PROJECT REPORT

ON

Multi-label Classification for Hate Speech Detection in Bengali and Hindi: A Comparative Analysis

Project II was Submitted in partial fulfillment 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

Dr. 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: “24/05/2024”, under the direct supervision and guidance of Dr. 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.

Dr. 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, Dr Susmita Das, for her guidance and support throughout the completion of our final year group project I on "Multi-label Classification for Hate Speech Detection in Bengali and Hindi: A Comparative Analysis". 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, Mouri Roy, Kingshuk Roy, Arnab Mukhopadhyay, Abir Mondal, and Ashutosh Kumar Jha

Date: 24/05/2024

Table of Contents

Introduction

The proliferation of social media platforms has democratized information sharing and fostered public discourse on various topics. Individuals of all demographics engage in online interactions, dedicating significant time to these virtual spaces. However, these platforms can also become breeding grounds for negativity, as evidenced by the prevalence of political bias, extreme religious viewpoints, and defamation. Disagreements frequently escalate, leading to the exchange of insults, threats, and offensive language.

To ensure a safe and inclusive online environment, it is imperative to remove these harmful comments. This task presents a significant challenge due to the sheer volume of information on social media platforms. Additionally, effective comment filtering necessitates a collaborative approach, leveraging both human expertise and automated systems.

While significant research efforts have been devoted to detecting toxic language in English, similar advancements lag for languages with limited resources. This is particularly true for prominent Indian languages like Hindi and Bengali. Despite the active participation of Hindi and Bengali users in generating social media content, filtering negative comments within these complex inflectional languages presents a significant obstacle.

Previously, comment filtering primarily relied on the manual creation of rules based on predefined lists of offensive words. However, the effectiveness of this approach has proven to be limited. Technological advancements have paved the way for automated systems utilizing Natural Language Processing (NLP) techniques. NLP leverages vast amounts of text data to train models for various tasks, including the identification of toxic comments.

Traditional machine learning models encounter difficulties in processing the unstructured nature of social media text. Deep learning models, however, exhibit the capacity to learn from such data and have demonstrated success in analyzing English language content. This report delves into the potential efficacy of deep learning models for detecting toxic comments within the Hindi and Bengali languages.

This report identifies a critical limitation in the field of comment detection for Hindi and Bengali languages. To address this knowledge gap, the report proposes a comprehensive analysis comparing nine transformer models. These models include XLM-RoBERTa, DistilBERT, TwHIN-BERT, MuRIL, BERT multilingual base (uncased and cased variants), HindBERT, RoBERTa Hindi, and BanglaBERT. Through this comparative study, the report strives to bridge the current knowledge gap and provide a practical solution for comment detection in these languages.

Firstly, we will delve into a rigorous hyperparameter tuning process. Transformer models contain internal settings known as hyperparameters that influence the learning process. By employing techniques like grid search or random search, we can fine-tune these hyperparameters for the most promising models identified during the initial evaluation. This fine-tuning has the potential to significantly improve the model's performance on the specific task of Hindi and Bengali comment detection.

Secondly, the project will explore the efficacy of ensemble learning techniques. Ensemble learning involves combining the predictions of multiple models to achieve a more robust and accurate outcome. In this context, we could potentially create an ensemble model specifically designed for Hindi and Bengali comment detection. This would involve combining the predictions from the top-performing models identified in the initial analysis. By leveraging the strengths of each model, the ensemble model could achieve superior performance compared to any single model.

Finally, to ensure the real-world applicability of our findings, the project will investigate the generalizability of the results. This will involve testing the performance of the best-performing models on unseen data sets from different domains relevant to comment detection. By evaluating how well the models perform on unseen data, we can gain valuable insights into their ability to effectively detect comments in Hindi and Bengali across various real-world scenarios.

Motivation

This report addresses a gap in current research by investigating toxic comment detection in Hindi and Bengali. These languages, while widely spoken, are considered low-resource in the field of Natural Language Processing (NLP), leading to a scarcity of existing methods for identifying harmful online content. The objective of this research is to develop automated systems capable of detecting toxic language, encompassing hate speech, defamation, and threats of violence. The prevalence of such online abuse necessitates robust detection methods to ensure a positive user experience and mitigate its negative impact on well-being.

