CSCI: B-535 DATA MINING

Predictive Modeling for Multi-Label Tagging in Algorithmic Challenges

~ Meet Palod, Aayush Jaiswal

Abstract

This project seeks to advance user interaction with CodeForce, a hub for algorithmic contests, by devising a predictive framework capable of assigning multiple descriptive labels to each posted problem based on its textual content. The study sets out to meticulously parse problem descriptions, fine-tune text processing methods, and critically assess several computational models, including logistic regression and random forest classifiers, as well as a BiLSTM network with an embedding feature. The end goal is to refine the tagging mechanism to significantly uplift the accuracy, exhaustiveness, and reduction of misclassification in the system to aid users in pinpointing relevant problems that match their proficiency and search requirements.

Introduction

The domain of competitive programming has long recognized the importance of effectively categorizing problem sets to enhance educational outcomes and user satisfaction. On platforms such as CodeForce, where the corpus of problems is extensive, the deployment of a predictive model that can accurately tag problems based on their textual descriptions is of paramount importance. This research delineates the development of a multi-label classification model that employs natural language processing and machine learning techniques to automate the assignment of tags to programming challenges. Drawing upon existing research that underscores the complexity of multi-label classification tasks (Zhang & Zhou, 2014; Tsoumakas et al., 2009), this study aims to bridge the gap between problem statements and their appropriate categorical tags, thereby refining the navigation and searchability of problems on the CodeForce platform.

Dataset

Our research project relied on a comprehensive dataset from Kaggle, originally obtained from the Codeforce platform's online programming competitions. The dataset is organized into four columns, namely 'contest', 'problem name', 'problem statement', and 'problem tags'. The two primary columns in the dataset are 'problem statement' and 'problem tags', which are crucial for our research work. The 'problem statement' column contains all the challenge descriptions, while the 'problem tags' column lists all tags assigned to each problem in a comma-separated format. The dataset is extensive, covering a total of 8434 entries, providing us with substantial data to analyze and draw conclusions.

Data Pre-Processing

During the data preprocessing phase, we encountered various elements that had the potential to affect the model's training effectiveness adversely. These elements included HTML codes, Latex symbols utilized for mathematical expressions, stop words, newline characters, non-ASCII characters, digits, punctuation, and Unicode characters. We identified these elements as extraneous due to their lack of importance in the dataset. Therefore, we removed them to reduce the data's complexity and increase its efficiency. In addition, we converted all text to lowercase to ensure consistency throughout the dataset.

To employ advanced preprocessing techniques, we applied both Lemmatization and Stemming to the 'problem statement' text. While Lemmatization adjusted words to their base form or Lemma, considering the context, Stemming reduced words to their root form, helping to capture the essence of the derived terms.

To address data cleanliness, we removed null entries in the 'tags' column and split the 'problem difficulty' text from the tags, forming two new columns: 'problem-difficulty' and 'problem-tags-values'. Additionally, we confirmed that no duplicate data existed within the dataset through our preprocessing efforts.

Exploratory Data Analysis

After completing the preprocessing stage, we proceeded with an in-depth descriptive statistical analysis of the data, specifically focusing on the tags used to categorize the problem statements. Our research uncovered some interesting insights, including the average number of tags assigned to each problem statement, the highest and lowest number of tags used, and other relevant metrics. Our study found that the average number of tags per problem statement was between 2 to 3, and the range of tags assigned to each problem statement varied widely, with a maximum of 11 tags and a minimum of 1 tag per problem.

