Ben-Sarc: A Corpus for Sarcasm Detection from Bengali Social Media Comments and Its Baseline Evaluation

Sanzana Karim Lora[a,*], G. M. Shahariar[a], Tamanna Nazmin[a], Noor Nafeur Rahman[a], Rafsan Rahman[a], Miyad Bhuiyan[a], Faisal Muhammad Shah[a]

[a]*Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh*

Abstract

Sarcasm detection research of the Bengali language so far can be considered to be narrow due to the unavailability of resources. In this paper, we introduce a large-scale self annotated Bengali corpus for sarcasm detection research problem in the Bengali language named 'Ben-Sarc' containing 25,636 comments, manually collected from different public Facebook pages and evaluated by external evaluators. Then we present a complete strategy to utilize different models of traditional machine learning, deep learning, and transfer learning to detect sarcasm from text using the Ben-Sarc corpus. Finally, we demonstrate a comparison between the performance of traditional machine learning, deep learning, and transfer learning models on our Ben-Sarc corpus. Transfer learning using Indic-Transformers Bengali BERT as a pre-trained source model has achieved the highest accuracy of 75.05%. The second highest accuracy is obtained by the LSTM model with 72.48% and Multinomial Naive Bayes is acquired the third highest with 72.36% accuracy for deep learning and machine learning, respectively. The Ben-Sarc corpus is made publicly available in the hope of advancing the Bengali Natural Language Processing community.

*Corresponding author

Email addresses: sanzanalora@yahoo.com (Sanzana Karim Lora), shahariar_shibli.cse@aust.edu (G. M. Shahariar), tamanna.naz98@gmail.com (Tamanna Nazmin), nafirahman27@gmail.com (Noor Nafeur Rahman), rafsanrahman35549@gmail.com (Rafsan Rahman), miyadbhuiyan@gmail.com (Miyad Bhuiyan), faisal.cse@aust.edu (Faisal Muhammad Shah)

Keywords: Sarcasm, Bengali sarcasm, sarcasm detection, Bengali sarcasm detection

1. INTRODUCTION

An ironic, stinging, sour, cutting statement or comment that indicates the reverse of what someone truly intends to express is Sarcasm [1]. The use of sarcastic language is a resentment concealed as humor and intended to provoke, annoy, or convey contempt. As the intention of sarcasm is often vague and misleading, people cannot discriminate between a true story and satire or irony [1]. Facebook, YouTube, and Twitter are influential social media platforms for sharing people's judgments, thoughts, opinions, sentiments nowadays [2]. The aforementioned large amount of available data offers the extent to research in Natural Language Processing (NLP).

Sarcasm detection in low resource language is a very narrow research area in Natural Language Processing. Sarcasm detection is a subset of sentiment analysis problems where the focus is on recognizing sarcasm rather than identifying a sentiment across the board [3]. Sarcasm detection researches are available for high resource languages such as English. But, despite being the world's seventh most spoken language with 240 million native speakers [4], research on sarcasm detection in the Bengali language is unexplored and overlooked. Due to the limited resources and the scarcity of large-scale sarcasm data, identifying sarcasm from Bengali text is currently a difficult challenge for the researchers of NLP [5].

Facebook is a popular free social networking website that allows registered users to upload photos, videos, send messages and keep in touch with friends, family, and colleagues[1]. Bangladesh has 41 million Facebook users since January 2021[2]. People socialize in the Facebook comment section to express their

[1] https://whatis.techtarget.com/definition/Facebook/

[2] https://www.statista.com/statistics/268136/top-15-countries-based-on-number-of-facebook-users/

perspectives, judgments, and opinions on the content of a post. Any automatic detection system that uses machine learning is large-scale dataset dependent as it requires rigorous training and testing. As far as we have noticed, there is no available Bengali text corpus for sarcasm detection. We have constructed a corpus named 'Ben-Sarc' that contains Facebook comments written in Bengali. Furthermore, we have classified the Bengali texts as Sarcastic and Non-sarcastic and proposed a sarcasm detection model using machine learning.

Our main contributions in this paper are summarized as follows:

- At first, we have constructed a large-scale self annotated Bengali corpus for sarcasm detection. The corpus can be found at `https://shorturl.at/oFJRZ`.
- Next, we have evaluated our constructed corpus by external human evaluators who are experts in this field.
- Then, we have conducted a comprehensive experiment on this corpus to detect sarcasm from Bengali texts with the help of traditional Machine Learning, Deep Learning, and Transfer Learning approaches to set a baseline for future researchers.

In the next section, we briefly discuss related works on high and low resource language sarcasm detection. Section 3 shows the dataset creation along with the annotation process. Moreover, Section 4 explains the proposed methodology. Section 5 contains the experimental results and their analysis, while Section 6 contains the conclusion and future work.

2. RELATED WORKS

The increasing engagement of social media users influences the quantitative and qualitative analysis of available data. Though most of the research is on the English language, sarcasm detection for low resource languages such as Indonesian [6], Hindi [7] [8] [9], Czech [10], and Japanese [11] are available. We

discuss some of the related approaches in the following literature review analysis.

2.1. English Language

[9] experimented with traditional machine learning algorithms such as SVM, KNN, and Random forest on 9104 tweets on Twitter. [12] worked on both sarcasm and irony detection separately. SVM, Naive Bayes (NB), Decision Tree, Random Forest (RF) were applied on the irony detection dataset whereas SVM and Random Forest (RF) algorithms were on the sarcasm detection dataset. The segregated experiments gained 64% accuracy on irony and 76% accuracy on sarcasm detection.

There exist a few models that use contextual information regarding the tweets on Twitter to detect sarcasm. [13] focused on the context of authors and audiences on Twitter posts to figure out sarcastic content with 85.1% accuracy. Binary logistic regression was applied to train the model upon 19534 tweets. [14] also aimed at the context for identifying sarcasm accurately. They collected 1500 tweets and derived 6774 history-based, 453 conversation-based, 2618 topic-based contextual tweets. Sequential SVM classifier exhibited a decent accuracy of 69.13%. [15] extracted 5000 tweets that include texts, labels, and contexts and analyzed the dataset through linear SVC, Logistic Regression (LR), Gaussian Naive Bayes (GNB), and Random Forest (RF) classifiers. They utilized BERT and GloVe embeddings to the algorithms. Logistic Regression with GloVe embeddings gained 69% accuracy on the dataset that involves context.

Hashtags exhibit a meaningful role in the content on Twitter. [9] extracted 9104 tweets containing hashtags such as "#sarcasm" and "#not" in Hindi and English. They implemented three SVM, KNN, and Random Forest (RF) classifiers. Random Forest (RF) showed an 81% accuracy on sarcasm detection.

[1] considered the impact of positive and negative situations on different sentiments to analyze sarcasm. They used a supervised SVM classifier and an N-gram classifier. To increase the accuracy, they optimized the RBF kernel, cost, and gamma parameters over 35000 tweets.

[16] inflicted four models: bidirectional LSTM, LSTM and CNN, SVM, and Multi-layer perception on 9400 data collected from Reddit and Twitter. Each model used 10-fold cross-validation. The ensemble method achieved the best F1 score. Very few research works executed deep learning models alongside the transformers models to improve the accuracy of the prediction of sarcasm detection models.

2.2. Bengali Language

Recently, emotion and specific sentiment analysis tasks like abusive text detection, toxicity detection, hateful speech detection from Bengali text have received extra attraction to many researchers involved in the Bengali Language Processing area.

[17] presented a deep learning approach to detect sentiment labels and emotions from Bengali, Romanized Bengali, and English YouTube comments. Skip-Gram and a continuous bag of words (CBOW) in Word2Vec are used to get the word embedding representation for CNN and LSTM model.

[18] presented a machine learning-based model and GRU-based deep neural network model to detect hateful speech from Facebook public pages' comments where GRU obtained a 70.10% accuracy. They collected 5126 comments, annotated them, and divided them into six classes.

