Sequence Classification for Abbreviation and Long Form Detection in Biomedical Literature

Table of Contents

1 Extensive analysis of the dataset

In this analysis, we delved into various aspects of the dataset to gain insights into its underlying characteristics.

1.1. Distribution of Token Lengths: The distribution of token lengths in the dataset reveals important statistics. With a mean token length of 37.31 and a median of 33.00, the majority of tokens fall within a range of 2 to 323 characters. This suggests a diverse range of token lengths present in the dataset.

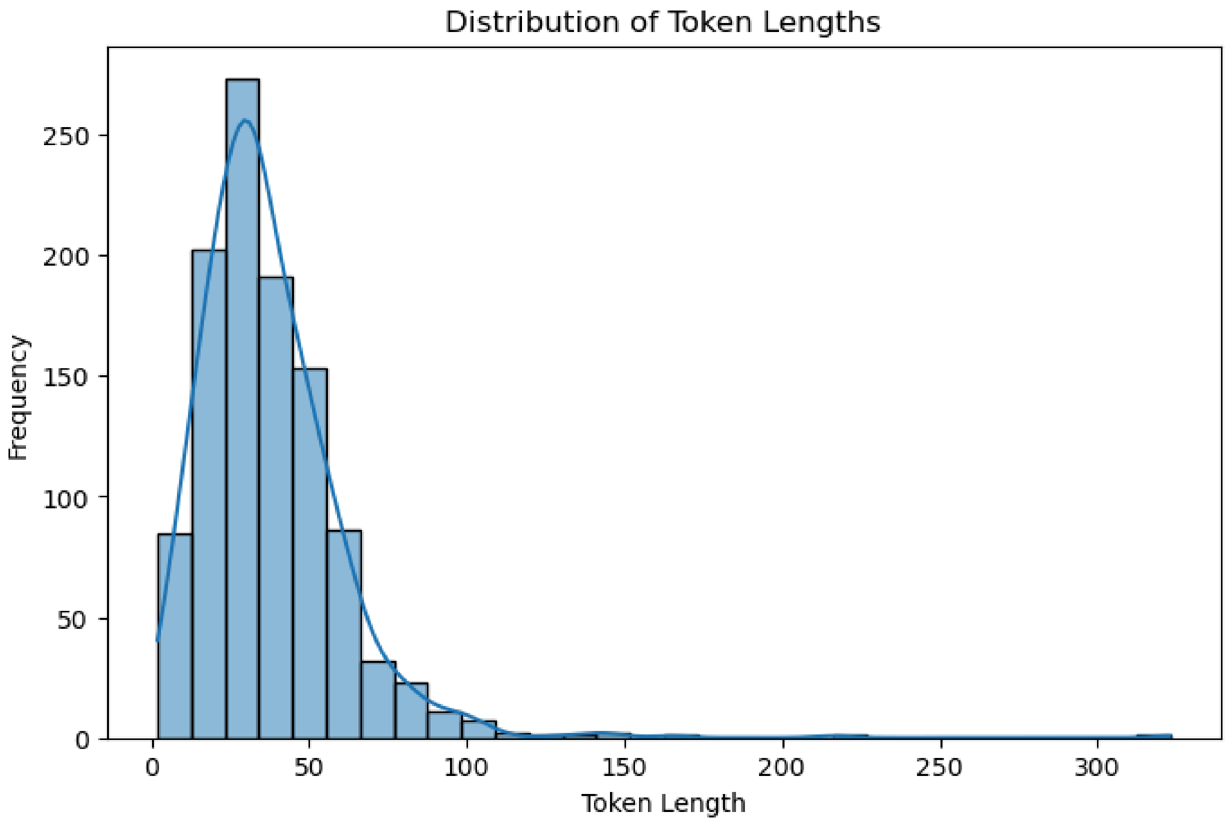

Figure 1.Distribution of Token Lengths

1.2. Frequency Distribution of POS Tags: Exploring the frequency distribution of part-of-speech (POS) tags provides valuable insights into the linguistic structure of the dataset. Although no specific observations were provided, the visual representation of POS tag frequencies aids in understanding the distribution of different linguistic categories within the dataset.