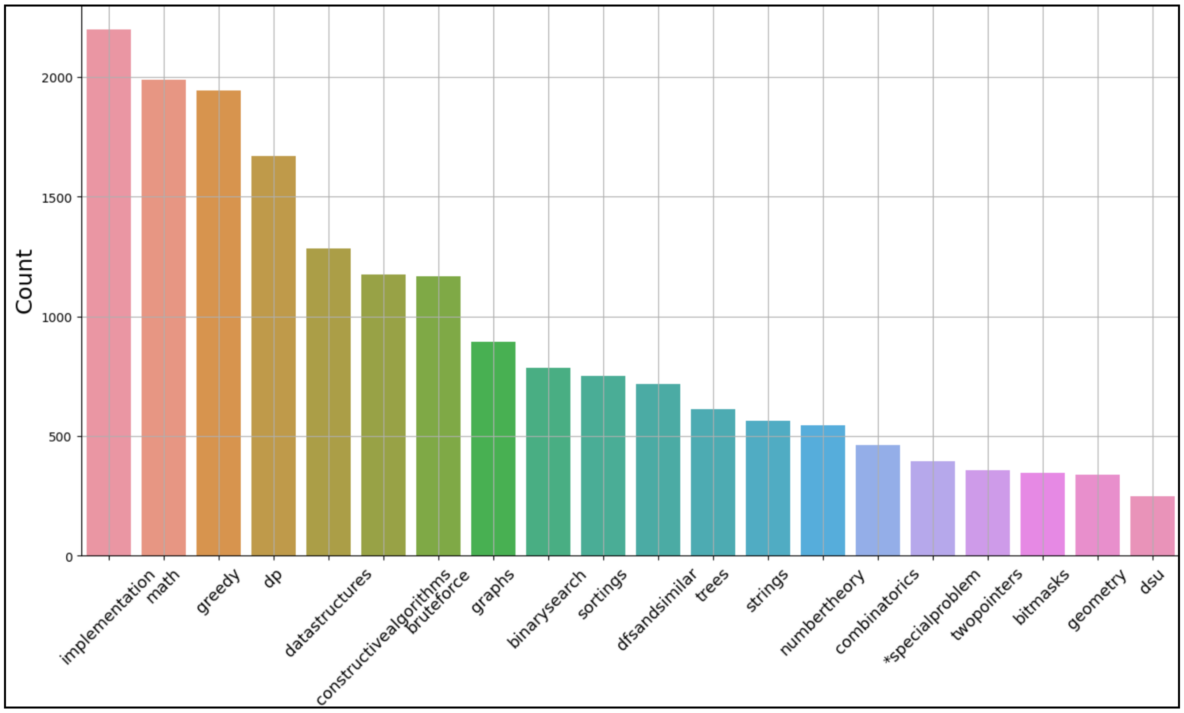


Fig. Top 20 Highest occurring Tags

As part of our analysis, we conducted a thorough examination of the problem tags using both uni-gram and multi-gram techniques. Our uni-gram analysis focused on single tags and revealed that they accounted for only 6% of the data. On the other hand, our analysis of multi-gram tags, which included two or more tags, constituted a significant 94% of the data.

