Ensemble of Gated Recurrent Unit and Convolutional Neural Network Sarcasm Detection in Bangla

Sarcasm is a form of emotional expression in which someone speaks or writes something entirely contrary to what is truly intended with humor underlying. Detecting sarcasm is a difficult task as the actual meaning of a text is not consistent. In Bangla Language, not enough work has been done in this field. Therefore, we experimented with some of the state of the art NLP's techniques and Machine Learning and Deep Learning models in an existing dataset. In this paper, we are proposing a Sarcasm Detection AI for Bangla Language based on Deep Learning Architecture with **96%** F_1 score and **96%** of accuracy which surpasses the existing traditional machine learning models performance.

CCS Concepts: • **Natural language processing**; • **Artificial intelligence**; • **Deep Learning**;

Additional Key Words and Phrases: Bangla, NLP, Sarcasm Detection, Gated Recurrent Unit, Convolutional Neural Network

1 INTRODUCTION

With the increasing users in social media , using sarcasm is becoming more popular day by day. Sarcasm mostly delivered as a verbal irony that actually means the opposite expression of what is being said. It is frequently used to mock or ridicule other people, either in humor or really in earnest, or to show contempt, and it is firmly ingrained in social media culture worldwide.

Sarcasm should be recognized on all social networking sites as well as in person situations like online conversations because it is typically employed negatively. It ensures more comprehensive analytics in online comments and reviews and that is why it has become very popular in recent days.

Bangla is still considered as a Low Resource Language (LRL) by NLP Research Communities. We have found only one publicly available dataset for sarcasm detection tasks. In that paper[1], they explained the technical difficulties they faced due to the scarcity of resources in order to create a vast dataset. However, they only used traditional machine learning models to evaluate the performance.

In our paper, we tested several popular NLP techniques to preprocess our data. First we used the traditional classification models like the Support Vector Machine (SVM), Logistic Regression and Random Forest model to monitor the effect of our preprocessing step. Then we implemented a model using deep learning architecture where an ensemble of Gated Recurrent Unit(GRU) and Convolutional Neural Network (CNN) was used. We split the dataset in 80:20 ratio. Finally, we evaluate our model using the test dataset and got **96%** F_1 score and **96%** of accuracy.

In the upcoming sections, we have briefly discussed regarding the previous works in section 2. After that, we have discussed about the dataset which we have used for training our model in section 3.1. Later, we have discussed about how we trained the models in section 3. The results we have gained through training the models are discussed in section 4. Lastly, we have concluded and explained future works in the section 5.

2 LITERATURE REVIEW

We have seen a notable amount of work in sarcasm detection in other resourceful languages. In Bangla we don't have sufficient work done despite it is the fifth most-spoken native language and the seventh most spoken language by total number of speakers in the world.[2]

In the BanglaSarc paper [1], the author introduced a sarcasm dataset that can aid in the process of automating sarcasm detection in Bangla language. They scraped the data from Facebook, YouTube and various online blog contents. They used seven different machine learning models: Kernel Support Vector Machine, K-Nearest Neighbor, Decision Tree, Random Forest, Multinomial Naive Bayes, and Stochastic Gradient Descent in order to evaluate their dataset. Among them, Stochastic Gradient Descent gained the lowest accuracy, only 53.86%. Accuracy ranging from 62.46% to 74.78%, Logistic Regression, Multinomial Naive Bayes, K-nearest Neighbor's performance was moderate. However, Random Forest and Decision Tree's performance was notable. The Support Vector Machine was 71.55% accurate in sarcasm detection. Decision Tree gained 83.58% accuracy whereas Random Forest managed to reach 89.93% accuracy with similar precision, recall and F_1 score.

In [3] paper, the author presented a systematic review of 31 studies on sarcasm detection. These studies are all based on machine learning solutions. They found out that over 22.58% of the adapted machine learning algorithms were SVM. Their study also revealed that using state of the NLP techniques like lexical, frequency, and POS tagging can contribute to the performance of SVM. Their work also addresses some of the main challenges faced by prior scholars.

The [4] paper introduced a multi-head attention-based bidirectional LSTM network to detect sarcasm. At first, they extracted features and built a feature rich SVM model in order to show the importance of handcrafted features in classification. Later, they created a neural network architecture. They implemented the BiLSTM layer right after the embedding matrix. Then it pass through a multi head attention system which provides a sentence embedding. Finally, they concatenated sentence embedding and auxiliary features and it goes through softmax classifier which gives final output. Their deep learning architecture surpasses feature rich SVM with a F_1 score of 77.48%.