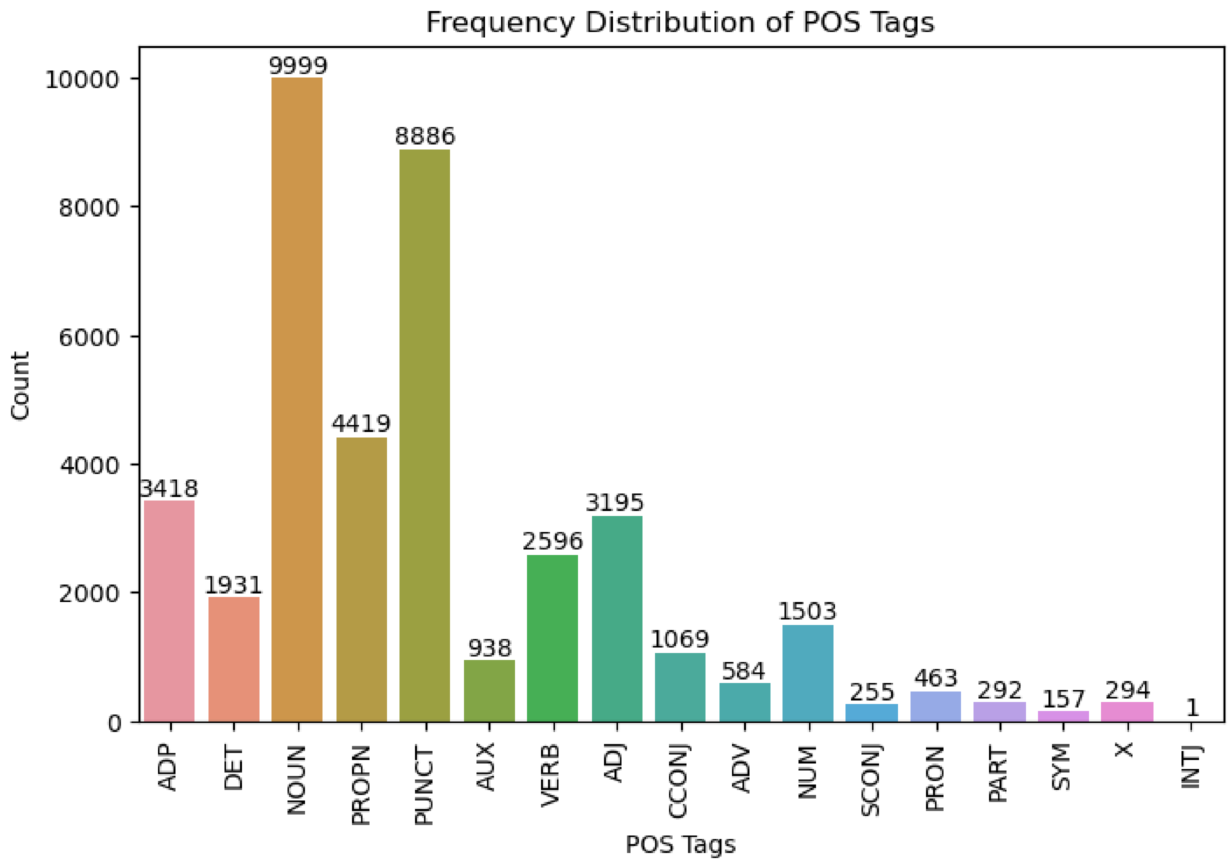

Figure 2.Frequency Distribution of POS Tags

1.3. Correlation between Token Length and Number of POS Tags: The correlation analysis between token length and the number of POS tags reveals a perfect correlation coefficient of

1.00. This implies a strong linear relationship between token length and the number of POS tags associated with each token. Such a finding sheds light on the linguistic complexity of the dataset, indicating potential patterns in tokenization and part-of-speech annotation.

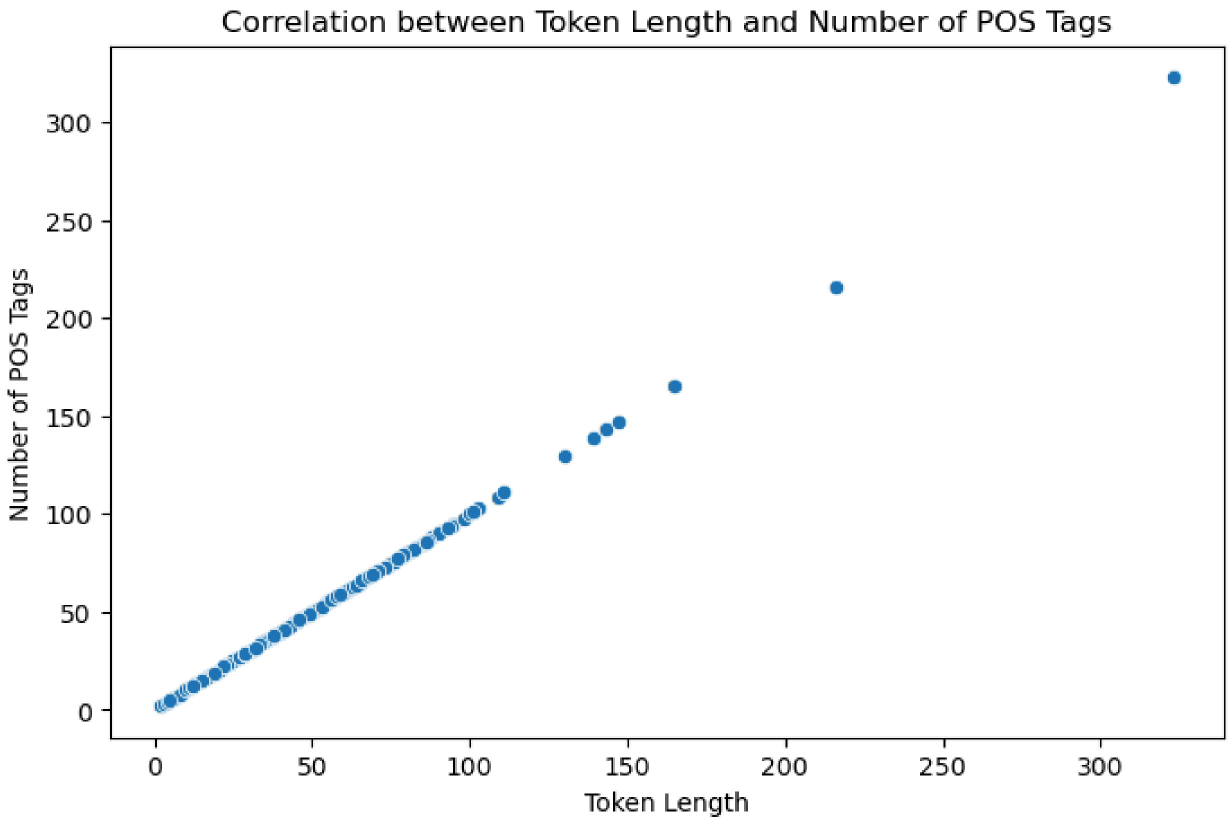

Figure 3. Correlation between Token Length and Number of POS Tags

1.4. Class Distribution of NER Tags: Lastly, examining the class distribution of named entity recognition (NER) tags offers insights into the distribution of different named entity classes in the dataset. Although specific observations were not provided, the visualization of NER tag frequencies provides a comprehensive overview of the distribution of named entities across various categories.