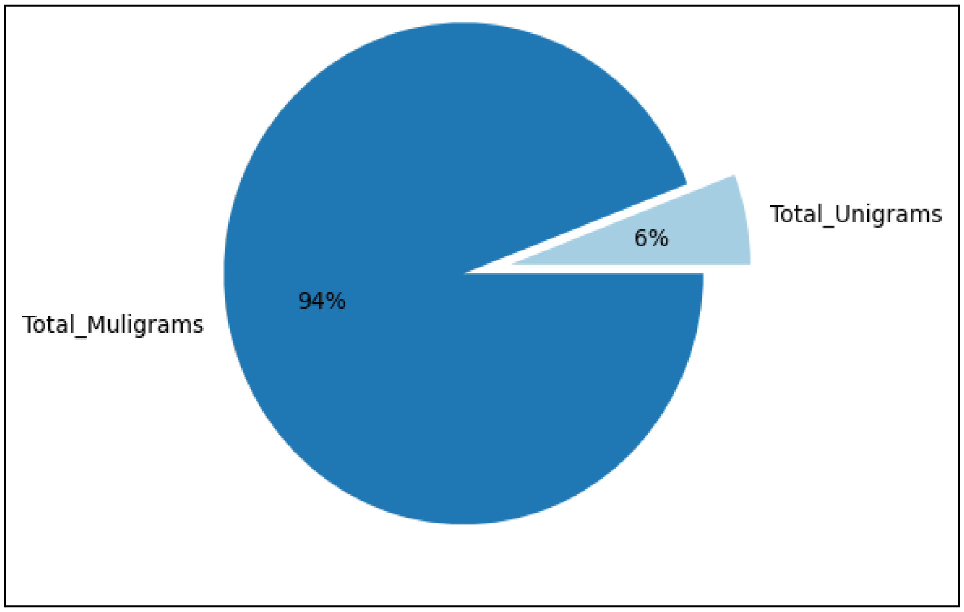


Fig. Uni-gram and Multi-gram Analysis

The below figure shows that the majority of the problems in the dataset have been tagged twice, with a nearly equal number of problems tagged thrice coming in second place. Problems with only one tag follow closely behind those with three. However, as the number of tags on a problem increases, the frequency of such problems decreases significantly. After five tags, there is a noticeable drop in the number of problems with that many tags, and very few problems have more than seven tags. The bars representing problems with 11 tags or more are so small that they indicate such instances are extremely rare in the dataset.

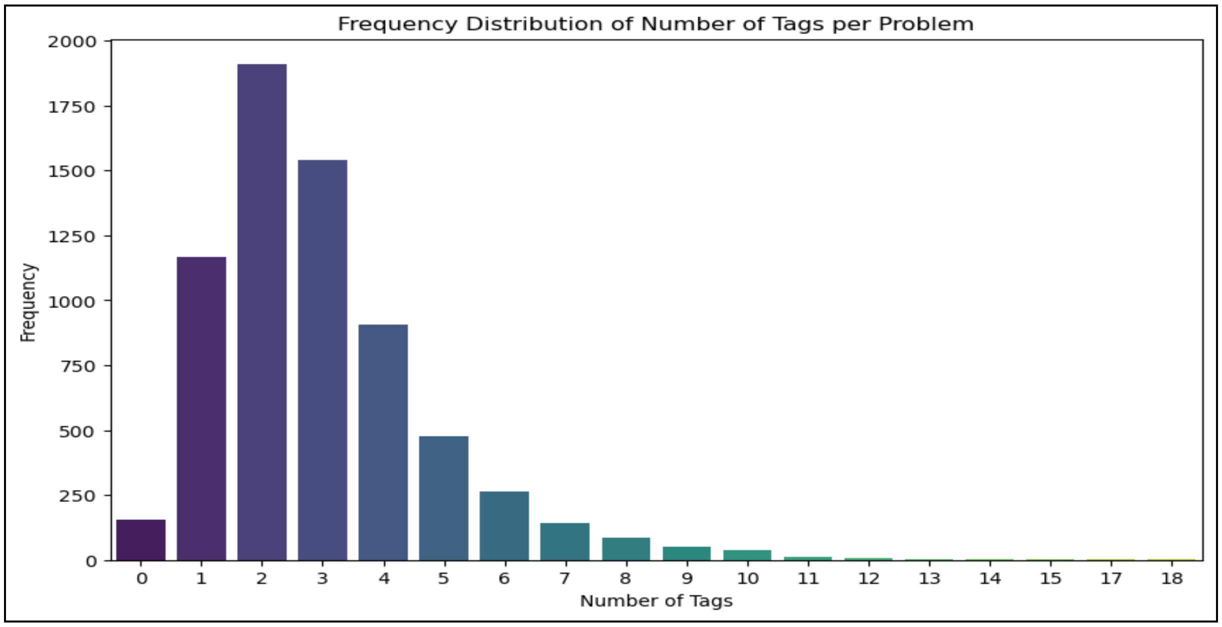


Fig: Distribution of Tags

Methods

Our project employs a mix of machine learning and deep learning approaches to categorize programming problem statements into their respective tags.

