PROJECT REPORT

ON

Detecting Toxic Comments in Bengali Language

Project I Submitted in partial fulfilment of the requirements

for the degree of

B. Tech in CSE (Artificial Intelligence and Machine Learning)

BY

Mouri Roy

Kingshuk Roy

Arnab Mukhopadhyay

Abir Mondal

Ashutosh Kumar Jha

UNDER THE GUIDANCE OF

Ms. Susmita Das

Assistant Professor

CSE (Artificial Intelligence and Machine Learning) Department

TECHNO MAIN SALT LAKE

EM 4/1 SALT LAKE CITY, SECTOR V

KOLKATA – 700091

West Bengal, India

Techno Main Salt Lake

[Affiliated by Maulana Abul Kalam Azad University of Technology (Formerly known as WBUT)]

FACULTY OF CSE (AI & ML) DEPARTMENT

Certificate of Recommendation

This is to certify that this group has completed their project report on: "**24/11/2023**", under the direct supervision and guidance of **Ms. Susmita Das**. We are satisfied with their work, which is being presented for the partial fulfilment of the degree of B. Tech in CSE (Artificial Intelligence and Machine Learning), Maulana Abul Kalam Azad University of Technology) (Formerly known as WBUT), Kolkata – 700064.

Ms. Susmita Das
(Signature of Project Supervisors)
Date: _ _ / _ _ / _ _ _ _

Dr. Sudipta Chakrabarty
(Signature of Project Coordinator)
Date: _ _ / _ _ / _ _ _ _

Dr. Soumik Das
(Signature of the HOD)
Date: _ _ / _ _ / _ _ _ _

Acknowledgement

We would like to express our sincere gratitude to our project supervisor, Ms. Susmita Das, for her guidance and support throughout the completion of our final year group project I on “Detecting Toxic Comments in Bengali Language”. Her valuable feedback and suggestions helped us to improve our work and to produce a project that we are proud of.

We would like to express our profound appreciation to our Department Head, Dr. Soumik Das, and the Project Coordinator, Dr. Sudipta Chakrabarty, for providing us with the resources and facilities we needed to complete the project.

We would also like to thank our group members for their hard work and dedication to this project. We could not have done it without them. Each member of the group brought their unique skills and talents to the project, and we learned a lot from each other along the way.

Finally, we would like to thank our families and friends for their support during this challenging but rewarding year. We are grateful for their encouragement and understanding.

Sincerely, Abir Mondal, Kingshuk Roy, Arnab Mukhopadhyay, Mouri Roy, and Ashutosh Kumar Jha

Date: 24/11/2023

Table of Contents

Introduction

Social media gives regular people a platform to express their thoughts, emotions, and responses on a wide range of subjects. Teens and adults alike develop regular habits of using this public forum to spend a great deal of time interacting with their peers. However, this online environment frequently gives rise to contentious issues like political propaganda, insane religious beliefs, and random hoaxes. Parties that are divided over this phenomenon exchange derogatory remarks, threats, and foul language in an attempt to harm one another directly. Such offensive language, often known as toxic comments, must be removed from the platform to ensure a safe user experience. Removing such harmful remarks from analysis in light of social media's role as a hub for information is a challenge that both humans and automated comment filter faces.

Most of the existing research on abusive language detection has focused on English or other widely spoken languages. However, there is a lack of resources and methods for detecting toxic comments in low-resource indic languages, such as Bengali, Assamese, Gujarati, Punjabi, Sanskrit, etc.. Bengali is the seventh most spoken language in the world, with over 250 million speakers. It is the official language of Bangladesh and one of the many languages widely spoken in India.

An increasing number of Bengali users of social media post a lot of status updates, along with images, comments, and other content, to which others can respond right away. Generally, this frequently produces text with negative comments that must be filtered out. With lists of offensive words and intricate regulations, manually filtering toxic comments is a difficult task, especially for inflectional languages like Bengali. Social media text is unstructured, which adds to its difficulty and low acceptability score. It was previously difficult to write handwritten rules using manual linguistic features, but automated systems and inexpensive computing resources have changed this situation. A component of Natural Language Processing (NLP) is the computational extraction of salient features from textual data. NLP requires an annotated corpus to convey information related to various applications, one of which is the detection of toxic comments. Scholars have attempted to identify sentence toxicity using statistical machine-learning models. However, these models don't work well with unstructured social media comments because they need frequency-based feature engineering or probabilistic phenomena. By capturing low-level features and combining them into layer-wise abstractions, deep learning-based models have demonstrated their efficacy in addressing these limitations. Various studies have demonstrated the significant superiority of these models over other supervised machine learning models in the context of English text analysis. We have examined that assertion for detecting Bengali toxic comment material in this academic work.

Our main contribution is that we have merged three datasets from different sources and domains, namely the Bangla-Abusive-Comment-Dataset, Bengali Hate Speech Dataset, and Bangla Online Comments Dataset, to create a large and diverse dataset for detecting abusive

content in Bengali. The merged dataset consists of 90,345 comments, out of which 38,412 are labeled as toxic and 52,089 are labeled as non-toxic. We trained a deep learning model using BanglaBERT [1], a modified version of BERT [2] architecture that is pre-trained on a large corpus of Bengali text, and LSTM [3], a recurrent neural network that can capture long-term dependencies in sequential data. We tested the performance of our model and compared it with four baseline classifiers, namely Random Forest, Naive Bayes, Support Vector Machine, and Logistic Regression, using various metrics such as accuracy, precision, recall, and F1-score. Our results show that our model outperforms the baseline classifiers in most of the metrics, demonstrating the effectiveness of BanglaBERT [1] and LSTM [3] for abusive content detection in Bengali.

Motivation

The motivation behind our project is to address the problem of toxic comments detection in Bengali, which is a low-resource language with limited approaches and methods for this task. Toxic language detection is an important and challenging task that aims to identify and flag harmful utterances in online conversations, such as hate speech, cyberbullying, harassment, and profanity. These forms of abusive language can have negative impacts on the mental health and well-being of the users and the online community. Therefore, it is essential to develop effective and robust methods for detecting abusive language in online social media platforms.

It is laborious to manually create rules to filter such offensive comments because they are disorganized and frequently contain vulgar words spelt incorrectly. Sentiment analysis includes automated machine learning-based models for classifying such toxic comments. These models are widely used for the English language and exhibit more promising outcomes than statistical models. Despite being a language that is spoken by many people, more research needs to be done to identify harmful remarks in Bengali. Therefore, to identify harmful Bengali comments from an annotated dataset, we compare four supervised learning models — Random Forest, Naive Bayes, Support Vector Machines, Logistic Regression, with our implemented model Bangla - Bidirectional Encoder Representations from Transformers [BanglaBERT] and Long Short Term Memory(LSTM) in this academic year.

In this project, we aim to fill this research gap and address this practical challenge by proposing an innovative technique for detecting abusive comments in Bengali from social media platforms. Our approach consists of three main components - *dataset design*, *model development,* and *model evaluation*. We will go into great detail about each element and will justify our decisions and choices, which will be based on the problem domain, the literature review, and the data characteristics. Additionally, we'll discuss how our approach responds to the research questions or hypotheses we formulated in the introduction. The following sections will provide further information on our project.

