Multilingual Hate Speech and Offensive Language Detection of Low Resource Languages

Abhinav Reddy Gutha[1], Nidamanuri Sai Adarsh[1], Ananya Alekar[1] and Dinesh Reddy[1]

[1]*Indian Institute of Technology, Goa - 403401*

Abstract

The last decade has seen a steep rise in the use and dependence of society on social media. The need for detection and prevention of hate and offensive speech is more than ever. The everchanging form of natural language makes the detection of hate speech challenging, involving code-mixed text. The task becomes even more daunting in a country like India, where different languages and dialects are spoken across the country. This paper details the Code Fellas team's approaches in the context of HASOC 2023 - Task 4: Annihilate Hate, an initiative aimed at extending hate speech detection to Bengali, Bodo, and Assamese languages. Here we describe our approaches which broadly involve Long Short Term Memory (LSTM) coupled with Convolutional Neural Networks (CNN) and pre-trained Bidirectional Encoder Representations from Transformers (BERT) based models like IndicBERT [1] and MuRIL [2]. Notably, our results showcase the effectiveness of these approaches, with IndicBERT achieving a remarkable F1 score of 69.726% for Assamese, MuRIL achieving 71.955% for Bengali, and a BiLSTM model enhanced with an additional Dense Layer attaining an impressive 83.513% for Bodo.

Keywords

Hate Speech, Offensive Language, LSTM, Convolutional Neural Networks, Transformers

1. Introduction

Social media allows users to express their opinions without disclosing their real identities. This leads to the misuse of social media platforms to generate hate among individuals and communities, often leading to hate crimes. In the past few years, platforms like Twitter, Facebook, and Reddit have seen a rising trend in the dissemination of offensive content and the coordination of hate-related actions. The hate affects not only the users but the general public, increasing the cases of depression, anxiety, and other mental health issues. Hence, effective hate speech detection is required.

Until a few years ago, hate and offensive speech were identified manually, which is now an impossible task due to the enormous amounts of data being generated daily on social media platforms. Detection of hate speech becomes a challenging task since filtering out certain words that express hate is not sufficient. The task also requires one to know the context and the background of the user. In addition, in a diverse country like India, where numerous languages are spoken, individuals often use their local languages when engaging through social media.

Forum for Information Retrieval Evaluation, December 15-18, 2023, India

abhinav.reddy.21031@iitgoa.ac.in (A. R. Gutha); nidamanuri.adarsh.21031@iitgoa.ac.in (N. S. Adarsh); ananya.alekar.21031@iitgoa.ac.in (A. Alekar); dinesh.reddy.21063@iitgoa.ac.in (D. Reddy)

CEUR Workshop Proceedings (CEUR-WS.org)

This becomes a major hurdle in hateful speech detection since two texts that have the same meaning literally can mean different things in their respective languages.

In this paper, we propose ways to address the above-mentioned issues and approaches like Machine Learning Algorithms, LSTM, and BiLSTM coupled with CNN and pre-trained BERT-based models to identify Hate Speech in Bengali, Bodo, and Assamese languages.

The rest of the paper is organized as follows. Section 3 describes the dataset, followed by the description of our proposed model in Section 4. Section 5 elaborates on the experimental setup used. In Section 6 we list out the results obtained by our model in the evaluations and finally conclude the paper.

2. Related works

The four major tasks in HASOC 2023 are Task-1[3], Task-2[4][5], Task-3[6], Task-4[7][8]. Task-1[8] focuses on Identifying Hate, offensive and profane content in Indo-Aryan languages with subtask Task-1A which focuses on Sinhala and subtask Task-2A focuses on Gujarati. Task-2 focuses on Identification of Conversational Hate-Speech in Code-Mixed Languages, this task focuses on the binary classification of such conversational tweets with tree-structured data. Task-3 focuses on Identification of Tokens Contributing to Explicit Hate in Text by Hate Span Detection. Task-4 of Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages aims to tackle the detection of hate speech within the Bengali, Bodo, and Assamese languages. The dataset used in this task is primarily derived from social media platforms, including Twitter, Facebook, and YouTube. It comprises lists of sentences, each annotated with a label indicating the presence or absence of offensive content. This task entails binary classification, with the core objective of predicting whether a given sentence contains offensive language.