[19] reported a deep learning approach for detecting abusive Bengali comments. Using RNN on 4700 Bengali text documents, they achieved an accuracy

of 82%. [2] used 300 Facebook comments without using any predictive algorithm to detect abusive Bengali text. [20] used Multinomial Naive Bayes (MNB), SVM, and Linear SVM to identify offensive text from 5644 posts and comments with emoticons where Linear SVM achieved 78% accuracy. [21] collected 2665 English texts from Youtube and translated them into Bengali to build the abusive text dataset. Naive Bayes (NB) classifier achieved 80.57% accuracy with a 39% f1 score using a 10 fold cross-validation.

[8] identified aggression and misogynistic aggression from English, Hindi, and Bengali text. They utilized En-BERT, RoBERTa, DistilRoBERTa, SVM for the English language but M-BERT, XLM RoBERTa, SVM for Bengali, and Hindi. [22] detected cyberbullying from Bengali text by NB, KNN, SVM using 2400 Bengali text collected from Facebook and Twitter. [23] tried machine learning and deep learning models for toxicity detection using 4255 Bengali comments.

The limitations of all these works symbolize the unavailability of a large-scale Bengali text corpus. For this reason, [24] constructed a dataset containing 44001 Facebook public posts' comments for helping the researchers to detect online harassment.

There is a limited number of contributions in the area of Satire, irony, or sarcasm detection. [25] detected satire in Bengali documents. They created their own Word2Vec model and achieved an accuracy of 96.4% by using the CNN model but the dataset had insufficient data. [26] identified sarcasm from 41350 Facebook posts considering public reactions and interactive comments and images. They utilized machine learning algorithms and a CNN-based model to detect sarcasm from images. Though the dataset is adequately large, the annotation process should have received special attention.

As far as we have seen, there is no comprehensive study that utilizes machine

Table 1: The Content of the Facebook Pages

News Channels	News Papers	TV Channels	Public Figures	Miscellaneous
Somoynews.tv	Prothom Alo	Zee Bangla	Shakib Al Hasan	Bdcrictime.com
Jamuna Television	Bangla Tribune	Star Jalsha	Pori Moni	Amari Dhaka
Ekattor				Udvash
BBC News Bangla				Lords Association

learning, deep learning and transfer learning to detect sarcasm. Therefore, in this paper, we have presented a comprehensive approach that includes machine learning, deep learning, and transfer learning. Besides, we have introduced a large-scale human-annotated dataset named "Ben-Sarc" containing 25636 comments written in Bengali collected from Facebook.

3. DATASET CONSTRUCTION

As far as we have seen, there is no available labeled dataset for sarcasm detection in Bengali. We felt the need to create our sarcasm detection dataset for the Bengali language. We defined our dataset as the Bengali Sarcasm dataset (Ben-Sarc). The duration of dataset construction is approximately three months. In the following subsections, we discuss the features of our Ben-Sarc dataset in detail.

3.1. Content Source

As Facebook is one of the major sources of textual data [27], we have targeted public Facebook pages to construct the Ben-Sarc dataset. We have collected Bengali Facebook comments from 14 different public pages from Bangladesh and India dated from 2013 to 2021. The content of the pages is shown in the table 1.

3.2. Content Search

Facebook comment section usually consists of the reaction of users based on the post. The commenters of targeted pages are mostly Bengali language people

and there are lots of comments written in Bengali, English, and Romanized Bengali. We have only taken the Bengali comments. All the comments have been scrapped manually by the authors of this paper.

3.3. Text Cleaning and Noise Removal

Text preprocessing is generally a vital phase of natural language processing (NLP) problems [28]. It converts text into a convenient format. The comment section of Facebook is very noisy and mostly contains errors, useless information [27]. A list of pre-processing steps has been executed on the texts collected to enrich the Ben-Sarc dataset. They are - removing non-Bengali words, duplicate texts, emojis, links, URLs; replacing #hashtag, all symbols, special characters (e.g. '\n', '%', '$', '&', '@') with a single space and multiple punctuations(e.g. '?', '।', ';', '!', ',') with single punctuation.

3.4. Annotation Process

Each text in the Ben-Sarc dataset has been annotated manually by us using '0' and '1' as we intend to work on a binary classification problem - sarcasm detection. '0' means non-sarcastic comments and '1' represents sarcastic comments. Each text in the Ben-Sarc dataset has been annotated by five annotators. The final choice on the polarity of a single text has been made using the majority voting method from five annotations. Facebook comments are frequently filled with harsh and filthy phrases, slang, and personal attacks [2] [24] [22]. As a result, we made sure that all annotators are of adult age and have domain knowledge.

3.5. Human Evaluation of Ben-Sarc Dataset

To maintain the quality of a labeled dataset, evaluation is a necessary step. We have tried to make sure the data in the Ben-Sarc dataset is not labeled vaguely keeping in mind that the researchers can use it for further applications without hesitation. The assessment process has been carefully accomplished in the Ben-Sarc dataset by two external human evaluators experts in this field.

Table 2: Inter Annotator Agreement of Ben-Sarc Assessed by Human Evaluators

Questions	Q1 in (%)	Q2 in (%)	Q3 in (%)	Q4 in (%)
Cohen's Kappa Score	98.16	87.65	16.32	30.88

Each evaluator is an adult, native Bengali speaker, and proficient in Bengali. Each evaluator has been provided the task of assessing the quality of the dataset by replying 'Yes' or 'No' to the given questions stated below:

Q1. *Is the text ironic, caustic, or biting without emoji and emoticons?*

Q2. *If Q1 is 'Yes', is the text written in the dialect, contains spelling mistakes or manipulated traditional phrases, sentences, songs, poems?*

Q3. *If Q1 is 'Yes', is there any totally opposite context in that text?*

Q4. *If Q1 is 'Yes', is there any information in the text that causes confusion to decide whether the text is sarcastic or not?*

The motivation of designing the questions for human evaluation of the Ben-Sarc dataset is from [29]. The recent advancement in the quality estimation of neural language generation (NLG) models has inspired the creation of these characteristics. [30] demonstrated that NLG models are sensitive to low-quality training samples. Thus, it is critical to evaluate the quality of comments using the characteristics of Q1. Moreover, to verify actual uniformity and fidelity, characteristics of Q2 and Q3 have been designed whereas Q4 determines if there is any ambiguous text or confusion to decide the polarity of the text. The text "আপনাদের ঠিকানা টা একটু দেন, গালি লিখে একটা চিঠি পোস্ট করব*(Please give me your address, I will post a letter with obscenities)*" creates confusion because someone may take it as an abusive text, or a threat which leads it to a non-sarcastic text where others may take it as a joke that leads to sarcasm.

The inter-annotator agreement is measured using Cohen's kappa coefficient[31] in table 2. Cohen's kappa measures annotator agreement and determines how well one annotator agrees with another. To evaluate the conventional inter-

annotator agreement, a pairwise kappa coefficient is computed using equation 1.

$$K = \frac{P_o - P_e}{1 + P_e} \tag{1}$$

where P_o represents relative observed agreement and P_e denotes the hypothesized probability of chance agreement. The quality assessment of Ben-Sarc is done on 5000 random samples of Ben-Sarc data. In most cases, the evaluators agree that the text seems ironic without any emoticons. Besides, a high percentage for Q2 indicates that dialect, manipulation of the traditional poems, songs, and spelling mistakes also express sarcasm from the text whereas a low percentage for Q3 determines the opposite context that is pretty normal. However, the Q4 raises an ambiguity to decide whether the text is sarcastic or not. In our situation, Q3 and Q4 should be in a very low percentage but the percentage of Q4 is comparatively higher than Q3 according to the inter-annotator agreement.

3.6. Dataset Description

A detailed description of our Ben-Sarc dataset has been presented in this section. The dataset contains a total of 25,636 Bengali comments where 12818 are sarcastic and 12818 are non-sarcastic. The visualization of the data distribution according to the labels is shown in fig 1.

