Natural Language Processing (2024), 1-26
doi:10.1017/nlp.2024.11

CAMBRIDGE UNIVERSITY PRESS

ARTICLE

Ben-Sarc: A self-annotated corpus for sarcasm detection from Bengali social media comments and its baseline evaluation

Sanzana Karim Lora, G. M. Shahariar, Tamanna Nazmin, Noor Nafeur Rahman, Rafsan Rahman, Miyad Bhuiyan and Faisal Muhammad Shah

Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh
Corresponding author: Sanzana Karim Lora; Email: sanzanalora@yahoo.com

(Received 6 July 2022; revised 13 September 2023; accepted 3 October 2023)

Special Issue on '**Natural Language Processing Applications for Low-Resource Languages**'

Abstract

Sarcasm detection research in 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 Bidirectional Encoder Representations from Transformers as a pre-trained source model has achieved the highest accuracy of 75.05%. The second-highest accuracy is obtained by the long short-term memory 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. The Ben-Sarc is available at https://github.com/sanzanalora/Ben-Sarc.

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 (Riloff *et al.* 2013). The use of sarcastic language is a resentment concealed as humor and intended to provoke, annoy, or convey contempt. Sarcasm, in the context of expression analysis, is a type of language expression in which the intended meaning of a remark differs from its literal interpretation. Because of its intrinsic ambiguity, it poses a distinct challenge and opportunity for investigation. Expression analysis attempts to decipher the complexities of sarcasm by evaluating verbal indicators such as tone, context, and grammatical structures. Expression analysis seeks to capture the nuances of sarcasm by interpreting the contrast between overt and implicit meanings, offering light on how language is used to transmit complex ideas and emotions. This provides useful insights into the complexity of human communication,

particularly in understanding how people utilize linguistic strategies to convey thoughts that go beyond the surface text. As the intention of sarcasm is often vague and misleading, people cannot discriminate between a true story and satire or irony (Riloff *et al.* 2013). Sarcasm, despite its sometimes vague or complex meanings, fulfills various functions in communication. Sarcasm is a rhetorical strategy used to communicate a message by exploiting the gap between the literal and intended meanings of words. Its use can be ascribed to a variety of circumstances, including humor or irony, expression of emotion, criticism, and so on. Facebook, YouTube, and Twitter are influential social media platforms for sharing people's judgments, thoughts, opinions, and sentiments nowadays (Hussain, Mahmud, and Akthar 2018). The aforementioned large amount of available data offers the extent to research in natural language processing (NLP).

Sarcasm detection in the low-resource language is a very narrow research area in NLP. Sarcasm detection is a subset of sentiment analysis problems where the focus is on recognizing sarcasm rather than identifying a sentiment across the board (Eke *et al.* 2020). Sarcasm detection research is available for high-resource languages such as English. But, despite being the world's seventh most spoken language with 240 million native speakers (Hossain *et al.* 2021), 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 (Romim *et al.* 2021).

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

In the next section, we highlight the objective of our research. Then, we briefly discuss related works on high and low-resource language sarcasm detection in Section 3. Section 4 shows the dataset creation along with the annotation process. Moreover, Section 5 explains the proposed methodology. Section 6 contains the experimental results and their analysis, while Section 7 contains the conclusion and future work.

2. Research objectives and our contribution

The purpose of this research is discussed in this section. Due to the limited resources and lack of a high-quality dataset, sarcasm detection from Bengali text is totally unexplored. To the best of our knowledge, there is no large-scale self-annotated dataset for sarcasm detection from Bengali text. Moreover, maintaining the quality of the dataset is required to produce satisfactory results. In this research, we introduce a large-scale self-annotated dataset 'Ben-Sarc' for Bengali sarcasm detection maintaining high quality by human evaluation. Then, we conduct a detailed experiment using machine learning, deep learning, and transfer learning to set a benchmark result on this dataset.

Our main contributions in this paper are summarized as follows:

- At first, we construct a large-scale self-annotated Bengali corpus for sarcasm detection. The corpus can be found at https://github.com/sanzanalora/Ben-Sarc.

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

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

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

- Then, we conduct 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.

3. 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 (Lunando and Purwarianti 2013), Hindi (Bharti, Babu, and Raman 2017; Baruah *et al.* 2020; Pawar and Bhingarkar 2020), Czech (Ptáček *et al.* 2014), and Japanese (Hiai and Shimada 2018) is available. We discuss some of the related approaches in the following literature review analysis.

3.1. English language

Pawar and Bhingarkar (2020) experimented with traditional machine learning algorithms such as Support Vector Machine (SVM), K Nearest Neighbors (KNN), and Random Forest (RF) on 9104 tweets on Twitter. Sentamilselvan *et al.* (2021) worked on both sarcasm and irony detection separately. SVM, Naive Bayes (NB), Decision Tree (DT), and RF were applied to the irony detection dataset whereas SVM and RF algorithms were to 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 tweets on Twitter to detect sarcasm. Bamman and Smith (2021) 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 on 19534 tweets. Wang *et al.* (2015) also aimed at the context for identifying sarcasm accurately. They collected 1500 tweets and derived 6774 history-based, 453 conversation-based, and 2618 topic-based contextual tweets. The sequential SVM classifier exhibited a decent accuracy of 69.13%. Khatri and P (2020) extracted 5000 tweets that include texts, labels, and contexts and analyzed the dataset through linear SVC, Logistic Regression (LR), Gaussian NB, and RF classifiers. They utilized Bidirectional Encoder Representations from Transformers (BERT) and Global Vectors for Word Representation (GloVe) embeddings in 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. Pawar and Bhingarkar (2020) extracted 9104 tweets containing hashtags such as '#sarcasm' and '#not' in Hindi and English. They implemented three SVM, KNN, and RF classifiers. RF showed an 81% accuracy on sarcasm detection.

Riloff *et al.* (2013) 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.

Lemmens *et al.* (2020) inflicted four models: bidirectional long short-term memory (LSTM), LSTM and Convolutional Neural Network (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.

Joshi *et al.* (2016) contended that sarcasm cannot be detected using current methods because they are unable to detect nuanced kinds of context incongruity. By utilizing semantic similarity/discordance between word embeddings, they suggested improving on earlier work. They examined four different word embeddings and found that sarcasm recognition has improved. The authors came to the conclusion that LSA and GloVe are less effective in sarcasm identification than Word2Vec and dependency weight-based features.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Ghosh, Fabbri, and Muresan (2018) explained how crucial it is to spot irony and sarcasm in user-generated material on social media networks. They emphasized that recognizing these applications of figurative language is essential for appreciating people's true thoughts and opinions. The research offers computer models for sarcasm identification in social media talks and looks into the efficacy of conversation context modeling in sarcasm detection.