Problem Statement

The communication languages used by multiple people on online platforms are wide and varied. English being the most widely used language, gets additional attention during language-based research. As other languages emerge on these online platforms, the offensive words being used in these languages are also increasing. Toxic language has been a fundamental issue in social media analysis recently. In the problem of toxic language detection, the primary focus has been given to classifying whether the posts on online social media platforms are abusive or non-abusive. In regards to language, initially, attention has been specified to Bengali, one of the widely used Indic languages. In our model, natural language processing and machine learning have been used for data preparation, feature engineering, selection, training and evaluation. The Bengali text string is considered abusive or toxic if it contains any form of offensive words, racism, sexism, hate speech or personal attacks. The output of the model is a binary label of abusive or non-abusive. The goal of the problem is to implement a method for accurately and efficiently detecting toxic comments in Bengali on social media platforms.

Literature Survey

A study [4] on hate speech detection in Bengali Social Media was conducted, addressing the gap in research for low-resource languages. A dataset of 10,000 Bengali social media posts, including both actual Bengali Script and Romanized Bengali text, was created. The study evaluated several baseline classification models and explored interlingual transfer mechanisms to enhance accuracy. XLM-Roberta performed best when trained separately, while MuRIL outperformed others in joint training and few-shot training. The dataset and code were made publicly available to facilitate further research.

The paper [5] introduces BanglaHateBERT, a model specifically designed to detect abusive language in Bengali. It was trained on a large corpus of offensive Bengali content and outperformed generic pre-trained language models on various datasets. The paper also makes available a 15,000-sample Bengali hate speech dataset and the BanglaHateBERT model for further research. This work contributes to addressing hate speech detection in languages with limited linguistic resources.

This research [6] introduces a new dataset of 30,000 user comments from YouTube and Facebook, tagged via crowdsourcing and verified by experts. The dataset is categorized into seven distinct categories and annotated with a high level of precision. The paper also presents baseline experiments and the application of various deep-learning models using pre-trained Bengali word embeddings. The results show that SVM outperforms other models, achieving an accuracy rate of 87.5%. The paper's most significant contribution is the provision of this benchmark dataset, which will facilitate further research and aid in addressing hate speech in Bengali.

This study [7] compares various feature engineering techniques and machine learning algorithms to assess their effectiveness in detecting hate speech messages. It evaluates their performance on a publicly available dataset that encompasses three distinct classes. The study demonstrates that utilizing bigram features in conjunction with the support vector machine algorithm yields the best performance, achieving an impressive overall accuracy rate of 79%. These findings not only offer practical implications for addressing the issue of hate speech but also provide a foundational reference for future research in the field of automated text classification. By serving as a benchmark, this study equips researchers with state-of-the-art techniques to compare and build upon in their endeavours to combat hate speech in the digital realm.

This research [8] proposes a multi-modal system for hate speech detection in video content. It extracts features from audio, text, and other relevant components, and uses machine learning and natural language processing to integrate the data for enhanced detection. This approach addresses the complexities of hate speech detection in video content and opens new avenues for improving the accuracy of detection models. It aligns with the evolving nature of digital communication and the need to mitigate hate speech online. Adopting a multi-modal approach to hate speech detection in video content is crucial for effectively combating this issue. Integrating diverse data sources and leveraging advanced technologies will lead to more robust and comprehensive solutions.

This paper [9] comprehensively reviews hate speech detection and monitoring, focusing on natural language processing and deep learning technologies. It covers key terminology, processing pipelines, core methods, and deep learning model architectures. The review adheres to PRISMA guidelines, covering a decade's worth of literature from ACM Digital Library and Google Scholar. It identifies limitations and outlines future research directions. The paper serves as a valuable resource for understanding the state of hate speech detection and monitoring, highlighting the need for ongoing research to enhance its performance and efficacy.

This work [10] provides a comprehensive overview of the background and context surrounding hate speech, exploring various detection approaches. It also delves into recent contributions and developments in hate speech detection and addressing anti-social behaviours in online spaces. The challenges unique to Arabic hate speech detection are highlighted, underscoring the complexity and nuances of the language. The paper concludes by offering recommendations and insights for addressing the pressing issue of Arabic hate speech detection, recognizing the importance of leveraging technology to mitigate its harmful effects in the digital age.

This analysis [11] addresses the issue of hate speech proliferation on social media and its potential to escalate into real-world violence and crimes. The authors proposed a hate speech detection model that leverages machine learning and text-mining techniques to combat this problem. The study focuses on English-Odia code-mixed data, sourced from a Facebook

public page, and manually categorized into three classes. These classes are further refined to create both binary and ternary datasets. The hate speech detection model is built by combining a range of machine learning algorithms with various feature extraction methods. Specifically, Support Vector Machine (SVM), Naive Bayes (NB), and Random Forest (RF) models are trained using the complete dataset, with features derived from word unigrams, bigrams, trigrams, combined n-grams, TF-IDF, combined n-grams weighted by TF-IDF, and word2vec for both binary and ternary datasets. The results reveal that SVM models utilizing word2vec features outperform the NB and RF models in binary and ternary classifications. Moreover, the ternary models exhibit a notable advantage in distinguishing between hate and non-hate speech, suggesting their potential to minimize confusion.

This work [12] addresses this gap by introducing a standard Arabic dataset tailored for hate speech and abuse detection. What sets this dataset apart is its multi-platform collection approach, drawing data from various social network platforms, including Facebook, Twitter, Instagram, and YouTube. To evaluate the dataset's efficacy, the study employs twelve machine learning algorithms and two deep learning architectures. Impressively, the Recurrent Neural Network (RNN) emerges as the top performer, achieving an accuracy rate of 98.7%. This work marks a significant step in mitigating hate speech and abusive content in Arabic-language online spaces, emphasizing the need for comprehensive datasets collected from multiple sources and harnessing advanced machine learning and deep learning techniques to combat this growing issue.

This study [13] explores the application of deep learning models and domain-specific embeddings for hate speech detection in English-Hindi code-mixed tweets. The proposed approach demonstrates significant improvements in hate speech identification compared to traditional methods, achieving a 12% increase in F1-score. This research highlights the potential of deep learning and domain-specific embeddings in enhancing hate speech detection capabilities, particularly in multilingual and code-mixed environments. The findings can contribute to the development of more effective tools for combating hate speech and fostering a safer online environment.

This research [14] proposes a new language-independent method to collect a large amount of offensive and hate tweets, regardless of their subject matter or style. The study leverages extralinguistic cues embedded in emojis to identify a significant number of offensive tweets in Arabic and English. The research contributes significantly by manually annotating and releasing the largest Arabic dataset for offensive content. The dataset is employed, along with an in-depth linguistic analysis, to assess the effectiveness of various transformer architectures in detecting offensiveness and hate speech. The study evaluates the models’ performance on external datasets. The competitive results on these datasets suggest that the approach captures universal attributes of offensive language, although the research highlights the challenges of considering cultural context and background, as well as handling nuances like sarcasm.