The unstructured nature and prevalence of misspelled profanities render manual content filtering impractical. Machine learning approaches, especially those utilizing sentiment analysis, provide a more effective solution than conventional statistical methods. While such techniques are well-established in the English language, identifying harmful content in languages like Hindi and Bengali necessitates the development of specialized methodologies.

In light of the identified research gap, this report endeavors to address the limitations in Hindi and Bengali comment detection through a comparative analysis of nine transformer models (XLM-RoBERTa, DistilBERT, TwHIN-BERT, MuRIL, BERT multilingual base (uncased), BERT multilingual base (cased), HindBERT, RoBERTa Hindi, BanglaBERT). This report aspires to bridge the current knowledge gap and offer a practical solution to this challenge by providing a comparative study of different transformers. The proposed approach is meticulously structured around three core components: dataset design, model development, and model evaluation. Each element will be comprehensively justified based on the specific problem domain, a thorough review of existing literature, and the inherent characteristics of the data employed. Furthermore, the project will meticulously address the research questions or hypotheses established in the introduction. A detailed exposition of the project methodology is provided in the subsequent sections of this report.

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 and Hindi, which are widely used Indic languages. In our model, natural language processing and deep learning have been used for data modification, feature engineering, selection, training and evaluation. The Bengali and Hindi text strings are considered abusive or toxic if they contain any form of offensive words, defaming, violence-provoking, vulgar or hate speech. 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

The paper [1] proposes an explainable approach for detecting hate speech in the Bengali language, which is under-resourced in computational resources for natural language processing (NLP). Their method, DeepHateExplainer, employs a neural ensemble of transformer-based architectures like monolingual Bangla BERT-base, multilingual BERT-cased and uncased, and XLM-RoBERTa. It includes comprehensive preprocessing of Bengali texts and classifies them into political, personal, geopolitical, and religious hate. The approach also uses sensitivity analysis and layer-wise relevance propagation (LRP) for providing human-interpretable explanations. The evaluation of DeepHateExplainer was conducted using several machine learning and deep neural network baselines, with the results showing F1 scores of 84%, 90%, 88%, and 88% for political, personal, geopolitical, and religious hates, respectively, during 3-fold cross-validation tests. These results demonstrate the effectiveness of the proposed approach in accurately detecting different types of hate speech in Bengali texts.

This study[2], which includes ~ 44,000 comments” collected for analysis. The dataset is designed to identify bullying expressions and categorize the extent of inappropriateness using Natural Language Processing. All the comments have been labeled with various harassment categories by experts and the dataset is available for download and is licensed under CC BY 4.0 This dataset is a valuable resource for researchers and developers working in the fields of sentiment analysis and online behavior monitoring.

This research [3] introduces a novel dataset for detecting hostility in Hindi-language social media posts. The dataset contains 8,200 manually annotated posts, categorized into fake news, hate speech, offensive, defamation, and non-hostile labels. The authors highlight the importance of detecting hostile content in low-resource languages like Hindi, especially given the rise in online hate speech. They also discuss the challenges in distinguishing between different types of hostility and their overlap. The paper presents baseline systems for benchmarking the dataset and evaluates them using traditional machine learning algorithms like SVM, Decision Tree, Random Forest, and Logistic Regression. The results show that SVM performed best in coarse-grained evaluation with an F1 score of 84.11%, while Logistic Regression excelled in detecting fake news with an F1 score of 68.15%. The study emphasizes the need for early detection of hostile posts to mitigate their harmful impact on society.

The paper [4] focuses on classifying Bengali toxic comments using a deep learning-based pipeline. Initially, a binary classification model determines if a comment is toxic, followed by a multi-label classifier to identify the type of toxicity. The dataset comprises 16,073 instances, with 8,488 labeled as toxic across six categories: vulgar, hate, religious, threat, troll, and insult. The study utilizes Long Short Term Memory (LSTM) with BERT Embedding for binary classification and a combination of Convolutional Neural Network and Bi-directional LSTM (CNN-BiLSTM) with an attention mechanism for multi-label classification. The binary classification achieved an accuracy of 89.42%, while the multi-label classifier attained 78.92% accuracy and a weighted F1 score of 0.86. The Local Interpretable Model-Agnostic Explanations (LIME) framework was used to interpret the predictions and understand word feature importance during classification. The paper also highlights the challenges of existing datasets and proposes improvements for future research. The authors have made their dataset publicly available for further study.