3.2. Bengali language

Recently, emotion and specific sentiment analysis tasks like abusive text detection, toxicity detection, sarcasm detection, and hateful speech detection from Bengali text have received extra attraction from many researchers involved in the Bengali Language Processing area. Therefore, in this subsection, we discuss all these research areas on Bengali text as these all are subsets of sentiment analysis.

Tripto and Ali (2018) 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 in Word2Vec are used to get the word embedding representation for CNN and LSTM models.

Ishmam and Sharmin (2019) presented a machine learning-based model and Gated Recurrent Unit (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.

Emon *et al.* (2019) reported a deep-learning approach for detecting abusive Bengali comments. Using RNN on 4700 Bengali text documents, they achieved an accuracy of 82%. Hussain *et al.* (2018) used 300 Facebook comments without using any predictive algorithm to detect abusive Bengali text. Chakraborty and Seddiqui (2019) used Multinomial NB (MNB), SVM, and Linear SVM to identify offensive text from 5644 posts and comments with emoticons where Linear SVM achieved 78% accuracy. Awal, Rahman, and Rabbi (2018) collected 2665 English texts from YouTube and translated them into Bengali to build the abusive text dataset. The NB classifier achieved 80.57% accuracy with a 39% f1 score using a 10-fold cross-validation.

Baruah *et al.* (2020) identified aggression and misogynistic aggression from English, Hindi, and Bengali texts. They utilized En-BERT, RoBERTa, DistilRoBERTa, and SVM for the English language but Multilingual BERT (M-BERT_, XLM RoBERTa, and SVM for Bengali, and Hindi. Akhter *et al.* (2018) detected cyberbullying from Bengali text by NB, KNN, and SVM using 2400 Bengali text collected from Facebook and Twitter. Banik and Rahman (2019) tried machine learning and deep learning models for toxicity detection using 4255 Bengali comments.

Lora *et al.* (2020) investigated deep learning algorithms for emotion recognition from Facebook comments. In this analysis, the distinction between positive and negative emotions was successfully produced using stacked LSTM, stacked LSTM with 1D convolution, CNN with pre-trained word embeddings, and RNN with pre-trained word embedding models, with the latter showing the greatest accuracy with 98.3%.

The limitations of all these works in the area of Bengali sentiment analysis symbolize the unavailability of a large-scale Bengali text corpus. For this reason, Ahmed *et al.* (2021) constructed a dataset containing 44001 Facebook public posts' comments to help the researchers detect online harassment.

There is a limited number of contributions in the area of Satire, irony, or sarcasm detection. Sharma, Mridul, and Islam (2019) 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. Das and Clark (2018) 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.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Table 1. Comparison of the performances and other aspects of our study with other existing studies

Author (Year)	Dataset Size	Model Used	Accuracy (highest)	Contribution	Limitation
Sharma *et al*. (2019)	2960	CNN + Hybrid feature extraction	96.4%	Hybrid feature extraction for satire detection	Small Dataset can led potential overfitting
Das and Clark (2018)	41350	ML algorithm and CNN-based model	93.11%	considered public reactions and interactive comments and images for sarcasm detection	Dataset annotation does not receive special attention
Our Study	25636	Traditional ML, DL and Transfer Learning	75.05%	Bengali Sarcasm detection dataset	—

Memes have lately gained popularity as a means of information dissemination on social media. A meme is an idea, habit, or style that circulates throughout a community through mimicry and frequently bears symbolic significance that refers to a specific occurrence or topic. Information can circulate through memes in a sarcastic way. Therefore, many researchers find interest in working on memes. Hossain, Sharif, and Hoque (2022a) developed a multimodal hate speech dataset with 4158 memes featuring Bengali and code-mixed captions. Hossain *et al.* (2022b) proposed a methodology for identifying multilingual offense and trolling from social media memes that uses the weighted ensemble technique to apply weights to the contributing graphical, textual, and multimodal models. Table 1 shows a comparative analysis of our study and the existing literature.

As far as we have seen, there is no research work on sarcasm detection from Bengali text as there is no publicly available dataset and no comprehensive study that utilizes machine learning, deep learning, and transfer learning to detect sarcasm from Bengali text. Therefore, in this paper, we present a comprehensive approach that includes machine learning, deep learning, and transfer learning. Besides, we introduce a large-scale human-annotated dataset named 'Ben-Sarc' containing 25636 comments written in Bengali collected from Facebook.

4. 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.

4.1. Content source

As Facebook is one of the major sources of textual data (Salloum *et al.* 2017), 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 Table 2.

4.2. Content search

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

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Table 2. 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

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

4.3. Text cleaning and noise removal

Text preprocessing is generally a vital phase of NLP problems (Hemalatha, Varma, and Govardhan 2012). It converts text into a convenient format. The comment section of Facebook is very noisy and mostly contains errors, and useless information (Salloum *et al.* 2017). A list of preprocessing steps has been executed on the texts collected to enrich the Ben-Sarc dataset. They are removing non-Bengali words, duplicate texts, emojis, links, and URLs; replacing #hashtag, all symbols, special characters (e.g. '\n', '%', '$', '&', '@') with a single space, and multiple punctuations (e.g. '?', '|', ';', '!', ',') with single punctuation.

4.4. Annotation process

The contrast between statements meant to communicate a genuine or literal meaning *versus* those intended to convey an opposite or ironic meaning is at the basis of the sarcastic and non-sarcastic labels in the sarcasm detection research challenge. In other words, sarcastic labels are assigned to utterances that are meant to be regarded as the inverse of their literal or surface-level meaning, whereas non-sarcastic labels are applied to utterances that are meant to be interpreted as their literal or surface-level meaning. For this reason, each text in the Ben-Sarc dataset has been annotated manually by the authors 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. The decision to employ five annotators in the annotation process stems from a desire for the robustness of the dataset. We hope to improve the dataset's annotations by having numerous annotators independently analyze each text. Multiple annotators' diverse viewpoints and judgments help to have a more thorough and well-rounded knowledge of the sarcasm instances in the dataset. Facebook comments are frequently filled with harsh and filthy phrases, slang, and personal attacks (Hussain *et al.* 2018; Ahmed *et al.* 2021; Akhter *et al.* 2018). As a result, we made sure that all annotators were of adult age and had domain knowledge.

