Rahul Gupta. Data augmentation for low resource sentiment analysis using generative adversarial networks. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7380–7384. IEEE, 2019.

[3] Ke-Wei Huang and Zhuolun Li. A multilabel text classification algorithm for labeling risk factors in sec form

10-k. ACM Transactions on Management Information Systems (TMIS), 2(3):18, 2011.

[4] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. In Proceedings of the ECML/PKDD, volume 18, page 5, 2008.

[5] Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. page 9.

[6] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 115–124. ACM, 2017.

[7] Andrew McCallum. Multi-label text classification with a mixture model trained by em. In AAAI workshop on Text Learning, pages 1–7, 1999.

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jersey Dean. Ecient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs], January 2013. arXiv: 1301.3781.

[9]https://aclanthology.org/Y18-1061.pdf

[10]https://arxiv.org/pdf/1802.00889.pdf

[11]https://scikit-learn.org/stable/modules/multiclass.html

[12]https://www.geeksforgeeks.org/an-introduction-to-multilabel-classification/

[13]https://www.pyimagesearch.com/2018/05/07/multi-label-classification-with-keras/

[14]https://builtin.com/data-science/random-forest-algorithm

[15]https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/