In [5] paper, the researcher proposed two novel deep learning models for sarcasm detection. Their model extends the architecture of BERT. Their model ACE1 scored 84.57% in SemEval-2018 dataset and 93.14% in IAC dataset.

3 METHODOLOGY

The whole workflow of our model is shown in figure 1.

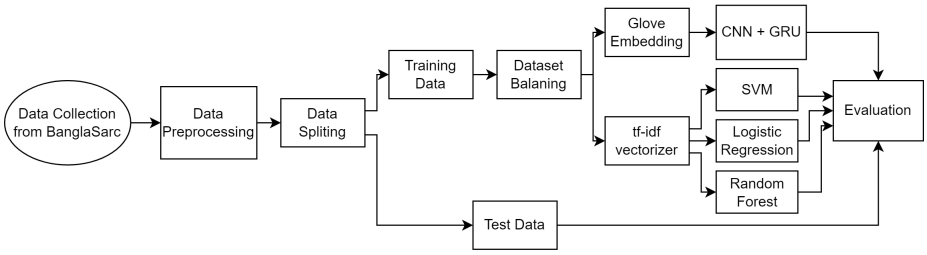

Fig. 1. Entire workflow

At first, we collected our dataset and applied various pre-processing. Then we split the preprocessed data in 80:20 ratio. Next, we balanced the training data and trained our data using four models SVM, Logistic Regression,

Random Forest and ensemble of Gated Recurrent Unit(GRU) and Convolutional Neural Network (CNN) model. Finally, we evaluate our model using test data.

3.1 Dataset

As we have mentioned earlier, the work on Bangla sarcasm detection is not rich. We have explored and decided to use BanglaSarc [1]. We chose this dataset because it has been scraped from Comments/status from Facebook, YouTube, and various online blog contents. BanglaSarc dataset can be downloaded from here [BanglaSarc Link]. This dataset contains 5112 data.

Table 1. Number of elements on each class

Label	**Number of Data**
Sarcastic	1953
Non sarcastic	3159
Total	5122

3.2 Pre-Processing

BanglaSarc dataset was mostly pre-processed. However, we experimented with a few common preprocessing techniques which improved the performance of our model. Firstly, we have removed the emoticons and other emojis by the help of RegEx. These are the following unicodes characters that we removed.

Table 2. Removed Unicode characters by RegEx

Unicode Range	**Description**
U0001F600 - U0001F64F	Emoticons
U0001F300 - U0001F5FF	Symbols & Pictographs
U0001F680 - U0001F6FF	Transport & map symbols
U0001F1E0 - U0001F1FF	Flags (iOS)
U00C0 - U017F	Latin
U2000 - U206F	General Punctuations

We also noticed that the BanglaSarc dataset is not balanced. To fix this issue, we oversampled the data only in the training split in order to prevent leakage between training and testing data. Oversampling may lead to overfitting to the training data which will be taken care of by early stopping.

3.3 Embedding

After preprocessing, we have experimented with a countvectorizer and tf-idf for our machine learning model (Support Vector Machine). For deep learning models we used GloVe [6] embedding for our deep learning model. As tf-idf and GloVe embedding performed better, we discarded countvectorizer and word2vec in the deep learning model. We have used BNLP's [7] pre-trained GloVe embedding. We chose 300 dimension embedding on 39 million tokens which was generated from wikipedia and crawl news articles. The embedding can be downloaded from here.

3.4 Machine Learning Model

We chose the Support Vector Machine, Logistic Regression and Random Forest Classifier as our machine learning models.

Support Vector Classifier: Support vector machine (SVM), which comes in different kernel functions. An SVM model's goal is to calculate a hyperplane (or decision boundary) based on a feature set in order to categorize data points. The goal is to locate the hyperplane that separates the data points of two classes with the greatest margin, which may take many different forms in an N-dimensional space. The cost function for the SVM model is illustrated in a mathematical form:

$$J(\theta) = \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 \tag{1}$$

$$\theta^T x^{(i)} \geq 1, \quad y^{(i)} = 1 \tag{2}$$

$$\theta^T x^{(i)} \leq -1, \quad y^{(i)} = 0 \tag{3}$$

A linear kernel is used in the code above. Kernels are typically used to fit multidimensional or difficult to easily separate data points. We have applied sigmoid SVM, kernel SVM (polynomial SVM), Gaussian SVM, and fundamental linear SVM models in our situation.