Researchers have used various techniques for the classification of text for hate speech detection. K.Ghosh, A.Senapati et al.[9] [10] [11] used baseline bert models for conversational hate speech detection in code-mixed tweets utilizing data augmentation and offensive language identification, compared mono and multilingual transformer model with cross-language evaluation and achieved transformer-based hate speech detection in assamese.

A primary challenge within this research domain revolves around the scarcity of available data, particularly in languages like Assamese, Bodo, and Bengali. The limited data availability has hindered comprehensive investigations into hate speech classification within these languages. In light of this, our work endeavors to bridge this critical gap by developing a model capable of effectively addressing hate speech detection in low-resource language contexts. Our approach holds relevance not only for the specific languages discussed but also as a valuable blueprint for addressing similar challenges in other low-resource languages.

[0]Github: https://github.com/16AbhinavReddy/Multilingual-Hate-Speech-and-Offensive-Language-Detection-of-Low-Resource-Languages

3. Task and Dataset Information

In this paper, we have used the multilingual datasets provided by Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC). The shared tasks of HASOC were provided for three languages (Assamese, Bengali, and Bodo) as part of task 4. There are training and test datasets provided for the three languages. Task 4 is a binary classification that needs respondents to classify the given tweets into two groups: Hate and Offensive (HOF) and Not Hate-Offensive (NOT).

1. HOF: This post includes hateful, offensive, or profane content.
2. NOT: This post contains neither Hate Speech nor offensive content.

Table 1 provides a detailed description of the training datasets used in this work.

Table 1
Details of Training Datasets

Language	HOF	NOT	Total
Assamese	2347	1689	4036
Bengali	766	515	1281
Bodo	998	681	1679

4. Preprocessing

Data processing is a crucial step in Natural Language Processing (NLP) tasks to clean and make the data suitable for analysis. However, it is recommended that minimal processing is applied, especially while dealing with BERT/LSTM-based approaches. The application of excessive preprocessing can strip away some of the information these models use to make accurate predictions. BERT and LSTM are both powerful language models that can learn complex relationships between words and phrases. However, our approach follows the steps [12] mentioned below to remove irrelevant text from the collected corpus and training datasets. The steps include:

1. Removing all the usernames using regular expressions.
2. Removing all the URLs and numerics using regular expressions.
3. Reducing elongated words that express screaming to their normal form. For example, **hellooooooooo** is reduced to **hello**.
4. Removing all the newline characters from the text.
5. Converting the text to tensors by applying tokenization to the tweets and padding the sequences accordingly. We have used the tokenizers that are specifically designed for BERT or LSTM models.

We experimented with various other preprocessing methods like replacing English emoticons and emojis[1] with their actual meaning using the emoji library, etc. But approaches like the

[1] https://pypi.org/project/emoji/

BERT and LSTM model are very powerful and can learn Unicode characters like emoticons, emojis, etc. This is because BERT models are already trained on a massive dataset of text and code, and they have learned to capture many of the important features of languages. Hence, applying excessive processing to the input text can introduce noise and can lead to the loss of hateful text like emojis, which although do not hold much meaning in normal embeddings, but can express hate in many contexts. Hence, removing important information like emojis can adversely affect the model's performance.

5. Methodology

As mentioned in the Data Preprocessing section, we first developed a pipeline to preprocess the data for model training.

5.1. Translation

The task involves detecting hate speech in various languages; hence, the most natural way to proceed is by converting the text from different languages into a single language, preferably English [13]. This would have been a good approach if the language translation would be able to capture the meaning of the text in its most appropriate context.

