Investigating the Effectiveness of Graph-based Algorithm for Bangla Text Classification

Farhan Noor Dehan, Md Fahim, Amin Ahsan Ali,
M Ashraful Amin, A K M Mahbubur Rahman
Center for Computational & Data Sciences
Independent University, Bangladesh
Dhaka-1229, Bangladesh
{1920269, md.fahim, aminali, aminmdashraful, akmmrahman}@iub.edu.bd

Abstract

In this study, we examine and analyze the behavior of several graph-based models for Bangla text classification tasks. Graph-based algorithms create heterogeneous graphs from text data. Each node represents either a word or a document and each edge indicates the relationship between any two words or word to document. We applied the BERT and different graph-based models including TextGCN, GAT, BertGAT, and BertGCN on five different Bangla text datasets including SentNoB, Sarcasm detection, BanFakeNews, Hate speech detection, and Emotion detection datasets. The performance with the BERT model surpassed the TextGCN and the GAT models by a large difference in terms of accuracy, Macro F1 score, and weighted F1 score. On the other hand, BertGCN and BertGAT outperformed the standalone graph models and the BERT. BertGAT excelled in the Emotion detection dataset and achieved a 1%-2% performance boost in Sarcasm detection, Hate speech detection, and BanFakeNews datasets from BERT's performance. Whereas BertGCN outperformed BertGAT by 1% for SentNoB and BanFakeNews datasets while beating BertGAT by 2% for Sarcasm detection, Hate Speech, and Emotion detection datasets. Furthermore, We examined different variations in graph structure and analyzed their effects.

1 Introduction

Natural language processing (NLP) has become very significant in recent years and text classification is one of the most crucial tasks in this domain. Text classification is the process of classifying text based on specific labels utilized in document categorization. It has applications in many diverse problems, including hate speech detection, spam detection, sentiment analysis, topic modeling, question answering, intent recognition, medical text analysis, legal document classification, social media analysis, fake news detection, and many more.(Zhou et al., 2020; Dwivedi and Arya, 2016; Patel and Mistry, 2015)

Graph algorithms can capture complex relationships and dependencies in various text structures and represent better semantic and syntactic relationships. Thus graph algorithms help create a more accurate understanding and interpretation of the text. (Wang et al., 2023) Moreover, these models can identify the grammatical relationship and contextual information between words and documents. By enabling these models to understand how the meaning of words changes based on context, for example, BertGCN performed better than Bidirectional Encoder Representations (BERT) (Devlin et al., 2018) and RoBERTa on different English datasets because of its contextual understanding. Graph-based models (Lin et al., 2021) can also identify sentiment-related relationships between words in sentiment analysis and can produce more accurate predictions based on sentiments. Furthermore, graph algorithms (Liang et al., 2022) can capture user interactions, mentions, and relationships during social media text analysis, hate speech detection, sentiment analysis, influence identification, and community recognition tasks. (Patel and Mistry, 2015; Akhter et al., 2018)

In recent times, extensive research has been conducted on Bangla text using different machine learning and deep learning models (Farhan et al., 2023; Bitto et al., 2023; Sadat Aothoi et al., 2023). However, graph-based Bangla text classification has remained largely unexplored. On the contrary, numerous graph-based models and structures have been implemented worldwide, particularly in English. The research works (Do et al., 2022; Zhang et al., 2023; Wu et al., 2023) motivate us to conduct our study on how different graph-based models with state-of-art models perform on Bangla datasets used for various tasks BERT. In this study, the previous best-performing models for each dataset were also observed alongside the implementation of various graph-based methods. Graph methods include Graph Convolutional Networks for text (TextGCN) (Yao et al., 2019), Graph Attention Networks(GAT) (Velickovic et al., 2017), BertGCN (Lin et al., 2021), and BertGAT (Lin et al., 2021). These algorithms were applied to five datasets containing SentNoB, Sarcasm detection, BanFakeNews, Hate speech detection, and Emotion detection datasets.

In this study, BERT outperformed the previously leading models and demonstrated significantly superior performance compared to GCN and GAT across all the datasets. BertGAT surpassed BERT's performance by 1%-5% for all datasets except SentNoB. However, BertGCN achieved better performance by 1%-2% over BertGAT for all datasets, which was attributed to BertGCN's superior ability to comprehend local contextual and global semantic relationships (Lin et al., 2021). As part of the ablation study, using BERT embeddings as node features showed better performance for GAT and GCN, while one-hot embeddings performed better for the integrated models. Two types of edge sets: document-word only (d2w) and document**to**word + wordtoword (d2w+w2w) were utilized for all the graph-based models. The edge set d2w+w2w outperformed all the models by 1%-2%. BertGCN outperformed all other models in this study. This research also aimed to find the right balance between graph models and BERT. BertGCN exhibited the highest performance when λ ranged from 0.3 to 0.7. BertGAT showed better performance when λ ranged from 0.1 to 0.5. Ultimately, this study was conducted to demonstrate that graph-based models can outperform traditional models for Bangla text classification and pave the way for future research in this domain.

Our main contributions are:

- We compare different graph methods for text classification and compare them with BERT. This study might be the first to compare graph-based models for Bangla text classification.

- We analyze results with several benchmarks of the datasets. We perform result analysis, ablation study, and identify the best graph-based models for Bangla text classification.

2 Related Work

Text classification is one of the classical problems for NLP. Naive Bayes, SVM, and other orthodox approaches for text classifications faced challenges in effectively learning meaningful text representations. Addressing these constraints, the application of deep learning models such as Convolutional Neural Networks (CNNs) (Kim, 2014) and Recurrent Neural Networks (RNNs) (Sherstinsky, 2020), materialized. These models demonstrated the ability to capture intricate features from text data.