This article [15] discusses the growing problem of hate speech on social media and the need for multilingual hate speech detection algorithms. The HASOC track was created to develop and improve hate speech detection algorithms for Hindi, German, and English. The best-performing algorithms achieved F1 measures of 0.51, 0.53, and 0.52 for task A, which involves binary classification for English, Hindi, and German. For task B, a fine-grained classification task, the top algorithms achieved F1 measures of 0.26, 0.33, and 0.29 for English, Hindi, and German, respectively. The results indicate that transformer-based architectures, particularly BERT variants, performed well, but other systems also showed promise. The HASOC track is a valuable initiative in advancing research and solutions for multilingual hate speech detection.

This study [16] proposes a new approach to hate speech detection using Deep Neural Network (DNN) structures as feature extractors. The goal is to capture the semantic nuances of hate speech, which can be more difficult to identify than simply classifying text as hateful or non-hateful. The proposed DNN structures are evaluated on a large dataset of Twitter hate speech and shown to outperform existing methods by up to 5% in macro-average F1 or 8% in identifying hateful content. This research has the potential to improve the effectiveness of hate speech detection tools and help combat the spread of hate speech online.

This analysis [17] proposes a "Multi-Class Classifier" to distinguish between hate speech, offensive language, and non-offensive content. It leverages crowd-sourced data to create a hate speech lexicon and labels tweets into three categories. A multi-class classifier is then trained to classify these categories. The research analyzes the accuracy of these classifications, finding that tweets with explicit hate keywords are more reliably classified as hate speech, while sexist tweets are typically labelled as offensive. Tweets without explicit hate keywords present more challenges in classification. The focus is on improving the precision of automatic hate-speech detection by separating it from other instances of offensive language on social media.

This paper [18] explores various approaches to hate speech detection, including rule-based, machine learning-based, deep learning-based, and hybrid approaches. It highlights the significance of implementing effective hate speech detection systems on social media platforms like Facebook and Twitter due to the pervasive nature of hate speech online. The paper delves into the contributions of various researchers who have studied and implemented these approaches to combat hate speech. While it doesn't specify a particular dataset, it emphasizes the need to address the ongoing issue of hate speech on the internet, particularly on major social media platforms, and underscores the diverse methods employed by researchers in this domain.

This study [19] explores two deep learning models, ULMFiT and mBERT-BiLSTM, for offensive language detection in code-mixed Tamil text on YouTube. ULMFiT utilizes transfer learning to fine-tune language models, while mBERT-BiLSTM combines multilingual BERT with BiLSTM networks. The study employs deep learning and transfer learning techniques to

identify intricate patterns in code-mixed Tamil text. The Tamil Code-Mix Dataset, curated from YouTube, serves as the research's foundation. This dataset highlights the challenges of offensive language detection in under-resourced languages like Tamil. The findings demonstrate the effectiveness of ULMFiT and mBERT-BiLSTM in addressing this issue, offering a promising avenue for improving offensive speech detection in Tamil.

This research [20] introduces a novel deep learning model called CNN-gram for hate-speech and offensive language detection in Roman Urdu (RU) social media content. The model is based on transfer learning and domain-specific embeddings, making it adaptable to Roman Urdu's nuances. The model is evaluated on the RUHSOLD dataset, which is a meticulously crafted dataset of 10,012 tweets in Roman Urdu with annotations for hate speech and offensive language. The findings show that the CNN-gram model is highly effective at detecting hate speech and offensive language in Roman Urdu social media content.

The study [21] introduces a robust ensemble approach for hate speech detection in Tamil, combining multiple techniques to achieve remarkable results. The study employs a diverse array of tools and methodologies, including TF-IDF for text representation, pre-trained transformer models, and enhanced stemming algorithms. The research leverages the HASOC 2021 dataset, comprising YouTube comments in Tamil, written in the Tamil script. The study achieves an F-score of 84% and an accuracy rate of 86% in effectively identifying offensive content within Tamil YouTube comments.

This paper [22] presents a Multiteacher Distillation Framework for offensive language detection. The framework includes building multiple teachers using publicly accessible datasets and using them to assign soft labels to the AugCOLD dataset, which is an augmented Chinese Offensive Language Dataset. The soft labels facilitate knowledge transfer from both AugCOLD and the multiteacher models to the student network, which is the final offensive language detector. This framework improves generalization and robustness in offensive language detection. The dataset introduced in this study is AugCOLD, a large-scale unsupervised dataset containing 1 million samples. It was gathered through data crawling and model generation and serves as a valuable resource for training and evaluating Chinese offensive language detectors.

This work [23] delves into the realm of tweet classification, examining the efficacy of various neural network architectures in categorizing tweets into three distinct classes: Hate, Offensive, and Neither. The authors employ a range of models, including BI-LSTM (Bidirectional Long Short-Term Memory) both with and without pre-trained Glove embeddings, as well as transfer learning approaches utilizing advanced language models such as BERT, DistilBert, and GPT-2. To optimize the BI-LSTM model, they engage in meticulous hyperparameter tuning. By leveraging these diverse neural network architectures and language models, the authors achieve an impressive accuracy of over 92%, effectively demonstrating the power of neural networks in the task of tweet classification. This study highlights the potential of neural networks to accurately classify tweets into meaningful

categories, paving the way for further advancements in sentiment analysis and social media monitoring.

Methods and Algorithms

In this section, we will explain the methods that we followed to solve the problem of detecting abusive content in Bengali. Our approach consists of four main steps - *data collection*, *data preprocessing*, *model development*, and *model evaluation*.

I. Data Collection

The text samples were collected from three sources. Among these datasets only one dataset was multi-labeled, so we decided to reclassify the data into a single label, where 1 is used to denote hate speech and 0 is used to denote non-hate speech. Reclassifying the texts from multiple datasets into a new set of categories is crucial for several reasons, including ensuring consistency, enhancing data quality, providing more insightful information about the nature and extent of toxicity, boosting the performance of machine learning models, and streamlining data management.

A. Dataset 1

The first dataset, BD-SHS [24] contains 50,281 comments. The authors collected those Bengali comments from online social media and streaming platforms based on controversial topics and keywords. They used an open-source tool called Facepager to scrape the comments and removed duplicates and highly similar ones using the Jaccard Index.

The authors annotated the comments in three levels - hate speech identification, target identification, and hate speech categorization. They used binary labels for the first level, four labels for the second level, and four labels for the third level. They also employed three professional annotators and calculated the inter-annotator agreement using Cohen’s kappa coefficient.

Using the dataset, the authors experimented with various machine learning models, including SVM and Bi-LSTM, to detect and analyse hate speech. To enhance performance, they adjusted hyperparameters and employed word embeddings that had already been trained. They mentioned the models' F1-score, recall, accuracy, and precision.

Table 1: Distribution of annotated comments across three levels.