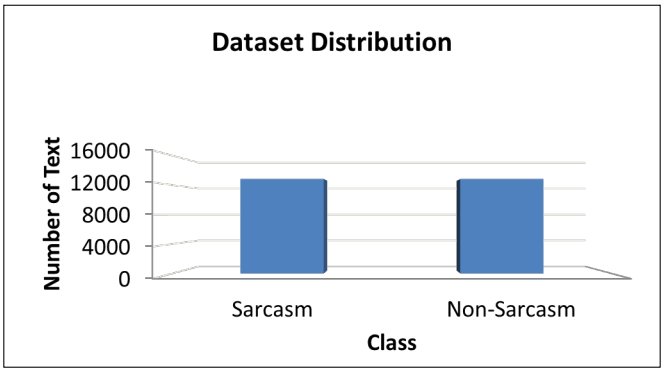

Figure 1: Data Distribution of Ben-Sarc Dataset According to Labels

Table 3 represents a short overview of our labeled dataset construction. The maximum length of a text in the Ben-Sarc dataset is 395 in words and the

Table 3: A Overview of the Ben-Sarc Dataset Preprocessing

Raw Text	Preprocessed Text	Polarity
আমার মন বলছে বাংলাদেশ জিতবে।	আমার মন বলছে বাংলাদেশ জিতবে।	0
বলছি কি এত ভালবাসা রাখবো কোথায়??? দয়া করে বাড়িতে ভালবাসা রাখবার জন্যে কিছু পাত্র পাঠিয়ে দেবেন...	বলছি কি এত ভালবাসা রাখবো কোথায়? দয়া করে বাড়িতে ভালবাসা রাখবার জন্যে কিছু পাত্র পাঠিয়ে দেবেন ।	1
এও একটা নায়িকা আর কাকও একটা গায়িকা hahaha	এও একটা নায়িকা আর কাকও একটা গায়িকা	1
করোনা ভাইরাস ওহ নাকি আবেদন করছে?? :p	করোনা ভাইরাস ওহ নাকি আবেদন করছে?	1
খুব ভাল অনুপ্রেরণার এক শাকিব!	খুব ভাল অনুপ্রেরণার এক সাকিব!	0
এই নয়া দামান গানের সঙ্গে নেচেছিলেন ঢাকা মেডিকেলের ডাক্তাররাও, কেন তারা ওই ভিডিও বানিয়েছিলেন জানতে পারবেন এই ভিডিওতে--> https://www.youtube.com/watch?v=geIKbjOBZtM	এই নয়া দামান গানের সঙ্গে নেচেছিলেন ঢাকা মেডিকেলের ডাক্তাররাও, কেন তারা ওই ভিডিও বানিয়েছিলেন জানতে পারবেন এই ভিডিওতে ।	0
বেসম্ভব সুন্দর ছবি□□□□	বেসম্ভব সুন্দর ছবি	1

Table 4: Overall Summary of Ben-Sarc Dataset

	Number of Comments	Number of Words	Number of Unique Words
Sarcasm	12818	195445	28056
Non-Sarcasm	12818	184535	24838
Total	25636	379980	52894

minimum is 3 in words. Thus, the average length of a comment is 15. The length-frequency distribution of the whole dataset has been shown in fig 2. For better visualization, the length of the text has limited to 100. The overall summary of the Ben-Sarc dataset including the number of comments, words, and unique words according to its classes has been shown in table 4. The visualization of the statistics of the Ben-Sarc dataset has been shown in fig 3.

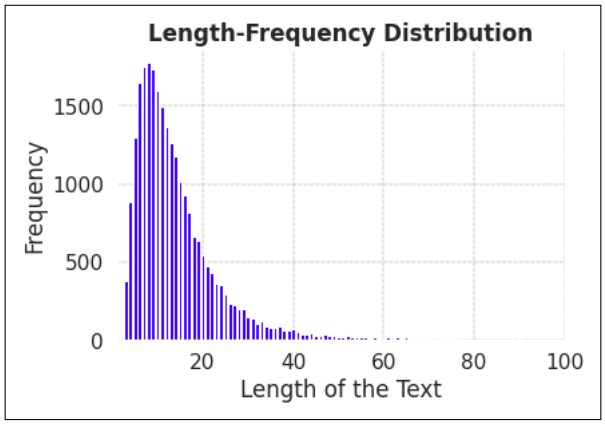

Figure 2: Length-Frequency Distribution of the Ben-Sarc Dataset

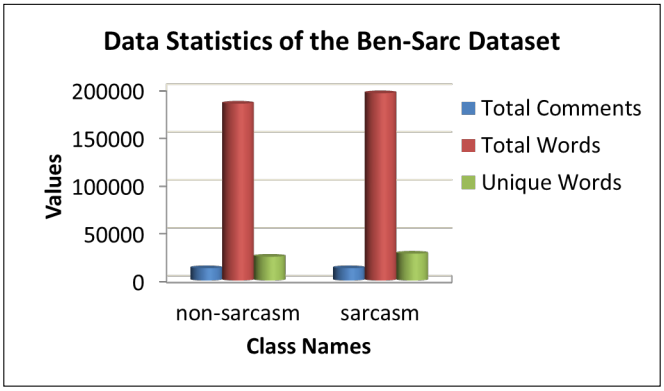

Figure 3: The Visualization of the Statistics of the Ben-Sarc Dataset

4. PROPOSED METHODOLOGY

In this section, we present our proposed methodology for sarcasm detection in brief. Figure 4 represents our proposed approach. We have distributed our proposed approach into five phases. The first phase comprises dataset construction. The second phase involves dataset preprocessing by utilizing a few natural language preprocessing techniques like punctuation removal and tokenization. The third phase incorporates the feature selection process. This process includes TF-IDF (Term-Frequency, Inverse Document Frequency) and n-grams for traditional machine learning models, word embeddings for deep learning models, and pre-trained transformer-based models for transfer learning. The fourth phase of our proposed method is the training phase. In this phase, we have employed traditional machine learning models, deep learning models, and transfer learning to classify text as sarcastic or non-sarcastic. We have examined the performance of each classifier and presented the best-performed classifier in the last phase. The details of all the phases are discussed in the following subsections.

4.1. Phase I - Dataset Construction

We have collected 25636 Facebook comments written in Bengali. The overall dataset construction process is described in section 3.