BERT from Transformers achieves superior performance on sentence-level and token-level tasks (Devlin et al., 2018). Global structure refers to the information of the whole document, and graph-based models utilize an adjacency matrix to capture this information. To address the constraint of BERT models, researchers explored the usage of Graph Neural Networks (GNNs) and graph embeddings. Some of the most used GNNs are Graph convolutional network (GCN)(Kipf and Welling, 2016), GAT, Graph Sample and Aggregated (Hamilton et al., 2017), and, MoNet (Thekumparampil et al., 2018). One such model used for text classification is the TextGCN. TextGCN focuses on global word co-occurrence information. However, TextGCN has significant drawbacks, including a small receptive field, a lack of edge characteristics, over-smoothing, and an inability to accommodate varying neighborhoods. GAT overcame these restrictions by utilizing self-attentional layers. The limitations associated with pretraining in GCN and GAT are noteworthy. To address this, an innovative approach known as BertGCN has been developed, strategically amalgamating the advantages of BERT and GCN. BertGCN has the power of Large-scale pretraining on enormous unrefined data. In addition, by spreading label influence through graph convolution, transductive learning concurrently learns representations for training data and unlabeled test data.(Lin et al., 2021)

In recent years, there has been a significant amount of research conducted on Bangla text classification because of its importance. Inverse class frequency along with TF-IDF (Dhar et al., 2018) was proposed for Bangla text classification. Machine learning algorithms such as Naive Bayes, J48, KNN, and SVM were also used for Bangla texts (Akhter et al., 2018; Chy et al., 2014). Alam and Islam (2018) used Logistic Regression, SVM, LIBLINEAR, and Neural Networks for a massive volume of data with 300k samples in datasets. Transformer was used on six different datasets in Bangla by Alam et al. (2020). Ahmed et al. (2022); Rahman and Chakraborty (2021) implemented deep learning RNN attention layer and RNN with BiLSTM. Furthermore, the BERT and ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) models were tested by Rahman et al. (2020). However, graph-based models are very rarely used in Bangla text classification, which motivated our investigation to check how graph-based algorithms perform on different Bangla text datasets.

3 Methodology

Graph-based models use graphs as the representation of textual information. These models build their graphs based on specific criteria. Usually, in these text-derived graphs, individual words and entire documents serve as nodes, while connections (edges) represent co-occurrences, semantic similarity, grammatical dependencies, or any other relevant relationship among the nodes.

3.1 Building Graph from Text

The construction of the graph serves as a pivotal precursor, setting the stage for essential operations like message passing and aggregation in the graph-based framework. Textual data is transformed into graph structures, establishing an organized format that graph-based models can directly process. Graph-based models produce embedding vectors for nodes by considering the characteristics of their neighboring nodes within the graph. Similar to figure 1, the graph-building process starts by pre-processing the textual data. Then, unique words and the entire document are converted into a set of nodes V for each document, while every node is assumed to be connected to other nodes and itself (creating a self-loop), meaning that for any node v there exists an edge $(v, v) \in E$ in the graph. The edges connect word nodes belonging to the same document and that document's document node. The connections (edges) E represent any relevant relationship between the nodes. Formally, a graph is denoted as $G = (V, E)$, where $V(|V| = N)$ represents the set of N number of nodes and E represents the set of edges within the graph.

Next, the graph-based models take a few things from the created graph. The models consider nodes with their associated features, while the relationships or edges between nodes are captured by the adjacency matrix A. Adjacency matrices represent relationships between nodes using a binary matrix of size $N \times N$ for the size of N number of nodes. During this time, edge weights (usually co-occurrence for words and TF-IDF for documents) of the graph are calculated. So, formally, the adjacency matrix is,

$$A_{i,j} = \begin{cases} \text{PMI}(i,j), & \text{if } i, j \text{ are words} \\ \text{TF-IDF}(i,j), & \text{if } i \text{ is doc \& } j \text{ is word} \\ 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The co-occurrence between two words is calculated using the PMI value. The PMI value of a word pair i, j is computed as follows:

$$\mathit{PMI}(i,j) = \max(\log \frac{p(i,j)}{p(i)p(j)}, 0) \tag{2}$$

$$p(i,j) = \frac{\#W(i,j)}{\#W} \tag{3}$$

$$p(i) = \frac{\#W(i)}{\#W} \tag{4}$$

$$p(j) = \frac{\#W(j)}{\#W} \tag{5}$$

Where $\#W(i)$ is the number of sliding windows in a corpus that contains word i, $\#W(j)$ is the number of sliding windows in a corpus that contains word j, $\#W(i,j)$ is the number of sliding windows that contain both word i and j, and $\#W$ is the total number of sliding windows in the corpus. For the BertGCN and BertGAT, the adjacency matrix is created almost similarly, and the only change is positive pointwise mutual information (PPMI) for word co-occurrence.