Level	Annotation Labels	Number of comments
Level 1	HS NH	24156 26125
Level 2	Ind Male Female Group	8815 7128 5443 3119
Level 3	Slander Religion Call to Violence Gender	16992 1562 7232 4244

B. Dataset 2

The second dataset [25] contains 10,219 Bengali comments. Experiments were performed with a human-annotated public domain dataset available at Github [26]. The dataset contains five tags for each Bengali social media comment toxic, threat, obscene, insult, and racism. However the number of tagged comments for all columns except toxic is considerably small.

Table 2: Dataset Statistics

Total Comments	10219
Toxic Comments	4255
Non-Toxic Comments	5964
Longest Comment Length (in words)	528
Smallest Comment Length (in words)	1
Average Comment Length (in words)	12
Unique Words	23600

The authors experimented with different machine learning and deep learning models such as Naive Bayes, Support Vector Machines, Logistic Regression, LSTM, and CNN. The top best accuracies were 94% and 95% for LSTM and CNN respectively.

C. Dataset 3

We followed the example of a previous paper [27] and combined various datasets to create a usable dataset for our work. The datasets that were used by the authors are - the multi-labelled Bangla-Abusive-Comment-Dataset [28], the multi-class Bengali Hate speech Dataset [29], and the Bangla Online Comments Dataset [30]. The authors also grouped the dataset into six categories: vulgar, hate, religious, threat, troll, and insult, to resolve the inconsistency in the original datasets. This [27] dataset was available in Github [31].

Table 3: Statistics of the toxic comments for multi-label classification

Class	Number of Instances	Kappa Score
Vulgar	2505	0.88
Hate	1898	0.79
Religious	1418	0.84
Threat	1419	0.71
Troll	1643	0.66
Insult	2719	0.74

Labeling of data was done using three annotators as mentioned in [32]. Those annotators have a trustworthiness score above 80%. To validate and assess the quality of the annotations, an evaluation of the inter-annotator agreement was done using Cohen's kappa coefficient [33]. The final dataset consisted of 16,073 instances in total, out of which 7,585 instances were labeled Non-toxic and 8,488 instances was labeled toxic.

The authors experimented with deep learning models. The models were LSTM + BERT embedding (accuracy 76.52%), MConv-LSTM + BERT embedding (accuracy 74.72%), BanglaBERT fine-tuned (accuracy 77.66%), and CNN-BiLSTM with Attention (accuracy 78.92%).

II. Dataset Preparation

A. Data Combination

The datasets listed above were all merged. Then the annotation was completed. A comment received a label of 0 if it was classified as not hateful or non-abusive and a label of 1 for toxic/hateful comments.

In conclusion, the final master dataset consists of a total of 90345 instances. Here is a tabular representation for better visualization.

Table 4: Merged Dataset Records

Total Comments	90345
Toxic Comments	38412
Non-Toxic Comments	52089
Unique Words	69370

B. Data Split

The dataset was split into three sections - the training set, the testing set, and the validation set. The sklearn.model_selection module was used to do the split using the train_test_split function.

First, the dataset was divided into two subsets - the training set and the testing set. 80% of the data were in the training set and the rest 20% were in the testing set. Following that, the training set was further split into two parts - the training set and the validation set. Now the training set contains 70% of the data and the validation set contains 10% data.

Table 5: Dataset Split details

Train Set	Test Set	Validation Set
63241 (70.00%)	18069 (20.00%)	9035 (10.00%)

III. Data Pre-processing

Social media comments typically lack structure and adhere to no particular rules. In the case of abusive and toxic comments, these situations become more complex when the commentators intentionally misspell words and repeatedly utilize characters that are close to one another to convey their feelings of rage and sadness. The size of comments can have some restrictions, although these vary depending on the platform where the comments are submitted. Emojis, careless punctuation, spelling mistakes, and haphazard repetition are a few other common issues with social network comments. Any manual text processing strategy involves several preprocessing steps to clean up the noisy data. But because punctuation and emoticons don't convey any meaning for toxicity, we merely remove them before doing tokenization due to the nature of toxic remarks. We do not apply stemming or lemmatization to our dataset because the commentators' attitudes can occasionally be conveyed by their lengthy wording or deliberate use of repetition.

The pre-processing tasks that are done on the dataset are as follows:

01. Removing comments with stars - Some comments had words that were replaced with stars, mainly abusive words, so those comments were removed from the dataset. This was done because if the abusive words are censored then the model will not be able to learn the pattern of abusive comments.
02. Removing special words - As the data was collected using web scraping, so, some unwanted HTML tags were also scrapped in the comments. Those were also removed.
03. Removing Links and Emojis - Social media comments can have links that do not contribute to the meaning of the comment. Also, emojis are difficult to handle for a deep learning model. That is why they were removed using an open-source library known as normalizer [34].
04. Removing single-letter words - Words that contained only one letter were removed because it was seen by manual testing that either those words did not contribute to the meaning of the comment or it was there by mistake from the commentator.
05. Translate English words to Bengali - Some comments had some English words. Those were translated using the Google Translator.
06. Strip Comments - Some comments had some extra, unnecessary spaces, so those were dropped out.

IV. Model Development

We will go over the suggested methodology we created for our project in this section. The two primary parts of our approach are model development and model evaluation. We will go into great detail about each element and provide detailed justification for our decisions and choices made during our project development.

Model Pipeline

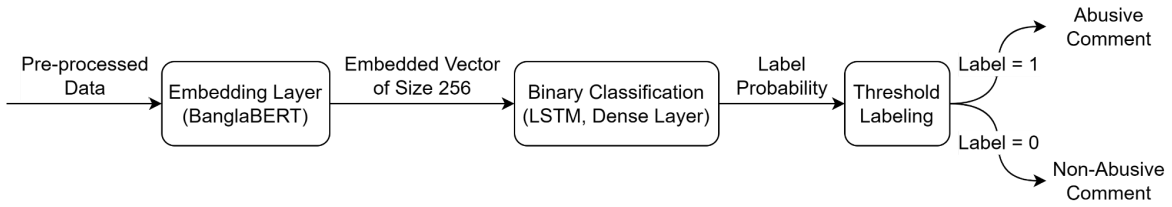

Figure 1

The model consists of three parts - *the embedding layer*, *the binary classification layer*, and *the Threshold Labelling*. All these layers will be discussed in the following sections.

Figure 1 shows the model pipeline. First, the pre-processed data enters into the Embedding Layer. Embedding Layer generates an embedding vector of size 256 for each comment. This embedding vector is passed on to the Binary Classification Layer. This layer learns the patterns and tries to learn how to detect Abusive Comments. The Threshold Labeling layer takes in the label probability to generate the label.

A. Embedding Layer

Deep Learning models do not understand text data. It can only accept numerical data. For that, the text data or the comments are converted into vectors which are called embedding vectors.