4.5. 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 having 3 to 5 years of experience that involves dataset annotation and validation. Each evaluator is an adult, a native Bengali speaker, proficient in Bengali, and active on social media having a habit of

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Table 3. 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

reading sarcastic comments. 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 for designing the questions for human evaluation of the Ben-Sarc dataset is from Hasan *et al.* (2021). The recent advancement in the quality estimation of neural language generation (NLG) models has inspired the creation of these characteristics. Belinkov and Bisk (2017) 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 (Cohen 1960) in Table 3. 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 and songs, and spelling mistakes also express sarcasm from the text whereas a low percentage for Q3 determines the opposite context 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.

4.6. 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 Figure 1.

Table 4 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 minimum is three in words. Thus, the

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Table 4. 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 #সাকিব!	খুব ভাল অনুপ্রেরণার এক সাকিব!	0
এই নয়া দামান গানের সঙ্গে নেচেছিলেন ঢাকা মেডিকেলের ডাক্তা-ররাও, কেন তারা ওই ভিডিও বানিয়েছিলেন জানতে পারবেন এই ভিডিওতে--> https://www.youtube.com/watch?v=geIKbjOBZtM	এই নয়া দামান গানের সঙ্গে নেচেছিলেন ঢাকা মেডিকেলের ডা-ক্তাররাও, কেন তারা ওই ভিডিও বানিয়েছিলেন জানতে পারবেন এই ভিডিওতে ।	0
বেসম্ভব সুন্দর ছবি□□□□	বেসম্ভব সুন্দর ছবি	1

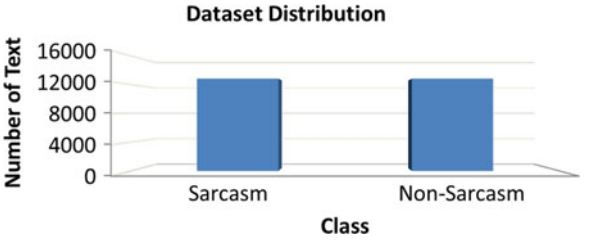

Figure 1. Data distribution of Ben-Sarc dataset according to labels.

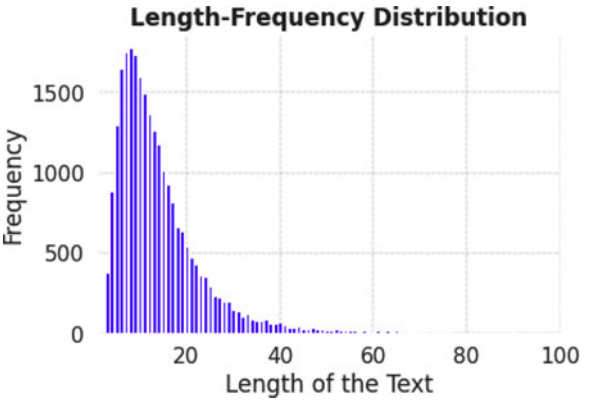

Figure 2. Length-frequency distribution of the Ben-Sarc dataset.

average length of a comment is fifteen. The length-frequency distribution of the whole dataset is shown in Figure 2. For better visualization, the length of the text is 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 5. The visualization of the statistics of the Ben-Sarc dataset has been shown in Figure 3.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Table 5. 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

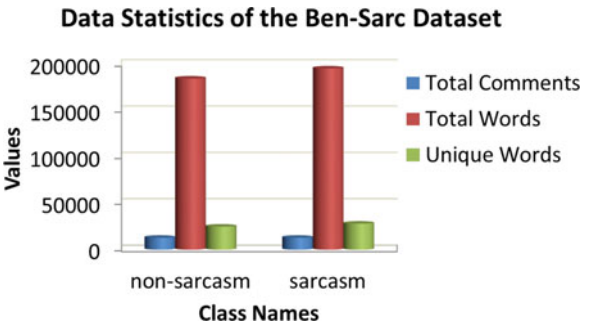

Figure 3. The visualization of the statistics of the Ben-Sarc dataset.

5. Proposed methodology

In this section, we provide a concise overview of our proposed methodology for detecting sarcasm. 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 NLP techniques like punctuation removal and tokenization. The third phase incorporates the feature selection process. This process includes Term-Frequency, Inverse Document Frequency (TF-IDF) 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.

5.1. Phase I—Dataset construction

We have collected 25636 Facebook comments written in Bengali. The overall dataset construction process is described in Section 4.

5.2. Phase II—Preprocessing

A few preprocessing steps have been executed before model training, which are punctuation removal (e.g. '!', '?') and tokenization.

Elongated words often contain some sentiment information. For example, "খুউউব মজাআআআর *(Veryyyy funnnnny)*" emphasizes more positive sentiment than "খুব মজার *(Very funny)*" (Tripto and Ali, 2018). So, we have not applied stemming and lemmatization to preserve the actual sense of the elongated words.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