3.2 GCN

For TextGCN, an identity matrix $X = I_{n_{\text{doc}}+n_{\text{word}}}$ is the initial node feature, where n_{doc} is the number of document nodes and n_{word} is the number of word nodes. Once the graph construction is complete, the initial input is then introduced to the primary GCN layer. Subsequently, this input undergoes the ReLU activation function. The outcome

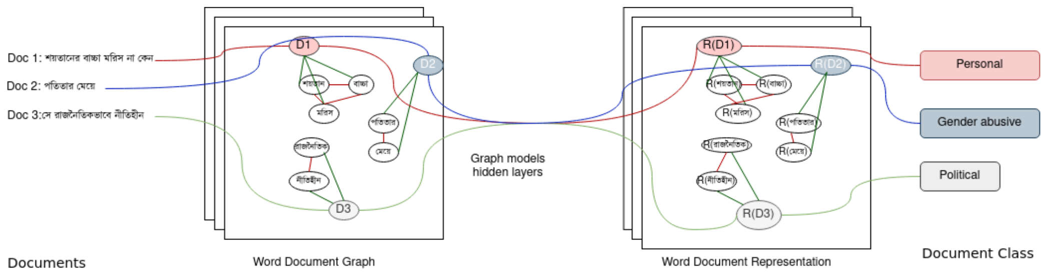


Figure 1: The process of text classification using Graph-based models

of this activation function is then directed to an additional GCN layer to ascertain logits. These logits are then transferred to the softmax function for classification, similar to the approach followed by the GCN model in the work by (Kipf and Welling, 2016),

$$Z = softmax\left(\tilde{A}ReLU\left(\tilde{A}XW_0\right)W_1\right) \quad (6)$$

Where $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix using degree matrix D, where $D_{ii} = \sum_j A_{ij}$ (i,j represent row and column of A,respectively) and W_0 and W_1 are weight parameters that are trained via gradient descent.(Yao et al., 2019) The loss function is defined as the cross-entropy error over all labeled documents,

$$\mathcal{L} = -\sum_{d\in\mathcal{Y}_D}\sum_{f=1}^{F} Y_{df}\ln Z_{df} \quad (7)$$

Where d is the variable that iterates over the set of document indices Y_D, Y_D is the set of document indices that have labels, F is the dimension of the output features, which is equal to the number of classes, Y is the label indicator matrix, Y_{df} is the element of the label indicator matrix Y at row d, and column f, and Z_{df} is the predicted probabilities or scores produced by the model for the document d being in class f.

3.3 GAT

The GAT model primarily operates on a collection of node features as its input.(Velickovic et al., 2017) The node features are expressed as:

$$\mathbf{h} = \left\{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\right\}, \vec{h}_i \in R^F \quad (8)$$

Where $\vec{h}_i$ is the node features of i^{th} node ($i \in N$), h is the set of node features, N is the number of nodes, and F is the number of features in each node. The model produces a new set of node features (of potentially different cardinality, F') as its output,

$$\mathbf{h}' = \left\{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\right\}, \vec{h}'_i \in R^{F'} \quad (9)$$

Firstly, to calculate the final representation in GAT, a weight matrix W, is applied to every initial node. After that, a shared attentional mechanism computes attention coefficients,

$$e_{ij} = a\left(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j\right) \quad (10)$$

Here, a represents the self-attention of the nodes, and the importance of node j's features to node i is calculated. Then, coefficients are normalized by using a softmax function,

$$\alpha_{ij} = \mathrm{softmax}_j\left(e_{ij}\right) = \frac{\exp\left(e_{ij}\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(e_{ik}\right)} \quad (11)$$

Here, k is a neighboring node of i from its neighborhood N_i. Finally, The attention coefficient α_{ij} for node i and its neighbor j is calculated using,

$$\alpha_{ij} = \sigma\left(LeakyReLU\left(\mathbf{a}^T\left[Wh_i \| Wh_j\right]\right)\right) \quad (12)$$

Where W is a learnable weight matrix, h_i and h_j are the feature vectors of nodes i and j, respectively, $\mathbf{a}$ is a learnable attention vector, LeakyReLU is the leaky rectified linear unit activation function, $\|$ denotes concatenation, ^{T} represents transposition and σ refers to the sigmoid function.

The feature representation of node i is updated by aggregating the features of its neighbors, weighted by the attention coefficients,

$$h_i^{(l+1)} = \sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)} \cdot Wh_j^{(l)} \quad (13)$$

Where $h_i^{(l+1)}$ is the updated feature vector of node i in layer $l+1$, $\alpha_{ij}^{(l)}$ is the attention coefficient between nodes i and j in layer l, $h_j^{(l)}$ is the feature vector of node j in layer l, and $N(i)$ represents the neighbors of node i. Where each final representation,

$$\vec{h}_i' = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^{k}\mathbf{W}^{k}\vec{h}_j\right) \tag{14}$$

Here, K depicts independent attention mechanisms (heads). An output layer in GAT models may translate the final node or graph representations into the required output format, such as class probabilities for classification tasks.

3.4 BERT-GCN

The BERT model generates the document embeddings for BertGCN.(Lin et al., 2021) The initial node feature matrix,

$$X = \begin{pmatrix} X_{\text{doc}} \\ 0 \end{pmatrix}_{(n_{\text{doc}}+n_{\text{word}})\times d} \tag{15}$$

Here, $X_{\text{doc}} \in R^{n_{\text{doc}}\times d}$ is the document node embeddings and d is the embedding dimension. X is the input of the GCN model and the output feature matrix of the i-th GCN layer,

$$L^{(i)} = \rho\left(\tilde{A}L^{(i-1)}W^{(i)}\right) \tag{16}$$

Where $L^{(i-1)}$ is the output feature matrix of the $i-1$ layer, ρ is an activation function, $\tilde{A}$ is the normalized adjacency matrix, and $W^{(i)} \in R^{d_{i-1}\times d_i}$ is a weight matrix of the layer i. Then, the outputs of GCN are fed into the softmax layer Z_{GCN}, similar to 3.2 for classification.