The model uses a two-step process to transform a text input into a numerical representation that can be used for deep learning. The first step is tokenization, which is the process of breaking down the text into its basic units, called tokens. Tokens are usually words or subwords that have a specific meaning in the language. The model uses the AutoTokenizer module of the transformer library, which is a Python package that provides various tools for natural language processing. The AutoTokenizer module can automatically detect the language of the text and apply the appropriate tokenization method. The second step is embedding, which is the process of mapping each token to a vector of numbers that captures its semantic and syntactic features.

We used BanglaBERT [1] to create the embedding vectors. It is based on BERT [2] architecture and pre-trained on Bengali text. It will generate an embedding vector of size 256. These embeddings can help deep learning models understand the similarity between two sentences. For computation constraints, we only kept the embedding vector size at 256, but this can be increased to 512 to incorporate more information into the vectors.

The Bidirectional Encoder Representations from Transformers (BERT) [2] is a seminal transformer-based language model that involves an attention mechanism that enables contextual learning relations between words in a text sequence. BanglaBERT [1] is a pre-trained language model for Bangla, a widely spoken language with over 265 million speakers. It is based on the BERT architecture, which has achieved state-of-the-art results on a variety of natural language processing (NLP) tasks. BanglaBERT was pre-trained on a massive dataset of Bangla text, including news articles, books, and social media posts. This training process allows BanglaBERT to learn the contextual relationships between words and phrases, which can be used to improve the performance of NLP tasks such as text classification and natural language inference.

For implementation, we used the Hugging Face transformer [35] to implement BanglaBERT [1].

B. Binary Classification Layer

This is the area where we have used deep learning and the architecture is designed by tuning different hyperparameters. It consists of three types of layers - LSTM, Dense, and Dropout.

Long Short-Term Memory(LSTM) [3] can learn long-term dependencies in sequential data, which is essential for many NLP tasks such as machine translation, text summarization, and

sentiment analysis. LSTMs work by using a gating mechanism to control the flow of information through the network. This gating mechanism allows LSTMs to learn which information is important to remember and which information can be forgotten. This is important for NLP tasks because text can often be long and complex, and the model needs to be able to focus on the most important information in the sequence.

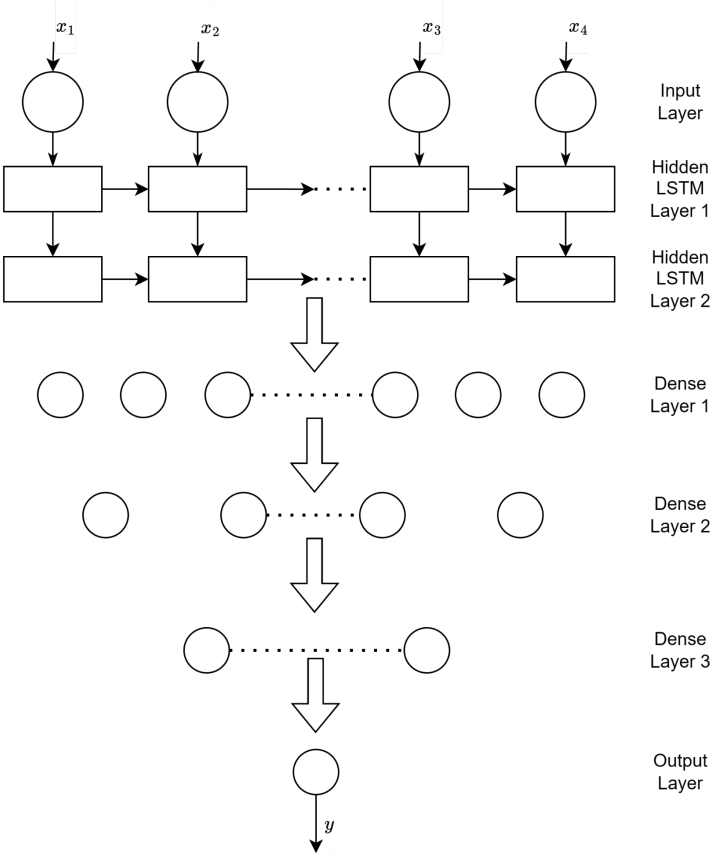

Figure 2

First, the embedding vector is passed to the binary classification model as input through the input layer. There are two LSTM layers each connected. There are a total of 256 LSTMs in each LSTM layer. LSTM Layer 1 takes the input of 256 size vector. Then LSTM Layer 2 takes 256 size vectors as input from LSTM Layer 1. Finally, the output of LSTM Layer 2 is passed to Dense Layer 1. The LSTM layers have a dropout rate of 0.1, and the activation function is *tanh*.

Dense Layer 1 has 512 nodes that take input from the LSTM Layers. The output from Dense Layer 1 is passed on to Dense Layer 2 which has 256 nodes. The output of Dense Layer 2 is passed to Dense Layer 3 which has 64 nodes. All the dense layers have leaky ReLU activation functions. After each dense layer, there is a dropout layer whose dropout rate is 0.5.

Finally, the output of Dense Layer 3 is passed to the Output Layer has only one node. The output layer has a sigmoid activation function. Output has a sigmoid activation function because we need to get a probability of how much the comment is abusive or not.

The model was trained for 100 epochs with a learning rate of 0.001 and Adam optimizer was used for optimizing the loss function. Accuracy was used as the metric to observe the model's performance while training.

C. Threshold Labelling

Threshold labelling is a type of layer in a neural network that is used to convert the output of an LSTM layer into a sequence of labels. It works by setting a threshold value, which is a probability level. All values in the output of the LSTM layer that are higher than the threshold value are converted to a 1, and all values that are lower than the threshold value are converted to a 0. Threshold labelling layers are often used in conjunction with LSTM layers because LSTM layers are good at learning long-range dependencies in data. This makes them well-suited for tasks such as sequence classification, segmentation, and tagging. The LSTM layer would learn to extract features from the sentences, and the dense layer would learn to combine these features to predict a probability that the sentence is positive. The threshold labelling layer would then convert this probability into a binary label (positive or negative).

V. Model Evaluation

The model was trained and evaluated based on accuracy, precision, recall, and f1-score. To compare the performances of our deep learning-based model, we have applied four classifiers named Random Forest, Naive Bayes, Support Vector Machines, and Logistic Regression as our baseline models. The dataset is trained on training data and tested on the testing data. No separate validation data is used for parameter tuning.

We used TF-IDF to represent each comment in our dataset as a vector of numerical values for the baseline machine learning models. TF-IDF stands for Term Frequency-Inverse Document Frequency and measures the importance of each word in a document relative to its frequency in the entire corpus. TF-IDF assigns higher weights to words that are more specific and informative and lower weights to words that are more common and generic.