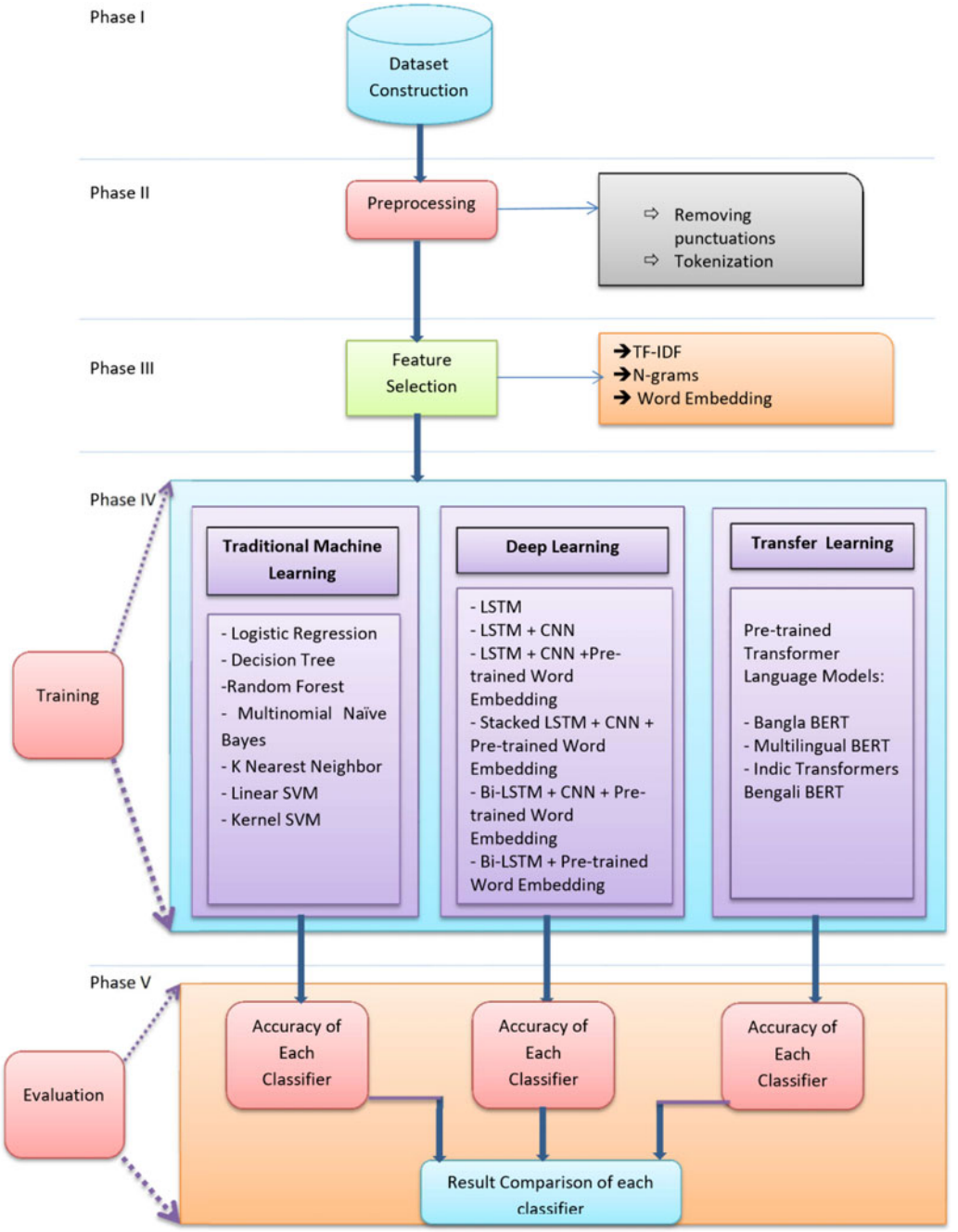

Figure 4. Workflow of the proposed approach.

5.3. 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 (Kumari, Jain, and Bhatia 2016). It is a

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

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.

$$(TFIDF)_{t,d} = tf_{t,d} \log \frac{N}{df_t} \tag{2}$$

where, $tf_{t,d}$ indicates frequency of term t in document d, df_t defines total number of documents containing term t, and 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 5.4.2. The selected features of transfer learning are pre-trained language models described in detail in Section 5.4.3. The use of n-grams, pre-trained word embeddings, and transformers in our task creates a connection with the field of grammar in NLP. N-grams capture sequential word patterns, which frequently reflect syntactic structures in language. These patterns can include grammatical links, assisting in the identification of sentences and context. Pre-trained word embeddings use massive language datasets to represent words in semantic space, capturing complex semantic and syntactic relationships. In doing so, they implicitly represent some grammatical connections based on co-occurrences. Transformers, with their attention mechanisms, succeed in capturing long-range relationships and complex contextual connections in text. This ability extends to capturing complicated sentence forms that represent grammatical syntax.

5.4. 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.

5.4.1. Traditional classifiers

We have initiated the sarcasm detection system by investigating traditional classifiers. We have used LR, DT, RF, MNB, KNN, Linear SVM, and Kernel SVM as traditional machine learning classifiers. Furthermore, we have applied all possible combinations of unigrams, 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 with the performance of the Ben-Sarc dataset with deep learning models in later.

5.4.2. 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. LSTM (Hochreiter and Schmidhuber 1997) 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.

5.4.2.1. 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.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

- **LSTM:** LSTMs interpret input sequences as pairs $(x_i, y_i) \ldots .(x_z, y_z)$. 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 (Graves, 2013) (Kalchbrenner, Danihelka, and Graves, 2015):

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

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

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

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

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

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

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

  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 (Kim 2014) 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 learned. It is a local feature extractor layer with well-trained kernels for weight modification utilizing the back-propagation approach (Rumelhart, Hinton, and Williams 1986). 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.

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

    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 (Boureau, Ponce, and LeCun 2010), 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.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

- **Embedding layer:** An embedding layer is learned alongside a neural network model on a particular NLP 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 are 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 $|V|$ indicates the vocabulary size.
- **Pre-trained word embeddings:** Pre-trained word embeddings are embeddings that are learned 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** (Pennington, Socher, and Manning 2014) 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.[c]
  – **Word2Vec** (Mikolov *et al.* 2013) 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.[c]
  – **BPEmb** (Heinzerling and Strube 2017) model based on Byte-Pair encoding, which gives a collection of pre-trained subword embedding models for 275 languages including Bengali.[d]
  – **FastText** (Grave *et al.* 2018) is a prediction-based embedding approach that conveys subword information. We have used FastText created for 157 languages including Bengali.[e]

5.4.2.2. 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 activation function. The number of nodes in the dense layer is two because of the binary classification task. The vocabulary size is 10000. The architecture of this model is shown in Figure 5.
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 5.4.2.2(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 a 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 5.4.2.2(b). The weights of the embedding layer are initialized with the weights of pre-trained word embedding. A dropout layer and then a dense layer are added after the embedding layer. After that, a 1D convolutional layer and a

[c] https://bnlp.readthedocs.io/en/latest/
[d] https://bpemb.h-its.org/
[e] https://fasttext.cc/docs/en/crawl-vectors.html

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

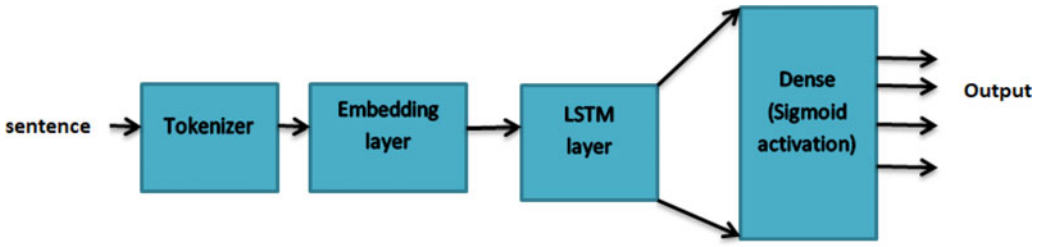

Figure 5. Long short-term memory model without pre-trained word embeddings.