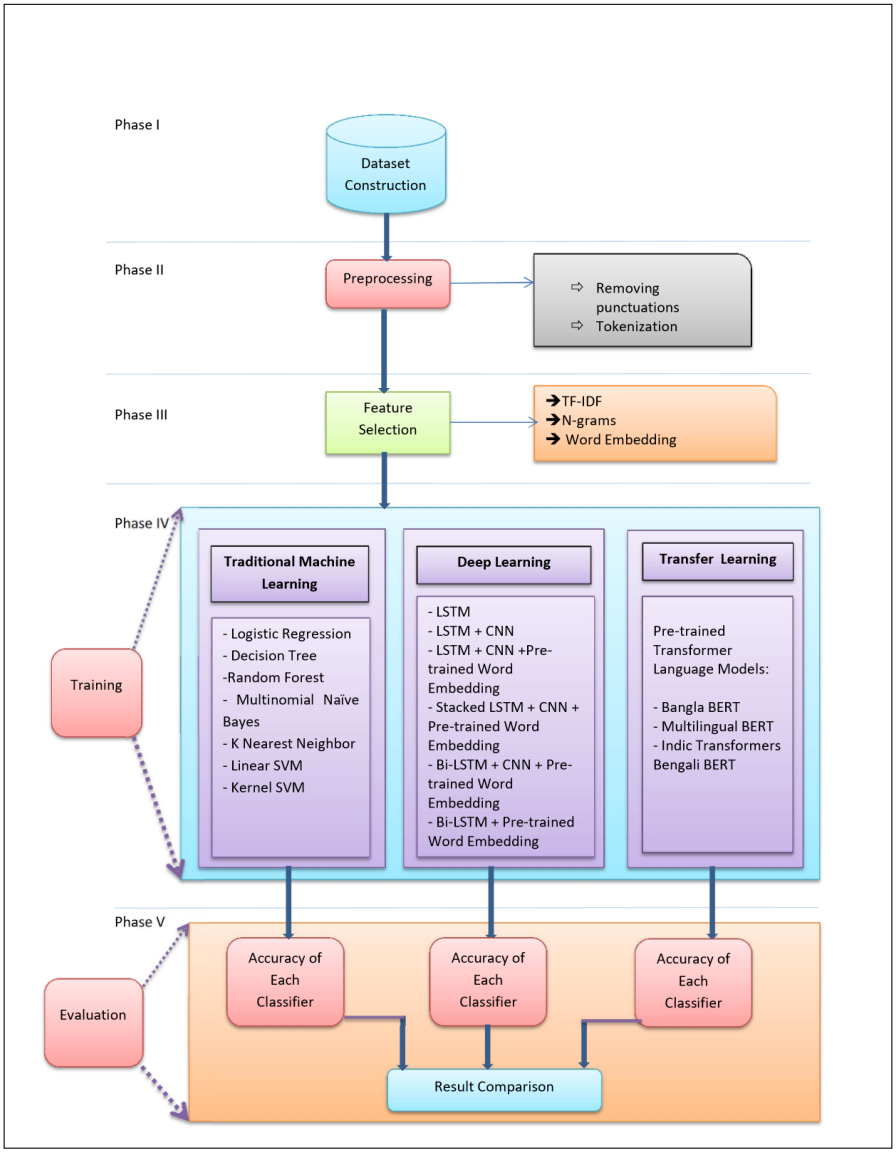

Figure 4: Work Flow of the Proposed Approach of This Paper

4.2. Phase II - Preprocessing

A few pre-processing steps have been executed before model training which is - punctuation removal (e.g. '!', '?'), tokenization.

Elongated words often contain some sentiment information. For example,

"খুউউব মজাআআআার *(Veryyyy funnnnny)*" emphasizes more positive sentiment than "খুব মজার *(Very funny)"* [17]. So, we have not applied stemming and lemmatization to preserve the actual sense of the elongated words.

4.3. Phase III - Feature Selection

Feature selection is the third phase of our proposed model. We have used three feature extraction approaches: n-grams, TF-IDF, and word embeddings. For traditional machine learning classifiers, we have used TF-IDF and n-grams methods. TF-IDF is the most extensively utilized traditional feature extractor approach in classification applications [32]. It is a mathematical statistic that reveals to us how essential a term is to a document in a collection. The increase in a word's TF-IDF value is directly proportional to the number of times that term appears in the document but is offset by the frequency of the term in the corpus, which helps to balance out terms that come more commonly in general.

$$(TF{-}IDF)_{t,d} = tf_{t,d} * \log \frac{N}{df_t} \tag{2}$$

where, $tf_{t,d}$ indicates frequency of term t in document d, df_t defines total number of documents containing term t, N means the number of documents. To pick the features for deep learning models different pre-trained word embedding for Bengali is used. All are explained in detail in section 4.4.2. The selected features of transfer learning are pre-trained language models described in detail in section 4.4.3.

4.4. Phase IV - Training

To classify whether a text is sarcastic or not, we have investigated traditional classifiers, deep learning classifiers, and transfer learning techniques. A comprehensive description of all the models is manifested in the following subsections.

4.4.1. ***Traditional Classifiers***

We have initiated the sarcasm detection system by investigating traditional classifiers. We have used Logistic Regression (LR), Decision Tree (DT), Random

Forest (RF), Multinomial Naive Bayes (MNB), K Nearest Neighbors (KNN), Linear Support Vector Machine (SVM), and Kernel SVM as traditional machine learning classifiers. Furthermore, we have applied all possible combinations of unigram, bigrams, and trigrams by extracting the features using TF-IDF for both 5 and 10 fold cross-validation.

The traditional classifiers are incapable of capturing the sequential information present in the text. Besides, these are unsuitable for enhancing performance with a large number of data. So, we will experiment the performance of the Ben-Sarc dataset with deep learning models in the later.

4.4.2. ***Deep Learning Models***

Recurrent neural networks (RNNs) are deep learning neural networks that are specially built to learn data sequences and are mostly used for textual data categorization. The learning process is carried out at hidden recurrent nodes based on their prior layers of nodes. However, when dealing with long sequences of data, RNNs suffer from the vanishing gradient problem. Long Short Term Memory (LSTM) [33] networks are a form of a recurrent neural network capable of learning order dependency in sequence prediction applications. LSTM has introduced a solution to the vanishing gradient problem and has shown to be efficient in various NLP-related applications. So, LSTM is chosen as our baseline model. Then, the LSTM model is expanded to understand the network's behavior.

4.4.2.1 Required Basic Components for Sarcasm Detection Models

. In this subsection, the basic components of sarcasm detection models are explained. If the reader is knowledgeable about these components, this subsection can be omitted.

- **LSTM:** LSTMs interpret input sequences as pairs $(x_i, y_i)....(x_z, y_z)$. An LSTM maintains a hidden vector $\mathbf{h}_t$ and a memory vector $\mathbf{m}_t$ for each pair (x_i, y_i) and at each time step t, which are responsible for regulating

state updates and outputs to create a target output y_i depending on the previous state of the x_i input. At time step t, the computations are as follows [34] [35]:

$$h_t = f(W_x t + U h_{t-1} + b) \tag{3}$$

$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \tag{4}$$

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \tag{5}$$

$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \tag{6}$$

$$g_t = \sigma(W^g x_t + U^g h_{t-1} + b^g) \tag{7}$$

$$c_t = f_t \odot c_t - 1 + i_t \odot g_t \tag{8}$$

$$h_t = o_t \odot tanh(c_t) \tag{9}$$

where σ indicates sigmoid function and $\odot$ indicates element-wise multiplication. W_i, U_i, and b_i are two weight matrices and a bias vector for input gate i respectively. The meaning is the same as for forget gate f, output gate o, tanh layer u, memory cell c, and hidden state h. The forget gate selects which past information should be forgotten on its own, whereas the input gate decides what new information should be placed in the memory cell. Finally, the output gate determines how much information from the internal memory cell is revealed. This gate unit assists an LSTM model in remembering important information over numerous time steps.

- **CNN:** A CNN [36] is mainly made up of convolutional layers and pooling layers. The convolutional layers include weights that must be taught, whereas the pooling layers change the activation using a fixed function.
  - **Convolutional Layer:** A convolutional layer is made up of a number of kernels whose parameters must be learnt. It is a local feature extractor layer with well-trained kernels for weight modification utilizing the back-propagation approach [37]. The kernels' height and

weight are less than those of the input volume. Every filter is convolved with the input volume to generate a neuron activation map. The convolutional layer's output volume is calculated by stacking the activation maps of all filters along the depth dimension. Convolution operation output is calculated by convolving an input (I) with a number of filters as follows.

$$x_k = I * W_k + b_k; k = 1, 2, 3, ..., F \tag{10}$$

where F is the number of filters, x_k is the output corresponding to the kth convolution filter, W_k is the weights of the kth filter, and b_k is the kth bias.

- **Global Max Pooling Layer:** A pooling layer is an additional layer that is inserted after the convolutional layer. Pooling layers give a method for downsampling feature maps by summarizing the existence of features in feature map patches. Maximum pooling, or max pooling [38], is a pooling operation that calculates the maximum value in each patch of each feature map. The Global Max Pooling layer is another form of pooling layer where the pool size can be fixed to the same as the input size so that the maximum of the total input is calculated as the output value.

- **Embedding Layer:** An embedding layer is learnt alongside a neural network model on a particular natural language processing application, such as language modeling or text categorization. If an input sentence s_i is given, the word sequences of this sentence w_1,w_2,w_3,...,w_i is fed into a word embedding layer to produce embedding vectors x_1,x_2,x_3,...,x_i before being sent to the next layer. The embedding layer is defined by an embedding matrix $E \epsilon R^{KX|V|}$, where K indicates the embedding dimension and $|V|$, the vocabulary size.