A. Support Vector Machines

SVMs [36] are a type of supervised machine-learning algorithm that can be used for both classification and regression tasks. However, they are most commonly used for classification problems. SVMs work by finding a hyperplane in the data that separates the different classes as widely as possible. This hyperplane is then used to classify new data points. They are particularly well-suited for classification problems in high-dimensional data. SVMs are relatively robust to noise in the data and can be trained on relatively small datasets. However,

SVMs can be slow to train on large datasets and can be sensitive to the choice of kernel function and other hyperparameters. Hyperparameters such as regularizer C, penalty l2, and hinge loss for linear svc were finetuned to find the best-performing combinations on the testing set. The general equation for a Support Vector Machine is

$$f(x) = w^{\mathrm{T}}\Phi(x) + b$$

where $f(x)$ is the predicted label for input x, w is the weight vector, $\Phi(x)$ is the feature vector of input x , and b is the bias term. The feature vector is created by transforming the text input into a numerical representation. This can be done using a variety of methods, such as bag-of-words, TF-IDF, or word embeddings. If the value $f(x)$ is positive then it is considered to be a toxic comment and if the value is negative, the comment is classified as non-toxic.

B. Naive Bayes

Naive Bayes [37] is a simple and effective supervised machine learning algorithm for classification tasks. It is a probabilistic classifier that uses Bayes' theorem to calculate the probability of a data point belonging to a particular class. The general equation for multinomial Naive Bayes (NB) classifier for text classification is as follows:

$$P(c \mid d) = P(c) * \prod_i P(x_i \mid c)^{n_i}$$

where $P(c)$ is the prior probability of class c, $P(x_i|c)$ is the probability of word $P(c)$ appearing in documents of class c, n_i is the number of times word x_i appears in document d and Π is the product symbol.
Naive Bayes classifier can be used to classify the sentiment of a text message or to classify the topic of a news article. The features for a text classification task could be the words in the text, or the features could be more complex features, such as the part-of-speech tags of the words.

C. Random Forest

Random forest [38] is an ensemble learning algorithm that combines multiple decision trees to produce more robust and accurate predictions. It is a popular machine-learning algorithm for both classification and regression tasks. The equation for Random Forest for text classification is as follows:

$$P(c \mid d) = Mode(T(d))$$

where $P(c \mid d)$ is the probability of class c in a given document d, $T(d)$ is the set of predictions made by the individual trees in the random forest. It is important to be aware of

the computational cost of training random forests and the difficulty of tuning their hyperparameters. Hypermaters such as number-of-estimators were set to 100 and the random state was set to 42 for the RandomForestClassifier sklearn module to find the best model performance.

D. Logistic Regression

Logistic regression [39] is a statistical model that can be used for classification tasks. It works by fitting a logistic function to the data, which allows it to predict the probability of a data point belonging to a particular class. Logistic regression is a simple and effective algorithm, and it is widely used in a variety of applications, including text classification. It is important to carefully select the features that will be used to represent the text documents and to address class imbalance in the training data.

The general equation for Logistic Regression is as follows:

$$P(c \mid d) = sigmoid(w^{\mathrm{T}}x + b)$$

where $P(c \mid d)$ is the probability of class c in a given document d, w is the weight vector, x is the feature vector, b is the bias term, $sigmoid()$ is the sigmoid function. The weight vector w and the bias term b are learned during training. The feature vector x is a representation of the document that is used to predict the class of the document. The sigmoid function is a non-linear function that maps the input to a value between 0 and 1. Hyperparameters such as a maximum number of iterations were set to 40224 for the solvers to converge for the LogisticRegression sklearn module to work finely.

Results and Discussions

We trained and tested our dataset in machine learning models namely Random Forest, Naive Bayes, Support Vector Machine, and Logistic Regression, and compared the result with our deep learning model and transformer BanglaBERT[1] and LSTM.

We conducted our experiments using Google Colab, a free Jupyter notebook environment that runs in the cloud. It provides a simple way to run Python code and supports a wide range of machine learning and data science libraries. The specifications of the hardware that we used are as follows:

- RAM: 13 GB
- CPU: Intel® Xeon® CPU @ 2.20GHz, with 2 virtual cores
- GPU: Tesla T4, with 16 GB of memory
- TPU: Google Cloud TPU, with 8 cores and 128 GB of memory

We used the GPU and TPU for training our deep learning models, as they offer faster and more efficient computation than the CPU.

Python 3.10 is used. We also used some popular Python libraries for our experiments, such as TensorFlow [40] and Hugging Face [35].

The results are as follows:

BanglaBERT: We used the pre-trained BanglaBERT[1] model in the embedding layer and LSTM [3], dense and dropout layers for the Binary Classification of Hate Speech data. Adam Optimizer and Binary Crossentropy were used as loss functions. The learning rate was set to 0.001, and number of epochs was set to 100 and the batch size was set to 128 for optimum results.

We were able to achieve an accuracy of 76.89%, weighted average precision of 71.07%, weighted average recall of 76.76%, and weighted average F1-score of 73.81%.

Random Forest Classifier: Similar to SVM we created vectors of each comment using the TF-IDF vectorizer and fed the created vectors to the Random Forest Classifier Model [38]. The number of estimators was set to 100. The model gave an accuracy score of 77.65%, weighted average precision of 75.16%, weighted average recall of 70.66%, and weighted average F1-Score of 72.84%.

Support Vector Machine (SVM): We created vectors of each comment using the Term Frequency Inverse Document Frequency (TF-IDF), and fed these vectors to the SVM [36] model. The training was done for 100 epochs. The penalty was set to l2, loss was set to hinge, and C=1. The model gave an accuracy score of 72.47%, weighted average precision of 73.39%, weighted average recall of 55.03%, and weighted average F1-score of 62.89%.

Logistic Regression: For experimentation purposes, we trained and tested two different models of Logistic Regression [39] with two different vectorization methods namely CountVectorization, which simply counts the frequency of each word in a document and creates a vector of these words, and TF-IDF vectorization, which is more advanced and takes into account the importance of each word in a document. The training was done for 100 iterations with C=1, and penalty=l2. Both of the models gave the same results, accuracy score of 72.37%, weighted average precision of 72.20%, and weighted average recall of 56.67%, and there was a slight difference in the F1 score, the model fed with count vectors gave weighted average F1 score of 62%, whereas the model fed with TF-IDF vectors gave weighted average F1 score of 63.50%.

Naive Bayes Classifier: Multinomial Naive Bayes Classifier [37] was used for TF-IDF vectorization. The model gave an accuracy score of 71.64%, with a weighted average precision score of 74.87%, a weighted average recall score of 49.88%, and a weighted average precision score of 59.87%.

The formulae used for calculating matrices:

- Accuracy: (TP + TN)/(TP+TN+FP+FN)
- Precision: TP/ (TP + FP)
- Recall: TP/(TP+FN)
- F1 Score: (2*Precision*Recall)/(Precision + Recall)
- Weighted Average Precision: (Support_0*Precission_0 + Support_1 * Precision_1) / (Support_0 + Support_1)
- Weighted Average Recall: (Support_0 * Recall_0+ Support_1 * Recall_1) / (Support_0 + Support_1)
- Weighted Average F1 score: (Support_0 * F1_0 + Support_1 * F1_1) / (Support_0 + Support_1)

Here, TP stands for True Positive, Instances that were positive and were classified positive by the model. TN stands for True Negative, instances that were negative and were classified as negative by the model. FP stands for False Positive, instances that were negative but were classified as positive by the model. FN stands for False Negative, instances that were positive but were classified as negative.

