Optimizing BERT for Nepali Text Classification: The Role of Stemming and Gradient Descent Optimizers

Arjun Singh Saud
Central Department of Computer Science and IT,
Tribhuvan University, Kirtipur.
arjunsaud@cdcsit.edu.np

Ajanta Dhakal
School of Mathematical Sciences,
Tribhuvan University, Kirtipur.
Patan Multiple Campus

Received: 5 May, 2025 / Accepted: 8 June, 2025 / Published: 9 July, 2025

Abstract

This study investigates the use of BERT for classifying Nepali news articles, addressing the specific challenges associated with Nepali as a low-resource language in natural language processing (NLP). While traditional text classification methods have proven effective for high-resource languages, they often fall short in capturing the contextual nuances necessary for accurate classification in Nepali. To address this gap, a pre-trained BERT model was fine-tuned on a balanced dataset of Nepali news articles sourced from various outlets. The study examined the effects of different preprocessing techniques, such as stemming, and optimization algorithms including Adam, AdamW, and Momentum, on classification performance. Experimental results demonstrate that the combination of stemming and the AdamW optimizer yielded the best performance, achieving a weighted accuracy of 93.67%, along with balanced macro precision, recall, and F1-scores of 0.94. These findings underscore the effectiveness of advanced optimization strategies-particularly AdamW-in enhancing model performance.

Keywords—BERT, Natural Language Processing, Nepali News Classification, Stemming, AdamW.

1. Introduction

Text classification is a fundamental Natural Language Processing (NLP) task that assigns predefined categories to text based on its content. By automating the organization of large-scale textual data, it plays a vital role in managing today's information-driven world. For instance, news platforms use classification to sort articles into domains like politics, sports, technology etc. This capability powers diverse applications from streamlined content management to personalized recommendations enhancing how users and systems interact with vast amounts of data (Kowsari et al., 2019; Minaee et al., 2021).

Traditional text classification methods primarily depended on manual rules or keyword-based approaches. While simple to implement, these techniques fail to capture the deeper semantic and contextual nuances of language. For example, relying solely on keywords can lead to misclassification, particularly when dealing with complex or ambiguous expressions. The advent of machine learning introduced more advanced models like Support Vector Machines (SVM), Naive Bayes, and decision trees, which improved classification by learning linguistic patterns from data. However, these models still relied on hand-engineered features such as bag-of-words or n-grams—that treat text as mere word frequencies without understanding meaning. Consequently, they perform poorly on polysemy or sentences where context determines interpretation (Gasparetto et al., 2022).

Recent advancements in deep learning, particularly neural networks and transformer architectures, have revolutionized text classification (Fields et al., 2024). Models like BERT (Bidirectional Encoder Representations from Transformers) leverage bidirectional context to interpret word meanings with remarkable precision, eliminating the need for manual feature engineering. Pre-trained on vast text corpora, BERT captures intricate semantic relationships, enabling state-of-the-art performance in understanding nuanced language (Garrido-Merchan et al., 2023).

Despite the success of advanced NLP models, their application to under-resourced languages like Nepali remains challenging. Limited labeled datasets, scarce pre-trained embedding’s, and a lack of specialized linguistic tools hinder the development of accurate classification systems. This study investigates the adaptation of transformer-based models specifically BERT for Nepali news classification under such constraints. By fine-tuning BERT on a curated Nepali news corpus, we aim to develop a robust classification framework capable of addressing the linguistic nuances and resource limitations unique to Nepali. The primary objective of this study is to evaluate the impact of stemming techniques and optimizer selection (Momentum, Adam, and AdamW) on BERT's performance for text classification. Through comparative analysis of accuracy, precision, recall, and F1-score, we aim to identify optimal configurations for effective model tuning.