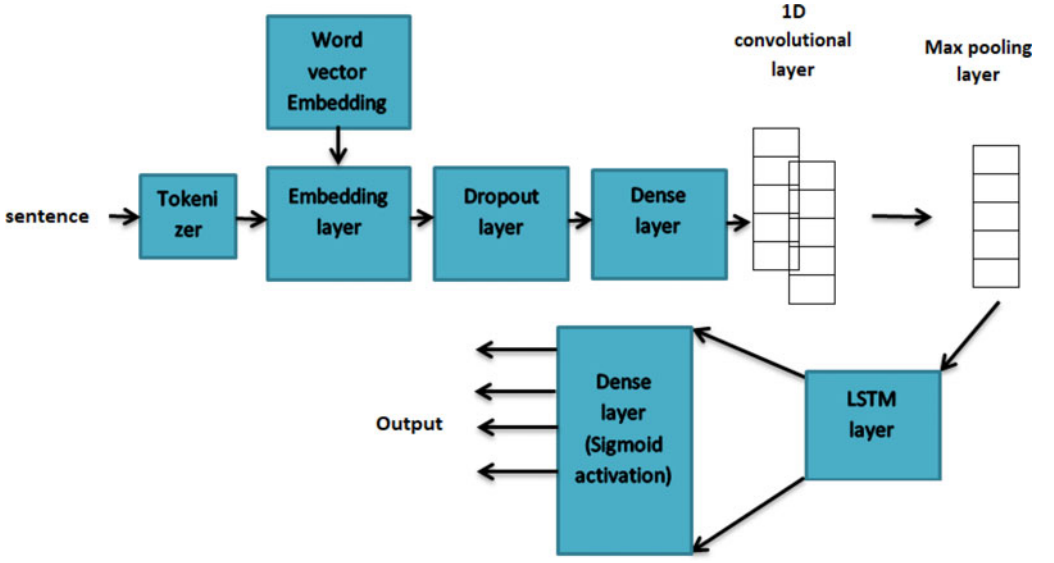

Figure 6. Long short-term memory + CNN + Pre-trained word embeddings model.

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 the sigmoid activation function as mentioned in Subsection 5.4.2.2(b). The architecture of this model is shown in Figure 6.

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 5.4.2.2(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 a sigmoid activation function and the number of output nodes is the number of labels in the dataset. The architecture of this model is shown in Figure 7.

e. **BiLSTM + CNN + Pre-trained word embedding:** A bidirectional LSTM (Schuster and Paliwal 1997), 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 5.4.2.2(c). Only the LSTM layer is replaced with the BiLSTM layer.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

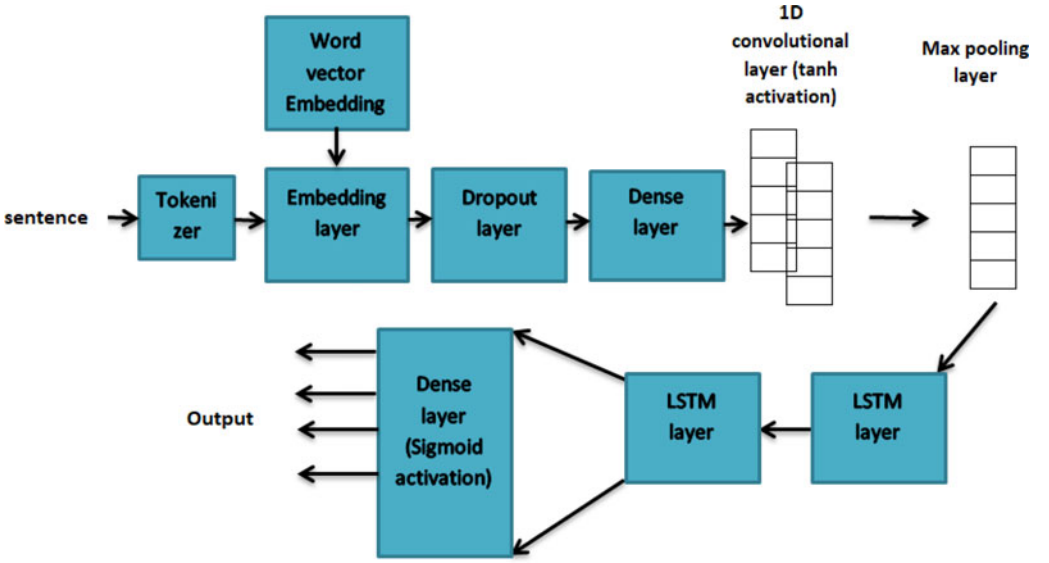

Figure 7. Stacked long short-term memory + CNN model + Pre-trained word embeddings model.

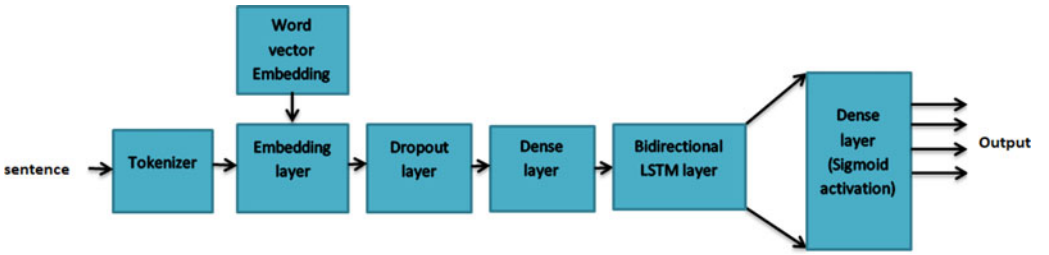

Figure 8. BiLSTM + Pre-trained word embedding model.

f. **BiLSTM + Pre-trained word embedding:** The architecture of the model remains the same as the model mentioned in Subsection 5.4.2.2(e) by dropping the CNN portion of the model. The architecture of this model is shown in Figure 8.

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 later to save computational costs.

5.4.3. 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 (Torrey and Shavlik 2010). Transfer learning approaches have been effectively used for speech recognition, document categorization, and sentiment analysis in NLP (Wang and Zheng 2015). Figure 9 represents an illustration of the transfer learning approach.

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 have 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

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

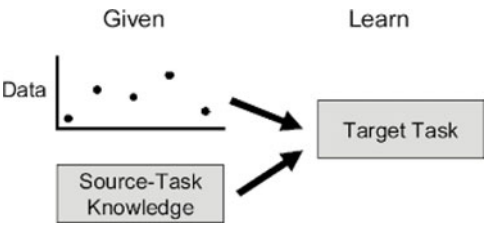

Figure 9. Illustration of transfer learning (Torrey and Shavlik 2010)

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.