Table 6: Metrics for different classification models

Model	**Accuracy**	**Weighted Average Precision**	**Weighted Average Recall**	**Weighted Average F1-Score**
BanglaBERT + LSTM	76.89	76.76	71.07	73.81
Random Forest	77.65	75.16	70.66	72.84
Support Vector Machine	72.47	73.39	55.03	62.89
Logistic Regression	72.37	72.20	56.67	63.50
Naive Bayes	71.64	74.87	49.88	59.87

As we can see from Table 6, the BanglaBERT + LSTM classifier and Random Forest Classifier have outperformed the rest of the models namely: the Naive Bayes Classifier and Logistic Regression Classifier, Linear Support Vector Classifier. The highest accuracy that was achieved in the BanglaBERT + LSTM Classifier is 76.89%, and for Random Forest it is 77.65%. Though our proposed model has less accuracy than the Random Forest classifier, the precision, recall and f1-score are significantly higher than any other classifier. So we can say that the abusive speech detection problem can be solved using deep learning networks in a much better way than the traditional machine learning classifiers.

Conclusion

In this project, we have developed a deep learning model to detect toxic comments in Bengali from social media platforms. Any promising social media site must prioritize comment filtering to give its users a secure environment. Due to the rise in Bengali-speaking users on social media, there is currently research being done on filtering toxic comments in Bengali. Deep learning-based models replaced traditional machine learning techniques, which were previously used to detect toxicity, because of the latter's poor performance scaling with large datasets. Our model is based on an innovative approach that consists of dataset combination, model development, comparison with the baseline models, and model evaluation. We have collected and merged three datasets of Bengali comments and annotated them with binary labels for toxic and non-toxic. We have also preprocessed the data, extracted features, and experimented with different machine learning and deep learning models, such as SVM, LSTM, and BERT. We have evaluated the performance of our model using various metrics, such as accuracy, precision, recall, and F1-score. Our model has achieved an accuracy of 76.89%, and a recall of 71.07% on the test set, which is better than the baseline machine learning models. The model can identify and flag harmful utterances in online conversations and facilitate a safe and respectful online environment for Bengali speakers. The project also contributes to the research on abusive language detection in low-resource languages and provides a valuable resource for future studies.

Future Scope

The problem statement of detecting abusive Bengali toxic comments is a challenging and important task that has many avenues for further improvement and innovation. One possible direction is to develop a new embedding system that can better handle toxic comments in Bengali, as the current BanglaBERT [1] model has some limitations in tokenizing them. For instance, BanglaBERT [1] may split a word into smaller units that lose the original meaning or context, or it may fail to recognize a word that is spelt differently but has the same abusive connotation. A new BERT [2] model that is specifically trained on abusive Bengali text could overcome these issues and provide more accurate embeddings for the deep learning model.

Another possibility is to create a new database that distinguishes between Indian and Bangladesh-style Bengali, which could reduce the confusion for both the BanglaBERT [1] and the other deep learning models. The current dataset contains both styles of Bengali, which may have different spellings, pronunciations, and meanings for the same words. This could affect the performance of the models, as they may not be able to generalize well across different dialects. A new database that separates the two styles of Bengali could help the models learn the specific features and patterns of each dialect and improve their accuracy.

Finally, a multi-lingual model or a model that can deal with English-coded Bengali comments could also be explored. Some of the comments in the dataset contain English words or

phrases, either mixed with Bengali or written entirely in English. These comments may pose a challenge for the BanglaBERT [1] and the deep learning model, as they may not be able to understand the meaning or sentiment of the English words. A multi-lingual model that can process both Bengali and English could handle these comments better and capture the nuances of the language. Alternatively, a model that can translate the English-coded Bengali comments into pure Bengali could also be useful, as it could simplify the input for the models and make them more consistent.

References

[1] S. Sarker, "bangla-bert." Sep. 2020. Accessed: Nov. 03, 2023. [Online]. Available: https://github.com/sagorbrur/bangla-bert

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv, May 24, 2019. Accessed: Nov. 03, 2023. [Online]. Available: http://arxiv.org/abs/1810.04805

[3] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[4] M. Das, S. Banerjee, P. Saha, and A. Mukherjee, "Hate Speech and Offensive Language Detection in Bengali," arXiv, 2022. doi: 10.48550/ARXIV.2210.03479.

[5] M. S. Jahan, M. Haque, N. Arhab, and M. Oussalah, "BanglaHateBERT: BERT for Abusive Language Detection in Bengali," in *Proceedings of the Second International Workshop on Resources and Techniques for User Information in Abusive Language Analysis*, Marseille, France: European Language Resources Association, Jun. 2022, pp. 8–15. Accessed: Sep. 15, 2023. [Online]. Available: https://aclanthology.org/2022.restup-1.2

[6] N. Romim, M. Ahmed, H. Talukder, and M. S. Islam, "Hate Speech detection in the Bengali language: A dataset and its baseline evaluation." arXiv, Dec. 17, 2020. Accessed: Sep. 09, 2023. [Online]. Available: http://arxiv.org/abs/2012.09686

[7] S. Abro, S. Shaikh, Z. Hussain, Z. Ali, S. Khan, and G. Mujtaba, "Automatic Hate Speech Detection using Machine Learning: A Comparative Study," *IJACSA*, vol. 11, no. 8, 2020, doi: 10.14569/IJACSA.2020.0110861.

[8] F. T. Boishakhi, P. C. Shill, and Md. G. R. Alam, "Multi-modal Hate Speech Detection using Machine Learning," in *2021 IEEE International Conference on Big Data (Big Data)*, Orlando, FL, USA: IEEE, Dec. 2021, pp. 4496–4499. doi: 10.1109/BigData52589.2021.9671955.

[9] M. S. Jahan and M. Oussalah, "A systematic review of hate speech automatic detection using natural language processing," *Neurocomputing*, vol. 546, p. 126232, Aug. 2023, doi: 10.1016/j.neucom.2023.126232.

[10] A. Al-Hassan and H. Al-Dossari, "DETECTION OF HATE SPEECH IN SOCIAL NETWORKS: A SURVEY ON MULTILINGUAL CORPUS," in *Computer Science & Information Technology(CS & IT)*, AIRCC Publishing Corporation, Feb. 2019, pp. 83–100. doi: 10.5121/csit.2019.90208.

[11] S. K. Mohapatra, S. Prasad, D. K. Bebarta, T. K. Das, K. Srinivasan, and Y.-C. Hu, "Automatic Hate Speech Detection in English-Odia Code Mixed Social Media Data Using Machine Learning Techniques," *Applied Sciences*, vol. 11, no. 18, p. 8575, Sep. 2021, doi: 10.3390/app11188575.

[12] A. Omar, T. M. Mahmoud, and T. Abd-El-Hafeez, “Comparative Performance of Machine Learning and Deep Learning Algorithms for Arabic Hate Speech Detection in OSNs,” in *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)*, vol. 1153, A.-E. Hassanien, A. T. Azar, T. Gaber, D. Oliva, and F. M. Tolba, Eds., in Advances in Intelligent Systems and Computing, vol. 1153. , Cham: Springer International Publishing, 2020, pp. 247–257. doi: 10.1007/978-3-030-44289-7_24.