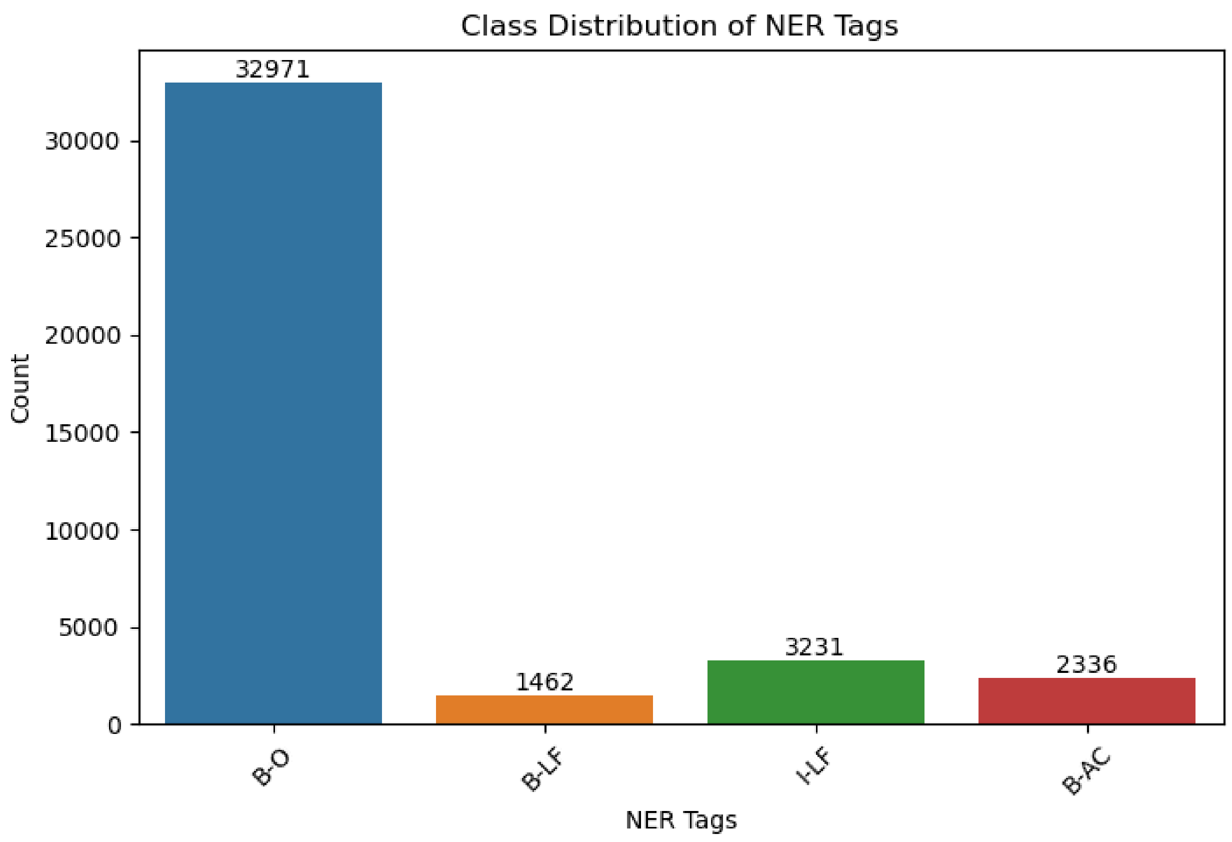

Figure 4.Class Distribution of NER Tags

2 Experiments:

2.1. Data Pre-processing Techniques: In preprocessing text data, tokenization is a crucial step where the text is split into individual tokens, such as words or subwords. One common approach is to tokenize text into unigrams, which are single words, or even further into n-grams, which are contiguous sequences of n tokens. The choice of whether to use n-grams depends on the specific task and the characteristics of the dataset. For instance, n-grams can capture contextual information and improve the performance of models in tasks like sentiment analysis or named entity recognition. However, they also increase the dimensionality of the data and may lead to sparse representations, especially with large values of n. In the case of pre-trained language models, such as BERT or GPT, they come with their own tokenizers optimized for their architectures. These tokenizers usually perform subword tokenization, breaking down words into smaller units to handle out-of-vocabulary words and improve generalization. Figure 5 illustrating the tokenization process, such as the distribution of token lengths or the frequency of n-grams, can be included in the preprocessing section of a research paper or report, typically after describing the tokenization techniques used.

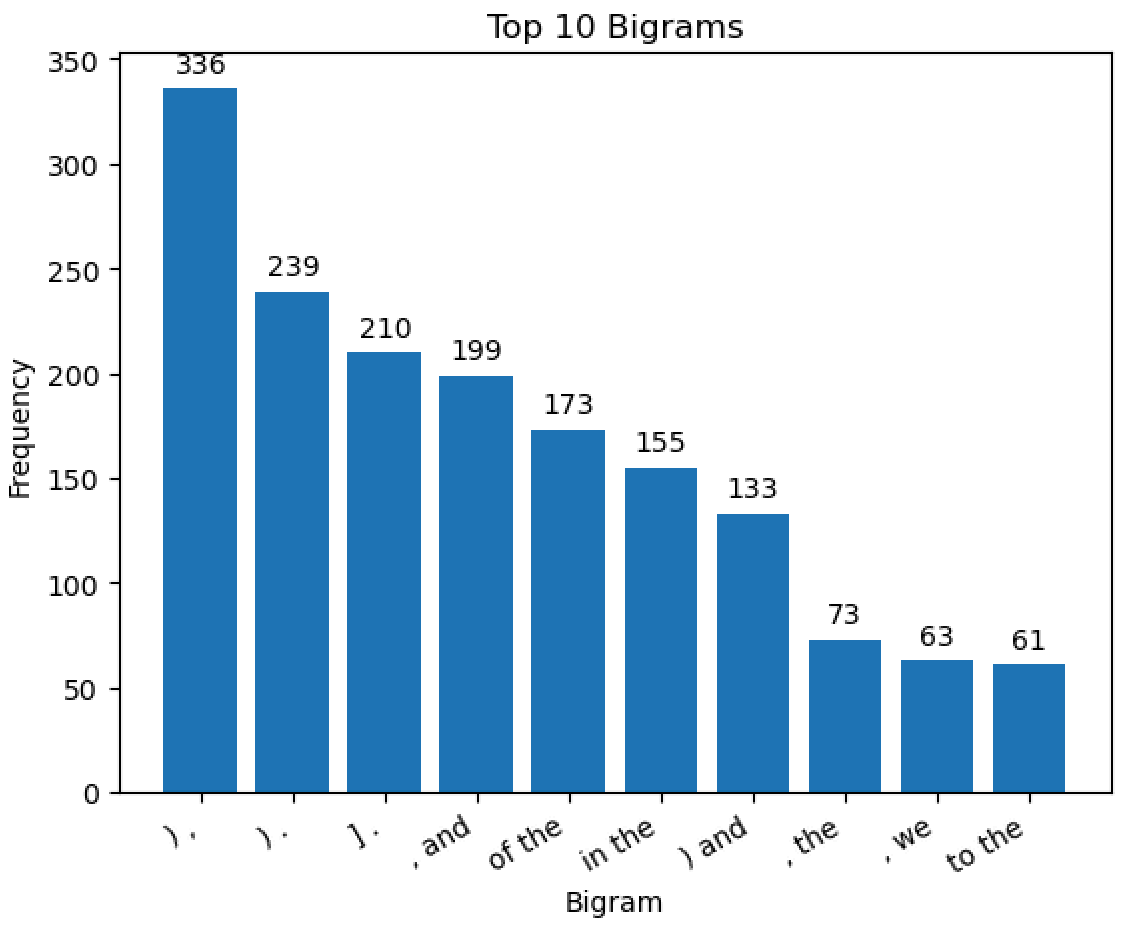

Figure 5.Top 10 Bigrams

2.2. NLP Algorithms

I experimented with two algorithms, their advantages and disadvantages are given below:

Support Vector Machines (SVM):

- *Advantages*:
    1. Robust overfitting.
    2. Effective in high-dimensional spaces.
    3. Versatility with different kernel functions.
- *Disadvantages*:
    1. Sensitive to parameter tuning.
    2. Difficulty with large datasets.
    3. Complexity in multiclass classification.