2. Related Theories and Models

2.1. Transformer

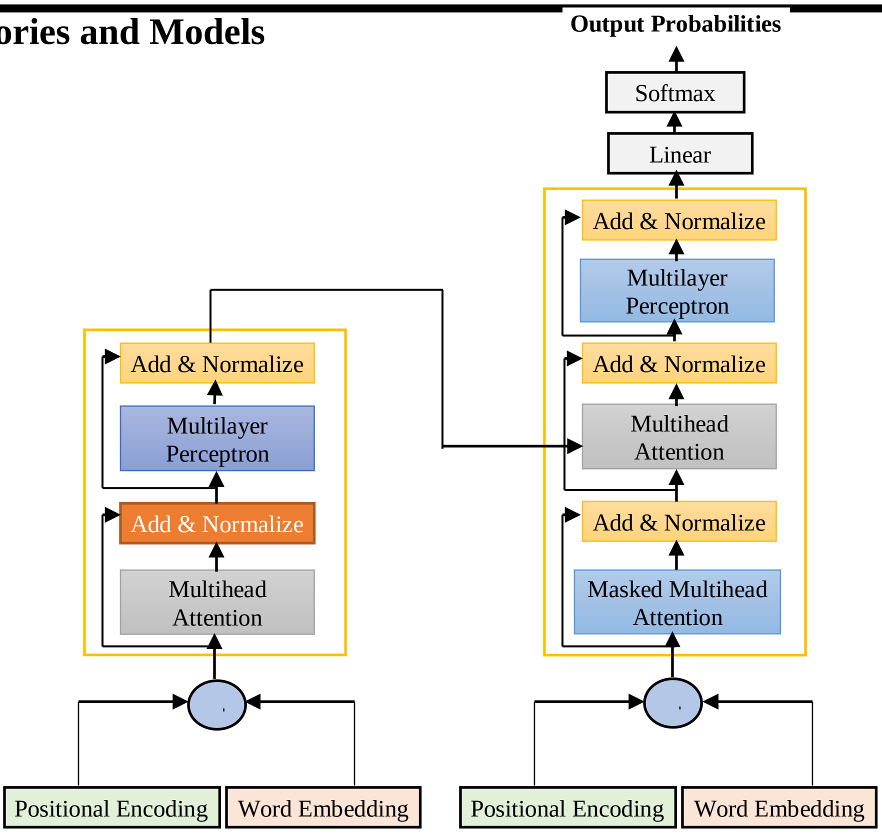

Figure 1: Architecture of Transformer

The Transformer architecture employs stacked encoder-decoder layers with self-attention mechanisms to effectively map input sequences to output sequences (Vaswani et al., 2017). Identical encoder layers process the input sequentially, while decoder layers iteratively generate outputs by attending to both the final encoder representation and previous decoder states. Figure 1 illustrates this architecture.

Word embeddings provide dense, low-dimensional vectors that capture meaning by placing similar words closer in vector space. Popular techniques like Word2Vec (with CBOW and Skip-Gram models) generate such embeddings effectively. For sequential data, positional embedding is essential—especially in parallel-processing models like Transformers—to retain word order information. Without it, Transformers would treat reordered sentences (e.g., "The cat sat on the mat" vs. "The mat sat on the cat") as identical. Common implementations, such as Sinusoidal Positional Encoding, explicitly encode token positions to address this limitation.

$$p(k,2i) = \sin\left(\frac{k}{n^{\frac{2i}{d}}}\right) \text{ and } p(k,2i+1) = \cos\left(\frac{k}{n^{\frac{2i}{d}}}\right) \tag{1}$$

Where, k is position of word in the input sequence, d is dimension of output embedding space, i is dimension index ($0 \leq i < d/2$), n is user defined scalar value which set to 10,000 in original paper (Vaswani et al., 2017) and p(k,j) is function for mapping position k of input sequence to index (k, j) of positional matrix.

The Transformer encoder, introduced by Vaswani et al. (2017), stacks multiple identical layers (originally six) to progressively refine input sequences into richer contextual representations. Each layer contains two key components: (1) a multi-head self-attention mechanism that captures global dependencies between all words in the sequence, and (2) a feedforward neural network that further processes these representations. For training stability, both components employ residual connections and layer normalization, which mitigate vanishing gradients and accelerate convergence.

The encoder's multi-head self-attention mechanism dynamically models relationships between all input sequence tokens, enabling each word to interact with and weigh the importance of every other word as formalized in Eq. (2). This approach captures long-range dependencies and nuanced contextual patterns by computing adaptive attention weights, unlike traditional methods that process words in isolation.

$$Attntion(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V \qquad (2)$$

Where, Q is query, K is key, V is value and d is dimensionality of key vector.

The self-attention mechanism represents each input word using three vectors: queries (Q) to seek relevant context, keys (K) to identify matching patterns, and values (V) to provide content for synthesis. The model computes attention scores by comparing queries to keys, then weights the corresponding values to generate context-aware representations. Multi-head attention extends this by running multiple parallel attention heads, each capturing distinct linguistic relationships (e.g., syntax, long-range semantics). Their combined output, formalized in Eq. (3), creates a richer understanding of the input sequence by integrating diverse contextual perspectives.

$$\begin{aligned} &Multihead(Q,K,V) \\ &= concat(h_{1,}h_2 \ldots\ldots h_k)W_0 \\ &h_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \qquad (3)$$

Where, all W are weight matrices

The Add & Normalize layer in Transformers integrates residual connections and layer normalization to stabilize deep networks. Residual connections (inspired by ResNet) enable direct gradient flow by adding the original input to transformed outputs as illustrated in Eq. (4) (He et al., 2016) mitigating vanishing gradients. Layer normalization standardizes activations for smoother training. Together, they enhance information preservation across layers in both encoder and decoder stacks, ensuring robust gradient propagation—a key innovation for training deep architectures effectively.

$$y = f(x) + x \qquad (4)$$

Where, $f(x)$ is target function.