We used Googletrans[2] library to translate the given text into English. Further to preprocess the text, we implemented the preprocessor pipeline and eventually we were able to proceed with classification. However, this methodology has certain limitations associated with it. Primarily, the translation loses the context, the original meaning, and the intentions of some offensive text, due to which the model is unable to perform well in classification. Generally, translation libraries are not available for some low-resource languages like Bodo, hence, we can conclude that this is not the optimal method.

While experimenting on offensive tweets in languages like Assamese and Bengali, we found that the offensive connotation is lost for some tweets as the translator cannot detect that information. Our observation is that the translation carries the errors forward into classification. Hence, we needed to identify an optimal method to complete the task without translation, while performing the task separately in all the languages.

5.2. Machine Learning Techniques

As the data from the given languages is sparse, the Term Frequency Inverse Document Frequency (TF-IDF) and CountVectorizer method is the best way to generate vector formats for the text data given.

5.2.1. Embedding Technique

While generating vector formats using TF-IDF and CountVectorizer for preprocessing, we applied basic preprocessing steps and removed URLs and usernames. As emojis indicate aggression in the context, we have retained them to capture the intended sentiment.

[2] https://pypi.org/project/googletrans/

After preprocessing, we applied TF-IDF and CountVectorizer techniques, considering N-grams with N-gram values 2 and 3, thus obtaining semantic relationships between two or three sequential words to some extent.

5.2.2. Model Application

After generating the vector embeddings for the text, we applied the following models for classifying hateful and offensive speech:

- Support Vector Machine
- Logistic Regression
- XGB classifier
- Decision Tree Classifier

Apart from this, we also tried using a Latent Semantic-based approach[3], where we took the number of components between 95 and 500. We used this as a variable when applying the RandomizedSearchCV method, which is further explained below.

The Latent Semantic Approach is a technique used in NLP and information retrieval to understand the underlying meaning of words and documents. It relies on statistical analysis to identify patterns and relationships between words based on their co-occurrence in a large text corpus. By doing so, it can capture the semantic similarity between words and documents, even if they don't share exact terms.

During Model Application, hyperparameter tuning is employed to reduce computation and calculate the best parameters for classification. In this process, RandomizedSearchCV is used over GridSearchCV to get better results with comparatively less computation.

Table 2, Table 3 and Table 4 tabulate the performance of the top eight models in the three languages.

Table 2
Results Obtained from Machine Learning Approaches for Assamese

Model	F1-Score
Logistic Regression + CountVectorizer + ngrams=(1,2)	**0.6308**
Logistic Regression + CountVectorizer + ngrams=(1,3)	0.6267
XGB Classifier + CountVectorizer + ngrams=(1,3)	0.5848
Decision Trees + CountVectorizer + ngrams=(1,3)	0.5610
Decision Tree + CountVectorizer + ngrams=(1,2)	0.5941
XGB Classifier + TfidfVectorizer + ngrams=(1,2)	0.5933
XGB Classifier + TfidfVectorizer + ngrams=(1,3)	0.5908
Decision Tree + TfidfVectorizer + ngrams=(1,2)	0.5857

5.3. Deep Learning Architectures

After employing machine learning models, our research turns its attention toward deep learning models, specifically LSTM and Bidirectional Long Short-term memory (BiLSTM).

[3]https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

Table 3
Results Obtained from Machine Learning Approaches for Bengali

Model	F1-Score
Logistic Regression + CountVectorizer + ngrams=(1,2)	**0.6243**
Logistic Regression + TfidfVectorizer + ngrams=(1,2)	0.6043
Decision Tree + TfidfVectorizer + ngrams=(1,2)	0.5801
Logistic Regression + CountVectorizer + ngrams=(1,3)	0.5572
XGB Classifier + TfidfVectorizer + ngrams=(1,3)	0.5512
XGB Classifier + CountVectorizer + ngrams=(1,2)	0.5659
Logistic Regression + TfidfVectorizer + ngrams=(1,3)	0.5480
SVM + TfidfVectorizer + ngrams=(1,2)	0.5147
SVM + CountVectorizer + ngrams=(1,2)	0.5004