The study [5] discusses the advances in natural language processing (NLP) driven by Transformer architectures and model pretraining. They introduce the open-source library 'Transformers' which provides state-of-the-art Transformer architectures with a unified API. The library includes a curated collection of pre-trained models for the community, designed to be extensible, simple for practitioners, and robust for industrial deployments. The paper highlights the Transformer's scalability with training data and model size, its efficient parallel training, and its ability to capture long-range sequence features. Datasets mentioned include GLUE, SST, and MNLI for classification tasks, SQuAD and NaturalQuestions for QA, and WMT, and IWSLT for translation tasks. The 'Transformers' library has facilitated the distribution of pre-trained models, making it easier for the machine learning community to access and utilize these models for a wide variety of tasks. The library supports industrial-strength implementations of popular model variants and allows for easy switching between models and frameworks. The authors also discuss the community model hub, which contains over 2,000 user models, both pre-trained and fine-tuned, demonstrating the library's significant impact and adoption within the NLP community.

The paper [6] proposes a novel network architecture called the Transformer, which is based solely on attention mechanisms, eliminating the need for recurrence and convolutions. This architecture is designed for sequence transduction tasks such as machine translation. The Transformer is more parallelizable and requires significantly less time to train compared to models based on complex recurrent or convolutional neural networks. In their experiments on machine translation tasks, the Transformer achieved a BLEU score of 28.4 on the WMT 2014 English-to-German translation task and 41.8 on the English-to-French translation task, setting new state-of-the-art results. The model was trained on datasets like Penn Treebank, CNN/Daily Mail, and WMT 2014, among others. The paper also demonstrates the Transformer's generalizability by applying it successfully to English constituency parsing.

This paper [7] introduces BERT, which stands for Bidirectional Encoder Representations from Transformers, which is designed to pre-train deep bidirectional representations from unlabeled text by conditioning on both left and right context in all layers. The paper discusses the use of the BooksCorpus and English Wikipedia as datasets for pre-training, involving tasks like Masked LM (Language Model) and Next Sentence Prediction (NSP). The approach is distinctive for its use of a "masked language model" pre-training objective, enabling the representation to integrate context from both directions and a "next sentence prediction" task that jointly pre-trains text-pair representations. BERT achieved state-of-the-art results on eleven natural language processing tasks, including significant improvements on the GLUE score, MultiNLI accuracy, and SQuAD v1.1 and v2.0 Test F1 scores. The results demonstrate the effectiveness of BERT's bidirectional pre-training and its ability to fine-tune with minimal task-specific architecture modifications for a wide range of language understanding tasks. The paper emphasizes the simplicity of the BERT model and its empirical power, marking a significant advancement in the field of NLP.

This paper [8] proposes a method for pre-training a smaller language representation model, DistilBERT, which retains most of the language understanding capabilities of BERT but is more resource-efficient. They introduce a triple loss function that combines language modeling, distillation, and cosine-distance losses to leverage the inductive biases learned by larger models during pre-training. DistilBERT is 40% smaller and 60% faster than its predecessor, BERT, while preserving 97% of its language understanding capabilities. This makes it more cost-effective for pre-training and suitable for on-device computations, as demonstrated in a proof-of-concept experiment and a comparative on-device study.

This study [9] presents a large-scale approach to pretraining multilingual language models, specifically a Transformer-based masked language model trained on over two terabytes of CommonCrawl data across one hundred languages. The model, known as XLM-R, aims to improve cross-lingual transfer task performance. XLM-R outperforms the multilingual BERT (mBERT) on various benchmarks, showing significant gains, especially in low-resource languages. For instance, it improved XNLI accuracy by 15.7% for Swahili and 11.4% for Urdu. The paper also explores the trade-offs between positive transfer, capacity dilution, and the performance of high and low-resource languages, demonstrating that multilingual modeling can be achieved without sacrificing per-language performance. The authors have made their code, data, and models publicly available for further research and development