Logistic Regression: The logistic regression function $p(x)$ is the sigmoid function of $f(x) : p(x) = 1/(1 + exp(f(x))$. As such, it's often close to either 0 or 1. A logistic regression (LR) model is used because it offers the intuitive equation to categorize issues into binary or multiple classes. This is because we are classifying text on the basis of a large feature set, with a binary output (true/false or true article/fake article). While several parameters were tested before obtaining the maximum accuracies from the LR model, we did hyperparameter tuning to acquire the best outcome for each particular dataset. The logistic regression hypothesis function can be defined mathematically as follows:

$$h_\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} \tag{4}$$

The output of logistic regression is converted to a probability value using a sigmoid function; the goal is to minimize the cost function to arrive at the best probability. As demonstrated in the cost function calculation:

$$\begin{aligned} &\text{Cost}\left(h_\theta(x), y\right) \\ &\quad = \begin{cases} \log\left(h_\theta(x)\right), & y = 1 \\ -\log\left(1 - h_\theta(x)\right), & y = 0 \end{cases} \end{aligned} \tag{5}$$

Random Forest Classifier: Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

FFNN: A Feed Forward Neural Network (FFNN) is a type of artificial neural network in which there is no cycle in the connections between the nodes.

In this model, we have experimented with different activation functions. We have used the relu function for the FFNN.

$$f(x) = max(0, x) \tag{6}$$

3.5 Deep Learning Model

The whole pipeline of our proposed deep learning model (BiLSTM + FFNN) is given below in figure 2:

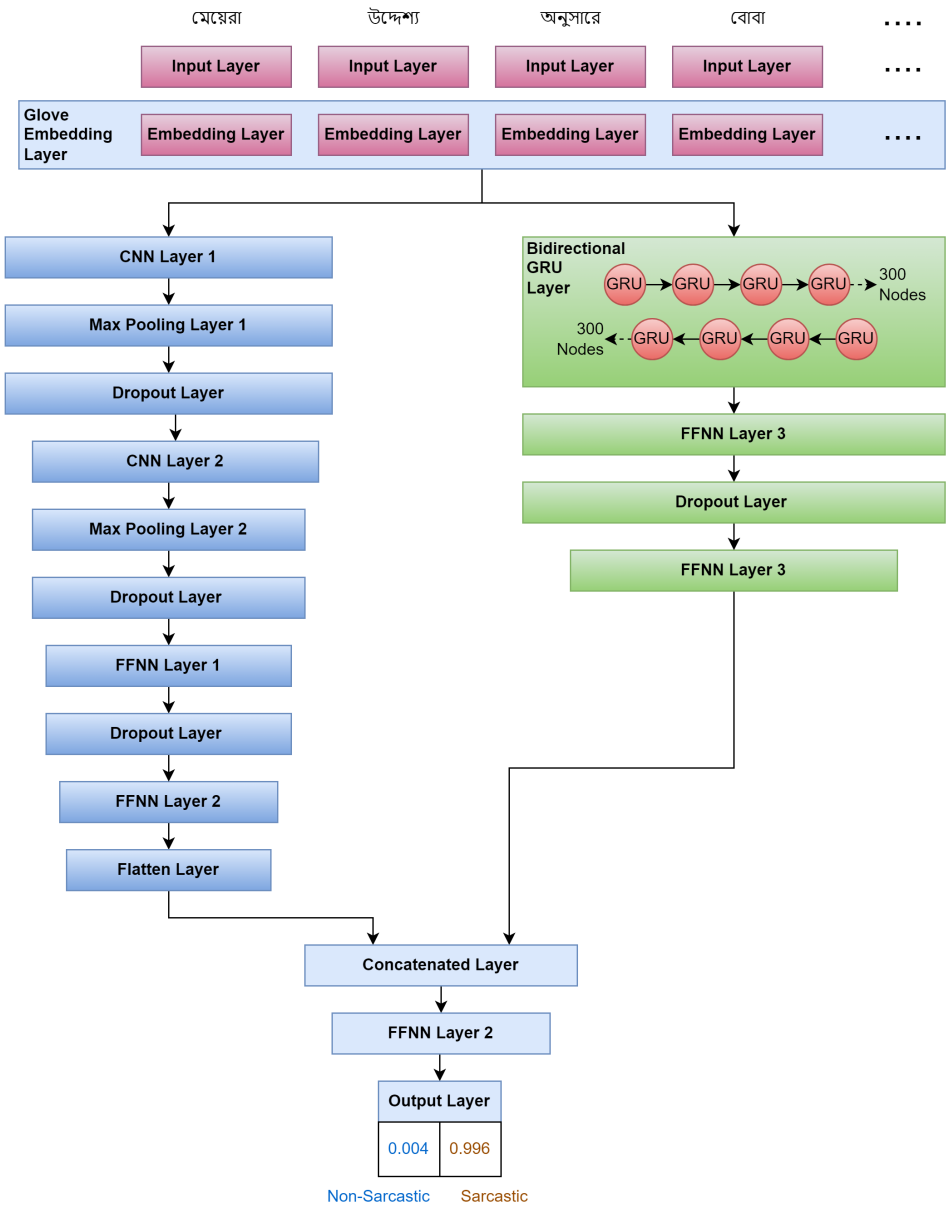

Fig. 2. Overview of the Deep Learning Model (GRU + CNN)

Convolutional Neural Networks (CNNs): CNNs are a type of neural network specifically designed for processing data with a grid-like topology, such as an image. They are made up of multiple ”convolutional” layers, which apply a specified number of filters to the input data. These filters are small matrixes that are used to detect specific patterns in the input data by sliding over it and performing element-wise multiplications with the input data and summing the results. The output of the convolutional layers is then passed through a non-linear activation function, such as ReLU, before being fed into the next layer. CNNs also include ”pooling” layers, which down-sample the output of the convolutional layers by applying a specified window function, such as max or average pooling, to the output data. This reduces the dimensionality of the data, making it more computationally efficient to process and also helps to reduce overfitting by reducing the number of parameters in the model.
Max Pooling: Max pooling is a technique used in CNNs to down-sample the output of the convolutional layers. It works by applying a max filter to the output of the convolutional layers, which looks at a specified window of data and outputs the maximum value within that window. This effectively reduces the size of the output data by discarding most of the values and only retaining the maximum ones. Max pooling has the effect of making the CNN more robust to small translations in the input data and also helps to reduce overfitting by reducing the number of parameters in the model.
Gated Recurrent Unit (GRU): A GRU [8] is a type of recurrent neural network (RNN) that is used to process sequential data, such as a time series or natural language. It is made up of ”gates,” which control the flow of information in and out of the unit, allowing it to selectively preserve or forget information from previous time steps. A GRU has two gates: a ”reset” gate, which determines what information to discard from the previous time step, and an ”update” gate, which determines what information to preserve. These gates are implemented using tanh activation functions, which output values between 0 and 1, indicating the strength of the gate.
Bidirectional GRU: A bidirectional GRU is a type of GRU that processes sequential data in both the forward and backward directions, allowing it to consider the context on both sides of a given time step. This can be useful for tasks like natural language processing, where the meaning of a word can depend on the words that come before and after it. To implement a bidirectional GRU, two separate GRUs are used: one processes the input data in the forward direction, and the other processes it in the backward direction. The output of these two GRUs is then concatenated, allowing the model to consider the context on both sides of the input data.
Both of the GRU layers of BiGRU use tanh activation function, whereas the output layer uses softmax activation.

$$Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{7}$$

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{8}$$

We have used binary cross entropy as our loss function and used Binary Accuracy as our metrics in the model.

$$L_{BCE} = -\frac{1}{n}\sum_{j=1}^{n}(Y_i \cdot log\hat{Y}_i + (1 - Y_i) \cdot log(1 - \hat{Y}_i)) \tag{9}$$

As we oversampled our training sample, we have suspected that the model can suffer from the overfitting issue. In order to solve this, we have monitored validation loss and stopped the model early.

4 EXPERIMENT AND RESULTS

We have done several experiment with our pre-processing and models. We tried using different regex in order to remove emoji, punctuations and other unnecessary strings. This improved our results. Our experiment showed that our preprocessing step worked effectively as we found a huge improvement in results in our machine learning models. We have achieved 89% accuracy in Logistic Regression and Random Forest model. It is a 26.5% increment than the BanglaSarc paper's Logistic Regression model. A 15% increase in accuracy can be seen in Support Vector Model too. However, our ensemble of Gated Recurrent Unit and Convolutional Neural Network model for Sarcasm Detection performed the best among all other models. We have used 20% of our training data as a validation set. The model had 94.89% training accuracy and 94.13% of validation accuracy. It achieved 96% accuracy in the test set with a Macro Average F_1 score of 96%. The detailed evaluation can be seen in the following table 3.