Table 4
Results Obtained from Machine Learning Approaches for Bodo

Model	F1-Score
SVM + CountVectorizer + ngrams=(1,2)	**0.7018**
XGB Classifier + CountVectorizer + ngrams=(1,2)	0.6902
XGB Classifier + CountVectorizer + ngrams=(1,3)	0.6901
Logistic Regression + CountVectorizer + ngrams=(1,2)	0.6901
Logistic Regression + CountVectorizer + ngrams=(1,3)	0.6469
SVM + CountVectorizer + ngrams=(1,3)	0.63
SVM + TfidfVectorizer + ngrams=(1,2)	0.61
Decision Tree + TfidfVectorizer + ngrams=(1,2)	0.61

5.3.1. LSTM

LSTM stands out as a specialized variant of Recurrent Neural Networks (RNNs), meticulously designed to apprehend and model extensive temporal dependencies within sequential data. It harnesses memory cells and gating mechanisms to selectively retain and retrieve information over protracted sequences. The rationale behind our exploration of LSTM stems from its manifold advantages, encompassing sequential data processing, the presence of memory cells that facilitate the storage and retrieval of information across extended sequences, and the incorporation of gating mechanisms.

However, we encountered a notable challenge with LSTM. It exhibits a tendency to demand a substantial volume of training data to achieve optimal performance. The given dataset does not possess the requisite scale. To address this limitation, we introduced a 1D CNN layer (CNN 1-D) to enhance the performance, since combining LSTM with 1D CNN amalgamates the strengths of both architectures.

Table 5 shows the performance of LSTM in all three languages.

5.3.2. LSTM with CNN 1-D

LSTM with CNN 1-D [14] is particularly advantageous when the input data exhibits local spatial patterns, which pertains to features or patterns within a sequence not contingent upon the order

of elements but related to their relative positions or proximity within the sequence. Additionally, it excels in capturing intricate temporal relationships, signifying how elements within a sequence relate to each other over time facilitating the modeling of long-range dependencies and temporal patterns.

However, in contrast to our initial expectations this configuration underperformed as compared to its performance in the preceding scenario. Overfitting turned out to be a prominent concern, primarily due to the heightened complexity of the model structure, which ultimately led to diminished accuracy.

Table 5 shows the performance of LSTM + CNN-1D in all three languages.

5.3.3. BiLSTM

Subsequently, we chose the BiLSTM approach, taking into account the issue of overfitting. BiLSTMs employ bidirectional processing, which differentiates them from traditional LSTMs that process sequences unidirectionally from left to right. BiLSTMs simultaneously engage two hidden states, one for processing sequences from the beginning to the end (forward direction) and another for processing sequences from the end to the beginning (backward direction).

Table 5 shows the performance of BiLSTM in all three languages.

5.3.4. BiLSTM with CNN-1D

A typical architectural configuration featuring the amalgamation of BiLSTM and CNN-1D [15] commences with the stacking of CNN-1D layers at the model's inception to extract local features. Subsequently, one or more BiLSTM layers capture temporal dependencies. The resultant outputs from these layers are now directed into additional fully connected layers, culminating in the final classification or regression tasks.

Table 5 shows the performance of BiLSTM + CNN-1D in all three languages.

Table 5
F1-Score of the different models on the datasets

Language	LSTM	LSTM + 1D CNN	BiLSTM	BiLSTM + 1D CNN
Bodo	0.8041	0.8178	**0.8351**	0.8141
Assamese	0.6770	0.6528	0.6554	**0.6630**
Bengali	0.6279	0.6279	0.5850	**0.6447**

5.4. Transformers based approach

After applying preprocessing techniques such as removing URLs, hashtags, usernames, etc, we employed various Transformer-based approaches [16], using pre-trained models and their tokenizers.

Bert-Base-Multilingual-Uncased:

Bert-Base-Multilingual-Uncased is a model [17] pre-trained on the top 102 languages with the largest Wikipedia using a masked language modeling objective. Uncased means all the

words are converted into lowercase. This model does not show the difference between 'english' and 'English'.