5.4.3.1. BERT.

BERT (Devlin *et al.* 2019) is one of the most prevalent transformer-based models that is used for pre-training a transformer (Vaswani *et al.* 2017). 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, 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[f] as these are mostly used in downstream works like text classification. The transformer language models are:

- **Bangla BERT(base)** (Sarker 2020), a pre-trained Bengali language model based on mask language modeling that has been pre-trained on Bengali Wikipedia Dump dataset[g] and a large Bengali corpus taken from Open Super-large Crawled Aggregated coRpus (OSCAR)[h]. The model follows the BERT-base-uncased model architecture, which means it has 12 layers, 768 hidden layers, 12 heads, and 110 M parameters.
- **Indic-Transformers Bengali BERT** (Jain *et al.* 2020), a BERT language model that has been pre-trained on about 3 GB of monolingual training corpus, majorly taken from OSCAR. It has achieved state-of-the-art performance in the Bengali language for the text classification tasks.
- **M-BERT** (Devlin *et al.* 2019), 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 110 M parameters.

[f] https://huggingface.co/
[g] https://dumps.wikimedia.org/bnwiki/latest/
[h] https://oscar-corpus.com/

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

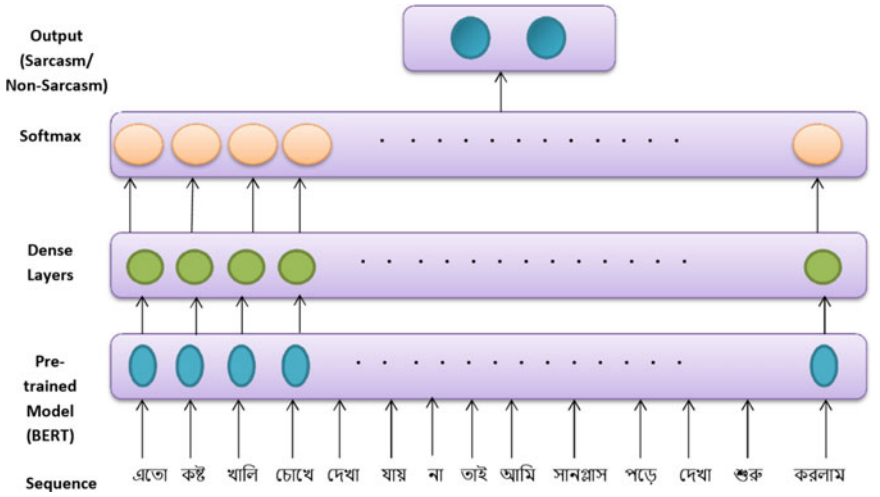

Figure 10. Model architecture of sarcasm detection for transfer learning.

5.4.3.2. Architecture of our model.
The detailed architecture of our model is shown in Figure 10. 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 Section 5.4.3.1 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 by freezing BERT's weight. This process prevents the updating of model weights during fine-tuning.
c. **Attaching our own neural network architecture:** A different number of dense layers with different activations mentioned in Section 6.4.3 and softmax as an output layer of our own is 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 6.4.3.

5.5. Phase V—Evaluation

In the last phase, we 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.

6. Experimental evaluation

6.1. 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

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Table 6. Performance(in %) of 5-fold and 10-fold cross-validation for experiment I

	5-Fold					10-Fold				
Classifier	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

testing. Experimental evaluation was conducted on a machine with an Intel Core i5 processor with 2.71 GHz clock speed and 4 GB 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.

6.2. Experiments

Our experiments are categorized into three parts. Experiment I is concerned with the experiments on traditional classifiers. Experiment II focuses on the experiments on deep learning classifiers and experiment III reports 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.

6.2.1. Experiment I

In this experiment, the performance of traditional machine learning classifiers mentioned in Section 5.4.1 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 experiment I has been demonstrated in Table 6. The process of choosing hyperparameters for experiment I has been discussed in Section 6.4.1. From Table 6, 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 among all traditional classifiers as it works well with high dimensional text data by taking advantage of probabilistic algorithm. It is 72.01% for 5-fold cross-validation and 72.36% for 10-fold cross-validation.

6.2.2. Experiment II

In this experiment, the performance of different deep learning classifiers mentioned in Section 5.4.2 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 Table 7.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

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

Table 8. 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

For LSTM models, LSTM units have been taken as 100, and for stacked LSTM, LSTM units have been taken as 128 and 64. Hundred 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 two epochs with no improvement in 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 8. The procedure for picking hyperparameters for experiment II is covered in Section 6.4.2.

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 decreased more after using pre-trained word embedding. After that, the performance of other models 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 the text is written in dialect, manipulating phrases, sentences, and spelling mistakes, which determines the text as sarcastic as mentioned in Section 4.5.

6.2.3. Experiment III

In this experiment, the performance of transfer learning techniques mentioned in Section 5.4.3 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 Table 9. The hyperparameter setting

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

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

Table 10. 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

is shown in Table 10. The approach for optimizing hyperparameters for experiment III is outlined in Section 6.4.3.

Here, for all cases, negative log-likelihood (Platt 1999) loss has been used as a loss function as it is the classic loss function used in any classification task (Ruan *et al.* 2020). 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 have obtained the highest accuracy among all pre-trained models. It has achieved 75.05% 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 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 experimented with another pre-trained model Bangla BERT, but the performance degraded significantly.

6.3. Result analysis

The highest accuracy from each experiment has been shown briefly in Table 11. From Table 11, 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 NB from traditional classifiers achieved a maximum 72.48% and 72.36% accuracy, respectively.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Table 11. 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**

Table 12. 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

6.4. 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.

6.4.1. For experiment I

For experiment I, we have applied 5-Fold and 10-Fold cross-validation on seven traditional classifiers listed in Table 6. Unigram, bigram, and trigram techniques have been applied for each classifier. Among them, the best results from each classifier have been demonstrated in Table 6 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.

6.4.2. For experiment II

For experiment II, all hyperparameters that have been tuned for several combinations across all models are mentioned in Table 12. We have experimented with all possible combinations of pre-trained word embedding, dense layer size, batch size, activation in the hidden layer, optimizer, and learning rate mentioned in Table 12 for all deep learning models for hyperparameter tuning.

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, and Nadam optimizer with 0.0001 learning rate. This is the second-highest accuracy for experiment II. Other models with different combinations cannot obtain better results.

6.4.3. For experiment III

For experiment III, all hyperparameters that have been tuned for several combinations for all models are mentioned in Table 13. We have experimented with all possible combinations of the pre-trained source model, batch size, number of hidden layers, number of nodes in hidden layers, activation, dropout, optimizer, and learning rate mentioned in Table 13 for all transfer learning models for hyperparameter tuning.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Table 13. 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

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, and 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, and batch size 4 with 30 epochs.

