# Aambela at BLP-2023 Task 2: Enhancing BanglaBERT Performance for Bangla Sentiment Analysis Task with In Task Pretraining and Adversarial Weight Perturbation

**Md Fahim**
Center for Computational & Data Sciences
Independent University, Bangladesh
Dhaka-1229, Bangladesh
fahimcse381@gmail.com

## Abstract

This paper introduces the top-performing approach of "Aambela" for the BLP-2023 Task 2: "Sentiment Analysis of Bangla Social Media Posts". The objective of the task was to create systems capable of automatically detecting sentiment in Bangla text from diverse social media posts. My approach comprised fine-tuning a Bangla Language Model with three distinct classification heads. To enhance performance, we employed two robust text classification techniques. To arrive at a final prediction, we employed a mode-based ensemble approach of various predictions from different models, which ultimately resulted in the **1st place** in the competition.

## 1 Introduction

In recent years, Natural Language Processing (NLP) has advanced significantly, highlighting the importance of sentiment analysis. This application provides insights into public opinion and social media trends. In the context of Bangla text, sentiment analysis is crucial, aiding businesses in interpreting customer feedback, assisting policymakers in understanding public sentiment, and boosting media engagement. Concerning the importance of sentiment analysis, the organziers of BLP-Shared Task 1 (Hasan et al., 2023a) provide one of the largest manually annotated datasets for sentiment analysis which encompasses sentiment across multiple platforms.

The proposed sentiment analysis approach involves fine-tuning the Bangla Language Model, such as BanglaBERT(Bhattacharjee et al., 2022), and utilizing three distinct classification heads to enhance model performance. To address overfitting and ensure robust generalization, strategies like cross-validation and adversarial perturbation techniques are employed. Task-specific pretraining of BanglaBERT on both the train and train+validation datasets is explored, yielding performance improvements. Different classification heads in various techniques focus on distinct aspects of sentiment classification reasoning. To capture these diverse perspectives, a mode-based ensemble technique is applied. The ensemble predictions prove to be the best-performing model in the experiments, securing the top position on the leaderboard.

## 2 Background

### 2.1 Task & Dataset Description

The primary aim of this task (Hasan et al., 2023a) is to conduct sentiment analysis on Bengali textual data, focusing on multi-class sentiment classification. In essence, it involves categorizing text into one of three distinct sentiment classes: Positive, Negative, or Neutral. The overarching objective is to create a model that can effectively and precisely assign text to these sentiment categories by discerning its emotional context.

| Data Splits | Total Samples | Class wise Samples | | |
|---|---|---|---|---|
| | | Negative | Positive | Neutral |
| ***Train*** | 35266 | 15767 | 12364 | 7135 |
| ***Dev*** | 3934 | 1753 | 1388 | 793 |
| ***Test*** | 6707 | 3338 | 2092 | 1277 |

Table 1: Dataset Statistics for Shared Task 2 (Sentiment Analysis Task).

The dataset for this shared task is a fusion of two distinct sources: MUBASE (Hasan et al., 2023b) and SentNob (Islam et al., 2021). SentNob encompasses public comments sourced from diverse social media platforms, spanning 13 domains including politics, education, and agriculture. Conversely, the MUBASE dataset comprises an extensive collection of multi-platform data, featuring manually labeled Tweets and Facebook posts. The dataset statistics along with class wise sample size is provided in Table 1.



*Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 317–323

### 2.2 Observations and Baselines

Upon analyzing the dataset, several key observations emerged. Firstly, despite the presence of numerous URLs, they appeared to have no substantial influence on the dataset's attributes. Additionally, there was an absence of class dependency linked to these URLs. Moreover, emojis within the dataset did not appear to significantly impact the analysis. And also, the dataset exhibited a notable prevalence of error words, a common feature in text collected from YouTube comments. These observations offer valuable insights into the dataset's nature and characteristics.

The organizers have also provided baseline results for this task on both the Dev-Test and Test Dataset. Three different methods were employed: the Random Baseline, Majority Baseline, and the n-gram Baseline. Notably, the n-gram Baseline demonstrated better performance, surpassing the other two methods by a good margin. In the test dataset, the n-gram Baseline achieved an impressive 55.14% micro F1 score, while on the Dev-Test dataset, it reached 57.36%.

## 3 Method Description

### 3.1 ITPT

withIn Task PreTraining (ITPT) is a popular approach while solving text classification problem. It was proposed by (Sun et al., 2019). We also use this ITPT techniques in our task. BanglaBERT undergoes training in a broad domain, characterized by a distinct data distribution when compared to the target domain. So we perform additional pretraining on BanglaBERT using data specific to the target domain. Actually, we futher perform Masked Langugae Modeling (MLM) (Devlin et al., 2019) using the pretrained BanglaBERT on our training corpus.

### 3.2 AWP

Adversarial Weight Perturbation (AWP) (Wu et al., 2020) is a regularization technique that encourages neural networks to have stable and robust weights by penalizing sensitivity to parameter perturbations. This regularization improves the model's generalization and robustness, making it less susceptible to adversarial attacks.

In the neural network, the loss function is denoted as $\mathcal{L}(\Theta)$, where $\Theta$ represents the model parameters. The objective is to minimize this loss on a training dataset. AWP introduces a regularization term to penalize the sensitivity of the model's output to small perturbations in its parameters. This is added to the loss function:

$$\mathcal{L}_{\text{AWP}}(\Theta) = \mathcal{L}(\Theta) + \lambda \cdot \text{AWP}(\Theta)$$

Here, $\lambda$ controls the regularization strength, and $\text{AWP}(\Theta)$ is the AWP term. The AWP term is designed to adversarially perturb the model's weights and is formulated as:

$$\text{AWP}(\Theta) = \frac{1}{2} \sum_i \left\| \frac{\partial \mathcal{L}}{\partial \Theta_i} \right\|_2^2$$

This term quantifies the sensitivity of the loss function to changes in each parameter $\Theta_i$ and encourages stable and robust weight values. During training, the combined loss function $\mathcal{L}_{\text{AWP}}(\Theta)$ is optimized. AWP's regularization helps prevent overfitting and enhances the model's resistance to adversarial attacks.

### 3.3 Classification Heads

For an input sentence $S$, we obtain $S = \{t_1, t_2, \ldots, t_n\}$ after processing the sentence with the BanglaBERT tokenizer, where $t_i$ represents the $i$-th token. Then the sentence $S$ through a BanglaBERT model, we obtain contextual representations of last layer for each token $t_i$, denoted as $H = \{h_1, h_2, \ldots, h_n\}$, where $h_i$ represents the contextual representation of token $t_i$.

#### 3.3.1 FFN Head on CLS Token

In order to obtain a fixed-size representation for the entire sentence to use in classification, we utilize the special [CLS] token representation, denoted as $h_{\text{CLS}}$ which is fed into a two-layer Feed Forward Neural Network (FFN). The resulting representation $z$ is employed for the classification process by following method.

$$z = W_2 \cdot (\text{ReLU}(W_1 \cdot h_{\text{CLS}} + b_1)) + b_2$$

#### 3.3.2 Mean, Max, Min Pooling

As our model does batchwise operations, so the sequence may contain padded values for equal length. BanglaBERT model provides an input mask vector $M$ for a sentence in a batch where $M = [m_1, m_2, \ldots, m_n]$ to indicate valid tokens. $m_i = 1$ for valid tokens and 0 for padded values. Then we apply *MeanPooling*, *MinPooling*, *MaxPooling* (Minaee et al., 2021) as followings:

| Techniques | Classification Head | CV Score | Performance Metrics | | | |
|---|---|---|---|---|---|---|
| | | | Dev Set | | Test Set | |
| | | Micro F1 | Accuracy | Macro F1 | Accuracy | Macro F1 |
| Without AWP | CLS + MLP | 72.36 | 72.67 | 68.82 | 71.05 | 66.29 |
| | Dropouts Enhanced MLP | 72.24 | 73.00 | 69.19 | 70.81 | 65.65 |
| | [Mean, Max, Min] Pooling | 72.38 | 71.73 | 68.72 | 71.15 | 67.33 |
| | Reinit Last Two Layers | 72.39 | 72.32 | 68.50 | 71.48 | 66.74 |
| With AWP | CLS + MLP | 73.21 | 74.12 | 70.05 | 72.64 | 67.58 |
| | Dropouts Enhanced MLP | 72.90 | 73.87 | 69.29 | 72.72 | 67.30 |
| | [Mean, Max, Min] Pooling | 73.24 | 72.34 | 69.83 | 71.72 | 68.42 |
| | Reinit Last Two Layers | 73.47 | 72.52 | 69.62 | 71.58 | 68.00 |
| Including Dev Dataset | CLS + MLP | 73.83 | - | - | 72.40 | 67.32 |
| | Dropouts Enhanced MLP | 73.77 | - | - | 72.42 | 67.41 |
| | [Mean, Max, Min] Pooling | 73.76 | - | - | 71.28 | 67.41 |
| | Reinit Last Two Layers | 73.91 | - | - | 71.50 | 67.34 |
| ITPT on Training Data | CLS + MLP | 73.49 | 74.17 | 70.26 | 72.76 | 67.97 |
| | Dropouts Enhanced MLP | 73.47 | 74.17 | 70.23 | 72.89 | 67.83 |
| | [Mean, Max, Min] Pooling | 73.60 | 72.88 | 70.18 | 71.76 | 68.29 |
| | Reinit Last Two Layers | 73.42 | 72.47 | 69.82 | 70.85 | 67.60 |
| ITPT on Train + Validation Data | CLS + MLP | 73.74 | 74.07 | 70.12 | 72.66 | 67.84 |
| | Dropouts Enhanced MLP | 73.79 | 74.10 | 70.07 | 72.51 | 67.63 |
| | [Mean, Max, Min] Pooling | 73.94 | 73.31 | 70.52 | 71.48 | 67.99 |
| | Reinit Last Two Layers | 73.59 | 72.93 | 70.27 | 71.39 | 68.03 |
| | **Ensemble** | - | - | - | **73.10** | **68.74** |

Table 2: Performance of BanglaBERT in Sentiment Analysis in Shared Task 2 with different Techniques. While experiments were done with including validation (dev) dataset the measurement on dev set were skipped. **Ensemble** model was the final model which place first in the leaderboard. Scores with underline can also be the top scorer.

$$
\begin{aligned}
\textit{Mean\_Pool} &= \text{MeanPooling}(X, M) \\
&= \frac{1}{\sum_{i=1}^{n} m_i} \sum_{i=1}^{n} m_i \cdot h_i \\
\textit{Min\_Pool} &= \text{MinPooling}(X, M) \\
&= \min_{i=1}^{n} (m_i \cdot h_i) \\
\textit{Max\_Pool} &= \text{MaxPooling}(X, M) \\
&= \max_{i=1}^{n} (m_i \cdot h_i)
\end{aligned}
$$

Then we concat those pooling and passed them into a two layer MLP for finding class logits for classification as follows:

$$
\begin{aligned}
z = W_2 \cdot (\text{PReLU}(W_1 \cdot [&Mean\_Pool, \\
&Min\_Pool, Max\_Pool]) + b_1)) + b_2
\end{aligned}
$$

#### 3.3.3 Dropout-Enhanced CLS Token Head

In this case, an expanded classification head is incorporated, which is an enhancement of the CLS_MLP head discussed in Section 3.3.1. In this variation, we apply dropout to the FFN layer. We explore a range of distinct dropout rates denoted as $D = d_1, d_2, \ldots, d_k$, where $d_i$ signifies the dropout rate for the $i$-th rate. For a specific dropout rate $d_i$, we calculate class representations $z_i$ using the subsequent equation:

$$z_i = W_2 \cdot (\text{DropOut}(d_i)(\text{ReLU}(W_1 \cdot h_{\text{CLS}}) + b_1)) + b_2$$

After acquiring $m$ unique class representations (logits), we calculate the final representation $z$ by taking the average of these representations, following the equation:

$$z = \frac{1}{m} \sum_{i=1}^{m} z_i$$

#### 3.3.4 Re-initialization of last 2 layers of BanglaBERT

In this case, we re-initialize the last two layers for BanglaBERT like (Zhang et al., 2020) did. If the original BERT model as $M_{BanglaBERT}$, which comprises multiple layers. When we say we are re-initializing the last two layers, it means we are modifying these layers to create a new model, which we'll call $M_{New}$. The re-initialized model $M_{New}$ can be defined as

$$M_{\text{New}} = M_{\text{BanglaBERT}}[: L-2] + \text{Reinitialize}(M_{\text{BanglaBERT}}[L-2 :])$$

For involving classification task in this case, we use MLP Head on CLS token similar to we describe in Section 3.3.1.

## 4 Result and Analysis

Different models and experiments were done during the development phase which are reproted in Appendix B. The experiment set up and hyper parameters details are described in Appendix A. Another experiments for model choice encompassed machine learning models (SVM, RandomForest, XGBoost) using TF-IDF feature extraction, deep learning models (LSTM, LSTM+Attention), and multilingual Transformer models (mBERT, mDeBerta, XLMRoberta base), with mDeBerta showing superior performance. Additionally, two Bangla Language Models were considered, with the *csebuetnlp-BanglaBERT* model emerging as the top performer. Table 3 summarizes the experimental results for model selection.

Table 3 displays key experiments using the *csebuetnlp/banglabert* model backbone for contextualized word representations, coupled with various classification heads as discussed in Section 3.3.

| Model Name | Acc ↑ | F1 ↑ |
|---|---|---|
| TF-IDF + SVM | 55.74 | 44.41 |
| TF-IDF + RandomForest | 58.41 | 50.65 |
| TF-IDF + XGBoost | 60.99 | 53.95 |
| LSTM | 65.91 | 61.88 |
| LSTM + Attention | 67.82 | 63.76 |
| mBERT-case | 66.29 | 62.19 |
| mDeBerta-v3 base | 70.84 | 64.35 |
| XLM-Roberta-base | 69.67 | 61.58 |
| SagorSarker-BanglaBERT | 67.08 | 61.30 |
| **csebuetnlp-BanglaBERT** | **72.57** | **66.42** |

Table 3: Different Types of Model Performance in Validation (Dev) Dataset.

These experiments employ 5-fold cross-validation. The inclusion of AWP (Section 3.2) enhances both cross-validation scores and generalization to validation and test datasets by approximately 1-2%. However, incorporating the validation data into training yields a slightly lower test set performance despite boosting the CV score. ITPT (Section 3.1) on training data significantly enhances performance across all classification heads. Conversely, including the validation data during ITPT yields mixed results, with slight improvements in some heads and minor reductions in others.

A final prediction is made by ensembling all classification heads from different techniques. The ensemble technique employed is a **Mode-based Ensemble**, aggregating predictions from all models across techniques and selecting the mode as the final prediction. This approach achieved an accuracy of **73.10%** (micro F1) and **68.74%** (macro F1) on the test set, placing it at the **top of the leaderboard**. Though the model has a highest score in the leaderboard, it has some limiations and scope for improvements which are describe in Limiation 5 section and Appendix C.

## 5 Conclusion

In this work,we have experimented with fine-tuning BanglaBERT in different aspects using different classification heads. The result showed that it gives a better score. Adverserial training and cross validation made the model more robust. In task pretraining helped the model to further investigate different classes of sentiment analysis. Our finding is that using adverserial training and in task pretraining we can improve our model further and build up a better model.

## Limitations

The proposed models are struggled to predict the Neutral samples. Besides a good amount of sentences have token length larager than 512. To fit those sentences, we need to truncate the token length to 512. More on error analysis and scopes for improvement can be found at Appendix C.

## References


Abhik Bhattacharjee, Tahmid Hasan, Wasi Ahmad, Kazi Samin Mubasshir, Md Saiful Islam, Anindya Iqbal, M. Sohel Rahman, and Rifat Shahriyar. 2022. BanglaBERT: Language model pretraining and benchmarks for low-resource language understanding evaluation in Bangla. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1318–1327, Seattle, United States. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Md. Arid Hasan, Firoj Alam, Anika Anjum, Shudipta Das, and Afiyat Anjum. 2023a. Blp-2023 task 2: Sentiment analysis. In *Proceedings of the 1st International Workshop on Bangla Language Processing (BLP-2023)*, Singapore. Association for Computational Linguistics.

Md. Arid Hasan, Shudipta Das, Afiyat Anjum, Firoj Alam, Anika Anjum, Avijit Sarker, and Sheak Rashed Haider Noori. 2023b. Zero- and few-shot prompting with llms: A comparative study with fine-tuned models for bangla sentiment analysis.

Khondoker Ittehadul Islam, Sudipta Kar, Md Saiful Islam, and Mohammad Ruhul Amin. 2021. SentNoB: A dataset for analysing sentiment on noisy Bangla texts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3265–3271, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, 54(3):1–40.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer.

Dongxian Wu, Shu-Tao Xia, and Yisen Wang. 2020. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*.


## A Experimental Setup & Hyperparameters

In every dataset, we conducted critical preprocessing steps for text, encompassing the elimination of punctuation, emojis, and any existing URLs. We applied different types of models including TF-IDF+SVM, TF-IDF+RandomForest, TF-IDF+XGBoost, LSTM, LSTM+Attention, mBERT-case, mDeBerta-v3 base, XLM-Roberta-base and SagorSarker-BanglaBERT and csebuetnlp-BanglaBERT for Dev dataset. To extract hidden representations from the text, we employed two distinct models: LSTM and BERT, as the text encoding methods.

When using the LSTM-based models, an embedding layer with an embedding dimension of 128 was employed to convert the tokens into vector representations. The LSTM model's hidden dimension was set to 256. We used a learning rate of $10^{-3}$ and a batch size of 8 for this configuration.

On the other hand, for the BERT model, we utilized the *Bangla-bert* variants that enables us to extract contextual representations through fine-tuning. Bert model along with other transformers models include the hidden dimension 768. The learning rate for BERT was $2 \times 10^{-5}$, max length was 512 and a batch size of 8 were used for the models. This token length was consider because of its performance shown for Dev dataset from table 4, while from table 5 for Dev dataset showed batch size 8 performed better than other batch size configurations. Which encouraged the usage of the batch size 8 along with maximum length 512 in this study for transformer based models.

Both configurations employed the AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. To ensure robustness, we performed five-fold cross-validation and three different random seeds. Additionally, we set $\lambda = 10$ for all experiments. An ablation study investigating the effect of different $\lambda$ values is presented in Table. All experiments were conducted using Python (version 3.8) and PyTorch, leveraging the free NVIDIA Tesla K80 GPU available in

Google Colab, as well as a single NVIDIA Tesla P100 GPU provided by Kaggle.

## B Ablation Study

In this section detailed ablation study was performed which contains max length effects, batch size effects and effects of different loss was measured and analyzed.

### B.1 Token Length analysis

In table 4, for validation (Dev) dataset BanglaBert with 'csebuetnlp/BanglaBert' was applied for different max lengths. For 512 max length maximum accuracy and f1 score was achieved with outperforming other variation by 1%-2%.

| Max Length | Dev Acc ↑ | Dev F1 ↑ |
|---|---|---|
| 64 | 72.24 | 67.12 |
| 128 | 71.2 | 66.1 |
| 256 | 72.22 | 66.93 |
| **512** | **73.34** | **68.52** |

Table 4: Token Length Effect of *csebuetnlp/BanglaBert* in Validation (Dev) Dataset. Epoch Size 3

### B.2 Batch Size Effects

Table 5 depicts the batch size for which the maximum accuracy and F1 score was achieved. For batch size 8 bested other variation by slight margin, ranging from 0.5% to 1%.

| Batch Size | Dev Acc ↑ | Dev F1 ↑ |
|---|---|---|
| **8** | **72.52** | **67.74** |
| 16 | 72.22 | 66.93 |
| 32 | 72.31 | 67.01 |

Table 5: Batch Size Effect of *csebuetnlp/BanglaBert* in Validation (Dev) Dataset while Max Length = 512 were considered. Epoch Size 3

### B.3 Ablation On Losses

In this study, table 6 represent the effects of different loss variation were measured. From the table, Cross Entropy Loss(CE Loss) outperformed other variations including Weighted CE Loss, and Focal loss along with Cross Entropy Loss. Cross Entropy Loss showed improvement in its performance matrices by 1%-2% for all other losses.

| Loss Name | Dev Acc ↑ | Dev F1 ↑ |
|---|---|---|
| **CE Loss** | **73.31** | **68.41** |
| Weighted CE Loss | 72.5 | 67.12 |
| 0.5*Focal + 0.5*CE | 72.72 | 67.41 |
| 0.3*Focal + 0.7*CE | 71.35 | 66.36 |

Table 6: Batch Size Effect of *csebuetnlp/BanglaBert* in Validation (Dev) Dataset while Max Length = 128 were considered with Batch Size = 16. Epoch Size 5. Here *CE* indicates the *Cross Entropy Loss* and *Focal* means the *Focal Loss*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Negative | 0.7803 | 0.8119 | 0.7958 | 3338 |
| Neutral | 0.5040 | 0.4957 | 0.4998 | 1277 |
| Positive | 0.7887 | 0.7457 | 0.7666 | 2092 |
| accuracy | | | 0.7310 | 6707 |
| macro avg | 0.6910 | 0.6844 | 0.6874 | 6707 |
| weighted avg | 0.7303 | 0.7310 | 0.7303 | 6707 |

Figure 1: The list of words that are considered as new tokens to the model.

### B.4 Text Preprocessing Effects

Different preprocessing variations were also considered for this research endeavour. Removing URLs, Punctuation & Emoji's, Removing Punctuation only and Removing Emoji's only showed least improvement in the performance matrices, containing almost similar values. No preprocessing and Removing punctuation showed improvement by 1% from the previous variations. Applying BN-Unicode Normalizer after removing URLs + HTML Tag showed, or adding Normalizer after removing URLs + HTML Tag showed the most improvement by 1%-2% from aforementioned models. While, normalizing after removing URLs + HTML Tag bested all other preprocessing variations in terms of performance matrices.

## C Error Analysis & Scope for Improvements

In Figure 1, the classification report for mode-base-ensembel, which gives the top performance score in leaderbaoard is reported. From the classification report it is easily seen that the proposed models are struggled when the class label is *Neutral*. One of reasons for the poor performance on the *Neutral* class labelled is that we have fewer samples for this class rather than the remaining classes 1. Besides there may overlapping words for the classses.

There are few scopes for improvements by which the model may be more efficient. Increasing class

| Preprocessing | Dev Acc ↑ | Dev F1 ↑ |
|---|---|---|
| No Preprocessing | 71.2 | 66.10 |
| Removing URLs, Punctuation & Emoji's | 70.67 | 66.56 |
| Removing Punctuation Only | 70.01 | 65.52 |
| Removing URLs & HTML Tags | 71.59 | 66.46 |
| Removing Emoji's Only | 70.87 | 66.84 |
| **Adding Normalizer after removing URLs + HTML Tag** | **72.57** | **67.12** |
| Adding BN-Unicode Noramlizer after removing URLs + HTML Tag | 72.22 | 66.93 |

Table 7: Effect of different preprocessing techniques in devset performance for BanglaBERT. Each experiment was trained for 3 epochs.

samples for *Neural* class may help. An external data can be used for this. As a good amount sentences have token length greater than 512, different techniques like (Chunking or Sliding Window, Document-Level Embeddings and so on) can be used. Besides different augmentation techniques can also be examined.