Assamese-Bert:
It is a BERT [18] model pre-trained on publicly available Assamese Monolingual datasets. This is used to classify for the Assamese Language task.

Bengali-Bert:
It is a BERT [18] model pre-trained on publicly available Bengali monolingual datasets. This is used to classify for the Bengali Language task.

Bengali-Abusive-Muril:
This is a MuRIL [2] model trained on Bengali abusive speech dataset. This model is trained with learning rates of $2e^{-5}$. We fine-tuned this to our Bengali dataset to obtain better results.

XLM-Roberta:
XLM-Roberta [19] is a model pre-trained on 2.5TB of filtered CommonCrawl data containing 100 languages.

We fine-tuned the given dataset and then applied hyperparameter tuning on an optimal batch size of 8, for accuracy, as very low and high batch sizes affect the training. Further, we incorporated early stopping and exponential learning rate decay with an initial learning rate of $1e^{-5}$ to fine-tune the above model for our task.

Indic-Bert:
IndicBERT[1] is a language model designed specifically for languages spoken in the Indian subcontinent, such as Hindi, Bengali, Tamil, and others. It underwent pre-training using a massive corpus of 9 billion tokens and we assessed it across various tasks. It is noteworthy that despite having significantly fewer parameters than models like m-BERT and XLM-Roberta, IndicBERT achieved top-tier performance in multiple tasks.

Distil-Bert:
DistilBERT [20] is like a smaller, faster version of the powerful BERT language model used for understanding and processing human language. It still works quite well but doesn't require as much computing power. It learns from BERT's knowledge instead of starting from scratch, making it good for tasks like figuring out what the text means or recognizing names in text.

Table 6 shows the performance of Transformers on Assamese, Bengali and Bodo.

6. Experimental Setup

Changing parameters in deep learning is a crucial part of the model development process. Before training the model, we split the training data into 2 parts, with 20% data as a test dataset and 80% as a training dataset using stratified sampling. We applied inbuilt CUDA GPU to our models, whenever it was available, else we used the CPU.

Hyperparameters are settings that are not learned during training but are instead configured before training begins. They have a significant impact on the model's performance and training

Table 6
Top Performances of Transformer Models

Model Architecture	F1-Score
Assamese	
Indic Bert	**0.6972**
Assamese Bert	0.68947
Distil Bert - base uncased	0.5872
XLM Roberta - large	0.3999
Bengali	
Bengali MuRIL	**0.7195**
m-Bert base uncased	0.6447
Bodo	
m-BERT base uncased	**0.8111**
XLM Roberta	0.8038
Indic Bert	0.6755

process. Here are some of the parameters we used to enhance the model's performance:

1. **Learning Rate:** Learning rate helps us measure the size of the updates made to the model's parameters during training. It influences how quickly or slowly a neural network learns. In this research, we have used an exponential learning rate[4], taking the initial value as 10^{-5} and gamma as 0.9. Gamma typically refers to a hyperparameter that controls the rate at which the learning rate decreases over time.
2. **Batch Size:** Batch size refers to the number of training examples used in each training iteration. A large batch size can lead to faster training but may require more memory. Hence, we changed the batch size based on memory usage. We took the batch size as 96 for Assamese and Bengali languages and as 64 for the Bodo corpus. It is recommended to choose a number that is a multiple of 32.
3. **Number of Epochs:** This refers to the number of times the model processes the entire training dataset. But if the number of epochs is less, then it may result in underfitting. Also, if the number of epochs is more, then it may cause the model to overfit for the given data. To handle this issue, we implemented an early stopping[5]method while training the model. Early stopping helps in finding a balance between model complexity and generalization. We chose the patience value as 2 for BERT-based approaches and 3 for Neural Network approaches. This value determines the number of training epochs the model should continue to train without seeing an improvement in the chosen validation metric before the early stopping mechanism is triggered. During this process, we found that early stopping is triggered for an epoch count of 6 in the LSTM model. When we implemented it with BERT models, early stopping triggered differently for different BERT models. Table 7 below lists the models and their corresponding epoch numbers.
4. **Network Depth:** This refers to the number of layers in a neural network. Deeper networks can capture more complex patterns but may be prone to overfitting if not