Table 3. Detailed Evaluation of Different Classifier

Model name		Precision	Recall	F_1 score
Logistic Regression	Non-Sarcastic	89%	94%	92%
	Sarcastic	88%	80%	84%
	Macro Average	89%	87%	88%
	Accuracy	89%		
Support Vector Classification	Non-Sarcastic	85%	98%	91%
	Sarcastic	95%	69%	80%
	Macro Average	90%	83%	85%
	Accuracy	87%		
Random Forest Classification	Non-Sarcastic	87%	98%	92%
	Sarcastic	95%	74%	83%
	Macro Average	91%	86%	88%
	Accuracy	89%		
GRU + CNN	Non-Sarcastic	97%	**97%**	**97%**
	Sarcastic	95%	94%	95%
	Macro Average	96%	96%	96%
	Accuracy	**96%**		

We may observe several implementations of our Bangla sarcasm detection model in this table. The table below compares four distinct methods, Logistic regression, Support vector classification, Random forest and ensemble of Gated Recurrent Unit(GRU) and Convolutional Neural Network (CNN) model.
Here we can see, with logistic regression and random forest model , we get the same accuracy of 89%. Support vector classification gives less acuuracy that is 87%. On the otherhand, with the ensemble of Gated Reccurent Unit model and Convolutional Neural Network model, the highest outcome has been obtained with the accuracy of 96%. Our ensemble of Gated Reccurent Unit model and Convolutional Neural Network (GRU+CNN) model achieved maximum precision, recall and F_1score.

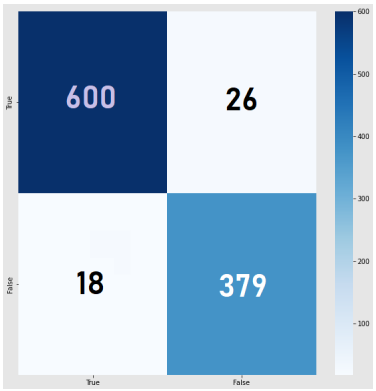

Fig. 3. Confusion Matrix of Deep Learning Model

Furthermore, we have plotted a confusion matrix in figure 3 to get better understanding of our model's performance. From the matrix, we can see that the true positive and true negatives are 600 and 379 respectively. False positives and false negatives are 26 and 18 only which is insignificant. We can see that our model is slightly overfitted which is also negligible.

5 CONCLUSION

In this paper, we presented an "Bangla Sarcasm Detection AI" by implementing the state of the art NLP techniques and Deep Learning architectures. As expected, better preprocessing in a balanced dataset gave us a higher score. Using modern embedding techniques like GloVe gave us much better performance even though we used a relatively simple Deep Learning model. For future work, it would be interesting to explore embedding like BERT, more complex Deep Learning models where multiple combinations of different architectures can be used. We are also proposing to make a high quality dataset for Sarcasm Detection and Offensive language detection tasks combined.

REFERENCES

[1] Tasnim Sakib Apon, Ramisa Anan, Elizabeth Antora Modhu, Arjun Suter, Ifrit Jamal Sneha, and MD. Golam Rabiul Alam. Banglasarc: A dataset for sarcasm detection, 2022.
[2] Wikipedia Contributors. Bengali language, 2022.
[3] Samer Muthana Sarsam, Hosam Al-Samarraie, Ahmed Ibrahim Alzahrani, and Bianca Wright. Sarcasm detection using machine learning algorithms in twitter: A systematic review. *International Journal of Market Research*, 62(5):578–598, 2020.
[4] Avinash Kumar, Vishnu Teja Narapareddy, Veerubhotla Aditya Srikanth, Aruna Malapati, and Lalita Bhanu Murthy Neti. Sarcasm detection using multi-head attention based bidirectional lstm. *IEEE Access*, 8:6388–6397, 2020.
[5] Nastaran Babanejad, Heidar Davoudi, Aijun An, and Manos Papagelis. Affective and contextual embedding for sarcasm detection. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 225–243, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.
[6] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.
[7] Sagor Sarker. Bnlp: Natural language processing toolkit for bengali language, 01 2021.
[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.