Logistic Regression with One-vs-Rest Classification

We applied Logistic Regression in conjunction with a One-vs-Rest classifier. This method is suited for situations where the target variable comprises three or more categories that have no inherent order. The One-vs-Rest strategy involves transforming the multi-class classification issue into multiple binary classification problems. For each binary problem, a model is trained and the one with the highest confidence prediction is chosen. The F1 score, a measure combining precision and recall, is used to evaluate the model, with multi-label cases considering a weighted average of each class's F1 score based on the specified average parameter.

Random Forest Classifier with One-vs-Rest Classification

Random Forest is another algorithm we used, which can perform both classification and regression tasks. It operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forest is known for its ability to rank the importance of variables and is also applied for feature selection.

Bi-Directional LSTM (Bi-LSTM)

For deep learning, we utilized a Bi-Directional Long Short-Term Memory (Bi-LSTM) network. Bi-LSTMs are an advancement over traditional Recurrent Neural Networks (RNNs), as they process data in both forward and backward directions, allowing the model to retain information from both the past and the future. This characteristic is especially beneficial for text classification tasks where understanding context is crucial. We use the rectified linear unit (ReLU) activation function in the output layer to predict the labels and implement our LSTM using PyTorch, an open-source deep-learning library in Python.

SGD Classifier with One-vs-Rest Classification

Finally, we incorporated the SGDClassifier with a One-vs-Rest approach. SGDClassifier is a linear classifier that implements stochastic gradient descent. It is flexible enough to fit different loss functions and penalties for classification. The technique is beneficial for large-scale and sparse machine-learning problems typically found in text classification and natural language processing. In the context of multi-class classification, it employs the "one versus all" method, wherein a separate binary classifier is trained for each class.

Metrics

Precision

Precision quantifies the accuracy of the positive predictions that the model makes. In other words, it's the ratio of correctly predicted positive observations to the total predicted positives. The formula for precision is:

$$\text{Precision} = TP / (TP + FP)$$

Recall

Recall, also known as sensitivity, measures the model's ability to correctly identify all relevant cases within the dataset. It is the ratio of correctly predicted positive observations to all observations in actual class - yes. The formula for the recall is:

$$\text{Recall} = TP / (TP + FN)$$

F1-Score
The F1-Score is a balanced measure that combines precision and recall. It is particularly useful when the class distribution is uneven. The F1-Score is the harmonic mean of precision and recall, providing a single score that balances both the concerns of precision and the recall in one number:

$$\text{F1-Score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

Hamming Loss
The Hamming loss is a metric for the multi-label classification that measures the fraction of labels that are incorrectly predicted, compared to the total number of labels. In binary classification, the Hamming loss is the complement of the accuracy. For multi-label classification, each label is weighted equally, and the Hamming loss is computed as:

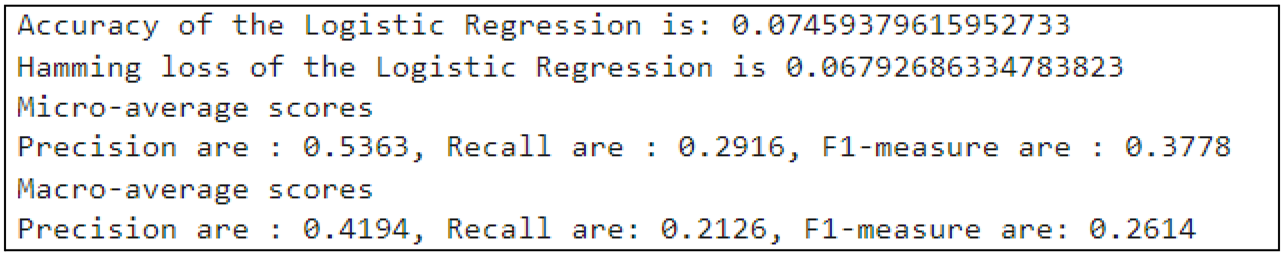
$$\text{Hamming Loss} = (1 / N) * \Sigma (\text{xor}(y_i, \hat{y}_i) / L)$$