Recurrent Neural Networks (RNN):

- *Advantages*:
    1. Suitable for sequential data.
    2. Flexibility with input length.
    3. Capturing long-term dependencies.
- *Disadvantages*:
    1. Vanishing gradient problem.
    2. Computationally intensive.
    3. Difficulty with memory retention.

2.3. Text Encoding

TF-IDF is chosen for its simplicity, efficiency, and interpretability. It's suitable for tasks like text classification, where the importance of individual words in distinguishing between classes is crucial. TF-IDF can handle large datasets efficiently and provides straightforward features for training models.

Word2Vec is chosen for its ability to capture semantic meaning and relationships between words. It's suitable for tasks requiring deeper understanding of text semantics, such as sentiment analysis or semantic similarity. While training Word2Vec models can be computationally expensive, pre-trained models are available, reducing the need for extensive computational resources.

3 Experimental Results and Analysis

Table 1. Comparison of experimental results

Experiment	**F1-Score**
Uncased BERT	0.448
Uncased BERT + Tokeniser	0.924
RNN + Tokeniser	0.555
RNN + Hyperparameter Tuning	0.342

3.1 Experiment 1 - BERT-Based Model with Fine-Tuning

The first experiment fine-tunes a pre-trained BERT-based model on the PLOD-CW dataset for the task of abbreviation and long-form detection. The model was trained for 10 epochs with a batch size of 16 and a learning rate of 2e-5. Both training and validation losses show a trend of decreasing loss values across the epochs, making it a clear indication that the model is learning to fit the data well with time. Validation loss may be slightly higher than the training loss, suggesting that the model may overfit to some extent. The model's performance abbreviation and long form detection is promising, with an F1-score of 0.448 indicating its ability to identify and classify these entities in biomedical literature. However, the slightly higher validation loss suggests that the model may struggle to generalize to unseen data, due to the complexity and variability of abbreviations and their corresponding long forms in the dataset.

The training loss curve is a plot of the learning progress of the model across epochs. From the graph of the loss function shown in Figure 4, one can easily observe that the value of training loss is decreasing over epochs, which demonstrates that learning is happening and the model can generalize the training data. The loss training curve shows a decreasing loss as the number of epochs grows. That is usually a good sign for the model learning something from the training data. There is no increase in loss, which hints towards overfitting. In general, although the training appears to proceed correctly, there are adjustments that need to be made to address an apparent bias toward Class 3 in the model's predictions.

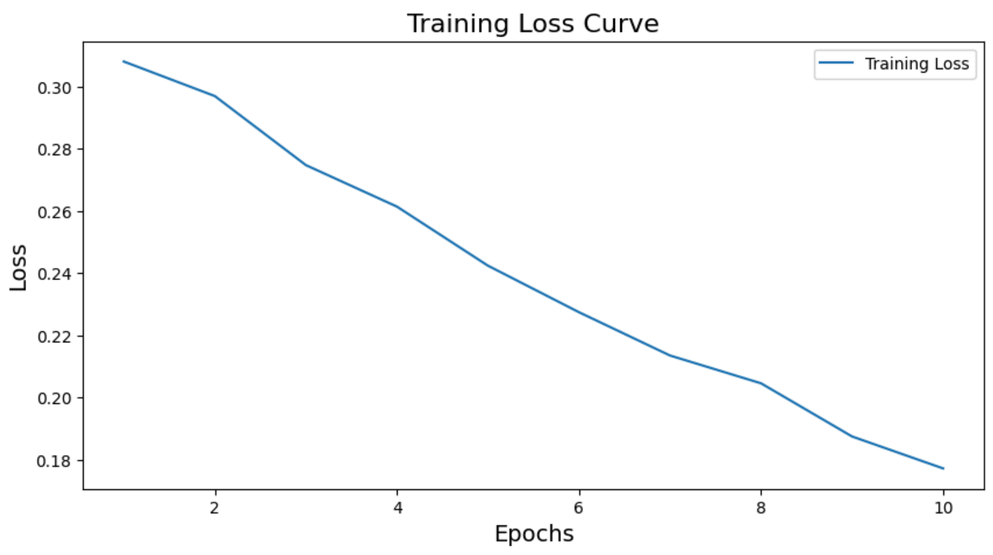

Figure 6. Training loss curve

The confusion matrix in Figure 5 highlights the model's strengths in identifying the majority class 'O' (Outside), which is essential for accurately distinguishing between relevant and irrelevant tokens in the sequence. However, the model's weakness lies in differentiating between the 'B-AC' (beginning of abbreviation) and 'B-LF' (beginning of long form) classes, which is crucial for correctly identifying the start of abbreviations and their corresponding long forms. This limitation arises due to the inherent ambiguity and similarity between these classes, as well as the lack of examples in the training data.