6.5. Error analysis

In this subsection, we discuss the error analysis of the three best-performing models from experiments I, II, and III—MNB, LSTM, and Indic-Transformer Bengali-BERT-based Transfer learning as we have shown in Table 11 for four types of sample input to demonstrate that which type of input deep learning can predict but traditional machine learning cannot. Table 14 shows the sample input-output with the predicted score of each best-performing model.

For SI 1, all models predict the text as sarcastic as all predicted scores are greater than 0.5 as it is a binary classification problem. The input SI 1 is clearly a funny text and there is no confusion about this text. In this text, the rhythm of the poem is used to express sarcasm which is very common in Bengali sarcasm. Similarly, input SI 2 is predicted as sarcasm with a very low score. Therefore, it is correctly classified as non-sarcasm by all models as the input text is clearly a non-sarcastic text.

For input SI 3, the MNB model misclassifies the text as non-sarcasm whereas the LSTM model classifies it correctly with the margin score. On the contrary, the transfer learning model with Indic BERT classifies it correctly. MNB model fails here as the traditional classifiers are unable to capture the sequential information present in the text. As LSTM is a sequential model, it classifies the text correctly though the predicted score is not so good. On the contrary, the Indic-Transformers Bengali-BERT-based transfer learning model successfully classifies it as it can use the knowledge and advantage of the transformer model.

For input SI 4, all models fail to classify it correctly as a sarcastic text. The text seems non-sarcastic but if the text contains one word বিষণ which refers to *poisonous* instead of ভীষণ which refers to *very*. By intentionally creating mistakes in spelling, this text indicates a sarcastic text. Therefore, MNB and LSTM fail to predict it correctly. Though the transfer learning model misclassifies it, it obtains almost the threshold of binary classification as a prediction score.

6.6. Statistical analysis

To determine whether the variations among the predictions produced by the traditional machine learning models, deep learning models, and transfer learning models are statistically significant or not, we performed a statistical test. We utilized Friedman test (Friedman 1937) to inspect

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Table 14. Error analysis for different input and output for the best performing model of experiment I, II, III.

	Sample Output (Predicted Label)				
Sample Input (SI)	MNB	LSTM	Indic-Transformers Bengali-BERT-based Transfer Learning	Actual Label	Remarks
তোমারে ভাবিয়া সারারাত জাগিয়া ঘুম মোর হয়েছে নষ্ট বুকের বামপাশে চিনচিন ব্যাথা করে SI 1: একি গ্যাস্ট্রিক না প্রেম হচ্ছে না স্পষ্ট । *Translation: I stayed up all night thinking about you I lost my sleep Pain in the left side of the chest A gastric or love is not clear*	Sarcasm (0.66)	Sarcasm (0.83)	Sarcasm (0.88)	Sarcasm	Yes (for All)
SI 2: আপিন খুব তাড়াতাড়ি সুস্থ হয়ে যান এইকামনাকির। Translation: Wish you a speedy recovery	Sarcasm (0.22)	Sarcasm (0.17)	Sarcasm (0.15)	Non-Sarcasm	Yes (for All)
SI 3: আপনার মন পাথেরর মত কোমল *Translation: Your heart is as soft as stone*	Sarcasm (0.35)	Sarcasm (0.51)	Sarcasm (0.61)	Sarcasm	No (for ML)
SI 4: চ্যাম্প ২২ গজে আপনাকে বিষণ মিস করছি *Translation: Champ 22 yards miss you*	Sarcasm (0.19)	Sarcasm (0.27)	Sarcasm (0.49)	Sarcasm	No (for All)

whether the observed differences in performances across different models were statistically significant or just due to chance. The obtained p-values for the evaluated paradigms–machine learning (0.42319008112684364), deep learning (0.4158801869955079), and transfer learning (0.36787944117144245)–in the context of the Friedman test indicate interesting observations into the potential differences between these methodologies. For example, all three p-values are more than the traditional threshold of 0.05, indicating a lack of solid evidence for significant variations in performance between approaches within the provided dataset and experimental setting. The findings indicate that, while there might be differences in performance outcomes, they are not significant enough to approach statistical significance at this threshold.

7. Conclusion and future work

In this paper, we have presented a benchmark dataset for Bengali sarcastic comments on Facebook to influence one of the low-resource languages named Bengali. Then, we 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 to 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 by adding various social media data like YouTube, Twitter, and comments from different newspapers or product websites. Besides, emojis and emoticons play a vital role in articulating the actual connotation of a comment on social media. So, we will consider emojis and emoticons along with the text.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

References

Ahmed M. F., **Mahmud Z.**, **Biash Z. T.**, **Ryen A. A. N.**, **Hossain A. and Ashraf F. B.** (2021). Bangla text dataset and exploratory analysis for online harassment detection. *CoRR*, abs/2102.02478.

Akhter S. and et al. (2018). Social media bullying detection using machine learning on Bangla text. In *2018 10th International Conference on Electrical and Computer Engineering (ICECE)*, IEEE, pp. 385–388.

Awal M., **Rahman M. S. and Rabbi J.** (2018). Detecting abusive comments in discussion threads using Naïve Bayes. In *International Conference on Innovations in Science, Engineering and Technology (ICISET)*, pp. 163–167.

Bamman D. and Smith N. (2021). Contextualized sarcasm detection on Twitter. In *Proceedings of the International AAAI Conference on Web and Social Media*, vol.**9**(1), pp. 574–577.

Banik N. and Rahman M. H. H. (2019). Toxicity detection on Bengali social media comments using supervised models. In *2019 2nd International Conference on Innovation in Engineering and Technology (ICIET)*, pp. 1–5.

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

Belinkov Y. and Bisk Y. (2017). *Synthetic and Natural Noise Both Break Neural Machine Translation*. CoRR, abs/1711.02173.

Bharti S. K., **Babu K. S. and Raman R.** (2017). Context-based sarcasm detection in Hindi tweets. In *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)*, pp. 1–6.

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

Chakraborty P. and Seddiqui M. (2019). Threat and abusive language detection on social media in Bengali language. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pp. 1–6.

Cohen J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* **20**(1), 37–46.

Das D. and Clark A. J. (2018). 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, pp. 1–5.

Devlin J., **Chang M.-W.**, **Lee K. and Toutanova K.** (2019). BERT: Pre-training of deep bidirectional transformers for language understanding (Long and Short Papers). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota: Association for Computational Linguistics, vol **1**, pp. 4171–4186, (Long and Short Papers)

Eke C., **Norman A.**, **Shuib L. and Nweke H.** (2020). Sarcasm identification in textual data: Systematic review, research challenges and open directions. *Artificial Intelligence Review* **53**(6), 4215–4258.