To compute the gradients, we calculate the partial derivative of the loss function with respect to the input x, as shown in Eq. (5). From the equation we can observe that even if gradient of ∂f(x) diminishes the overall gradient will not vanish due to an additional ∂L/∂y.

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} = \frac{\partial L}{\partial y}\left(\frac{\partial f(x)}{\partial y} + 1\right) \qquad (5)$$

Layer normalization then standardizes these combined outputs, maintaining stable activations and accelerating convergence. Together, these operations form a robust foundation that enables Transformers to learn efficiently despite their depth and complexity. This is achieved using Eq. (6).

$$\hat{x}_i = \frac{x_i - \mu_i}{\sigma_i + \varepsilon} \qquad (6)$$

After normalization, it is passed through a multilayer perceptron, which typically consists of a linear layer followed by a ReLU activation layer. This network introduces non-linearity and captures complex patterns in the data. The Decoder in Transformers mirrors the Encoder but integrates masked multi-head attention to enforce autoregressive generation. During training, this mechanism prevents the model from accessing future tokens by masking positions beyond the current step ($j > i$). Specifically, future token attention scores are set to $-\infty$, forcing their softmax outputs to zero. This ensures the model generates text sequentially—relying only on past and current tokens—and maintains causality in predictions.

The transformer computes scores for every word in the vocabulary, with higher scores indicating greater likelihood of being the next word in the sequence. These scores are processed by a linear layer (acting as a classifier) whose dimension equals the vocabulary size. Finally, a softmax layer normalizes the scores into probabilities between 0 and 1, as given by Eq. (7).

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{i=1}^{k} e^{z_i}} \tag{7}$$

2.2. Bidirectional Representations from Transformer (BERT)

Devlin et al. (2019) introduced BERT, a model designed to pretrain deep bidirectional representations from unlabeled text. Unlike previous approaches, BERT jointly conditions on both left and right context across all layers, enabling it to capture comprehensive word context from surrounding words. This bidirectional architecture makes BERT highly effective for a wide range of natural language processing (NLP) tasks. BERT stacks multiple Transformer encoders, each utilizing self-attention mechanisms to dynamically weigh word importance, capturing intricate linguistic patterns. There are two main steps in BERT: pre-training and fine-tuning.

BERT's pretraining phase leverages two unsupervised objectives to develop its robust language understanding capabilities. The model first employs Masked Language Modeling (MLM), where it randomly masks 15% of input tokens and learns to predict them based on contextual clues, thereby mastering word-level semantics and syntactic relationships. Simultaneously, through Next Sentence Prediction (NSP), BERT analyzes pairs of sentences to determine their original sequential relationship, cultivating an understanding of discourse-level connections. This dual-task approach - combining fine-grained word analysis with broader textual comprehension - enables BERT to construct rich, bidirectional representations that surpass traditional language models in capturing linguistic subtleties. The resulting architecture forms a versatile foundation that can be fine-tuned for numerous downstream NLP applications while maintaining strong contextual awareness.

Following pre-training, BERT undergoes task-specific fine-tuning using labeled datasets tailored to particular NLP applications like text classification, named entity recognition, or question answering. During this phase, the model's pre-trained parameters are carefully optimized through supervised learning on domain-specific data, enabling it to adapt its broad linguistic knowledge to specialized tasks while maintaining its deep contextual understanding. This two-stage training approach -

combining general-purpose language pretraining with targeted fine-tuning - allows BERT to consistently achieve state-of-the-art performance across diverse NLP benchmarks, often surpassing previous task-specific architectures. The BERT architecture comes in two primary variants distinguished by their model depth: BERT-Base and BERT-Large. BERT-Base employs a 12-layer transformer block architecture, while BERT-Large utilizes a more complex 24-layer structure.

2.3. Stemming

Stemming is a basic NLP technique that simplifies words to their root form by cutting off prefixes and suffixes (e.g., "running" → "run"). It enhances text processing efficiency in tasks like search and sentiment analysis by grouping word variants. However, stemming can yield incorrect roots (e.g., "business" → "busi") because it uses rule-based trimming rather than deep linguistic analysis. Common algorithms like the Porter and Snowball stemmers vary in their truncation intensity. Though less precise than lemmatization, stemming is favored for its speed and simplicity in handling large datasets.

A Nepali Stemmer is an NLP tool that reduces words to their root forms by removing suffixes and inflections, which is particularly challenging due to Nepali's complex morphology involving conjugations, gender agreements, and derivational suffixes (Upadhyaya et al., 2021). Unlike English stemmers, Nepali stemmers often use rule-based or machine learning approaches customized for the language's grammar (e.g., “गर्दै”, “गर्छ” and “गरेको” stemmed to “गर”).

While essential for tasks like search and text classification in Nepali, current stemmers face difficulties with exceptions, compound words, and dialects.