- **Pre-trained Word Embeddings:** Pre-trained Word Embeddings are embeddings that are learnt in one task and then applied to solve another

related problem. In this paper, we have used the following pre-trained word embeddings available in Bengali.

- **GloVe** [39] creates the feature vector based on global and local word counts, word-word co-occurrence, and local context with the center word. GloVe's semantic and syntactic features can be extracted more effectively. However, owing to matrix factorization, it takes a long time. In our task, we have used Bengali-GloVe[3].
- **Word2Vec** [40] is a prediction-based embedding approach that generates an embedding vector from the center word to the context word or vice versa. In this paper, we have used Bengali-Word2Vec[3].
- **BPEmb** [41] model based on Byte-Pair encoding, which gives a collection of pre-trained subword embedding models for 275 languages including Bengali[4].
- **FastText** [42] is a prediction-based embedding approach that conveys sub-word information. We have used FastText created for 157 languages including Bengali[5].

4.4.2.2 Sarcasm Detection Models Architecture

. A detailed description of all deep learning models for sarcasm detection is provided below.

a. **LSTM:** A single hidden LSTM layer is followed by a typical feedforward output layer in the original LSTM model. After preprocessing, texts are passed through a tokenizer and a one-hot encoding vector of length 100 is generated because Facebook comments are usually long. These vectors are then fed into the embedding layer. The output of the embedding layer is fed into the LSTM layer. Finally, a dense layer is added with a sigmoid

[3] https://bnlp.readthedocs.io/en/latest/

[4] https://bpemb.h-its.org/

[5] https://fasttext.cc/docs/en/crawl-vectors.html

activation function. The number of nodes in the dense layer is two because of the binary classification task. The vocabulary size is 10000.

b. **LSTM + CNN:** This model is the combination of the LSTM and CNN models. The architecture of LSTM and the input of the embedding layer are the same as the model mentioned in subsection 4.4.2.2(a). After the embedding layer, a 1D convolutional layer with 100 filters and kernel size 4 is added to speed up the longer training time. Next, a global max pooling layer with pool size 5 is used to extract the maximum value from each filter and the output is the input of the LSTM layer. This vector is directly passed to a dense layer which is the output layer with sigmoid activation function and the number of output nodes is the number of labels in the dataset.

c. **LSTM + CNN + Pre-trained Word Embedding:** The architecture of this model is the same as the model mentioned in subsection 4.4.2.2(b). The weights of the embedding layer are initialized with the weights of pre-trained word embedding. A dropout layer, then a dense layer is added after the embedding layer. After that, a 1D convolutional layer and a global max-pooling layer are added and the output is the input of the LSTM layer. This vector is directly passed to the output layer with sigmoid activation function as mentioned in subsection 4.4.2.2(b).

d. **Stacked LSTM + CNN + Pre-trained Word Embedding:** This model is the combination of stacked LSTM and CNN models. The Stacked LSTM is a variation of the LSTM model that includes multiple hidden LSTM layers, each of which contains multiple memory cells. The architecture of CNN is the same as the model mentioned in subsection 4.4.2.2(c). The output of the LSTM with 1D convolution is passed to another LSTM layer before being used as the input of a dense layer. The obtained vector is directly passed to a dense layer which is the output layer with sigmoid activation function and the number of output nodes is the number of labels in the dataset.

e. **Bi-LSTM + CNN + Pre-trained Word Embedding:** A bidirec-

tional LSTM [43], often known as biLSTM, is a sequence processing model that consists of two LSTMs, one of which takes the input forward and the other backward. BiLSTMs effectively improve the quantity of data available to the network, allowing the algorithm to understand the context better (knowing what words immediately follow and precede a word in a sentence). The architecture of the model remains the same as the model mentioned in subsection 4.4.2.2(c). Only the LSTM layer is replaced with the BiLSTM layer.

f. **Bi-LSTM + Pre-trained Word Embedding:** The architecture of the model remains the same as the model mentioned in subsection 4.4.2.2(e) by dropping the CNN portion of the model.

Deep learning models require a longer training time as these process input sequence token by token. As a result, we will monitor how the Ben-Sarc dataset performs on transfer learning in the later to save the computational cost.

4.4.3. ***Transfer Learning***

Transfer learning is a machine learning procedure in which the starting point of a new task is an already produced model for similar tasks [44]. Transfer learning approaches have been effectively used for speech recognition, document categorization, and sentiment analysis in natural language processing [45]. Figure 5 represents an illustration of the transfer learning approach.

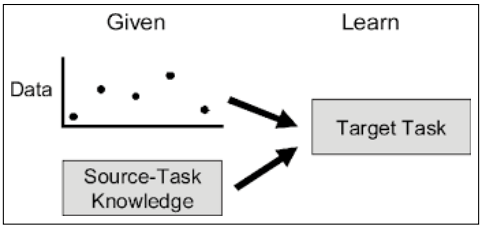

Figure 5: Illustration of Transfer Learning [44]

In transfer learning, we can utilize pre-trained source models available for developing new models. A plethora of transformer-based models for various

NLP tasks has recently emerged. The significant improvement of transformer-based models over RNN-based models is that these models accept the complete sequence as input all at once instead of analyzing an input sequence token by token. For this reason, we have utilized BERT. But BERT is a large neural network architecture with a massive number of parameters that may vary from 100 million to over 300 million. As a result, training a BERT model from scratch on a limited dataset would result in overfitting. Moreover, the computational cost of pre-training a BERT model is very high. As a result, as a starting point, it is preferable to employ a pre-trained BERT model that was trained on a large dataset. We can then further train the model using our relatively small dataset for fine-tuning. This approach is called fine-tuning. That is why we have used transfer learning approaches.

4.4.3.1 BERT

. Bidirectional Encoder Representations from Transformers(BERT) [46] is one of the most prevalent transformer-based models which is used for pre-training a transformer [47]. BERT generates deep bidirectional word representations in unlabeled text based on the words' contextual relationships to their surroundings. Depending on its vocabulary, it generates word-piece embeddings. BERT pre-training is carried out using a masked language model (MLM), which randomly masks words that the model will estimate and compute the loss, and a next sentence prediction task, in which the model can predict the next sentence from the present sentence.

Let $a1$,$a2$,....,$a6$ be sentence words. $a5$ is randomly masked with the $[MASK]$ token. The output of the sentence's words is thus $b1$, $b2$,..., $b6$. The outputs are then routed through a block that includes two Fully Connected Layers, a GELU layer, and a normalization layer. The sentence and the anticipated value of the masked token are both outputs of the block. Three pre-trained BERT-based transformer language models are used as source models available in Hugging

Face Transformer's library[6] as these are mostly used in downstream works like text classification. The transformer language models are-

- **Bangla BERT(base)** [48], a pre-trained Bengali language model based on mask language modeling that has been pre-trained on Bengali Wikipedia Dump dataset[7] and a large Bengali corpus taken from Open Super-large Crawled Aggregated coRpus (OSCAR)[8]. The model follows the bert-base-uncased model architecture that means it has 12 layers, 768 hidden layers, 12 heads, and 110M parameters.

- **Indic-Transformers Bengali BERT** [49], a BERT language model that has been pre-trained on about 3 GB of monolingual training corpus, majorly taken from OSCAR[8]. It has achieved state-of-the-art performance on the Bengali language for the text classification tasks.

- **Multilingual BERT(m-BERT)** [46], a pre-trained model on 102 languages with the largest Wikipedia including Bengali. We have used the model 'bert-base-multilingual-uncased'. It has 12 layers with 768 hidden layers, 12 multi-headed attention layers, and 110M parameters.