In scenarios where there are multiple classes to be classified, a one-vs-rest approach is used. This approach distinguishes each class individually from all the others. Algorithms such as Random Forest Classifier, SVM, and Logistic Regression utilize this technique to evaluate their performance. The results are presented through confusion matrices that are ordered based on the unique labels from the combined set of actual and predicted values. These matrices provide a detailed view of the true positives, false positives, true negatives, and false negatives for each label. This offers insights into the precision and robustness of the machine learning models across the various classes they aim to predict.

Results

Logistic Regression with OnevsRestClassifier

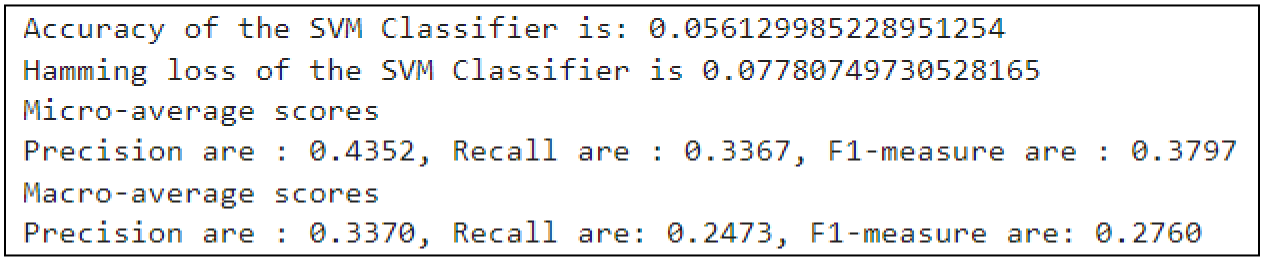
We used ‘sag’ solver and achieved a micro F1 score of 0.37. Despite hyperparameter tuning, which included adjusting the learning rate to 1.2 and employing 'liblinear' solver with an L2 penalty, the performance improvement was marginal.



```
Accuracy of the Logistic Regression is: 0.07459379615952733
Hamming loss of the Logistic Regression is 0.06792686334783823
Micro-average scores
Precision are : 0.5363, Recall are : 0.2916, F1-measure are : 0.3778
Macro-average scores
Precision are : 0.4194, Recall are: 0.2126, F1-measure are: 0.2614
```

SGD Classifier with One-vs-Rest Classification

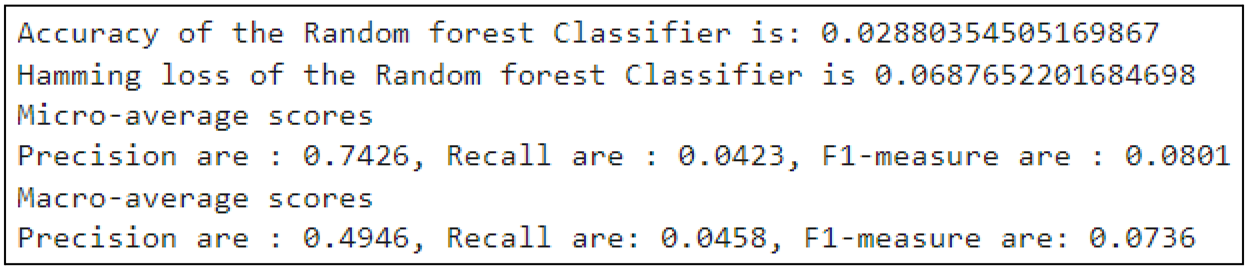
We trained the classifier using ‘hinge’ loss, and L1 penalty, later tuned to an L2 penalty and different alpha value, which resulted in a micro F1 score of 0.37, which was not significantly improved from the untuned version.



```
Accuracy of the SVM Classifier is: 0.05612998522895l254
Hamming loss of the SVM Classifier is 0.07780749730528165
Micro-average scores
Precision are : 0.4352, Recall are : 0.3367, F1-measure are : 0.3797
Macro-average scores
Precision are : 0.3370, Recall are: 0.2473, F1-measure are: 0.2760
```