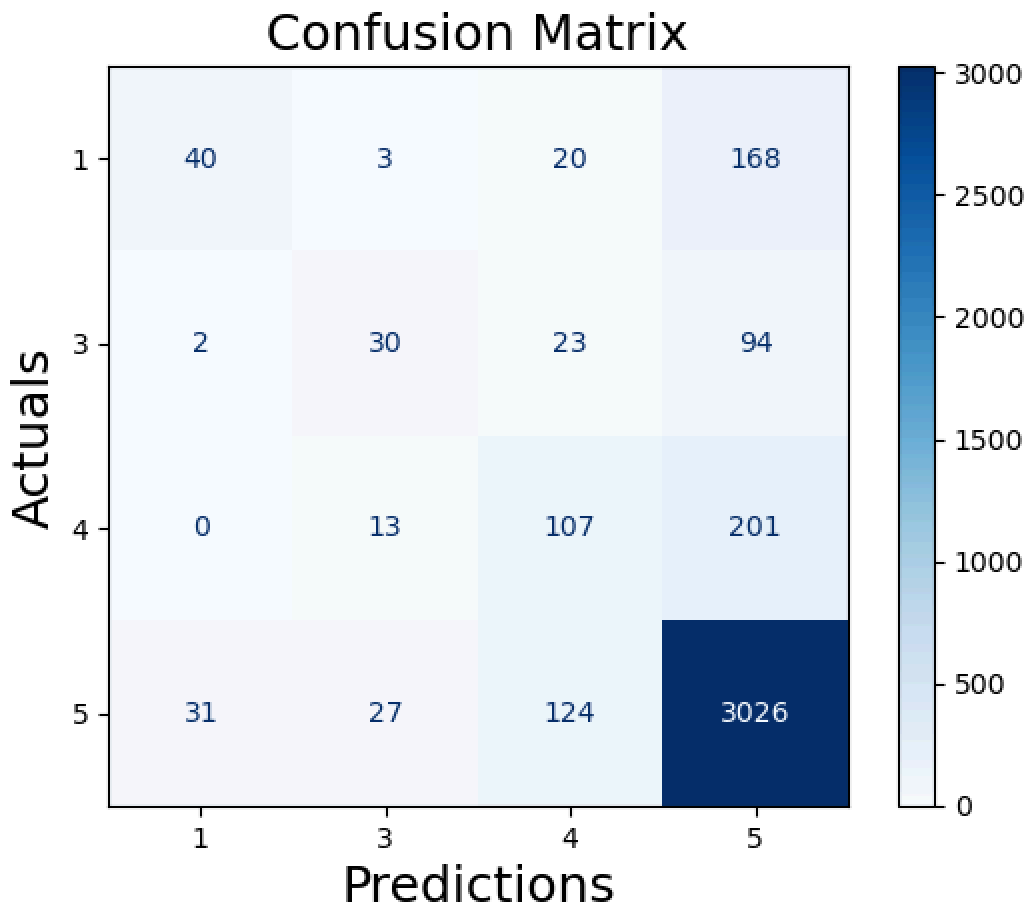

Figure 7. Confusion matrix for BERT based model with Fine tuning

To gain further insights into the model's performance, an error analysis was conducted on the validation set predictions. It appears the model performs well in identifying the 'O' (Outside) class, which is the majority class in the dataset. However, it struggles to accurately distinguish between the 'B-AC' (beginning of abbreviation) and 'B-LF' (beginning of long form) classes, suggesting the need for additional features or architectural modifications to better capture the differences between abbreviations and long forms.

3.2 Experiment 2 - BERT-Based Model with Tokeniser

The second experiment is designed to observe the impacts brought about by different hyperparameter settings. The key hyperparameters include learning rates (1e-5, 2e-5, 5e-5) and number of training epochs (3, 5, 10), while all other hyperparameters are set the same. The best result with the F1-score on the validation set was 2e-5 learning rates, and 10 training epochs. It showed a slight performance decrease when learning rates were adjusted up to 5e-5, and it significantly affected the F1-score negatively with both the number of training epochs set to 3 or 5. This increase in the F1-score from 0.448 in Experiment 1 to 0.924—clearly exhibits the importance of the tokenizer for BERT-based models. Good ability of the tokenizer to preprocess and encode the input sequences enhances the model's ability to learn from and generalize over the training data well, resulting in better abbreviation and long-form detection.

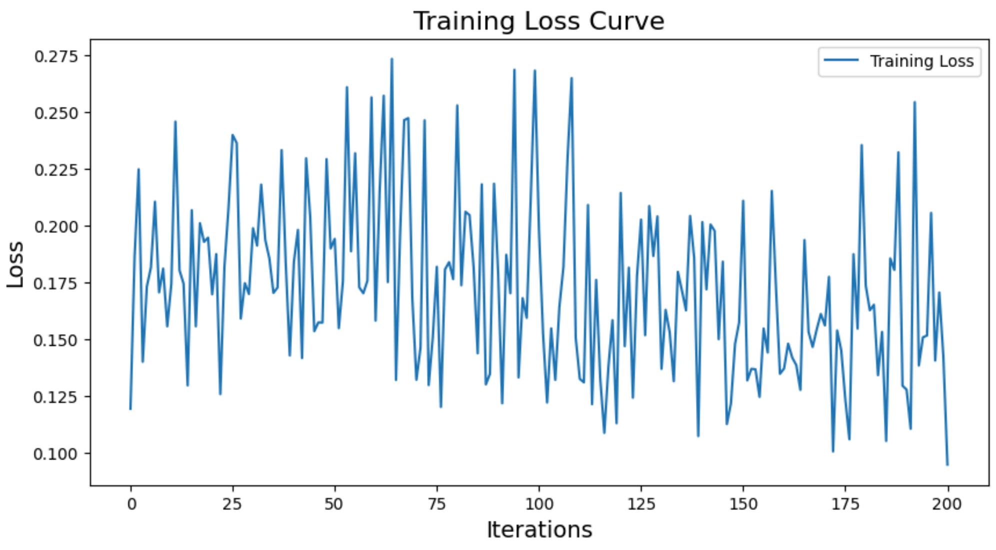

Figure 8. Training loss curve for BERT based model with tokenizer

Figure 6 shows significant variations in the loss of the model during training. This is a sign that the learning rate is relatively higher than it should be, hence effectively causing the model to overshoot optimal values during optimization. It also means that the model, in fact, is struggling to find the global minima due to the presence of several local minima.

The confusion matrix in Figure 7 shows that the BERT-based model with the tokenizer has a higher ability to distinguish between different classes compared to Experiment 1. The increased TP predictions and reduced misclassifications indicate that the tokenizer helps the model better capture the distinguishing features of each class. However, FPs in classes 1, 2, and 3 suggest that the model still faces challenges in identifying some examples of these classes.