In practice, improving BertGCN with an auxiliary classifier that directly acts on BERT embeddings leads to rapid convergence and improved outcomes. An auxiliary classifier is created by directly feeding document embeddings (denoted by X) to a dense layer activated using softmax:

$$Z_{\text{BERT}} = \mathit{softmax}(WX) \tag{17}$$

The final training objective is the linear interpolation of the prediction from BertGCN and the prediction from BERT, which is given by:

$$Z = \lambda * Z_{\text{GCN}} + (1-\lambda) * Z_{\text{BERT}} \tag{18}$$

Where λ is the balance between BertGCN model and BERT module.

3.5 BERT-GAT

BertGAT(Lin et al., 2021) follows the structure and implementation process specified in Section 3.4. Notably, it is fundamentally comparable to BertGCN, comprising similar methodologies from initial node embedding to final output generation. The primary difference is that it uses the GAT model, as described in Section 3.3, rather than the GCN model, for its underlying graph modeling architecture. In BertGAT, λ is also used to control the influence of GAT and BERT.

4 Experiment Setup

4.1 Dataset

- **BanFake News** - Hossain et al. (2020) introduces a dataset for detecting fake news. The dataset consists of 48K authentic and 1K fake news articles from different categories. The tasks are classification tasks to find out if news is fake or not.

- **Sarcasm Detection** - This is Kaggle Competition Dataset [1] where the organizer curated a dataset comprised of around 50K news headlines labeled in two categories: Sarcastic (1) or Not-Sarcastic (0).

- **HateSpeech Detection** - Dataset provided by Karim et al. (2020) has raw texts collected from different sources with around 3k samples for training and 1k samples for testing. This dataset categorized into political, personal, gender-abusive, geopolitical, and religious hates

- **SentNoB** - In SentNoB(Islam et al., 2021), public comments on news and videos were collected from social media to detect the sentiment. The sentiments were labeled as Positive, Negative, and Neutral. The training dataset size is 13.5K, whereas the validation and test dataset size are 1.5K.

- **Emotion Detection** - For the emotion detection task, we use an emotion detection dataset provided by Trinto and Ali (2018) on which Bengali text data were extracted from

[1] https://www.kaggle.com/competitions/nlp-competition-cuet-ete-day-2022/data

YouTube comments in different kinds of Bengali videos. The emotion dataset contains around 3k samples and 5 classes representing different emotions such as anger/disgust, fear/surprise, joy, sadness, and none.

4.2 Preprocessing & Setup

In this experiment, we preprocessed the datasets by removing the number, URL, other language symbols or words, punctuation, and emojis. Five different models were used in this experiment including four graph-based models. BERT, BertGAT, and BertGCN Model used *csebuetnlp/banglabert* base (Bhattacharjee et al., 2022). BERT model contains hidden dimension 768, learning rate 1×10^{-5}, batch-size 16, and the maximum length of each sequence considered was 128. For the GCN model used in this experiment, the layers considered were 3, hidden dimension 200, drop-out rate 0.5, and learning rate considered 1×10^{-3}. GAT was also used in this experiment and includes 8 heads, learning rate 1×10^{-3}, batch size 64, hidden dimension 200, and epochs used 200. On the other hand, the BERT model in BertGCN and BertGAT includes max length for inputs 128, batch size 128, the learning rate of 1×10^{-5}, and 60 epochs. GCN model integrated into BertGCN includes layers 3, hidden dimension 200, drop-out rate 0.5, and learning rate considered 1×10^{-3}. Finally, the GAT model combined with BERT in BertGAT contains 8 heads, learning rate 1×10^{-3}, batch size 64, hidden dimension 200, and epochs used 200. Different ablation studies were done to find the right graph structures for GCN and GAT in Table 3 and 4. This experiment was done using Python 3.10 and experimented on Google Colab with NVIDIA Tesla $T4$ GPU and Kaggle with a single NVIDIA Tesla $P100$ GPU. To evaluate the model performance we use Accuracy, Macro-F1 Score & Weighted F1 score matrices in this experiment.

4.2.1 Computational Efficiency Analysis

Graph-based algorithms impose substantial computational demands. The feasibility of deploying these models at scale relies on several critical factors, encompassing model complexity, graph dimensions, and scalability considerations. Increasing the complexity of BertGCN models, especially through the enlargement of BERT embeddings, inevitably mandates a significant allocation of computational resources. The size of the graph serves as a pivotal determinant influencing computational efficiency. Addressing very large graphs poses significant challenges due to heightened computational and memory requirements, potentially leading to scalability issues. It's worth noting that graph-based models require a longer computation time (more than 2 to 5 times) compared to the BERT model.

4.2.2 Assessing Past Leading Models

In Table 1, an overview of the performance of previous state-of-the-art models is provided. These outcomes offer valuable insights and establish a foundation for performance bench marking.

Dataset	**Performance**	
	Model	Macro F1
SentNoB	n-gram fusion	**64.61**
Sarcasm	BERT	**89.93**
HateSpeech	SVM	**60.78**
BanFakeNews	SVM	**91.00**
Emotion	LSTM	**59.23**

Table 1: Performance of Previous Leading Models

5 Result & Discussion

In this section, we measured performance metrics across all five models for each of the five distinct datasets, facilitating a comprehensive comparative analysis. We evaluated edge features, where we assessed the effects of d2w-only relationships exclusively, as well as the d2w+w2w relationships. Subsequently, we turned our attention to a thorough investigation into the utilization of both one-hot embeddings and BERT embeddings, with a focus on how these variations influenced the overall model performance. Finally, we embarked on the quest to identify the optimal values for the parameter λ, particularly within the BertGCN and BertGAT models. Our pursuit aimed to unravel the intricacies of their behavior and performance under varying λ values.