Random Forest Classifier with One-vs-Rest Classification

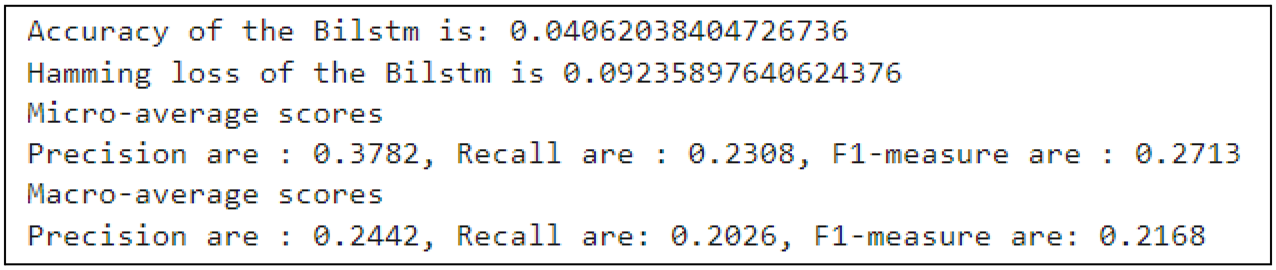
The model underperformed on our dataset with an F1 micro score of 0.08, falling short of both SGDClassifier and Logistic Regression.



```
Accuracy of the Random forest Classifier is: 0.02880354505169867
Hamming loss of the Random forest Classifier is 0.0687652201684698
Micro-average scores
Precision are : 0.7426, Recall are : 0.0423, F1-measure are : 0.0801
Macro-average scores
Precision are : 0.4946, Recall are: 0.0458, F1-measure are: 0.0736
```

Bi-Directional LSTM (Bi-LSTM)

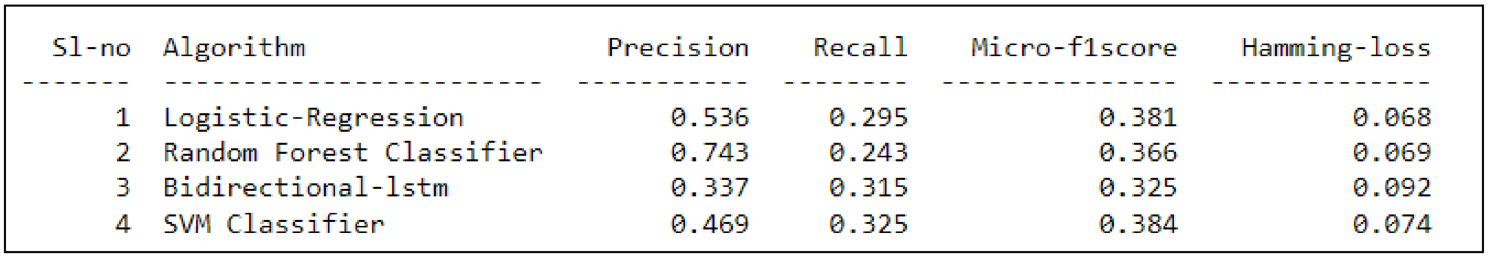
The Bi-LSTM model with a final sigmoid activation layer yielded an F1 score of 0.27 after 40 epochs.



```
Accuracy of the Bilstm is: 0.04062038404726736
Hamming loss of the Bilstm is 0.09235897640624376
Micro-average scores
Precision are : 0.3782, Recall are : 0.2308, F1-measure are : 0.2713
Macro-average scores
Precision are : 0.2442, Recall are: 0.2026, F1-measure are: 0.2168
```