4.4.3.2 Architecture of Our Model

. The detailed architecture of our model is shown in fig 6. The fine-tuning strategies of our new models can be divided as follows:

a. **Selecting Pre-trained Source Model:** As explained earlier, BERT-based transformer models mentioned in section 4.4.3.1 are taken to this experiment as pre-trained source models to observe how these models work on transfer learning.

b. **Freezing the Entire Architecture of Source Model:** Before fine-tuning, all the layers of each pre-trained language model are kept frozen

[6] https://huggingface.co/

[7] https://dumps.wikimedia.org/bnwiki/latest/

[8] https://oscar-corpus.com/

by freezing BERT's weight. This process prevents updating of model weights during fine-tuning.

c. **Attaching Our Own Neural Networks Architecture:** A different number of dense layers with different activations mentioned in 5.4.3 and softmax as output layer of our own are appended to the architecture to train this new model. Softmax can be expressed as-

$$a_i = \frac{e^{z_i}}{\sum_{k=1}^{c} e^{z_k}} where \sum_{i=1}^{c} a_i = 1 \tag{11}$$

The weights of the appended layers are updated during model training. Different optimizers and learning rates are experimented with to get the optimized hyperparameter which is explained in section 5.4.3.

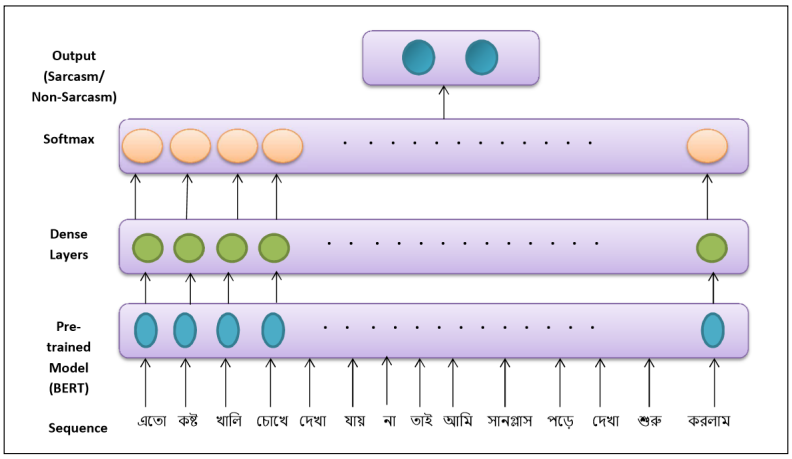

Figure 6: Model Architecture of Sarcasm Detection for Transfer Learning

4.5. Phase V - Evaluation

In the last phase, we have measured the performance of all models of phase IV. Then, the achieved results are compared and the best-performed mode is reported. The details of measuring the performance of the models are discussed briefly in the experiment section.

5. EXPERIMENTAL EVALUATION

5.1. Experimental Setup

Python Keras framework with Tensorflow is used as a background to implement all deep learning models and Pytorch library is used for transfer learning models for training, tuning, and testing. Experimental evaluation was conducted on a machine with an Intel Core i5 processor with 2.71GHz clock speed and 4GB RAM. Tensorflow based experiments can utilize GPU instructions. Google Colaboratory has been used for developing all the models described in this paper in later sections as we have used Python language.

5.2. Experiments

Our experiments are categorized into three parts. Experiment I is concerned with the experiments on traditional classifiers. Experiment II is focused on the experiments on deep learning classifiers and experiment III is reported on the experiments on transfer learning approaches.

To judge the effectiveness of the models, accuracy, precision, recall, and f1-score measurements are taken into account. After hyperparameter tuning, the variation of results has been obtained for each model. So, only the better-performed model from each experiment has been taken in the Result Analysis section.

5.2.1. Experiment I

In this experiment, the performance of traditional machine learning classifiers mentioned in 4.4.1 for the Ben-Sarc dataset has been evaluated. For this experiment, 20% of our data is used for testing purposes. The rest is used for training. A full overview of the performance with necessary evaluation metrics for the experiment I has been demonstrated in the table 5. The process of choosing hyperparameters for the experiment I has been discussed in section 5.4.1. From table 5, it is shown that the MNB classifier has achieved the highest accuracy for the bigram technique with both 5-fold and 10-fold cross-validation

Table 5: Performance(in %) of 5-Fold and 10-Fold Cross-Validation for Experiment I

Classifier	5-Fold					10-Fold				
	Technique	Accuracy	Precision	Recall	F1 Score	Technique	Accuracy	Precision	Recall	F1 Score
LR	bigram	71.80	71.01	73.71	72.33	bigram	72.06	71.29	73.88	72.56
DT	bigram	61.92	61.58	63.44	62.49	bigram	62.42	62.17	63.37	62.76
RF	trigam	70.52	69.93	72.03	70.96	trigram	70.89	70.11	72.89	71.47
MNB	**bigram**	**72.01**	**72.54**	**70.81**	**71.67**	**bigram**	**72.36**	**72.92**	**71.14**	**72.02**
KNN	unigram	64.52	67.72	55.50	61.00	unigram	64.67	67.91	55.63	61.14
Linear SVM	unigram	71.03	69.04	76.28	72.47	unigram	71.29	69.47	75.99	72.58
Kernel SVM	unigram	71.51	70.15	74.95	72.46	unigram	71.84	70.59	74.90	72.67

among all traditional classifiers as it works well with high dimensional text data by taking the advantage of probabilistic algorithm. It is 72.01% for 5 fold cross-validation and 72.36% for 10 fold cross-validation.

5.2.2. Experiment II

In this experiment, the performance of different deep learning classifiers mentioned in subsection 4.4.2 for the Ben-Sarc dataset has been evaluated. For this experiment, 20% of our data is used for testing purposes. The rest is further divided into 60% for training and 20% for the validation set. A full overview of the performance with necessary evaluation metrics for experiment II has been demonstrated in the table 7. For LSTM models, LSTM units have been taken as 100 and for stacked LSTM, LSTM units have been taken as 128 and 64. 100 epochs have been used for all deep learning models. Though the epoch number was set as 100, it was stopped earlier due to early stopping criteria for monitoring 2 epochs with no improvement of the model's performance. Binary cross-entropy is used as the loss function for all cases as the task is a binary classification problem. For all cases, dropout probability, and recurrent dropout probability have been set as 0.2. The hyperparameter setting is shown in the table 6. The procedure for picking hyperparameters for experiment II is covered in section 5.4.2.

From table 7, it is clear that LSTM without pre-trained word embedding has achieved the highest accuracy of 72.48%. When an extra CNN and max-pooling layer is added to this model, the performance of the model has decreased slightly. Then, the performance of this LSTM+CNN model has decreased more after

Table 6: Hyperparameter Setting of Each Best Performed Model for Experiment II

Model	Dense Layer Size	Batch Size	Activation in Hidden Layers	Optimizer	Learning Rate
LSTM	1000	16	tanh	Nadam	0.0001
LSTM+CNN	1000	16	tanh	Nadam	0.0001
LSTM+CNN+GloVe	1000	16	tanh	Nadam	0.00001
Stacked LSTM+CNN+GloVe	1000	16	tanh	Nadam	0.00001
BiLSTM+CNN+GloVe	1000	16	tanh	Nadam	0.00001
BiLSTM+GloVe	1000	16	tanh ReLU	Nadam	0.00001

Table 7: Performance(in %) of Each Model for Best Setting of Experiment II

Model	Accuracy	Precision	Recall	F1 Score
LSTM	**72.48**	**72.52**	**72.53**	**72.53**
LSTM+CNN	69.40	69.31	69.49	69.39
LSTM+CNN+GloVe	66.09	65.99	65.92	65.95
Stacked LSTM+CNN+GloVe	65.91	65.80	65.88	65.84
BiLSTM+CNN+GloVe	65.58	65.67	65.84	65.75
BiLSTM+GloVe	61.64	62.04	58.66	60.24

using pre-trained word embedding. After that, the performance of other models has decreased gradually by adding or removing certain parts of the model. The reason for decreasing the models' accuracy using pre-trained word embedding is that pre-trained word embeddings are mainly trained on a large dataset like Wikipedia where most of the language is very formal. But in the Ben-Sarc dataset, 87.65% of text are written in dialect, manipulating phrases, sentences, and spelling mistakes which determines the text as sarcastic as mentioned in section 3.5.