5.1 Performance Analysis of Graph NLP Models

In the study, table 2 represents performance matrices of different models and datasets. Models include BanglaBert which is generally used for Bangla language classification tasks. BanglaBert is compared with various graph-based models including TextGCN, GAT, BanglaBertGAT, and BanglaBertGCN. Different datasets including

Dataset	Model	Performance Metrics		
		Accuracy	Macro F1 Score	Weighted F1
SentNoB	BanglaBERT	74.46	69.55	73.03
	GCN	41.60	29.66	33.69
	GAT	42.03	33.98	36.97
	BanglaBERT-GAT	74.65	70.65	74.65
	BanglaBERT-GCN	**75.66**	**71.70**	**74.72**
Sarcasm Detection	BanglaBERT	93.30	49.00	98.31
	GCN	61.40	44.22	50.91
	GAT	77.59	44.69	87.38
	BanglaBERT-GAT	95.85	48.94	97.88
	BanglaBERT-GCN	**98.22**	**49.55**	**99.10**
HateSpeech Detection	BanglaBERT	69.33	41.65	65.41
	GCN	44.07	14.57	30.87
	GAT	47.37	25.63	42.58
	BanglaBERT-GAT	71.44	57.05	70.32
	BanglaBERT-GCN	**73.33**	**60.80**	**72.81**
BanFakeNews	BanglaBERT	96.65	92.99	96.51
	GCN	84.60	75.12	77.54
	GAT	87.10	77.16	78.09
	BanglaBERT-GAT	97.14	92.91	97.02
	BanglaBERT-GCN	**98.55**	**96.69**	**98.55**
Emotion Detection	BanglaBERT	70.78	41.26	65.52
	GCN	46.68	14.79	34.97
	GAT	47.29	16.96	35.42
	BanglaBERT-GAT	75.30	45.67	71.63
	BanglaBERT-GCN	**76.81**	**46.70**	**72.67**

Table 2: Performance of Graph base NLP Model in Different Bangla Text Classificaiton Dataset

SentNoB, Sarcasm detection, hate speech detection, BanFakeNews, Emotion detection, and sentiment analysis were used for text classification using these models. In table 2, BERT's accuracy, macro F1 score, and weighted F1 score are very superior to TextGCN and GAT models for all the datasets. BERT shows this excellence due to a strong contextual understanding of text and pretraining on a large number of data. GCN and GAT perform well when the data is a graph. However, these models are not able to properly understand and represent local contextual information. Thus, they didn't perform well in the classifying task of the datasets. GAT was better compared to TextGCN.

GAT was able to use attention mechanisms to identify the importance of neighbors. This enables GAT to capture complex relationships and local sequences better than GCN. Bangla BertGAT outperformed BERT by 1% to 5% for Sarcasm Detection, Hate speech detection, Emotion detection, and Sentiment analysis datasets. For the SentNoB dataset, BertGAT shows a very slight improvement over BERT. The reason is BertGAT's attention mechanism of GAT and Bangla BERT's pretraining. Finally, BanglaBertGCN bested all the models for the datasets. BanglaBertGCN outperformed BanglaBertGAT and gained superior results by 1% to 3% for all the datasets. Bangla BertGCN captures local contextual information as well as the global relationship among all the words and documents. Which accounts for its greater accuracy,

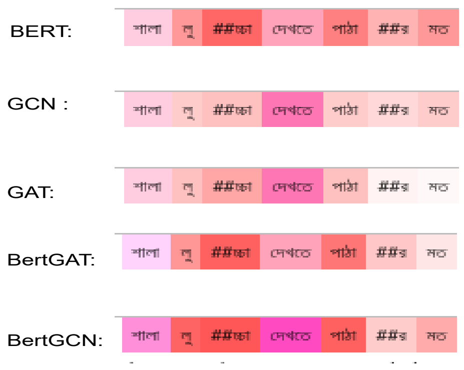


Figure 2: Attention heatmap for all models of the sentence

macro F1 score, and weighted F1 score.

Figure 2 represents each model's focus on individual word tokens from text "শালা লুচ্চা দেখতে পাঠার মত" for hate speech detection. The tokens 0: '[CLS]', 1: 'শালা', 2: 'লু', 3:'##চ্চা', 4:'দেখতে', 5:'পাঠা', 6:'##র', 7:'মত', 8: '[SEP]' are generated. For graph models, the word tokens were also considered as nodes. The BertGCN model provided more attention scores on the hate words (highlighted by deeper colors).

5.2 Edge Features Effect in Graph based Models

Table 3 depicts the effect of different edge feature structures on the graph-based models. Two types of edge features were evaluated: (1) d2w only, which is the edges created from only the word and document edges, and (2) d2w+w2w, which contains edges from word and document relationships as well as word and word relationships. The effects are measured in terms of accuracy. The d2w+w2w edge features showed better performance than the d2w-only structure for all the datasets in all four graph models.

Word-word edge structure creates a similar semantic cluster of similar words, providing more information about the context.(Han et al., 2022) Thus, an edge set containing a d2w+w2w structure captures more contextual information from a text, providing it with a greater performance. BertGCN bested all the models for both edge feature structures. BertGAT slightly lags behind BertGCN in terms of performance.