[13] S. Kamble and A. Joshi, “Hate Speech Detection from Code-mixed Hindi-English Tweets Using Deep Learning Models.” arXiv, Nov. 13, 2018. Accessed: Nov. 04, 2023. [Online]. Available: http://arxiv.org/abs/1811.05145

[14] H. Mubarak, S. Hassan, and S. A. Chowdhury, “Emojis as Anchors to Detect Arabic Offensive Language and Hate Speech.” arXiv, May 18, 2022. Accessed: Nov. 04, 2023. [Online]. Available: http://arxiv.org/abs/2201.06723

[15] T. Mandla *et al.*, “Overview of the HASOC track at FIRE 2020: Hate Speech and Offensive Content Identification in Indo-European Languages.” arXiv, Aug. 12, 2021. Accessed: Nov. 04, 2023. [Online]. Available: http://arxiv.org/abs/2108.05927

[16] Z. Zhang and L. Luo, “Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter.” arXiv, Oct. 25, 2018. Accessed: Nov. 08, 2023. [Online]. Available: http://arxiv.org/abs/1803.03662

[17] T. Davidson, D. Warmsley, M. Macy, and I. Weber, “Automated Hate Speech Detection and the Problem of Offensive Language.” arXiv, Mar. 11, 2017. Accessed: Nov. 10, 2023. [Online]. Available: http://arxiv.org/abs/1703.04009

[18] Mr. Mohiyaddeen and S. Siddiqi, “Automatic Hate Speech Detection: A Literature Review,” *SSRN Journal*, 2021, doi: 10.2139/ssrn.3887383.

[19] C. Vasantharajan and U. Thayasivam, “Towards Offensive Language Identification for Tamil Code-Mixed YouTube Comments and Posts,” *SN COMPUT. SCI.*, vol. 3, no. 1, p. 94, Jan. 2022, doi: 10.1007/s42979-021-00977-y.

[20] H. Rizwan, M. H. Shakeel, and A. Karim, “Hate-Speech and Offensive Language Detection in Roman Urdu,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, 2020, pp. 2512–2522. doi: 10.18653/v1/2020.emnlp-main.197.

[21] R. Rajalakshmi, S. Selvaraj, F. M. R., P. Vasudevan, and A. K. M., “HOTTEST: Hate and Offensive content identification in Tamil using Transformers and Enhanced STemming,” *Computer Speech & Language*, vol. 78, p. 101464, Mar. 2023, doi: 10.1016/j.csl.2022.101464.

[22] J. Deng *et al.*, “Enhancing Offensive Language Detection with Data Augmentation and Knowledge Distillation,” *Research*, vol. 6, p. 0189, Jan. 2023, doi: 10.34133/research.0189.

[23] B. Wei, J. Li, A. Gupta, H. Umair, A. Vovor, and N. Durzynski, “Offensive Language and Hate Speech Detection with Deep Learning and Transfer Learning.” arXiv, Aug. 22, 2021. Accessed: Nov. 13, 2023. [Online]. Available: http://arxiv.org/abs/2108.03305

[24] N. Romim, M. Ahmed, M. S. Islam, A. S. Sharma, H. Talukder, and M. R. Amin, “BD-SHS: A Benchmark Dataset for Learning to Detect Online Bangla Hate Speech in Different Social Contexts.” arXiv, Jun. 01, 2022. Accessed: Sep. 09, 2023. [Online]. Available: http://arxiv.org/abs/2206.00372

[25] N. Banik and M. Rahman, *Toxicity Detection on Bengali Social Media Comments using Supervised Models*. 2019. doi: 10.13140/RG.2.2.22214.01608.

[26] aimansnigdha, “Bangla-Abusive-Comment-Dataset.” Mar. 11, 2022. Accessed: Sep. 15, 2023. [Online]. Available: https://github.com/aimansnigdha/Bangla-Abusive-Comment-Dataset
[27] T. A. Belal, G. M. Shahariar, and M. H. Kabir, “Interpretable Multi Labeled Bengali Toxic Comments Classification using Deep Learning,” in *2023 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, Feb. 2023, pp. 1–6. doi: 10.1109/ECCE57851.2023.10101588.
[28] Y. Luan and S. Lin, “Research on Text Classification Based on CNN and LSTM,” Mar. 2019, pp. 352–355. doi: 10.1109/ICAICA.2019.8873454.
[29] M. R. Karim, B. R. Chakravarthi, J. P. McCrae, and M. Cochez, “Classification Benchmarks for Under-resourced Bengali Language based on Multichannel Convolutional-LSTM Network.” arXiv, Apr. 19, 2020. Accessed: Sep. 09, 2023. [Online]. Available: http://arxiv.org/abs/2004.07807
[30] M. F. Ahmed, Z. Mahmud, Z. T. Biash, A. A. N. Ryen, A. Hossain, and F. B. Ashraf, “Cyberbullying Detection Using Deep Neural Network from Social Media Comments in Bangla Language.” arXiv, Jun. 08, 2021. doi: 10.48550/arXiv.2106.04506.
[31] deepu099cse, “Multi_labeled_toxic_comments.” Aug. 21, 2023. Accessed: Nov. 07, 2023. [Online]. Available: https://github.com/deepu099cse/Multi-Labeled-Bengali-Toxic-Comments-Classification
[32] J. Cohen, “A Coefficient of Agreement for Nominal Scales,” *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, Apr. 1960, doi: 10.1177/001316446002000104.
[33] O. Sharif and M. M. Hoque, “Tackling cyber-aggression: Identification and fine-grained categorization of aggressive texts on social media using weighted ensemble of transformers,” *Neurocomputing*, vol. 490, pp. 462–481, Jun. 2022, doi: 10.1016/j.neucom.2021.12.022.
[34] B. C. N. Group, “normalizer.” Aug. 30, 2023. Accessed: Nov. 03, 2023. [Online]. Available: https://github.com/csebuetnlp/normalizer
[35] T. Wolf *et al.*, “HuggingFace’s Transformers: State-of-the-art Natural Language Processing.” arXiv, Jul. 13, 2020. Accessed: Nov. 07, 2023. [Online]. Available: http://arxiv.org/abs/1910.03771
[36] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach Learn*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.
[37] J. Ren, S. D. Lee, X. Chen, B. Kao, R. Cheng, and D. Cheung, “Naive Bayes Classification of Uncertain Data,” in *2009 Ninth IEEE International Conference on Data Mining*, Dec. 2009, pp. 944–949. doi: 10.1109/ICDM.2009.90.
[38] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Aug. 1995, pp. 278–282 vol.1. doi: 10.1109/ICDAR.1995.598994.
[39] D. R. Cox, “The Regression Analysis of Binary Sequences,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 20, no. 2, pp. 215–242, 1958.
[40] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems”.