5.2.3. Experiment III

In this experiment, the performance of transfer learning techniques mentioned in 4.4.3 for the Ben-Sarc dataset has been evaluated. A full overview of the performance with necessary evaluation metrics for experiment III has been demonstrated in the table 9. The hyperparameter setting is shown in the table 8. The approach for optimizing hyperparameters for experiment III is outlined in section 5.4.3.

Here, for all cases, Negative Log-Likelihood (NLL) [50] Loss has been used as a loss function as it is the classic loss function used in any classification task [51].

Table 8: Hyperparameter Setting of Each Best Performed Model for Experiment III

Pre-trained Model	No of Hidden Layers	No of Nodes in Hidden Layers	Activation in Hidden Layer	Dropout	Optimizer	Learning Rate	Batch Size	No of Epoch
M- BERT	7	512, 256, 128, 64, 32, 16, 8	tanh	0.1	Adam	0.0001	8	30
Indic-Transformers Bengali BERT	7	512, 256, 128, 64, 32, 16, 8	tanh	0.1	Adam	0.0001	8	30
Bangla BERT	7	512, 256, 128, 64, 32, 16, 8	tanh	0.1	Adam	0.0001	8	30

Table 9: Performance(in %) of Each Model for Best Setting of Experiment III

Pre-trained Model	Accuracy	Precision	Recall	F1 Score
M- BERT	66.00	67.00	64.00	65.47
Indic-Transformers Bengali BERT	**75.05**	**74.00**	**77.00**	**75.47**
Bangla BERT	68.00	69.00	64.00	66.41

Softmax activation has been used in the output layer for all cases. From table 9, it can be concluded that transfer learning approaches for Indic-Transformers Bengali BERT pre-trained model has obtained the highest accuracy among all pre-trained models. It has achieved 75.05% of accuracy by using seven hidden layers and the settings mentioned above.

For the transfer learning approach, at first, we have taken the m-BERT transformer model as a pre-trained model. But the overall performance was not satisfactory. Then, we have replaced the pre-trained model with Indic-Transformers Bengali BERT keeping the same hyperparameter setting. Here, a significant increase in all the performance measurement metrics has been observed. Almost 9% accuracy has been increased by changing only the pre-trained model from m-BERT to Indic-Transformers Bengali BERT. Then, we have experimented with another pre-trained model Bangla BERT, but the performance has degraded significantly.

Table 10: A Short Overview of the Best Performed Model(in %) from Each Experiment

Experiment No	Model	Accuracy	F1 Score
Experiment I	MNB	72.36	72.02
Experiment II	LSTM	72.48	72.35
Experiment III	**Indic-Transformers Bengali BERT**	**75.05**	**75.47**

5.3. Result Analysis

The highest accuracy from each experiment has been shown briefly in table 10. From the table 10, it can be concluded that the performance of experiment III, which means the transfer learning approach is slightly better than traditional machine learning and deep learning classifiers. By using Indic-Transformers Bengali BERT as a pre-trained model, transfer learning has obtained the highest accuracy of 75.05% for the Ben-Sarc dataset where LSTM without pre-trained word embeddings from deep learning classifiers and multinomial Naive Bayes from traditional classifiers achieved a maximum 72.48% and 72.36% accuracy respectively.

5.4. Hyperparameter Tuning

Hyperparameter tuning is a necessary stage in each experiment to boost performance. The hyperparameter tuning has been carried out on all of our experiments I, II, and III. A thorough explanation of all of the models is provided in the following subsections.

5.4.1. For Experiment I

For the experiment I, we have applied 5-Fold and 10-Fold cross-validation on seven traditional classifiers listed in table 5. Unigram, bigram, and trigram techniques have been applied for each classifier. Among them, the best results from each classifier have been demonstrated in table 5 for both cross-validation techniques. MNB classifier for 5-fold cross-validation has attained 71.85% accuracy and 72.13% for 10-fold cross-validation for the unigram technique. These are the second-best results for both 5-fold and 10-fold cross-validation, respectively. The performance of other classifiers cannot surpass these results.

Table 11: Hyperparameters with Their Values Which Tuned across All Models for Experiment II

Pre-trained Word Embedding	Dense Layer Size	Batch Size	Activation in Hidden Layers	Optimizer	Learning Rate
GloVe, Word2Vec, BPEmb, fastText, and without Pre-trained word embedding	1000, 2000	8, 16, 32	tanh, ReLU,	Adam, Nadam	0.01, 0.001, 0.0001, 0.00001

Table 12: Hyperparameters with Their Values Which Tuned across All Models for Experiment III

Pre-trained Source Models	Batch Size	No of Hidden Layers	No of Nodes in Hidden Layers	Activation in Hidden Layers	No of Dropout Layers	Dropout Probabilities	Optimizer	Learning Rate	No of Epoch
Bangla BERT, Indic-Transformers Bengali BERT, m-BERT	4,8, 16, 32	1,2,3, 5,7,9	1024,512, 256,128, 64,32,16,8,4	tanh, ReLU, ReLU6,sigmoid, selu	1,2,3	0.1,0.2,0.5	Adam, AdamW, Adamsx, SGD, RMSprop	0.01, 0.001, 0.0001, 0.00001, 0.0005	10,20,30

5.4.2. For Experiment II

For experiment II, all hyperparameters which have been tuned for several combinations across all models are mentioned in table 11. Among them, LSTM without pre-trained word embedding has achieved 71.37% on dense layer 1000, the number of LSTM layers 2, batch size 16, hidden layer activation tanh, Nadam optimizer with 0.0001 learning rate. This is the second-highest accuracy for experiment II. Other models with different combinations cannot obtain better this result.

5.4.3. For Experiment III

For experiment III, all hyperparameters which have been tuned for several combinations for all models are mentioned in table 12. The second highest accuracy from experiment III is 74.00% which is achieved from two settings - first one : 3 hidden layers with 512,256, 128 hidden layer nodes, tanh hidden layer activation, 0.1 dropout, SGD optimizer with 0.01 learning rate, batch size 8 with 30 epochs and the second one: 4 hidden layers with 512,256, 128, 64 hidden layer nodes, sigmoid hidden layer activation, 0.2 dropout, Adam optimizer with 0.001 learning rate, batch size 4 with 30 epochs.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented a benchmark dataset for Bengali sarcastic comments on Facebook to make an influence in one of the low resources languages named Bengali. Then we have demonstrated a thorough and comprehensive strategy to utilize different models of machine learning, deep learning, and transfer learning. This is an attempt to make a contribution in the discipline of sentiment analysis on the Bengali language domain to achieve a boon in the branch of consumer research, opinion mining, branding, and so on. In the future, we wish to improve the quality of our work by increasing the size of our Ben-Sarc dataset. Besides, emoji and emoticons play a vital role to articulate the actual connotation of a comment on social media. So, we will consider emoji and emoticons along with the text.

References

[1] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, R. Huang, Sarcasm as contrast between a positive sentiment and negative situation, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Seattle, Washington, USA, 2013, pp. 704–714.

[2] M. G. Hussain, T. A. Mahmud, W. Akthar, An approach to detect abusive bangla text, in: 2018 International Conference on Innovation in Engineering and Technology (ICIET), 2018, pp. 1–5. doi:10.1109/CIET.2018.8660863.

[3] C. Eke, A. Norman, L. Shuib, H. Nweke, Sarcasm identification in textual data: systematic review, research challenges and open directions, Artificial Intelligence Review 53. doi:10.1007/s10462-019-09791-8.

[4] M. Hossain, M. Hoque, N. Siddique, I. Sarker, Bengali text document categorization based on very deep convolution neural network, Expert Systems with Applications 184 (2021) 115394.

[5] N. Romim, M. Ahmed, H. Talukder, M. S. Islam, Hate Speech Detection in the Bengali Language: A Dataset and Its Baseline Evaluation, 2021, pp. 457–468. doi:10.1007/978-981-16-0586-4_37.