5.3 Node Features Effect in Graph based Models

One hot embedding is usually used to determine node features for graph-based models. BERT embedding was also used in this study to compare with One hot embedding. BERT embedding is learned during pretraining. In this study, the evaluation of test accuracy for One-hot embedding and BERT embeddings were used as initial node features on five datasets for four different Graph models. Specifically, In table 4 comparison between test accuracy against One hot embedding and BERT embeddings for the SentNoB dataset is shown. Firstly, TextGCN and GAT models give better results with BERT embedding than one-hot embedding. This may be because sentiment analysis tasks gain better leverage from the broader semantics knowledge learned from an extensive external text.(Han et al., 2022)

Model Name	**Edge Features**	
	d2w only	d2w+w2w
TextGCN	41.31	**41.60**
GAT	41.55	**42.03**
BERT-GAT	74.46	**74.65**
BERT-GCN	74.84	**75.66**

Table 3: Comparing different SentNoB Edge Features architecture

Model Name	**Node Features**	
	One Hot	BERT
TextGCN	41.60	**41.98**
GAT	42.03	**56.18**
BERT-GAT	**74.65**	58.13
BERT-GCN	**75.66**	62.55

Table 4: Comparing different SentNob Node Features architecture

Moreover, GCN and GAT aren't able to capture local semantic features with one hot embedding. BERT provides local attention as well as identifies long-term dependencies in a text.(Devlin et al., 2018) Thus, BERT features to improve the overall performance of GCN and GAT. Secondly, BertGAT and BertGCN show better performance in one hot embedding than BERT embedding. This finding can be ascribed to the hypothesis that providing BERT embedding results in additional redundancies and complexity. Thus, resulting in poor performance when compared with one hot embedding. Finally, in this research endeavor edge

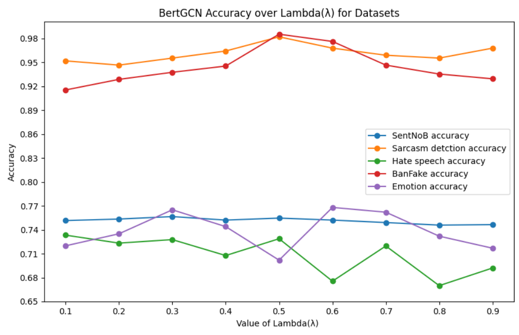


Figure 3: Effects of λ in BanglaBERT-GCN over accuracy for all datasets

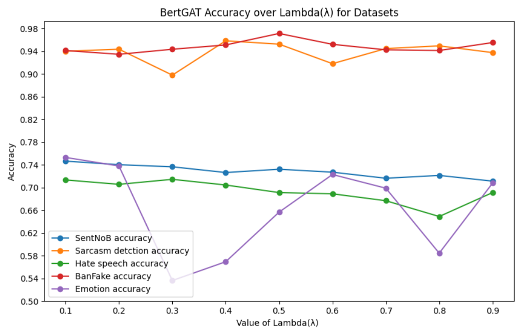


Figure 4: Effects of λ in BanglaBERT-GAT over accuracy for all datasets

sets (d2w+w2w) were used because they gave the best result in a full environment.

5.4 Effects of Lambda(λ)

The values of lambda λ have a significant effect on the overall performance of both BertGCN and BertGAT models. λ has values from 0 to 1. BertGCN/BertGAT's final output is determined by a linear interpolation of predictions from BERT and BertGCN (or BertGAT), which is defined by the parameter (λ). The value of λ varies for different tasks. When λ is set to 1, it represents sole dependence on BertGCN or BertGAT, whereas a λ value of 0 represents exclusive reliance on the BERT model.

In this study, the λ values were measured against accuracy to determine the effects it has on all five different datasets. In figure 3 and figure 4, the curves represent the influence of λ on each dataset for BertGCN and BertGAT models. Each of the curves contains accuracy for λ value from 0.1 to 0.9. Usually, very high or very low values of λ are ignored. Because it removes a significant portion of either BERT or GCN (or GAT) influence from the BertGCN (or BertGAT) structure. In figure 4 we also examined similar phenomena for the BERTGAT model. It is observed in the study that the highest accuracy values can be observed for λ values from 0.3-0.7. This is observed because of the balanced performance of graph-based and BERT methods. However, when the BertGAT model was considered, the picture was slightly different. In figure 4, maximum accuracy was obtained for BertGAT throughout a range of λ values ranging from 0.1 to 0.5. While the SentNoB, Hate Speech, and Emotion datasets performed best at $\lambda = 0.1$. The overall behavior can be explained by the high performance of the BERT method in the integrated structure.

6 Conclusion

This study evaluates various graph-based models for Bangla text classification and assesses their performance. TextGCN and GAT exhibit comparatively lower performance when compared to BERT. However, the integration of these models with BERT yields superior results in comparison to other models for classification tasks. BertGCN incorporates BERT's large-scale pretraining and fine-tuning, enhanced by transductive learning. BertGCN and BertGAT exhibit improved comprehension of local semantics through their integration with BERT. We advocate for the adoption of graph-based models, particularly BertGCN and BertGAT, for Bangla text classification, given their comparatively heightened predictive accuracy when contrasted with traditional text classification models.

In conclusion, it's noteworthy that the domain of Bangla text remains relatively unexplored in the context of graph-based algorithms and concepts. Numerous unexplored avenues including knowledge graphs and alternative graph models beyond GCN and GAT demand further exploration. However, it's essential to acknowledge that graph-based models entail significant computational resources, leading us to consider these avenues for future research endeavors.

Limitations

Graph models exhibit certain limitations that should be considered in academic research.

First, their scalability is often constrained, as handling large-scale graphs can be computationally intensive. Additionally, these models may struggle with sparse or incomplete data, impacting their accuracy in real-world scenarios. Interpretability can be challenging, making it hard to discern the rationale behind their predictions. Moreover, graph models may not effectively capture temporal dynamics, limiting their applicability in time-dependent problems. Lastly, they may require significant domain-specific expertise for effective deployment, posing a barrier to their widespread adoption in diverse fields.