2.4. Gradient Descent Optimizers

2.4.1. Momentum

Momentum enhances gradient-based optimization by calculating an exponentially weighted average of past gradients, which is then used for weight updates as shown in Eq. (8) instead of relying solely on the current noisy gradient. This approach helps the optimization algorithm follow a smoother, more direct path toward the minimum while reducing vertical oscillations, leading to faster and more stable convergence (Ruder, 2017).

$$\begin{aligned} v_t &= \beta v_{t-1} + (1-\beta) dw_t \\ w_{t+1} &= w_t + \alpha v_t \end{aligned} \tag{8}$$

2.4.2. Adaptive Moment Estimation (Adam)

Adam is an advanced optimization algorithm widely used in deep learning that combines the benefits of momentum and adaptive learning rates. It works by computing exponentially decaying averages of both past gradients (first moment, like momentum) and squared gradients (second moment, like RMSProp). These estimates are used to adaptively adjust the learning rate for each parameter as shown in Eq. (9), allowing for faster convergence and better handling of sparse gradients. Adam also includes bias correction to account for initialization effects in the early stages of training (Ruder, 2017).

$$\begin{aligned} v_t &= \beta_1 v_{t-1} + (1-\beta_1) dw_t \\ s_t &= \beta_2 s_{t-1} + (1-\beta_1) dw_t^2 \\ w_{t+1} &= w_t + \frac{\alpha}{\sqrt{s_t + \epsilon}} v_t \end{aligned} \tag{9}$$

2.4.3. Adam with Weight Decay (AdamW)

In standard Adam, L2 regularization, often implemented as weight decay, is added to the loss function, which can affect the adaptive learning rates, potentially hindering optimal convergence. AdamW decouples weight decay from the gradient update step, applying it separately after the gradient update, leading to more stable training and better generalization, especially for large models (Loshchilov & Hutter, 2019).

3. Related Works

Delvin et al. (2019) introduced Bidirectional Encoder Representations from Transformers (BERT), a pre-trained language model based on transformer architecture that excels at capturing contextual relationships between words in a text sequence. This model achieved state-of-the-art performance across multiple natural language processing (NLP) tasks, particularly in text classification. The study demonstrated that fine-tuning BERT for specific downstream tasks led to substantial performance improvements.

Alam et al. (2020) fine-tuned multilingual transformer models for Bangla text classification across multiple domains, including sentiment analysis, emotion detection, news categorization, and authorship attribution. Their approach achieved state-of-the-art performance on six benchmark datasets, surpassing previous methods by 5–29% in accuracy across different tasks.

Shaheen et al. (2020) evaluated multiple transformer-based models—including BERT, RoBERTa, DistilBERT, XLNet, and M-BERT—alongside strategies such as generative pretraining, gradual unfreezing, and discriminative learning rates to achieve competitive classification performance. Their work established new state-of-the-art results. When comparing architectures, the study demonstrated that transformers, with their attention mechanisms, outperform LSTMs in identifying classification-relevant aspects, particularly in long documents.

Subba et al. (2019) compared the performance of Recurrent Neural Networks (RNNs) and Multilayer Neural Networks (MNNs) for Nepali news document classification. Their experiments showed that RNNs consistently outperformed MNNs, achieving a peak accuracy of 63% compared to MNNs' 48%. Notably, while MNNs exhibited significant variability with accuracy dropping as low as 4.3%, RNNs maintained more stable performance, with their lowest accuracy at 59%.

Singh (2018) performed a comparative study on Nepali news classification, evaluating traditional machine learning algorithms against deep learning approaches. The study assessed traditional classifiers—including Logistic Regression (LR), Support Vector Machine (SVM), Multinomial Naive Bayes (MNB), Bernoulli Naive Bayes (BNB), K-Nearest Neighbors (KNN), and Multi-Layer Perceptron (MLP)—against deep learning models such as Recurrent Neural Networks, LSTM, GRU, and Adaptive GRU architectures. All models utilized Word2Vec embeddings for text representation.

Timilsina et al. (2022) developed NepBERTa, a specialized BERT-based model trained on the largest monolingual Nepali corpus to date – containing 0.8 billion words from 36 major Nepali news sources. Their results demonstrate that NepBERTa substantially enhances performance across multiple NLP tasks, particularly in news classification, by effectively utilizing this rich linguistic dataset.

Munikar et al. (2019) leveraged BERT for fine-grained sentiment analysis of Nepali text, utilizing the Stanford Sentiment Treebank (SST) dataset. Their findings reveal that BERT surpasses conventional approaches—including convolutional and recursive neural networks—in detecting subtle sentiment nuances in Nepali language. This work highlights BERT's versatility, extending its effectiveness beyond news classification to broader NLP applications.

Joshi et al. (2019) performed a systematic evaluation of deep learning approaches for Hindi text classification, comparing CNN, LSTM, and attention-based architectures. The study also assessed multilingual embeddings, BERT and LASER, for this morphologically rich, low-resource language. Using translated English datasets, their work revealed transformer models with attention mechanisms consistently outperformed traditional CNN/LSTM approaches, offering important insights for processing Devanagari-script texts.