[4]https://keras.io/api/optimizers/learning_rate_schedules/exponential_decay/
[5]https://keras.io/api/callbacks/early_stopping/

Table 7
Early Stopping Details

Model	Early Stopping Triggered	Epoch with Best Parameters
Assamese		
BERT-base uncased	12	10
XLM RoBERTa	8	6
Indic BERT	6	4
Distil BERT	11	9
Assamese BERT	9	7
Bengali		
BERT-base uncased	10	8
Bengali MuRIL	8	6
Indic BERT	9	7
Bengali BERT	8	6
Bodo		
BERT-base uncased	6	4
XLM RoBERTa	8	6
Indic BERT	10	7

regularized properly. While working with neural network approaches like LSTM, BiLSTM, and BiLSTM with CNN1D, we took a network depth of 4.

5. **Number of Neurons:** While working with the neural networks, we set the input length as the number of neurons in the first input layer. In the layer where we applied RNN-based networks, we set the number of neurons as 128. In the third layer, where we applied a dense layer, we set it as 256. For the final layer, we set it as 1 because it determines the model's outcome. It is recommended to choose a number that is a multiple of 32.
6. **Dropout Rate:** This gives us the probability of dropping a neuron from the neural network during training, which helps prevent overfitting. While working with recurrent neural networks, we used two dropout rates. One being the regular dropout, which we applied to the inputs and/or the outputs. Another being the recurrent dropout, which removes the connections between the recurrent units. During our work, we set both of these values as 0.2.
7. **Optimizers:** We utilized Adam and AdamW optimizers while working with the model. Adam optimizer combines the advantages of two other popular optimization algorithms, stochastic gradient descent (SGD) and RMSprop, whereas AdamW incorporates weight decay directly into the optimization process, which helps prevent overfitting.
8. **Activation Functions:** While working with neural networks, we used ReLU activation function for every layer except the last layer where we used sigmoid activation function. The purpose being, dealing with vanishing gradients and exploding gradient problems. The problem of vanishing gradient occurs when the gradients of the loss function diminish significantly with respect to the model's parameters as they are propagated backwards through the layers of a deep neural network during training. The exploding gradient problem is the opposite of the vanishing gradient problem. It occurs when gradients grow significantly during backpropagation, often to the point where they become null or cause

numerical instability in the training process.

7. Results and Conclusion

The methodologies discussed in Section 5 and the comparison models are used to evaluate the performance of the three languages, namely Assamese, Bengali, and Bodo. Based on our research, we observed the following:

1. Notably, for Assamese and Bengali, BERT-based models exhibit the highest F1 scores, suggesting their efficacy in these contexts. Specifically, IndicBERT [1] emerges as the top-performing model for the Assamese corpus, while Bengali MuRIL [2] demonstrates superior performance for the Bengali corpus. These results underscore the effectiveness of leveraging pre-trained Bert models that cater to languages commonly spoken in the northeastern region of India, where Assamese and Bengali are prevalent.
2. In contrast, when evaluating the Bodo language, which is characterized by limited linguistic resources, we observed that a Neural Network-based approach outperforms other methodologies. Among the neural network architectures tested, the BiLSTM with an additional Dense layer yields the highest F1 score for Bodo. This result highlights the adaptability of neural network models for low-resource languages like Bodo, where dedicated pre-trained models may be scarce. Table 8 displays more details.