Acknowledgements

This project has been jointly sponsored by Independent University, Bangladesh and the ICT Division of the Bangladesh Government.

References

Mostaq Ahmed, Partha Chakraborty, and Tanupriya Choudhury. 2022. Bangla document categorization using deep rnn model with attention mechanism. In Cyber Intelligence and Information Retrieval: Proceedings of CIIR 2021, pages 137--147. Springer.

Shahin Akhter et al. 2018. Social media bullying detection using machine learning on bangla text. In 2018 10th International Conference on Electrical and Computer Engineering (ICECE), pages 385--388. IEEE.

Md Tanvir Alam and Md Mofijul Islam. 2018. Bard: Bangla article classification using a new comprehensive dataset. In 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), pages 1--5. IEEE.

Tanvirul Alam, Akib Khan, and Firoj Alam. 2020. Bangla text classification using transformers. arXiv preprint arXiv:2011.04446.

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.

Abu Kowshir Bitto, Md Hasan Imam Bijoy, Md Shohel Arman, Imran Mahmud, Aka Das, and Joy Majumder. 2023. Sentiment analysis from bangladeshi food delivery startup based on user reviews using machine learning and deep learning. Bulletin of Electrical Engineering and Informatics, 12(4):2282--2291.

Abu Nowshed Chy, Md Hanif Seddiqui, and Sowmitra Das. 2014. Bangla news classification using naive bayes classifier. In 16th Int'l Conf. Computer and Information Technology, pages 366--371. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Ankita Dhar, Niladri Sekhar Dash, and Kaushik Roy. 2018. Classification of bangla text documents based on inverse class frequency. In 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), pages 1--6. IEEE.

Phuc Do, Trung Phan, Hung Le, and Brij B Gupta. 2022. Building a knowledge graph by using cross-lingual transfer method and distributed minie algorithm on apache spark. Neural Computing and Applications, pages 1--17.

Sanjay K Dwivedi and Chandrakala Arya. 2016. Automatic text classification in information retrieval: A survey. In Proceedings of the second international conference on information and communication technology for competitive strategies, pages 1--6.

Niloy Farhan, Ishrat Tasnim Awishi, Md Humaion Kabir Mehedi, MD Mustakin Alam, and Annajiat Alim Rasel. 2023. Ensemble of gated recurrent unit and convolutional neural network for sarcasm detection in bangla. In 2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC), pages 0624--0629. IEEE.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. Advances in neural information processing systems, 30.

Soyeon Caren Han, Zihan Yuan, Kunze Wang, Siqu Long, and Josiah Poon. 2022. Understanding graph convolutional networks for text classification. arXiv preprint arXiv:2203.16060.

Md Zobaer Hossain, Md Ashraful Rahman, Md Saiful Islam, and Sudipta Kar. 2020. Banfakenews: A dataset for detecting fake news in bangla. arXiv preprint arXiv:2004.08789.

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.

Md. Rezaul Karim, Bharathi Raja Chakravarthi, Mihael Arcan, John P. McCrae, and Michael Cochez. 2020. Classification benchmarks for under-resourced bengali language based on multichannel convolutional-lstm network. 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), pages 390--399.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Bin Liang, Hang Su, Lin Gui, Erik Cambria, and Ruifeng Xu. 2022. Aspect-based sentiment analysis via affective knowledge enhanced graph convolutional networks. Knowledge-Based Systems, 235:107643.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. Bert-gcn: Transductive text classification by combining gcn and bert. arXiv preprint arXiv:2105.05727.

Priyanka Patel and Khushali Mistry. 2015. A review: Text classification on social media data. IOSR Journal of Computer Engineering, 17(1):80--84.

Md Mahbubur Rahman, Md Aktaruzzaman Pramanik, Rifat Sadik, Monikrishna Roy, and Partha Chakraborty. 2020. Bangla documents classification using transformer based deep learning models. In 2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI), pages 1--5. IEEE.

Saifur Rahman and Partha Chakraborty. 2021. Bangla document classification using deep recurrent neural network with bilstm. In Proceedings of International Conference on Machine Intelligence and Data Science Applications: MIDAS 2020, pages 507--519. Springer.

Mehzabin Sadat Aothoi, Samin Ahsan, Najeefa Nikhat Choudhury, and Annajiat Alim Rasel. 2023. Supervised hybrid model for rumor classification: A comparative study of machine and deep learning approaches. In International Conference on Big Data Analytics and Knowledge Discovery, pages 281--286. Springer.

Alex Sherstinsky. 2020. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. Physica D: Nonlinear Phenomena, 404:132306.

Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. 2018. Attention-based graph neural network for semi-supervised learning. arXiv preprint arXiv:1803.03735.

Nafis Irtiza Trinto and Mohammed Eunus Ali. 2018. Detecting multilabel sentiment and emotions from bangla youtube comments. 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), pages 1--6.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. stat, 1050(20):10--48550.

Kunze Wang, Yihao Ding, and Soyeon Caren Han. 2023. Graph neural networks for text classification: A survey. arXiv preprint arXiv:2304.11534.

Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. 2023. Graph neural networks for natural language processing: A survey. Foundations and Trends� in Machine Learning, 16(2):119--328.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In Proceedings of the AAAI conference on artificial intelligence, 01, pages 7370--7377.

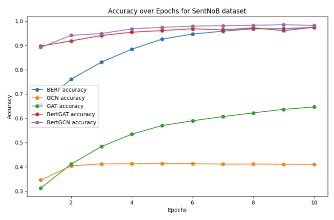
Yazhou Zhang, Dan Ma, Prayag Tiwari, Chen Zhang, Mehedi Masud, Mohammad Shorfuzzaman, and Dawei Song. 2023. Stance-level sarcasm detection with bert and stance-centered graph attention networks. ACM Transactions on Internet Technology, 23(2):1--21.