[6] E. Lunando, A. Purwarianti, Indonesian social media sentiment analysis with sarcasm detection, in: 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS), 2013, pp. 195–198. doi:10.1109/ICACSIS.2013.6761575.

[7] S. K. Bharti, K. S. Babu, R. Raman, Context-based sarcasm detection in hindi tweets, in: 2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR), 2017, pp. 1–6. doi:10.1109/ICAPR.2017.8593198.

[8] A. Baruah, K. Das, F. Barbhuiya, K. Dey, Aggression identification in English, Hindi and Bangla text using BERT, RoBERTa and SVM, in: Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying, European Language Resources Association (ELRA), Marseille, France, 2020, pp. 76–82.

[9] N. Pawar, S. Bhingarkar, Machine learning based sarcasm detection on twitter data, in: 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 957–961. doi:10.1109/ICCES48766.2020.9137924.

[10] T. Ptácek, I. Habernal, J. Hong, Sarcasm detection on czech and english twitter, in: COLING, 2014.

[11] S. Hiai, K. Shimada, Sarcasm Detection Using Features Based on Indicator and Roles, 2018, pp. 418–428. doi:10.1007/978-3-319-72550-5_40.

[12] K. Sentamilselvan, P. Suresh, G. K. Kamalam, S. Mahendran, D. Aneri, Detection on sarcasm using machine learning classifiers and rule based approach, IOP Conference Series: Materials Science and Engineering 1055 (1) (2021) 012105. doi:10.1088/1757-899x/1055/1/012105.

[13] D. Bamman, N. Smith, Contextualized sarcasm detection on twitter, Proceedings of the International AAAI Conference on Web and Social Media 9 (1) (2021) 574–577.

[14] Z. Wang, Z. Wu, R. Wang, Y. Ren, Twitter sarcasm detection exploiting a context-based model, in: Proceedings, Part I, of the 16th International Conference on Web Information Systems Engineering — WISE 2015 - Volume 9418, Springer-Verlag, Berlin, Heidelberg, 2015, p. 77–91. doi:10.1007/978-3-319-26190-4_6.

[15] A. Khatri, P. P, Sarcasm detection in tweets with BERT and GloVe embeddings, in: Proceedings of the Second Workshop on Figurative Language Processing, Association for Computational Linguistics, Online, 2020, pp. 56–60. doi:10.18653/v1/2020.figlang-1.7.

[16] J. Lemmens, B. Burtenshaw, E. Lotfi, I. Markov, W. Daelemans, Sarcasm detection using an ensemble approach, in: Proceedings of the Second Workshop on Figurative Language Processing, Association for Computational Linguistics, Online, 2020, pp. 264–269. doi:10.18653/v1/2020.figlang-1.36.

[17] N. I. Tripto, M. E. Ali, Detecting multilabel sentiment and emotions from bangla youtube comments, 2018 International Conference on Bangla Speech and Language Processing (ICBSLP) (2018) 1–6.

[18] A. M. Ishmam, S. Sharmin, Hateful speech detection in public facebook pages for the bengali language, in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019, pp. 555–560. doi:10.1109/ICMLA.2019.00104.

[19] E. A. Emon, S. Rahman, J. Banarjee, A. K. Das, T. Mittra, A deep learning approach to detect abusive bengali text, in: 2019 7th International Conference on Smart Computing Communications (ICSCC), 2019, pp. 1–5. doi:10.1109/ICSCC.2019.8843606.

[20] P. Chakraborty, M. Seddiqui, Threat and abusive language detection on social media in bengali language, 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT) (2019) 1–6.

[21] M. Awal, M. S. Rahman, J. Rabbi, Detecting abusive comments in discussion threads using naïve bayes, 2018 International Conference on Innovations in Science, Engineering and Technology (ICISET) (2018) 163–167.

[22] Abdhullah-Al-Mamun, S. Akhter, Social media bullying detection using machine learning on bangla text, 2018, pp. 385–388. doi:10.1109/ICECE.2018.8636797.

[23] N. Banik, M. H. H. Rahman, Toxicity detection on bengali social media comments using supervised models, in: 2019 2nd International Conference on Innovation in Engineering and Technology (ICIET), 2019, pp. 1–5. doi:10.1109/ICIET48527.2019.9290710.

[24] M. F. Ahmed, Z. Mahmud, Z. T. Biash, A. A. N. Ryen, A. Hossain, F. B. Ashraf, Bangla text dataset and exploratory analysis for online harassment detection, CoRR abs/2102.02478. arXiv:2102.02478.

[25] A. S. Sharma, M. A. Mridul, M. S. Islam, Automatic detection of satire in bangla documents: A CNN approach based on hybrid feature extraction model, CoRR abs/1911.11062. arXiv:1911.11062.

[26] D. Das, A. J. Clark, Sarcasm detection on facebook: A supervised learning approach, in: Proceedings of the 20th International Conference on Multimodal Interaction: Adjunct, ICMI '18, Association for Computing Machinery, New York, NY, USA, 2018. doi:10.1145/3281151.3281154.

[27] S. Salloum, M. Al-Emran, A. Monem, K. Shaalan, A survey of text mining in social media: Facebook and twitter perspectives, Advances in Science, Technology and Engineering Systems Journal 2 (2017) 127–133. doi:10.25046/aj020115.

[28] I. Hemalatha, G. S. Varma, A. Govardhan, Preprocessing the informal text for efficient sentiment analysis, International Journal of Emerging Trends Technology in Computer Science (IJETTCS) 1 (2012) 127–133.

[29] T. Hasan, A. Bhattacharjee, M. S. Islam, K. Samin, Y. Li, Y. Kang, M. S. Rahman, R. Shahriyar, Xl-sum: Large-scale multilingual abstractive summarization for 44 languages, CoRR abs/2106.13822. arXiv:2106.13822.

[30] Y. Belinkov, Y. Bisk, Synthetic and natural noise both break neural machine translation, CoRR abs/1711.02173. arXiv:1711.02173.

[31] J. Cohen, A coefficient of agreement for nominal scales, Educational and psychological measurement 20 (1) (1960) 37–46.

[32] M. Kumari, A. Jain, A. Bhatia, Synonyms based term weighting scheme: An extension to tf. idf, Procedia Computer Science 89 (2016) 555–561.

[33] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[34] A. Graves, Generating sequences with recurrent neural networks, arXiv preprint arXiv:1308.0850.

[35] N. Kalchbrenner, I. Danihelka, A. Graves, Grid long short-term memory, arXiv preprint arXiv:1507.01526.

[36] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1746–1751. doi:10.3115/v1/D14-1181.

[37] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.

[38] Y.-L. Boureau, J. Ponce, Y. LeCun, A theoretical analysis of feature pooling in visual recognition, in: Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 111–118.

[39] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[40] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.

[41] B. Heinzerling, M. Strube, Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages, arXiv preprint arXiv:1710.02187.

[42] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, T. Mikolov, Learning word vectors for 157 languages, arXiv preprint arXiv:1802.06893.

[43] M. Schuster, K. K. Paliwal, Bidirectional recurrent neural networks, IEEE transactions on Signal Processing 45 (11) (1997) 2673–2681.

[44] L. Torrey, J. Shavlik, Transfer learning, in: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques, IGI global, 2010, pp. 242–264.

[45] D. Wang, T. F. Zheng, Transfer learning for speech and language processing, in: 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), IEEE, 2015, pp. 1225–1237.

[46] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[48] S. Sarker, Banglabert: Bengali mask language model for bengali language understading (2020).
URL https://github.com/sagorbrur/bangla-bert

[49] K. Jain, A. Deshpande, K. Shridhar, F. Laumann, A. Dash, Indic-transformers: An analysis of transformer language models for indian languages (2020). arXiv:2011.02323.

[50] J. C. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in: ADVANCES IN LARGE MARGIN CLASSIFIERS, MIT Press, 1999, pp. 61–74.

[51] W. Ruan, Y. Nechaev, L. Chen, C. Su, I. Kiss, Towards an asr error robust spoken language understanding system., in: INTERSPEECH, 2020, pp. 901–905.