Table 8
Leaderboard best run in All the Languages

Language	Top-Performing Model	F1-Score
Assamese	Indicbert (BERT-based)	**0.69726**
Bengali	Bengali MuRIL (BERT-based)	**0.71955**
Bodo	BiLSTM with extra Dense Layer (Neural Network-based)	**0.83513**

3. A noteworthy observation from our research is the advantage of leveraging specialized BERT models pre-trained on languages such as Assamese and Bengali. These models are tailored to the linguistic nuances and characteristics of these languages, particularly relevant in the context of the northeastern region of India. Our findings demonstrate that utilizing these specialized models led to superior performance for Assamese and Bengali, showcasing the significance of language-specific pre-training in NLP tasks.
4. We encountered unique challenges when working with Bodo, a low-resource language in India. Unlike Assamese and Bengali, which benefit from pre-trained BERT models, Bodo lacks dedicated pre-trained models. Consequently, our research favors neural network-based methodologies for Bodo, as they outperform BERT models in this particular context. While it's possible to adapt existing BERT models to the Devanagari script used in Bodo, our results indicate that these adaptations may not match the performance achieved through neural network based approaches.
5. Apart from the given models, we are also experimenting with large transformer models [21] within the BERT family. However, due to the relatively small size of our dataset, these models tend to overfit during training.

In summary, our research highlights the importance of tailoring NLP methodologies to the linguistic characteristics and available resources of specific languages. While BERT-based models excel in well-resourced languages, low-resource languages like Bodo may benefit more from neural network-based approaches. Additionally, utilizing specialized pre-trained models for languages like Assamese and Bengali can significantly enhance performance, but it's crucial to consider the limitations posed by dataset size, especially when working with large transformer models. These findings contribute towards valuable insights into optimizing NLP approaches for diverse linguistic contexts. We secured 11th, 7th, and 11th position in Assamese, Bengali, and Bodo languages, respectively.

8. Acknowledgments

The authors would like to convey their sincere thanks to Clint Pazhayidam George, Vigneshwaran Shankaran, and Rajesh Sharma for helping us with our research. Throughout our research journey, we have been deeply inspired by their unwavering commitment to our project and our individual development. Their knowledge and teaching methods have made a big difference in our research journey.

We would also like to extend our heartfelt gratitude to Koyel Ghosh and her dedicated team for their exceptional organization of HASOC 2023. Throughout our research, they exhibited remarkable support and approachability, which greatly contributed to the success of our work.

References

[1] D. Kakwani, A. Kunchukuttan, S. Golla, G. N.C., A. Bhattacharyya, M. M. Khapra, P. Kumar, IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages, in: Findings of EMNLP, 2020.

[2] M. Das, S. Banerjee, A. Mukherjee, Data bootstrapping approaches to improve low resource abusive language detection for indic languages, arXiv preprint arXiv:2204.12543 (2022).

[3] S. Satapara, H. Madhu, T. Ranasinghe, A. E. Dmonte, M. Zampieri, P. Pandya, N. Shah, M. Sandip, P. Majumder, T. Mandl, Overview of the hasoc subtrack at fire 2023: Hate-speech identification in sinhala and gujarati, in: K. Ghosh, T. Mandl, P. Majumder, M. Mitra (Eds.), Working Notes of FIRE 2023 - Forum for Information Retrieval Evaluation, Goa, India. December 15-18, 2023, CEUR Workshop Proceedings, CEUR-WS.org, 2023.

[4] H. Madhu, S. Satapara, P. Pandya, N. Shah, T. Mandl, S. Modha, Overview of the hasoc subtrack at fire 2023: Identification of conversational hate-speech, in: K. Ghosh, T. Mandl, P. Majumder, M. Mitra (Eds.), Working Notes of FIRE 2023 - Forum for Information Retrieval Evaluation, Goa, India. December 15-18, 2023, CEUR Workshop Proceedings, CEUR-WS.org, 2023.

[5] S. Satapara, S. Masud, H. Madhu, M. A. Khan, M. S. Akhtar, T. Chakraborty, S. Modha, T. Mandl, Overview of the HASOC subtracks at FIRE 2023: Detection of hate spans and conversational hate-speech, in: Proceedings of the 15th Annual Meeting of the Forum for Information Retrieval Evaluation, FIRE 2023, Goa, India. December 15-18, 2023, ACM, 2023.