This research [10] introduces TwHIN-BERT. TwHIN-BERT is a novel multilingual language model developed for Twitter, trained on 7 billion tweets in over 100 languages. It stands out because it incorporates a social objective based on Twitter's heterogeneous information network (TwHIN), alongside the usual text-based self-supervision. This approach allows the model to better handle the short, noisy text typical of social media. The model demonstrates significant improvements in multilingual social recommendation and semantic understanding tasks over existing pre-trained language models. The authors have open-sourced TwHIN-BERT and provided curated hashtag prediction and social engagement benchmark datasets to the research community. However, the paper does not specify the datasets used, suggesting that they may be proprietary or not publicly disclosed.

Methods and Algorithms

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

I. Data Collection

The text samples were collected from two sources. Both datasets were multi-labelled, but the classification labels were different. So we decided to reclassify the data into five labels - defamation, hate, non-hate, violence, and vulgar. 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. Bengali Dataset

The Bengali Dataset is a compilation of three individual datasets that were merged. There is one multi-label abusive comment dataset: *Bangla-Abusive-Comment-Dataset*[1] and two multi-class hate speech and cyberbullying datasets: *Bengali Hate Speech Dataset* [1], *Bangla Online Comments Dataset* [2] publicly available. We have collected the text samples from these three datasets and categorized them manually into six classes: vulgar, hate, religious, threat, troll, and insult where each sample text can be in multiple labels. For annotation purposes, we took help from three expert annotators and on the annotator's judgments for each class, we used the majority vote. There are in total 16,073 instances in the dataset. Among them, 7,585 instances are Non-toxic and 8,488 instances are Toxic.

Table 1: Dataset Statistics of the toxic comments for multi-label classification

Class	**Number of Instances**
vulgar	2505
hate	1898
religious	1418
threat	1419
troll	1643
insult	2719

B. Hindi Dataset

Hostility Detection Dataset in Hindi [3] - collected and manually annotated ~ 8200 online posts. The annotated dataset covers four hostility dimensions: fake news, hate speech, offensive, and defamation posts, along with a non-hostile label. The hostile posts are also considered for multi-label tags due to a significant overlap among the hostile classes. This dataset was used in the CONSTRAINT-2021 shared task on hostile post-detection.

[1] https://github.com/aimansnigdha/Bangla-Abusive-Comment-Dataset

Table 2: Dataset Statistics and Label Distribution. (* denotes total hostile posts.)

	Fake	Hate	Offense	Defame	Total*	Non-hostile
Train	1144	792	742	564	2678	3050
Validation	160	103	110	77	376	435
Test	334	237	219	169	780	873
Overall	1638	1132	1071	810	3834	4358

II. Dataset Preparation

A. Data Modification

The Bengali dataset [4] has six labels. They are *vulgar*, *hate*, *religious*, *threat*, and *insult*. To refine the classification scheme, a data pre-processing step was conducted. In this step, the labels "insult" and "troll" were merged to create a new label termed "defamation." Similarly, the labels "religious content" and "threats" were combined to form a new label category "violence." Following this consolidation, a new column titled "non-hate" was introduced. This column was populated with zeros initially. Subsequently, data points classified under any of the pre-existing labels (vulgar, defamation, or violence) were assigned a value of 1 in the "non-hate" column. Conversely, data points not categorized under any of the existing labels remained assigned a value of 0 in the "non-hate" column. This addition facilitates the identification of neutral content within the dataset.

The Hindi dataset [2] employed a five-point label system for categorizing text: defamation, fake, hate, non-hostile, and offensive. The initial processing steps involved transforming the data into a more suitable format to facilitate the machine learning model. *Label Binarization* where original text labels were converted into a binary representation. This conversion process is essential for specific machine learning algorithms that require a binary classification task. *Column Renaming* where the column containing the actual text data was renamed from "Post" to "text" for clarity and consistency within the dataset. *Label Consolidation* to achieve a more balanced label distribution and potentially reduce model complexity, the labels "vulgar" and "non-hate" were merged into a single category named "offensive" and "non-hostile," respectively. *Data Augmentation* since the dataset lacked a specific label for "violence," a new column named "violence" was added and initialized with blank values. This step serves as a placeholder for potential future data collection or labeling efforts that might incorporate violence classification.