Dai et al. (2022) investigated transformer-based approaches for long document classification, tackling key computational challenges in processing extended texts. The study systematically compared two efficient strategies: (1) sparse attention, which reduces computational costs by focusing on token subsets, and (2) hierarchical encoding, which processes documents through segmented analysis. Their work not only demonstrated performance improvements across multiple datasets but also provided actionable guidelines for implementing these methods in long-document classification tasks.

Thapa et al. (2024) developed specialized transformer models for Nepali language processing by pre-training BERT, RoBERTa, and GPT-2 architectures on a newly compiled large-scale Nepali corpus. Their evaluation using the Nep-gLUE benchmark – covering diverse NLP tasks including text classification, sentiment analysis, and text generation – revealed these models substantially outperformed existing solutions. This work demonstrates the critical importance of language-specific pre-training for achieving state-of-the-art performance in low-resource languages like Nepali.

Maskey (2022) investigated auto-encoding transformer models (DistilBERT, DeBERTa, and XLM-R) for Nepali text classification, demonstrating their effectiveness in capturing linguistic nuances for tasks like sentiment analysis and news categorization. Through comparative evaluation, the study revealed these transformer architectures significantly improve classification accuracy over traditional approaches, underscoring their value for low-resource language processing.

4. Methodology

Figure 2 illustrates the schematic framework of the methodology employed in this study. The following subsections provide a concise overview of each component.

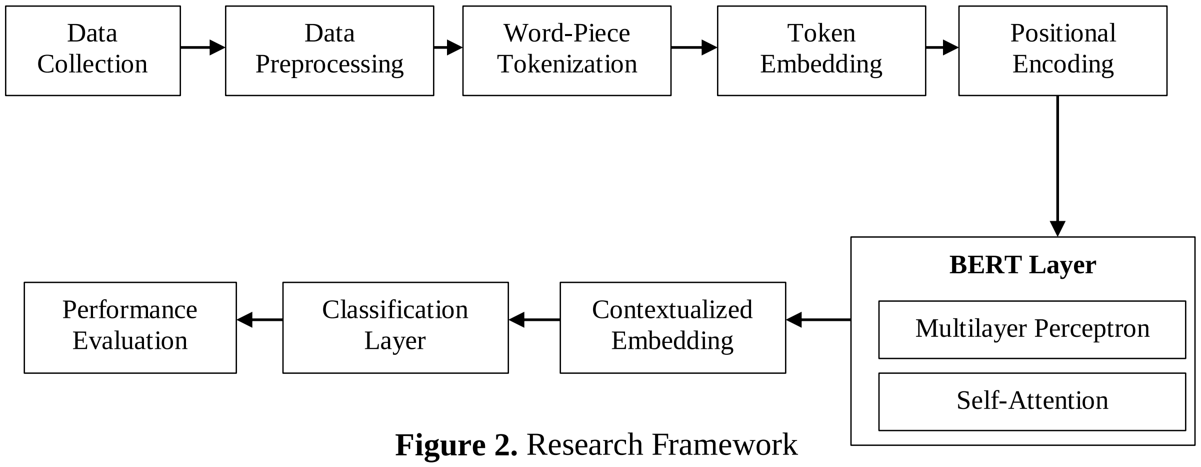

Figure 2. Research Framework

4.1. Data Collection

The research dataset was constructed by merging publicly available data from GitHub (Neupane, 2022) with articles scraped from prominent Nepali news platforms, including Ratopati, Ekantipur, Onlinekhabar, and Techpana. Initial analysis revealed a significant class imbalance, with categories like politics and sports containing disproportionately higher numbers of articles compared to underrepresented domains such as technology and education. To mitigate potential model bias and improve classification accuracy for minority classes, a uniform sampling approach was implemented. This involved selecting 2,500 articles from each of the six target categories—business, education, sports, health, technology, and entertainment—resulting in a balanced dataset of 15,000 articles for training.

4.2. Data Preprocessing

The scraped news data, comprising a mix of digits, English characters, and special symbols, underwent rigorous cleaning to retain only linguistically relevant Nepali text. The preprocessing pipeline included the following steps: Removal of Non-Nepali Elements, Whitespace Standardization, Stop Word Removal, Stemming, Lemmatization, Category Mapping of News articles. The balanced dataset was partitioned into training and testing subsets, following an 80-20 ratio. This division allocates 80% of the data for model training and reserves 20% for testing.