Xujuan Zhou, Raj Gururajan, Yuefeng Li, Revathi Venkataraman, Xiaohui Tao, Ghazal Bargshady, Prabal D Barua, and Srinivas Kondalsamy-Chennakesavan. 2020. A survey on text classification and its applications. In Web Intelligence, 3, pages 205--216. IOS Press.

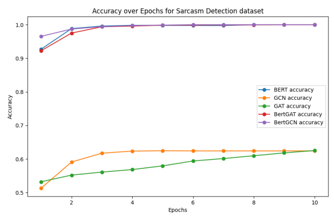
A Accuracy & Loss Plots

A.1 Accuracy Plots

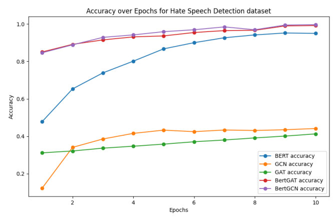
Figure 5 illustrates the training accuracy versus epochs for all the different datasets. Initially, the training accuracy for these datasets applying all the models ranges from 0.1-0.9. GCN and GAT show very low training accuracies compared to all the models for all the epochs. GAT shows better performance due to its attention mechanism.



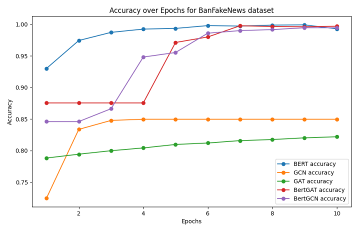
SentNOB Accuracy Plot



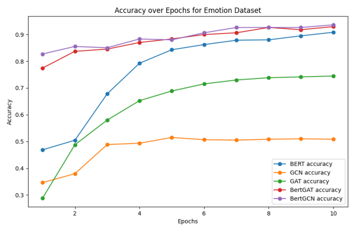
Sarcasm Accuracy Plot



Hate Accuracy Plot



BanFake Accuracy Plot



Emotion Accuracy Plot

Figure 5: Accuracy Plots of Different Models in Aforementioned Datasets

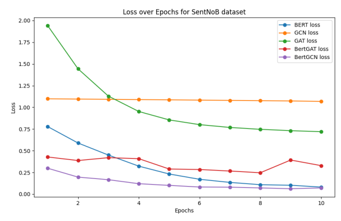
BERT, BertGAT, and BertGCN show the highest accuracies. BertGCN and BertGAT outperform the BERT model. One of the reasons for this outperformance is the fine-tuning of the BERT model before integrating it with the graph-based model. In the fine-tuning phase, the model adapts to the specific characteristics of each dataset. In this figure, a consistent increase in accuracy can be seen as training continues until the highest training accuracy is achieved. It is crucial to highlight that beyond this point, there is a risk of over-fitting.

A.2 Loss Plots

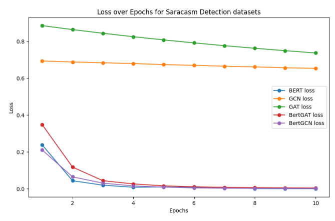
In figure 6, training loss versus epochs for each dataset was observed. The curves represent different models used in this study. The training loss starts from a very high value initially. With the increase of epochs, the training loss sharply decreases.

Then, the losses decrease and become steady until it reaches a point where the training accuracy reaches its maximum value. If the training continues overfitting may occur resulting in a large gap between training loss and test loss. For GAT and GCN, the loss curves are consistently situated higher on the graph across all datasets.

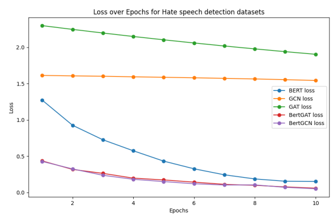
The training loss for all the datasets for GCN as well as Sarcasm, BanFake, and Hate Speech for GAT is not decreasing significantly, showing that the training loss may be not learning effectively from the data. For SentNOB, BanFake, and Emotion datasets, GAT's loss curves decrease below those of GCN, while Sarcasm detection and Hate speech datasets exhibit the opposite behavior. Particularly, for the SentNOB dataset, the BERT loss decreases below BertGAT, and BertGCN's loss curve decreases below BERT's.



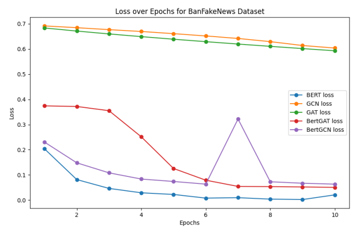
SentNOB Loss Plot



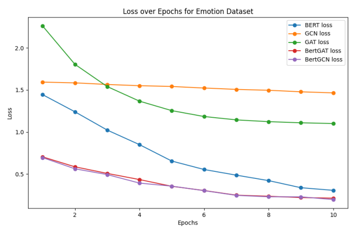
Sarcasm Loss Plot



Hate Loss Plot



BanFake Loss Plot



Emotion Loss Plot

Figure 6: Loss Plots of Different Models in Aforementioned Datasets

On the other hand, in the Emotion and Hate Speech datasets, the loss curves decrease in the following order: BERT, BertGAT, and BertGCN. Finally, Sarcasm and BanFake datasets, BertGAT and BertGCN both exhibit decreased losses compared to BERT. But, overall BERT, BertGAT, and BertGCN show a steep fall in the initial epochs suggesting that the model is learning quickly. However, as training progresses, the pace of decline may drop, suggesting that the model is approaching an ideal answer.