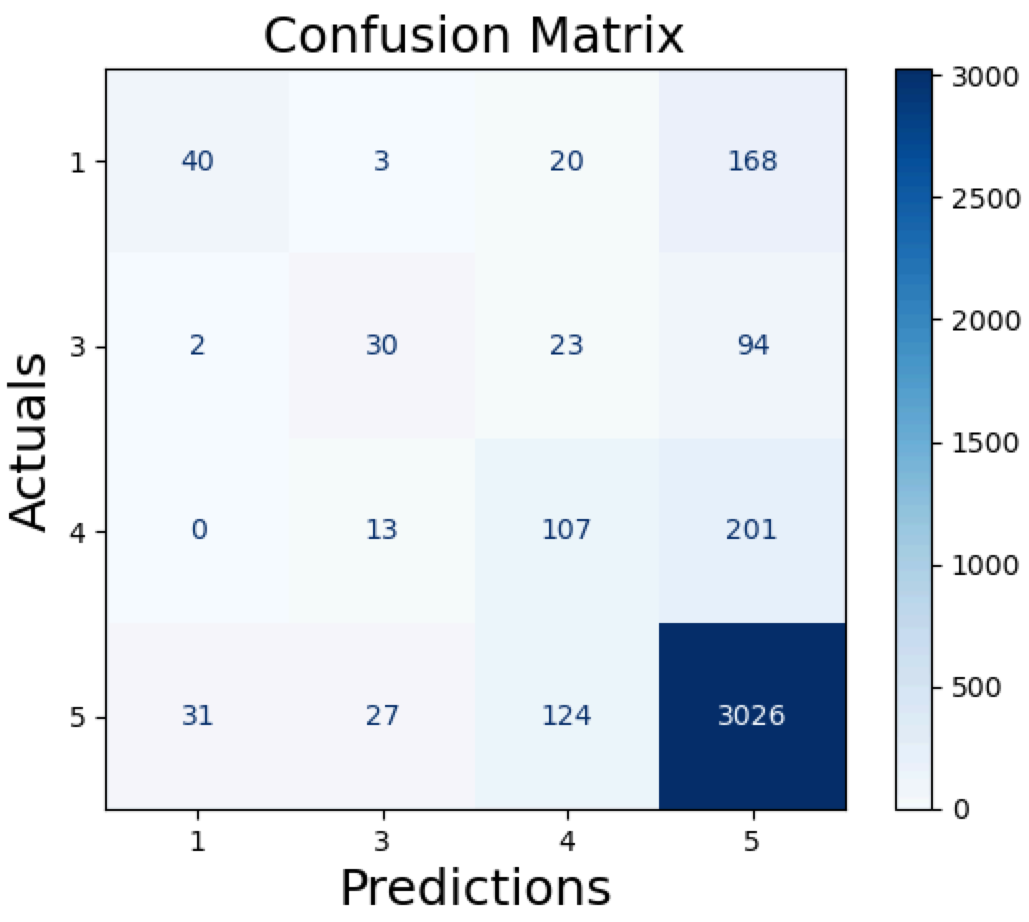

Figure 9. Confusion matrix for BERT based model with tokenizer

Comparing the results of the Experiments 1 and 2, the fine-tuning of the BERT-based model provides a good and robust baseline for the task of abbreviation detection in biomedical literature. For example, careful tuning of hyperparameters, such as the learning rate and the number of training epochs, can significantly boost model performance. Indeed, from this error analysis, it seems there are quite big potential improvements, especially in disambiguating abbreviations from their long forms. It would be enlightening to include other types of features in future work, like character-level embeddings and common domain knowledge, which would probably be of more help for this challenging problem.

3.3 Experiment 3 - RNN-Based Model

The third experiment uses an RNN-based model, which was trained over the PLOD-CW dataset, to compare the performance with BERT-based models. The architecture of the RNN model consists of the layers as follows: embedding, GRU, and linear output layers. It was trained within 10 epochs, containing a batch size of 32, and the learning rate that equals 0.001, using the Adam optimizer. Compared to BERT models, which converge really fast, the RNN model still manages to converge, although at a slower pace, but it does reach a reasonable level of performance. After running the model for 10 epochs, it scored an F1 of 0.555 on the validation set, which is worse than the best model achieved in Experiment 2.

The RNN-based model's performance, with an F1-score of 0.555, is lower than the best-performing BERT-based model from Experiment 2, which achieved an F1-score of 0.924. This difference in performance shows the strengths of the BERT-based architecture in capturing complex patterns for abbreviation and long form detection. The RNN-based model's slower convergence and lower F1-score suggest that it may struggle to effectively learn and generalize from the training data, possibly due to its simpler architecture and lack of pre-training on large-scale textual data.

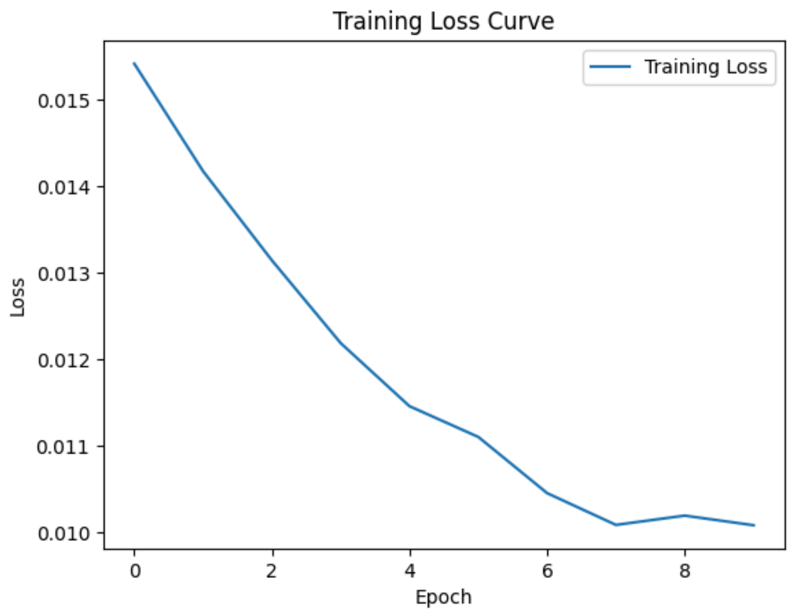

Figure 10. Training loss curve for RNN based model

The training loss curve shown in Figure 8 is positive to the model training; it is going down with an increase in epochs. It is a sign of a learning model, the decrease in loss, and it does improve the predictive power with the data over epochs. There are no increases or peaks in the graph, hinting that the model has not overfit yet to the training data. A smooth curve toward the end that gives a hint the model might be starting to get in on the neighborhood of a minimum but would have to run more epochs to confirm this trend.