4.3. Hyperparameter Settings

In this study, we investigated the effect of different optimization algorithms on the performance of a BERT-based text classification model, keeping all other hyperparameters constant. Specifically, we compared the Adam, AdamW, and Momentum optimizers. A batch size of 32 was chosen to balance computational efficiency with generalization, and the number of training epochs was fixed at 3 to allow adequate learning while minimizing overfitting. For Adam and AdamW, a learning rate of 2e-5 and weight decay regularization were used. The Momentum optimizer was configured with a learning rate of 0.03 and a momentum value of 0.9. By holding all other hyperparameters fixed, this study aimed to isolate and analyze the impact of the optimizer on convergence speed, training stability, and classification accuracy in the context of news article categorization.

4.4. Performance Measures

The news classification strategies investigated in this research are evaluated using accuracy, precision, recall, and F1-score which are formulated as below.

$$accuracy = \frac{Number\ of\ correct\ predictions}{Number\ of\ total\ predictions} = \frac{TP+TN}{Total} \quad (11)$$

$$\text{Weighted Average Accuracy} = \sum_{i=1}^{N} w_i \times \text{Accuracy}_i \quad (12)$$

Where, N is the total number of classes, w_i is the weight of class i, calculated as the proportion of true samples for class i in the dataset and Accuracy_i is the accuracy for class i.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive+False Positive}} \quad (13)$$

$$\text{Macro Precision} = \frac{1}{N}\sum_{i=1}^{N} Precision_i \quad (14)$$

$$\text{Micro Precision} = \frac{\sum_{i=1}^{N}\text{True Positive}_i}{\sum_{i=1}^{N}(\text{True Positive}_i\text{+False Positive}_i)} \quad (15)$$

Where, N is the total number of classes and $Precision_i$ is the precision for class i.

$$\text{Recall(Sensitivity)} = \frac{\text{True Positive}}{\text{True Positive+False Negative}} \quad (16)$$

$$\text{Macro Recall} = \frac{1}{N}\sum_{i=1}^{N} Recall_i \quad (17)$$

$$\text{Micro Recall} = \frac{\sum_{i=1}^{N}\text{True Positive}_i}{\sum_{i=1}^{N}(\text{True Positive}_i\text{+False Negative}_i)} \quad (18)$$

Where, N is the total number of classes and $Recall_i$ is the recall for class i.

$$\text{F1 Score} = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision+Recall}} \quad (19)$$

$$\text{Macro F1 Score} = \frac{1}{N}\sum_{i=1}^{N} F1\ \text{Score}_i \quad (20)$$

$$\text{Micro F1 Score} = \frac{2*\text{Micro Precision}*\text{Micro Recall}}{\text{Micro Precision+Micro Recall}} \quad (21)$$

Where, N is the total number of classes and $F\ Score_i$ is the F1 for class i.

4.5. Tools Used

This research was implemented using Python and various libraries within the Google Colab environment, which offers free GPU support for machine learning tasks. Key tools included Pandas for data manipulation, scikit-learn for preprocessing and evaluation, Hugging Face Transformers for BERT-based text classification, and PyTorch for model training and data handling.

5. Results and Discussion

This study explores the use of BERT for Nepali news classification, combining various optimizers (Momentum, Adam, and AdamW) with stemming techniques to analyze their impact on model performance. The evaluation metrics included weighted average accuracy, macro-averaged precision, recall, and F1-score, enabling a comprehensive assessment of how stemming and optimizer selection influence the effectiveness of transformer-based models. These experiments highlight the interplay between text normalization strategies and optimization algorithms in enhancing BERT's capability for multilingual text classification tasks, offering practical insights for improving NLP workflows in low-resource languages.

Table 1. Performance of Various Models

Metric / Model	Weighted Avg. Accuracy	Macro Recall	Macro Precision	Macro F1
BERT-Momentum	87.87	0.88	0.88	0.88
BERT-St-Momentum	86.57	0.87	0.87	0.87
BERT-Adam	92.7	0.93	0.93	0.93
BERT-St-Adam	92.63	0.93	0.93	0.93
BERT-AdamW	91.87	0.92	0.92	0.92
BERT-St-AdamW	93.67	0.94	0.94	0.94