[6] S. Masud, M. A. Khan, M. S. Akhtar, T. Chakraborty, Overview of the HASOC Subtrack at FIRE 2023: Identification of Tokens Contributing to Explicit Hate in English by Span Detection, in: Working Notes of FIRE 2023 - Forum for Information Retrieval Evaluation, CEUR, 2023.
[7] K. Ghosh, A. Senapati, A. S. Pal, Annihilate Hates (Task 4, HASOC 2023): Hate Speech Detection in Assamese, Bengali, and Bodo languages, in: Working Notes of FIRE 2023 - Forum for Information Retrieval Evaluation, CEUR, 2023.
[8] T. Ranasinghe, K. Ghosh, A. S. Pal, A. Senapati, A. E. Dmonte, M. Zampieri, S. Modha, S. Satapara, Overview of the HASOC subtracks at FIRE 2023: Hate speech and offensive content identification in assamese, bengali, bodo, gujarati and sinhala, in: Proceedings of the 15th Annual Meeting of the Forum for Information Retrieval Evaluation, FIRE 2023, Goa, India. December 15-18, 2023, ACM, 2023.
[9] K. Ghosh, A. Senapati, U. Garain, Baseline bert models for conversational hate speech detection in code-mixed tweets utilizing data augmentation and offensive language identification in marathi, in: Fire, 2022. URL: https://api.semanticscholar.org/CorpusID:259123570.
[10] K. Ghosh, D. A. Senapati, Hate speech detection: a comparison of mono and multilingual transformer model with cross-language evaluation, in: Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation, De La Salle University, Manila, Philippines, 2022, pp. 853–865. URL: https://aclanthology.org/2022.paclic-1.94.
[11] K. Ghosh, D. Sonowal, A. Basumatary, B. Gogoi, A. Senapati, Transformer-based hate speech detection in assamese, in: 2023 IEEE Guwahati Subsection Conference (GCON), 2023, pp. 1–5. doi:10.1109/GCON58516.2023.10183497.
[12] S. Mundra, N. Mittal, Cmhe-an: Code mixed hybrid embedding based attention network for aggression identification in hindi english code-mixed text, Multimedia Tools and Applications 82 (2023) 11337–11364. doi:10.1007/s11042-022-13668-4.
[13] I. Bhat, V. Mujadia, A. Tammewar, R. Bhat, M. Shrivastava, Iiit-h system submission for fire 2014 shared task on transliterated search, in: Proceedings of the [conference name], 2015. doi:10.1145/2824864.2824872.
[14] A. Joshi, A. Prabhu, M. Shrivastava, V. Varma, Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text, in: Proceedings of COLING 2016, the 26th International conference on computational linguistics: Technical papers, The COLING 2016 Organizing Committee, 2016, pp. 2482–2491. URL: https://aclanthology.org/C16-1234.
[15] P. Kapil, A. Ekbal, D. Das, Investigating deep learning approaches for hate speech detection in social media, 2020.
[16] M. Das, S. Banerjee, P. Saha, A. Mukherjee, Hate speech and offensive language detection in Bengali, 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 (Volume 1: Long Papers), Association for Computational Linguistics, Online only, 2022, pp. 286–296. URL: https://aclanthology.org/2022.aacl-main.23.
[17] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: http://arxiv.org/abs/1810.04805. arXiv:1810.04805.
[18] R. Joshi, L3cube-hindbert and devbert: Pre-trained bert transformer models for devanagari

based hindi and marathi languages, arXiv preprint arXiv:2211.11418 (2022).

[19] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, Unsupervised cross-lingual representation learning at scale, CoRR abs/1911.02116 (2019). URL: http://arxiv.org/abs/1911.02116. arXiv:1911.02116.

[20] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, ArXiv abs/1910.01108 (2019).

[21] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized BERT pretraining approach, CoRR abs/1907.11692 (2019). URL: http://arxiv.org/abs/1907.11692. arXiv:1907.11692.