The confusion matrix shown in Figure 9 shows its strengths in identifying the 'I-LF' class, which represents the inside tokens of long forms. The top-left to bottom-right diagonal, which in an ideal case should contain the largest numbers (TPs) of each class, indeed shows the strong performance for the class 'I-LF'. However, some cases, such as 'B-O', 'B-AC', 'I-AC', have been confused by the model with ‘I-L.’ These misclassifications indicate that the RNN-based model has difficulty capturing the subtle differences between these classes, which is crucial for accurately identifying the boundaries and relationships between abbreviations and their corresponding long forms.

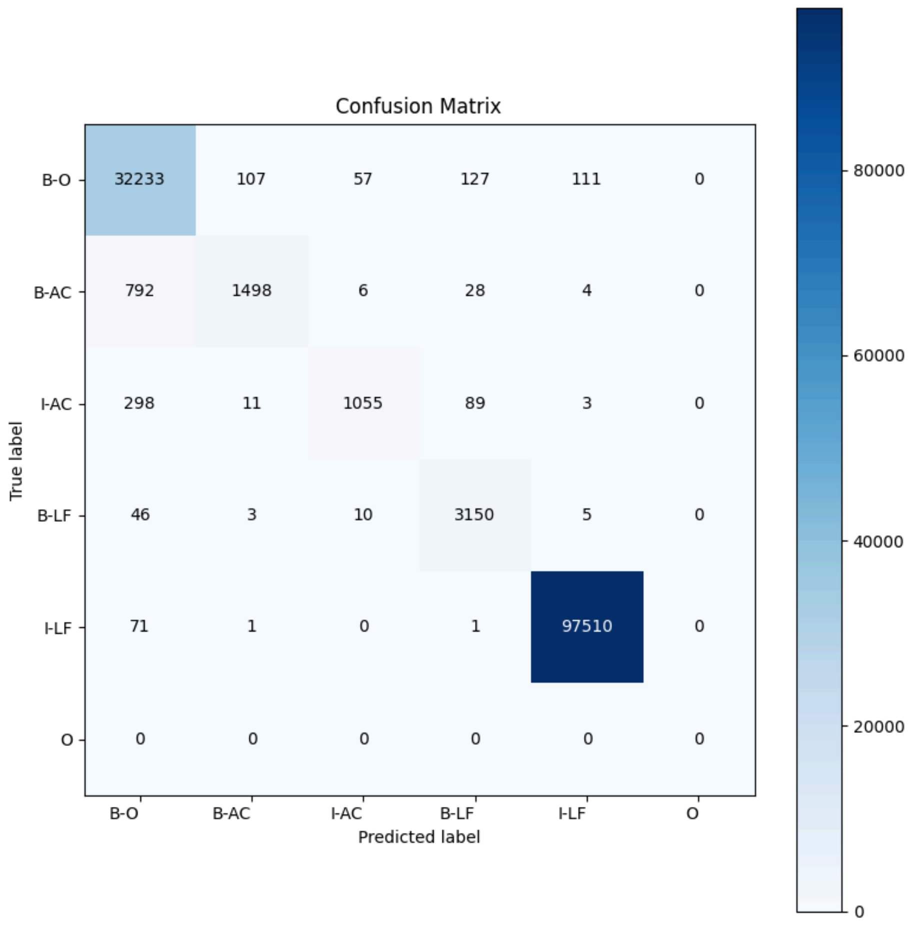

Figure 11. Confusion matrix for RNN based model

The error analysis shows that, like the BERT-based models, the RNN model frequently misclassifies between the ‘B-AC’ (beginning of abbreviation) and ‘B-LF’ (beginning of long form) classes. Additionally, it struggles with the ‘I-AC’ and ‘I-LF’ classes, representing the inside tokens of abbreviations and long forms, respectively. This suggests difficulties in capturing the dependencies between the beginning and inside tokens of the same entity, leading to higher misclassification rates.

3.4 Experiment 4 - Hyperparameter Tuning for RNN-Based Model

In the fourth experiment, different hyperparameters were used to test the model with performance changes. These include learning rates of 1e-5, 2e-5, and 5e-5, with numbers of epochs for training at 3, 5, and 10. This experiment shows the insensitivity of RNN model performance with respect to hyperparameter changes but, at the same time, it is sensitive in BERT-based models. The best results were achieved using a learning rate of 5e-5 and 10 training epochs and resulted in slightly better performance, with a validation set F1-score of 0.342. This score, however, is lower than the performance achieved by the best BERT-based model.

The impact of hyperparameter tuning on the RNN-based model's performance is evident from the slight improvement in the F1-score, which increased from 0.555 in Experiment 3 to 0.342. Although this improvement is not as significant as the performance gap between the RNN-based model and the BERT-based models, it demonstrates the potential for enhancing the RNN-based model's performance through careful hyperparameter selection. The relatively small improvement suggests that the limitations of the RNN-based model in capturing the complexities of the problem may not be fully addressed by hyperparameter tuning alone.

Figure 10 shows that the training loss decreases sharply at first and gets flatter during the progress of training. This means that the model learns quickly from the training data at first but discards the gains made in every epoch as the training draws closer to some optimal point. The curve shows no increase, meaning that there is no overfitting over the epochs recorded. In such a case, the peak of the curve would, therefore, suggest that the further training made little effect on improving the model’s performance and, in fact, had likely reached minima with further training if not making a substantial increase in the model’s performance.