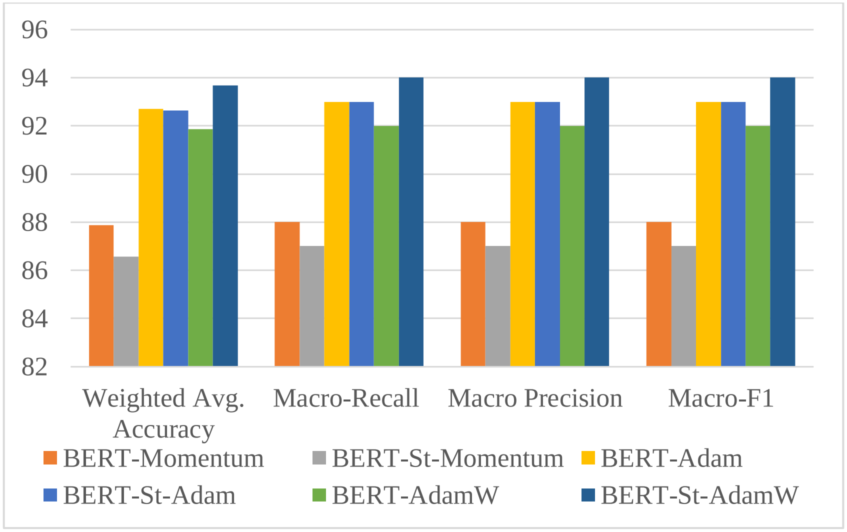

Figure 3. Performance of Various Models

For Model 1 (BERT with Momentum optimizer), Categories 2 (Sports) and 5 (Entertainment) achieved the highest classification accuracies at 92% and 93%, respectively. In contrast, Category 0 (Business) recorded the lowest accuracy at 78%, with 12% of its samples misclassified as Category 4 (Technology), likely due to semantic similarities between business and technology-related content. Additionally, 6% of samples from Category 1 (Education) were misclassified as Category 3 (Health), suggesting an overlap in terminology between educational and health-related topics. In the case of Model 2 (BERT with Stemming and Momentum), improvements were observed across several categories. Category 5 (Entertainment) achieved the highest accuracy at 95%, while Categories 0 (Business) and 4 (Technology) showed notable improvements, reaching accuracies of 87% and 88%, respectively. Despite these gains, misclassifications persisted—7% of Category 1 (Education) samples were misclassified as Category 0 (Business), and Category 3 (Health) attained an accuracy of 81%, with 10% of its samples misclassified as Category 4 (Technology). Based on the above observations, stemming did not improve performance when used with the Momentum optimizer, as evidenced by the slight decline in accuracy and other evaluation metrics compared to Model 1. This suggests that stemming may cause minor information loss, as reducing words to their root forms can eliminate important linguistic features.

For Model 3 (BERT with Adam), Category 5 (Entertainment) achieved the highest accuracy at 98%, followed by Category 3 (Health) at 96% and Category 1 (Education) at 95%, demonstrating the model's effectiveness in distinguishing these categories. Category 0 (Business) attained an accuracy of 84%, though it experienced notable misclassifications into Category 4 (Technology) at 6% and Category 1 at 3%, indicating some feature overlap. Category 2 (Sports) achieved 92% accuracy, with minor confusion involving Categories 1 and 4. Similarly, Category 4 (Technology) reached 91% accuracy, but with some misclassification into Categories 0 and 1. For Model 4 (BERT with Stemming and Adam), Categories 5 (Entertainment) and 3 (Health) achieved the highest accuracies at 98% and 96%, respectively, followed closely by Category 2 (Sports) at 95% and Category 4 (Technology) at 94%. However, Category 0 (Business) continued to pose challenges, with a lower accuracy of 80% and significant misclassifications into Category 4 (13%) and Category 1 (3%). Category 1 (Education) achieved a solid 93% accuracy, though it exhibited minor misclassifications into Categories 0 and 3.

In Model 5 (BERT with AdamW), Categories 1 (Education) and 4 (Technology) achieved the highest accuracies at 96%, closely followed by Categories 2 (Sports) and 5 (Entertainment) at 95%. Despite these strong performances, Category 0 (Business) remained challenging, with an accuracy of 78% and significant misclassifications into Category 4 (11%) and Category 1 (4%). Category 3 (Health) achieved a solid accuracy of 91%, though it showed minor misclassifications into Categories 1 and 4. Whereas in Model 6, Category 0 (Business) achieved an accuracy of 91%, with minor misclassifications into Category 4 (6%). Category 1 (Education) followed with 93% accuracy, showing slight misclassifications into Category 0 (3%) and Category 3 (2%). Categories 3 (Health), 4 (Technology), and 5 (Entertainment) recorded the highest accuracies at 95%, 96%, and 96%, respectively, demonstrating the model’s strong capability to effectively differentiate these categories.

The results clearly show that the AdamW optimizer outperformed both Momentum and Adam across all evaluation metrics. Models trained with AdamW consistently achieved higher weighted average accuracy, macro precision, macro recall, and macro F1-score compared to those using Momentum, and slightly outperformed models using Adam. Adam also demonstrated superior performance over Momentum in every metric, underscoring its effectiveness as an optimizer. These findings highlight AdamW as the most effective optimizer for training deep learning models in text classification tasks, with Adam as a strong alternative. The impact of stemming on model performance was mixed. For models using the Momentum optimizer, stemming led to a slight decline in performance, suggesting sensitivity to information loss. In contrast, stemming had minimal impact on models using the Adam optimizer, with metrics remaining largely stable. Notably, stemming enhanced performance when paired with the AdamW optimizer, indicating that advanced optimizers like AdamW can better leverage the simplified input resulting from stemming. This suggests that stemming is most beneficial when used alongside optimizers capable of effectively managing reduced linguistic complexity, particularly AdamW, and to a lesser extent, Adam.