Emon E. A., **Rahman S.**, **Banarjee J.**, **Das A. K. and Mittra T.** (2019). A deep learning approach to detect abusive Bengali text. In *2019 7th International Conference on Smart Computing Communications (ICSCC)*, pp. 1–5.

Friedman M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american Statistical Association* **32**(200), 675–701.

Ghosh D., **Fabbri A. R. and Muresan S.** (2018). Sarcasm analysis using conversation context. *Computational Linguistics* **44**(4), 755–792.

Grave E., **Bojanowski P.**, **Gupta P.**, **Joulin A. and Mikolov T.** (2018). Learning word vectors for 157 languages, arXiv preprint arXiv:1802.06893.

Graves A. (2013). Generating sequences with recurrent neural networks, arXiv preprint arXiv:1308.0850.

Hasan T., **Bhattacharjee A.**, **Islam M. S.**, **Mubasshir K.**, **Li Y.-F.**, **Kang Y.-B.**, **Rahman M. S. and Shahriyar R.** (2021). XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational*. Linguistics: ACL-IJCNLP, Online. Association for Computational Linguistics, pp. 4693–4703.

Heinzerling B. and Strube M. (2017). Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages, arXiv preprint arXiv:1710.02187.

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

Hiai S. and Shimada K. (2018). *Sarcasm Detection Using Features Based on Indicator and Roles*. Cham: Springer International Publishing, pp. 418–428.

Hochreiter S. and Schmidhuber J. (1997). Long short-term memory. *Neural Computation* **9**(8), 1735–1780.

Hossain E., **Sharif O. and Hoque M. M.** (2022a). Mute: A multimodal dataset for detecting hateful memes. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: Student Research Workshop*, pp. 32–39.

Hossain E., **Sharif O.**, **Hoque M. M.**, **Dewan M. A. A.**, **Siddique N. and Hossain M. A.** (2022b). Identification of multilingual offense and troll from social media memes using weighted ensemble of multimodal features. *Journal of King Saud University-Computer and Information Sciences* **34**(9), 6605–6623.

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

Hussain M. G., **Mahmud T. A. and Akthar W.** (2018). An approach to detect abusive Bangla text. In *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, pp. 1–5.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

Ishmam A. M. and Sharmin S. (2019). Hateful speech detection in public facebook pages for the Bengali language. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 555–560.

Jain K., **Deshpande A.**, **Shridhar K.**, **Laumann F. and Dash A.** (2020). Indic-transformers: An analysis of transformer language models for Indian languages.

Joshi A., **Tripathi V.**, **Patel K.**, **Bhattacharyya P. and Carman M.** (2016). Are word embedding-based features useful for sarcasm detection? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, pp. 1006–1011.

Kalchbrenner N., **Danihelka I. and Graves A.** (2015). Grid long short-term memory, arXiv preprint arXiv:1507.01526.

Khatri A. and P. P. (2020). Sarcasm detection in tweets with BERT and GloVe embeddings. In *Proceedings of the Second Workshop on Figurative Language Processing*, Online. Association for Computational Linguistics, pp. 56–60.

Kim Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, pp. 1746–1751.

Kumari M., **Jain A. and Bhatia A.** (2016). Synonyms based term weighting scheme: An extension to TF. IDF. *Procedia Computer Science* **89**, 555–561.

Lemmens J., **Burtenshaw B.**, **Lotfi E.**, **Markov I. and Daelemans W.** (2020). Sarcasm detection using an ensemble approach. In *Proceedings of the Second Workshop on Figurative Language Processing*, Online. Association for Computational Linguistics, pp. 264–269.

Lora S. K., **Jahan N.**, **Antora S. A. and Sakib N.** (2020). Detecting emotion of users' analyzing social media Bengali comments using deep learning techniques. In *2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, IEEE, pp. 88–93.

Lunando E. and Purwarianti A. (2013). Indonesian social media sentiment analysis with sarcasm detection. In *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 195–198.

Mikolov T., **Sutskever I.**, **Chen K.**, **Corrado G. S. and Dean J.** (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pp. 3111–3119.

Pawar N. and Bhingarkar S. (2020). Machine learning based sarcasm detection on Twitter data. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 957–961.

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

Platt J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*. MIT Press, pp. 61–74.

Ptáček T., **Habernal I. and Hong J.** (2014). Sarcasm detection on Czech and English Twitter. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical papers*, pp. 213–223.

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

Romim N., **Ahmed M.**, **Talukder H. and Saiful Islam M.** (2021). Hate speech detection in the Bengali language: A dataset and its baseline evaluation. In *Proceedings of International Joint Conference on Advances in Computational Intelligence*, Singapore: Springer Singapore, pp. 457–468.

Ruan W., **Nechaev Y.**, **Chen L.**, **Su C. and Kiss I.** (2020). Towards an asr error robust spoken language understanding system. In *INTERSPEECH*, pp. 901–905.

Rumelhart D. E., **Hinton G. E. and Williams R. J.** (1986). Learning representations by back-propagating errors. *Nature* **323**(6088), 533–536.

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

Sarker S. (2020). Banglabert: Bengali mask language model for Bengali language understading.

Schuster M. and Paliwal K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* **45**(11), 2673–2681.

Sentamilselvan K., **Suresh P.**, **Kamalam G. K.**, **Mahendran S. and Aneri D.** (2021). Detection on sarcasm using machine learning classifiers and rule based approach. *IOP Conference Series: Materials Science and Engineering* **1055**(1), 012105.

Sharma A. S., **Mridul M. A. and Islam M. S.** (2019). Automatic detection of satire in Bangla documents: A cnn approach based on hybrid feature extraction model. In *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–5.

Torrey L. and Shavlik J. (2010). Transfer learning. In *Handbook of Research On Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI global, pp. 242–264.

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

Vaswani A., **Shazeer N.**, **Parmar N.**, **Uszkoreit J.**, **Jones L.**, **Gomez A. N.**, **Kaiser Ł. and Polosukhin I.** (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008.

https://doi.org/10.1017/nlp.2024.11 Published online by Cambridge University Press

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

Wang Z., **Wu Z.**, **Wang R. and Ren Y.** (2015). Twitter sarcasm detection exploiting a context-based model. In *Proceedings, Part I, of the 16th International Conference on Web Information Systems Engineering — WISE. 2015*, Berlin, Heidelberg: Springer-Verlag, vol **9418**, pp. 77–91.

Cite this article: Lora S K, Shahariar G. M., Nazmin T, Rahman N N, Rahman R, Bhuiyan M and Shah F M. Ben-Sarc: A self-annotated corpus for sarcasm detection from Bengali social media comments and its baseline evaluation. *Natural Language Processing* https://doi.org/10.1017/nlp.2024.11