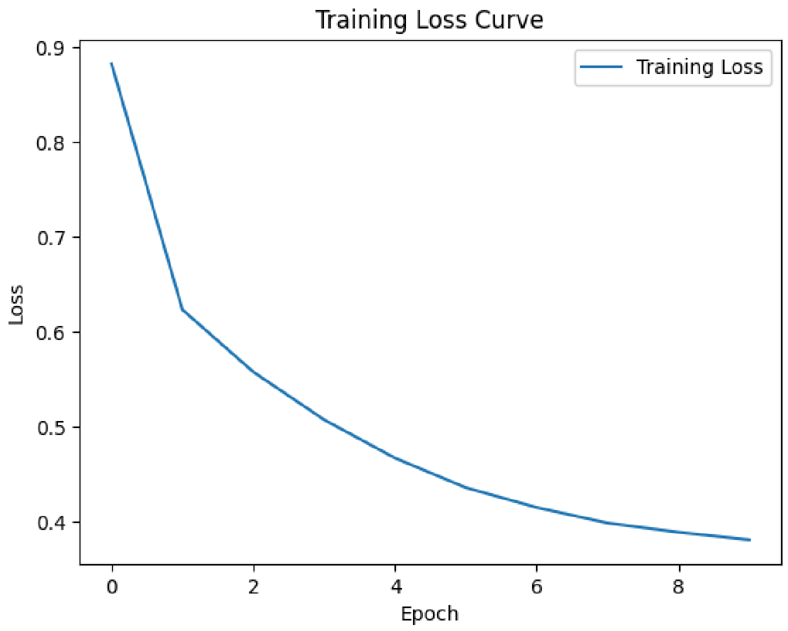

Figure 12. Training loss curve of hyper parameter tuning for RNN model

The confusion matrix for the RNN-based model with tuned hyperparameters shows improvements in the TP predictions for some classes, such as ‘I-LF’. However, the model still struggles with misclassifications between the ‘B-LF’ and ‘I-LF’ classes, as well as between the ‘B-AC’ and ‘B-O’ classes. These persistent misclassifications indicate that the RNN-based model, even with tuned hyperparameters, has difficulty capturing the subtle differences and dependencies between these classes. This limitation may be attributed to the inherent complexity of the problem, the variability in the representation of abbreviations and long forms, and the lower number of representative examples in the training data.

Comparing Experiments 3 and 4 with Experiments 1 and 2, it is evident that BERT-based models outperform the RNN-based models in detecting abbreviations and long forms in biomedical literature. The RNN models, although capable of learning the task to some extent, struggle to match the performance of BERT-based models, even after optimal hyperparameter tuning. This disparity is likely due to the extensive pre-training of BERT models on large-scale textual data, which equips them to better capture complex linguistic patterns and relationships than RNN models trained from scratch. Future research might explore using pre-trained RNN models like ELMo to potentially bridge this performance gap.

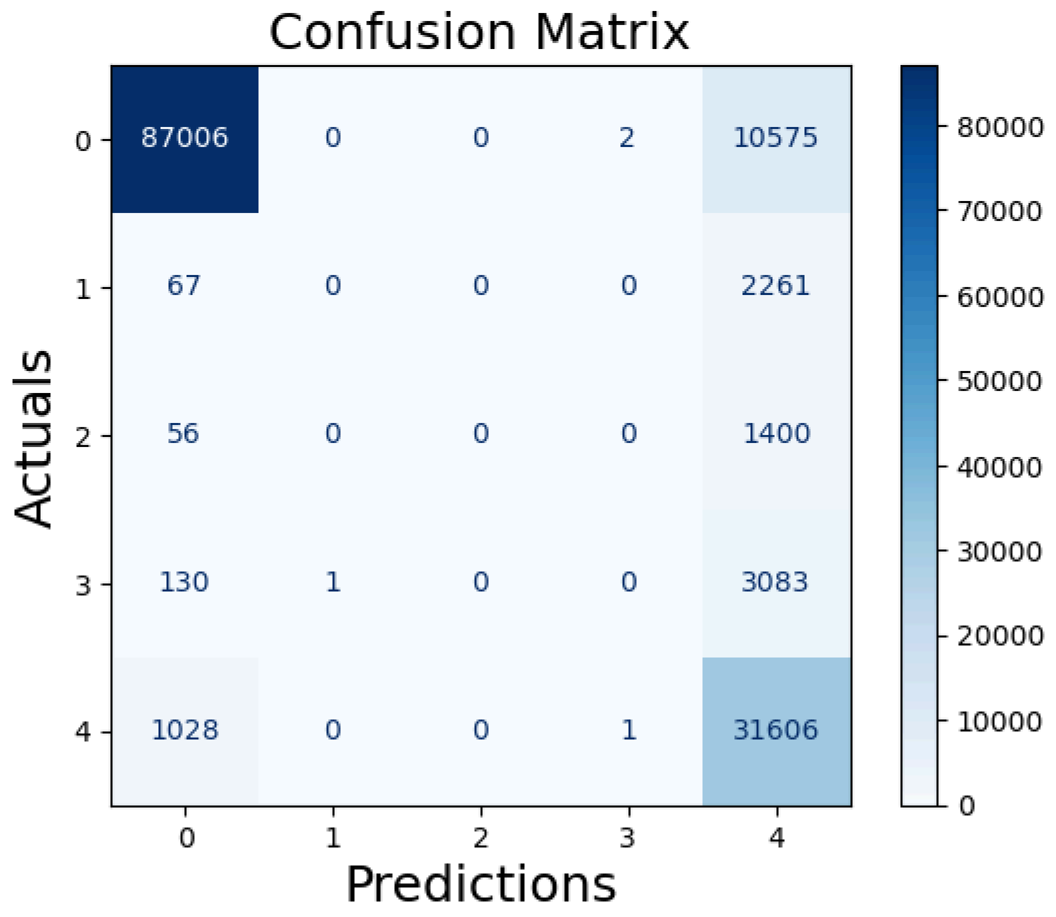

Figure 13. Confusion matrix for RNN based model

4 Outcomes

Experiments compared different models for detecting abbreviations and their definitions in medical text. A BERT-based model with a tokenizer achieved the best results, reaching an F1-score of 0.924 on the validation set. This significantly outperformed RNN-based models (0.555 and 0.342 F1-score). Fine-tuning the BERT model's learning rate (2e-5) and training epochs (10) was crucial for optimal performance. The tokenizer likely helped with accurate text processing. RNN-based models struggled despite hyperparameter adjustments, suggesting they might not capture the complexity of abbreviations in biomedical text. Overall, adjusting learning rates and epochs was important for all models. While BERT excelled, further exploration is needed to address misclassifications and potentially improve performance with other models or architectures.

5 Articulated Evaluation and Effective Solution

a. Varied Performance:

- BERT with tokenizer achieved high accuracy, suggesting its real-world potential.
- RNN-based models lagged behind, highlighting room for improvement in those architectures.

b. Defining "Good Enough" Performance:

- The acceptable F1-score depends on the application's needs.
- In biomedical tasks, high precision and recall are crucial, so a higher F1-score (>0.9) is ideal.

c. Improving Underperforming Models:

- RNN models can benefit from architectural changes (attention mechanisms, complex recurrent units), larger and more diverse training data, or transfer learning techniques.

d. Balancing Efficiency and Performance:

- For exceptional models (like BERT with tokenizer), sacrificing some accuracy for efficiency might be possible (smaller model, optimized inference).
- The trade-off between efficiency and performance depends on the application's needs.