6. Conclusion

The study conducts a detailed examination of optimizer efficacy and stemming influences on text classification models. Findings reveal that Adam and AdamW optimizers markedly outperformed Momentum, demonstrating consistently superior weighted average accuracy, macro precision, macro recall, and macro F1-scores across all experimental setups. While AdamW achieved marginally better results than Adam, both optimizers emerged as ideal choices for training deep neural networks in NLP tasks.

Stemming slightly diminished performance when paired with Momentum and showed minimal impact with Adam. However, AdamW models experienced significant performance gains when stemming was applied, suggesting this technique enhances outcomes specifically when combined with optimizers adept at managing simplified data structures, such as AdamW's adaptive gradient mechanisms.

The optimal configuration integrated stemming with AdamW, achieving 93.67% weighted accuracy and balanced metric performance. This combination exhibited strong generalization and operational efficiency, positioning it as a practical solution for Nepali news categorization. Notably, Adam also delivered robust results regardless of stemming, reinforcing its versatility in text classification workflows. These insights underscore the importance of aligning stemming with optimizer selection to maximize model effectiveness.

References

Alam, T., Khan, A., & Alam, F. (2020). *Bangla Text Classification using Transformers*. https://doi.org/10.48550/arxiv.2011.04446.

Dai, X., Chalkidis, I., Darkner, S., & Elliott, D. (2022). Revisiting transformer-based models for long document classification. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2204.06683.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (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, 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics. doi: https://doi.org/10.18653/v1/n19-1423.

Fields, J., Chovanec, K., & Praveen Madiraju. (2024). A Survey of Text Classification with Transformers: How wide? How large? How long? How accurate? How expensive? How safe? *IEEE Access*, 1–1. https://doi.org/10.1109/access.2024.3349952

Garrido-Merchan, E. C., Gozalo-Brizuela, R., & Gonzalez-Carvajal, S. (2023). Comparing BERT against traditional machine learning models in text classification. Journal of Computational and Cognitive Engineering, 2(4), 352–356. https://doi.org/10.47852/bonviewjcce3202838

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). A Survey on Text Classification Algorithms: From Text to Predictions. *Information*, *13*(2), 83.

https://doi.org/10.3390/info13020083

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. doi:https://doi.org/10.1109/cvpr.2016.90

Joshi, R., Goel, P., & Joshi, R. (2020). Deep Learning for Hindi Text Classification: A Comparison. *Intelligent Human Computer Interaction*, 94–101. https://doi.org/10.1007/978-3-030-44689-5_9.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text Classification Algorithms: A Survey. *Information*, *10*(4), 150. https://doi.org/10.3390/info10040150

Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. ArXiv.org. https://doi.org/10.48550/arXiv.1711.05101.

Maskey, U., Bhatta, M., Bhatt, S., Dhungel, S., & Bal, B. K. (2022). Nepali Encoder Transformers: An analysis of auto encoding transformer language models for Nepali text classification. ACL Anthology. https://aclanthology.org/2022.sigul-1.14/.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2021). Deep Learning--based Text Classification. ACM Computing Surveys, 54(3), 1–40. https://doi.org/10.1145/3439726

Munikar, M., Shakya, S., & Shrestha, A. (2019). Fine-grained Sentiment Classification using BERT. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.1910.03474.

Ruder, S. (2017). An Overview of Gradient Descent Optimization Algorithms. ArXiv.org. https://doi.org/10.48550/arXiv.1609.04747.

Shaheen, Z., Wohlgenannt, G., & Filtz, E. (2020, October 24). *Large Scale Legal Text Classification Using Transformer Models*. ArXiv.org. https://doi.org/10.48550/arXiv.2010.12871.

Singh, O. (2018). Nepali Multi-Class Text Classification. https://oya163.github.io/assets/resume/NepaliText_Classification.pdf.

Subba, S., Paudel, N., & Shahi, T. B. (2019). Nepali Text Document Classification Using Deep Neural Network. *Tribhuvan University Journal*, *33*(1), 11–22. https://doi.org/10.3126/tuj.v33i1.28677.

Thapa, P., Nyachhyon, J., Sharma, M., & Bal, B. K. (2024). Development of Pre-Trained Transformer-based Models for the Nepali language. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2411.15734.

Timilsina, S., Gautam, M., & Bhattarai, B. (2022). NepBERTa: Nepali Language Model Trained in a Large Corpus. 273–284. https://doi.org/10.18653/v1/2022.aacl-short.34.

Upadhyaya, B., Sharma, K., & Gurung, S. (2021). A Survey on Various Stemming Techniques for Hindi and Nepali Language. *Lecture Notes in Electrical Engineering*, 137–142. https://doi.org/10.1007/978-981-16-2911-2-14

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). Attention Is All You Need. ArXiv. https://arxiv.org/abs/1706.03762