Comparison

During our analysis of machine learning models for multi-label text classification, we found that OnevsRestClassifier using 'sag' solver with Logistic Regression performed the best, achieving a micro F1 score of 0.38 and a hamming loss of 0.068. Despite hyperparameter tuning, which included adjusting the learning rate to 1.2 and employing 'liblinear' solver with an L2 penalty, the improvement in performance was minimal. RandomForestClassifier, when adapted to OnevsRest, underperformed in our dataset with an F1 micro score of 0.36, falling short of both SGDClassifier and Logistic Regression. For the deep learning approach, a Bi-LSTM model with a final sigmoid activation layer yielded an F1 score of 0.325 after 40 epochs. Similarly, SGDClassifier using OnevsRest with 'hinge' loss, and L1 penalty, later tuned to an L2 penalty and different alpha value, resulted in a micro F1 score of 0.38, which was comparable to Logistic Regression but had a worse hamming loss.



```
  Sl-no  Algorithm                    Precision    Recall    Micro-f1score    Hamming-loss
-------  -------------------------  -----------  --------  ---------------  --------------
      1  Logistic-Regression              0.536     0.295            0.381           0.068
      2  Random Forest Classifier         0.743     0.243            0.366           0.069
      3  Bidirectional-lstm               0.337     0.315            0.325           0.092
      4  SVM Classifier                   0.469     0.325            0.384           0.074
```

Fig. Model Results

Conclusion

Our investigation into multi-class multi-label text classification revealed that machine learning models, particularly Logistic Regression, showed strong performance on our dataset, potentially due to its size and diversity. However, deep learning techniques demonstrated the capacity to outperform traditional Information Retrieval (IR) methods like tf-idf when sufficient data was available. Experimentation with various text representations and neural network architectures led us to discover that a combination of a Tokenizer with an LSTM network yielded the most favorable results. The challenge of limited data, a well-known issue in deep learning research, became evident during our study, highlighting the need for more robust datasets. Future research could benefit from collecting more data from various programming challenge websites or related domains, or by employing methods like data augmentation with synonyms or generative adversarial networks to expand the dataset. With more data, there is the potential to leverage advanced text embeddings such as BERT and XLNet, which represent the forefront of NLP technology. Additionally, while deep learning models outperformed our baseline tf-idf with the Decision Tree model in terms of Weighted Hamming Score and Random Classifier, they showed a significantly higher performance across all metrics evaluated. This suggests a promising direction for future work in employing and refining deep learning models for text classification tasks.

References

[1] Zhang, M. L., & Zhou, Z. H. (2014). A Review on Multi-Label Learning Algorithms. IEEE Transactions on Knowledge and Data Engineering, 26(8), 1819-1837.

[2] Tsoumakas, G., Katakis, I., & Vlahavas, I. (2009). Mining Multi-label Data. In Data Mining and Knowledge Discovery Handbook (pp. 667-685). Springer US.

[3] Institution Name/CodeForce [Data set]. Kaggle. https://www.kaggle.com/immortal3/codeforces-dataset

[4] Gupta, R. (2019). 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) (pp. 7380-7384). IEEE.

[5] Huang, K.-W., & Li, Z. (2011). A Multilabel Text Classification Algorithm for Labeling Risk Factors in SEC Form 10-K. ACM Transactions on Management Information Systems (TMIS), 2(3), Article 18.

[6] Katakis, I., Tsoumakas, G., & Vlahavas, I. (2008). Multilabel Text Classification for Automated Tag Suggestion. Proceedings of the ECML/PKDD, 18, 5.

[7] Le and Tomas Mikolov. Distributed Representations of Sentences and Documents.

[8] Liu, J., Chang, W.-C., Wu, Y., & Yang, Y. (2017). Deep Learning for Extreme Multi-label Text Classification. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 115-124). ACM.

[9] McCallum, A. (1999). Multi-label Text Classification with a Mixture Model Trained by EM. In AAAI workshop on Text Learning (pp. 1-7).

[10] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs].