Aambela at BLP-2023 Task 1: Focus on UNK tokens: Analyzing Violence Inciting Bangla Text with Adding Dataset Specific New Word Tokens

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

Abstract

The BLP-2023 Task 1 aims to develop a Natural Language Inference system tailored for detecting and analyzing threats from Bangla YouTube comments. Bangla language models like BanglaBERT have demonstrated remarkable performance in various Bangla natural language processing tasks across different domains. We utilized BanglaBERT for the violence detection task, employing three different classification heads. As BanglaBERT's vocabulary lacks certain crucial words, our model incorporates some of them as new special tokens, based on their frequency in the dataset, and their embeddings are learned during training. The model achieved the **2nd position** on the leaderboard, boasting an impressive macro-F1 Score of 76.04% on the official test set. With the addition of new tokens, we achieved a 76.90% macro-F1 score, surpassing the top score (76.044%) on the test set.

1 Introduction

In recent times, transformer models have gained popularity through pretraining on diverse text data (Zhang et al., 2022). Pretraining imparts context awareness, linguistic patterns, and word knowledge (Yenicelik et al., 2020). Combining it with fine-tuning vastly outperforms traditional models. These models have a vocabulary from pretraining, representing known words. During fine-tuning the pretrained in a task, there might be some words not in the vocabulary. Tokenizer furhtermore tries to split the word into multiple subwords. Still, if a word is missing or unsegmentable, they use a *[UNK]* token (Nayak et al., 2020).

BanglaBERT (Bhattacharjee et al., 2022) model is one of the most popular pretrained language models in Bangla text classification which is trained on a large corpus of Bangla text. It faces trouble because of *[UNK]* tokens. This problem arises because the pretrianed BERT model sometimes can't give representation to the words that are very important to understand the context. Table 1 represents a small sample taken from the dataset. Here, it can be seen that the significant words which are important for the contextual understanding, assigned as *[UNK]* token in token representation.

Sentence	Tokens
সুন্দর সিদ্ধান্ত নেওয়া হয়েছে	['সুন্দর', 'সিদ্ধান্ত', '[UNK]', '[UNK]']
বয়কট নিউমার্কেট	['[UNK]', 'নিউমার্কেট']
ঢাকা কলেজ মানেই অন্যায়ের বিরুদ্ধে প্রতিবাদ।	['ঢাকা', 'কলেজ', 'মানেই', '[UNK]' 'বিরুদ্ধে', 'প্রতিবাদ','।']

Table 1: The table depicts word-wise tokenization for some example sentences using BanglaBERT tokenizer. Here [UNK] means unknown token

In this study, the main focus was to identify the most frequent words for which *[UNK]* tokens are assigned and add these words to the pretrained vocabulary as a special token. It is shown in the study that this approach improved performance for text classification. To further improve the model's performance, three different classification heads were used. These heads improved the model's prediction by focusing on different words. Three classification heads, along with the proposed approach, achieved better performance than the previous approach.

2 Background

2.1 Task and Dataset Description

The preliminary task of shared task 1 (Saha et al., 2023b) is to detect violence-inciting text (VITD), particularly focusing on identifying threats that could incite further violence. The dataset (Saha et al., 2023a), comprised of Bangla-language YouTube comments, is centered around the top nine violent incidents in the Bengal region over the past decade. This task involves three categories: Direct violence, encompassing explicit

Proceedings of the First Workshop on Bangla Language Processing (BLP-2023), pages 201–207

threats to individuals or communities; Passive violence, representing violence through derogatory language, abusive remarks, or slang; and Non-violence, which pertains to content unrelated to violence, including discussions on social rights and general topics. The primary objective is to develop models for automated detection, contributing to online safety and, foster responsible and constructive discourse.

Data Splits	Total Samples	Label wise Samples		
		Label 0	Label 1	Label 2
Train	2700	1389	922	389
Dev	1330	717	417	196
Test	2016	1096	719	201

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

The dataset,**Vio-lens**, for shared task 1 (Saha et al., 2023a) comprises texts explicitly associated with violence, each annotated with a corresponding label. Labels are assigned as follows: direct violence is labeled as 2, passive violence is labeled as 1, and non-violence is designated as 1. The dataset statistics are given in Table 2, with mentioning label wise sample sizes for different splits of the dataset.

The dataset (for all splits) contains a significant number of emojis, and these emojis exhibit a notable influence on class dependencies. For instance, violent texts often feature angry emojis.

2.2 Related Work and Baselines

BanglaHateBERT (Jahan et al., 2022), a BERT model for Bangla abusive language detection, was trained on a large-scale offensive text corpus. They also provide a 15K manually annotated Bangla hate speech dataset to the research community. By retraining BanglaBERT (Bhattacharjee et al., 2022) with 1.5 million offensive posts, BanglaHateBERT consistently outperforms the generic pre-trained language model in all datasets. (Mridha et al., 2021) address the rise of offensive Bangla and Banglish texts in online communication. They propose an offensive message detection mechanism using BanglaBERT (Sarker, 2020) combining AdaBoost (Hastie et al., 2009) and LSTM (Hochreiter and Schmidhuber, 1997) models. This proposed **L-Boost** model outperforms baseline classifiers.

Vio-lens dataset provided by the organizer for this shared task introduces a novel dataset related to violence detection tasks in Bangla consists of different forms of violence. The organizer also provided the baselines for this tasks, where fine-tuned model of BERT multilingual base (Devlin et al., 2019) gets 68.19% macro-F1 score where as fine-tuned model of XLM-Roberta (Conneau et al., 2020) gets 72.92% and the fine-tuned model of BanglaBERT (Bhattacharjee et al., 2022) gets 78.79% in the validation test dataset.

3 Method Description

3.1 Adding New Tokens to Vocabulary

Table 1 provides insights into the behavior of the BanglaBERT tokenizer. Notably, the tokenizer occasionally represents highly informative words as *[UNK]* tokens. These words are pivotal for context comprehension, and their conversion to *[UNK]* tokens can pose challenges for the model's predictive capabilities. Identifying the words that result in *[UNK]* tokens from the tokenizer presents a notable challenge. This complexity arises from the tokenizer's utilization of subword tokenization techniques, wherein token lengths may not align with the number of words in a sentence.

To address this challenge effectively, we restrict our analysis to samples without subwords in their tokenization. Within this subset, we extract the specific words that are tokenized as *[UNK]* by the BanglaBERT tokenizer. These words are then ranked by their frequency of occurrence, and we select the top p words to be introduced as new tokens, precisely as special tokens, into the pre-trained vocabulary.

$$Vocab_{new} = Vocab_{original} + \{w_1, w_2, \ldots, w_p\}$$

where $w_1, w_2, \ldots, w_p$ denotes the those frequent words. For a given sentence S, the original tokenization process as:

$$S_{\text{original}} = \{t_1, t_2, \ldots, t_n\}$$

Here, t_i represents the i-th token obtained using the BanglaBERT pretrained tokenizer. While considering new vocabulary for tokenization, the tokenization process becomes:

$$S_{\text{updated}} = \{t_1, t_2, \ldots, t_l\}$$

During the fine-tuning process of the BanglaBERT model, it adapts its internal representations to consider new tokens as valid tokens. This enables the model to encode the

contextual information of words. As a result of fine-tuning, the BanglaBERT model generates contextual embeddings for each token, including those new tokens. These embeddings capture the semantic meaning and context of each token in the input sequence.

3.2 Model Classification Heads for Enhanced Performance

For an input sentence S, we will get $S = \{t_1, t_2, \ldots, t_n\}$ after passing the sentence into the BanglaBERT tokenizer, where t_i represents the i-th token. In this case, we also incorporate the new tokens added to the BanglaBERT tokenizer as we discuss in Section 3.1.

After passing the sentence S through a BanglaBERT model, we obtain contextual representations 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. In this case, we consider the last layer hidden representations of the BanglaBERT encoder.

3.2.1 MLP Head on CLS Token

To obtain a fixed-size representation for the entire sentence for classification, we typically use the special [CLS] token representation h_{CLS}. This representation can be extracted as: $h_{\text{CLS}} = h_1$. Then we pass this h_{CLS} representation through a two-layer Feed Forward Neural Network (FFN) for classification the get class logits.

3.2.2 Dropout-Enhanced CLS Token Head

We introduce an extended classification head, an expansion of the CLS_MLP head detailed in Section 3.2.1. In this variant, we apply dropout to the FFN layer. We explore a set of distinct dropout rates denoted as $D = \{d_1, d_2, \ldots, d_m\}$, where d_i represents the i-th dropout rate. For a given dropout rate d_i, we compute class representations z_i from the MLP. Once we obtain m distinct class representations (logits), we derive the final representation z by averaging these representations, as defined by the equation:

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

3.2.3 Attention-Based Head

For this classification head, once contextual representations H are obtained for a sentence S, an additional attention layer is added to compute learnable attention scores α_i for each token t_i in H, and its calculation is as follows:

$$\alpha_i = \textit{softmax}(W \cdot h_i + b), \quad i = 1, 2, \ldots, n$$

This results in a set of $\textit{attention\_scores} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ corresponding to the tokens in sentence S. These attention scores collectively represent the overall attention distribution across the sentence, indicating the relative importance or relevance of each token to the context of the entire sentence. After finding attention scores for each token, we find the context vector for the sentence S by multiplying the contextual representations of token t_i with its attention score α_i.

$$c = \sum_{i=1}^{n} \alpha_i \cdot h_i$$

After obtaining the context vector, it is further processed through a linear layer to perform the classification task.

4 Result and Analysis

During the development phase, we conducted various experiments, all of which are detailed in Appendix B. The experiment setup and hyperparameter specifics can be found in Appendix A. Our experiments for model selection encompassed a wide range, including machine learning models (SVM, RandomForest, XGBoost) with TF-IDF feature extraction, deep learning models (LSTM, LSTM+Attention), and multilingual Transformer models (mBERT, mDeBerta, XLM-Roberta base). Notably, mDeBerta exhibited superior performance. Additionally, we evaluated two Bangla language models, with the *csebuetnlp-BanglaBERT* model emerging as the top performer. For a concise summary of the experimental outcomes related to model selection, please refer to Table 4.

It's important to note that the LSTM and LSTM+Attention models were trained for 5 epochs, while all transformer-based models underwent finetuning for 3 epochs with the utilization of the [CLS]-based classification head, as detailed in Section 3.2.1.

Table 3 displays the main experimental findings. Each experiment employed a 5-fold cross-validation technique, with the Macro-F1 score as

Techniques	Classification Head	CV Score	Performance Metrics			
			Dev Set		Test Set	
		Macro F1	Accuracy	Macro F1	Accuracy	Macro F1
Without Adding New Tokens	CLS + MLP	79.13	82.63	80.82	80.01	75.96
	Dropouts Enhanced MLP	***79.26***	***81.80***	***80.07***	***80.10***	***76.04***
	Attention Pool	79.76	82.26	80.20	80.36	76.59
Including Dev Dataset	CLS + MLP	80.45	-	-	78.77	74.51
	Dropouts Enhanced MLP	80.49	-	-	78.52	74.14
	Attention Pool	80.29	-	-	79.86	75.80
With Addition of New Tokens	CLS + MLP	79.28	**83.38**	**81.55**	**80.86**	76.76
	Dropouts Enhanced MLP	79.20	82.78	80.94	80.31	76.79
	Attention Pool	79.39	82.86	80.60	80.65	**76.90**

Table 3: Performance of different classification heads at the top of the BanglaBERT with different techniques is shown here. ***Dropouts Enhanced MLP*** without new token addition indicates that the best performing model scores that were submitted to the competition. All the experiments with new token addition techniques and attention-based heads weren't submitted to the competition. But the experiments with new token addition + attention-based classification head give the beat the top leaderboard score, which is marked in underline.

Model Name	Acc ↑	F1 ↑
TF-IDF + SVM	62.26	53.76
TF-IDF +RandomForest	61.88	51.76
TF-IDF + XGBoost	62.83	52.49
LSTM	67.89	62.37
LSTM + Attention	70.76	66.31
mBERT-case	72.33	68.06
mDeBerta-v3 base	75.04	72.27
XLM-Roberta base	73.61	71.68
SagorSarker-BanglaBERT	71.35	67.63
csebuetnlp-BanglaBERT	**81.20**	**79.12**

Table 4: Different Types of Model Performance in Validation (Dev) Dataset. Epoch Size 3

the evaluation metric. Three distinct scenarios were examined with different classification heads, as outlined in Section 3.2. In the first scenario, the use of new token additions (described in Section 3.1) was omitted. In this context, we observed that the *CLS+MLP* configuration outperformed others in the development set. However, the *Dropouts Enhanced MLP* head demonstrated notable improvements, not only in cross-validation scores but also in the test set performance. The Attention based head had showed significant enhancements in both cross-validation scores and test set results, despite a slightly lower performance in the development set. Interestingly, incorporating the development dataset with the training dataset did not yield superior results in the test dataset, despite achieving a better cross-validation score.

Fascinating findings emerged when we incorporated new token additions as new special tokens (described in Section 3.1) into the pretrained BanglaBERT vocabulary. The words that are considered as new tokens are mentioned in Appendix C. In this experiment, we observed approximately a 1% improvement in both dev set and test set performance, measured by accuracy and macro-F1 metrics across all heads. Remarkably, the Attention pool combined with the addition of new tokens yielded the highest macro-F1 score. Notably, the *Dropouts Enhanced MLP* model without the new tokens addition , which secured the **2nd position on the leaderboard**, emerged as the top-performing model among the submissions.

All heads with new token addition and attention pool head without new tokens addition beat the top leaderboard score, which was 76.044% macro-F1 score. Unfortunately, those models weren't submitted during the competition. The models that beat the top leaderboard score are marked as underlined in Table 3.

5 Conclusion

In this study, an analysis is done when we add dataset-specific tokens (most frequent) to the pretrained vocabulary of BanglaBERT for which the BanglaBERT tokenizer gives the *[UNK]* token.

The addition tokens learn their embeddings during the finetuning. From the experiment, it has been seen that the addition of those type of tokens boosts the model's performance. To enhance the model prediction's further, different classification heads are applied.

Limitations and Future Plan

The aforementioned approach, adding dataset-specific most frquent tokens for which the pre-trained tokenizer gives *[UNK]* tokens, helps in this task. A proper investigation is needed to analyse if this approach performs better in some other tasks.

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.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. 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.

Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. 2009. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Md Saroar Jahan, Mainul Haque, Nabil Arhab, and Mourad Oussalah. 2022. BanglaHateBERT: BERT for abusive language detection in Bengali. In *Proceedings of the Second International Workshop on Resources and Techniques for User Information in Abusive Language Analysis*, pages 8–15, Marseille, France. European Language Resources Association.

Muhammad F Mridha, Md Anwar Hussen Wadud, Md Abdul Hamid, Muhammad Mostafa Monowar, Mohammad Abdullah-Al-Wadud, and Atif Alamri. 2021. L-boost: Identifying offensive texts from social media post in bengali. *Ieee Access*, 9:164681–164699.

Anmol Nayak, Hariprasad Timmapathini, Karthikeyan Ponnalagu, and Vijendran Gopalan Venkoparao. 2020. Domain adaptation challenges of BERT in tokenization and sub-word representations of out-of-vocabulary words. In *Proceedings of the First Workshop on Insights from Negative Results in NLP*, pages 1–5, Online. Association for Computational Linguistics.

Sourav Saha, Jahedul Alam Junaed, Arnab Sen Sharma Api, Nabeel Mohammad, and Mohammad Ruhul Amin. 2023a. Vio-lens: A novel dataset of annotated social network posts leading to different forms of communal violence and its evaluation. In *Proceedings of the 1st International Workshop on Bangla Language Processing (BLP-2023)*, Singapore. Association for Computational Linguistics.

Sourav Saha, Jahedul Alam Junaed, Nabeel Mohammed, Sudipta Kar, and Mohammad Ruhul Amin. 2023b. Blp-2023 task 1: Violence inciting text detection (vitd). In *Proceedings of the 1st International Workshop on Bangla Language Processing (BLP-2023)*, Singapore. Association for Computational Linguistics.

Sagor Sarker. 2020. Banglabert: Bengali mask language model for bengali language understading. *textsIGitHub*.

David Yenicelik, Florian Schmidt, and Yannic Kilcher. 2020. How does BERT capture semantics? a closer look at polysemous words. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 156–162, Online. Association for Computational Linguistics.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2022. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*.

A Experimental Setup and Hyperparameters

In this research endeavour, various text preparation procedures, such as eliminating punctuation, emojis, numeric characters, and potential web addresses, were implemented. LSTM and Bert were used as the text encoders. Their generated representation were used as the hidden layer representation for the text. Different models including TF-IDF combinations, LSTM variants and Bert variations were applied on Dev dataset. Then utilizing the representation, different models

including combinations involving TF-IDF, various LSTM variants, and variations of Bert were used in this study.

LSTM-based models including standalone LSTM and LSTM+Attention used embedding dimension of 128 for an embedding layer.The models include hidden dimension of 256, learning rate of 10^{-3}, batch size of 16 for this configuration. As it can be seen from table 7, this batch size bested all other variation for Dev dataset. Thus, batch size 16 was consider for the models to gain the optimal performance for all the models.

For the BERT model used in this study, we utilized the *Bangla BERT* variant that enables us to extract contextual representations and long-term dependency through fine-tuning and pretraining. In this case, the hidden dimension of the BERT model was set to 768. The learning rate for BERT was 2×10^{-5}, maximum token length was 64 and batch size was 16. From table 5, it can be seen that the Bert models outperformed all other variations for token length 64. Therefore, token length 64 was considered.

Both configurations included 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 λ values is presented in Table. All experiments were conducted using Python (version 3.10) and PyTorch, leveraging the free NVIDIA Tesla T4 GPU available in Google Colab, as well as a single NVIDIA Tesla P100 GPU provided by Kaggle.

B Ablation Study

This section represents the ablation studies performed in this study. Which includes token cutoff analysis, batch size effect analysis and Loss analysis. For each of the analysis we compare the variations in terms of the optimal value of accuracy and F1 score.

B.1 Token Length Effects

In this study, different token length were considered to gain the highest performance for all the performance matrices. Token length 512 showed poor performance for the implemented models. The value of token length 64 outperforms all other variations, while improving the lowest accuracy and F1 score by 1%-2%. Rest of the variations slightly lags behind.

Token Length	Dev Acc ↑	Dev F1 ↑
32	80.15	77.28
64	**80.75**	**78.35**
128	80.32	77.92
256	80.3	77.36
512	79.1	76.72

Table 5: Token Cutoff Experiment of *csebuetnlp/BanglaBert* in Validation (Dev) Dataset.

B.2 Batch Size Influence

Batch size effects were also considered to gain optimal results. Batch size 16 showed superior performance than all other variations. It bested the lowest performance of batch size 64 by 2% in accuracy, and 4% in F1 score. While batch size 8 and 16 closely tails behind in terms of performance matrices.

B.3 Loss Analysis

Loss has significant effects on overall outcome of the study. Thus, a detailed investigation was performed on different variant of loss. Cross Entropy loss bested all other loss variations. The Cross Entropy Loss and Focal loss variation (0.5*Focal + 0.5*CE) performed vary poorly among the losses. Weighted Cross Entropy Loss showed slight improvement. While The standalone Cross Entropy loss, and the combination of Cross Entropy Loss and Focal loss (0.3*Focal + 0.7*CE) showed improvement by 1% to 3% from the lowest values of accuracy and F1 score for all the performance matrices. Finally, the Cross Entropy Loss showed superior performance to all other variations.

C The words which are considered as new tokens

As per discussion in Section 3.1, it is very challenging to figure out the words for which the tokenizer is giving [UNK] tokens. The reason behind this, the tokenizer that BanglaBERT uses sub-

Preprocessing	Dev Acc ↑	Dev F1 ↑
No Preprocessing	80.30	77.92
Removing Punctuation & Emoji's	79.10	76.72
Removing Emoji's Only	78.50	76.23
Removing Punctuation Only	78.95	75.96
Adding Normalizer in the text	81.20	79.12
Adding BN-Unicode Noramlizer	80.15	79.00
Converting Emoji's into Text	80.90	78.44
Adding most frequent [UNK] tokens as new tokenizers	**83.01**	**81.40**

Table 6: Effect of different preprocessing in dev set. For experiment BanglaBERT is used with 4 epochs of training

Batch Size	Dev Acc ↑	Dev F1 ↑
8	79.77	77.59
16	**80.08**	**77.99**
32	80	76.95
64	78.12	73.22

Table 7: Batch Size Effect of *csebuetnlp/BanglaBert* in Validation (Dev) Dataset while Token Length = 64 were considered. Epoch Size 5

word tokenization for which the no. of tokens and white space based basic tokenizer word list of a sentence aren't equal. To takcle this issues we extracted those samples for which token legnth from tokenizer is equal to length white space based basic tokenizer word list. We find only 531 samples considering both train and validation dataset for which the condition is followed. For those 531 samples, the most frequent words for which the BanglaBERT tokenizer gives *[UNK]* tokens. The tokens that are considered as new tokens is shown in Table 8.

D Preprocessing Analysis

In Table 6, several experiment are done with different preprocessing techniques. The table shows that punctuation of emoji's carry some contextual information while classifying the texts. So, removing them didn't help the model. For bangla text, normalizer plays a vital role. Two differnet normalization techniques were experimented where *csebuetnlp/normalizer* proven effective rather than BN-unicode normalizer. Another experiment were done converting emoji's into but it didn't help. Finally, adding most frquent words for which pretrained BanglaBERT gives [UNK] tokens become more helpful for the model.

Word	Count
হয়	21
হয়ে	21
সময়	21
দেওয়া	17
মিডিয়া	16
নিয়ে	14
বয়কট	13
যায়	12
আওয়ামী	11
ভয়	10
বড়	10
হায়রে	9
দেয়া	8
দায়	8
আওয়ামিলীগ	8
হয়েছে	7
দিয়ে	7
এগিয়ে	7

Table 8: The list of words that are considered as new tokens to the model