B. Data Combination

The two datasets [4] and [3] were merged. However, these datasets were maintained in separate files to facilitate ease of use and experimentation within the context of language-specific transformers. This approach allows for independent testing and analysis of the datasets' suitability for the chosen deep-learning models.

Table 3: Merged Dataset Records

Total Comments	24162
Defamation	4962
Hate	3034
Non-hate	11943
Violence	2598
Vulgar	3569

C. Data Split

The dataset[2] 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 4: Dataset Split details

Train Set	Test Set	Validation Set
16889 (70.00%)	4856 (20.00%)	2417 (10.00%)

[2] https://www.kaggle.com/datasets/abirmondal/modified-hate-speech-bengali-hindi.

D. Custom Dataset

A dataset scraped from social media platforms can be a treasure trove of information. To combat online hate speech, we have created our custom dataset by web scraping comments from Youtube and used the Twitter API to collect tweets. This dataset focuses on text content, aiming to capture hateful language. It might include text from tweets and comments, along with video data from YouTube. Text data could be the content of tweets, including usernames and hashtags, or comments left on YouTube videos. They would include the comments themselves, along with usernames (if available) and timestamps. Scraped data might also contain replies to comments, which can provide context for identifying hate speech. This collection of text would then be manually labeled as containing hate speech or not, allowing us to develop algorithms for automatic detection and testing our BERT models to ascertain their applicability in real-world scenarios.

Table 5: Manually annotated Bengali dataset details

Total Comments	6253
Defamation	1337
Hate	1093
Non-hate	1975
Violence	701
Vulgar	1147

Table 6: Manually annotated Hindi dataset details

Total Comments	12209
Defamation	4538
Hate	5502
Non-hate	4520
Violence	4548
Vulgar	3190

III. Model Implementation

This section details the methodological framework developed for this project. The approach consists of two key components: model implementation and model evaluation. Each component will be examined in depth, with a thorough justification provided for the decisions and choices made throughout the project's development.

A. Model Pipeline

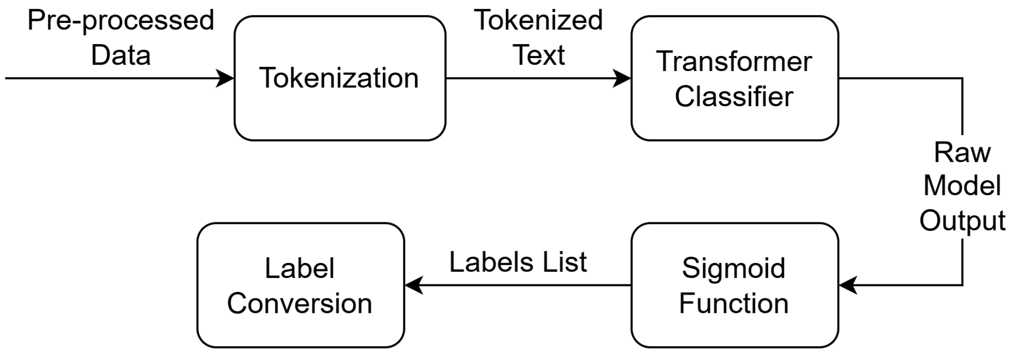

Figure 1: Model Pipeline

The model pipeline consists of four parts - *the tokenization*, *the transformer classifier*, *the sigmoid function*, and the *label conversion*. All these layers will be discussed in the following sections.

Figure 1 shows the model pipeline. First, the pre-processed data is tokenized. The transformer classifier classifies the text data into a set of 5 probabilities. These probabilities are passed on to the sigmoid function. This layer converts the probabilities into labels based on a threshold value. Finally, the label for the text helps to convert the binary label into text labels.

B. Tokenization

Neural networks and transformers, while powerful tools for natural language processing, do not inherently understand the meaning of text data. To facilitate processing, raw text undergoes a two-step preprocessing stage. First, the text is segmented into smaller units such as words, phrases, or characters, known as tokens. This process is referred to as *tokenization*. Subsequently, these tokens are converted into numerical representations, typically vectors, using a process called *embedding*. These embeddings capture the semantic and syntactic properties of the tokens, allowing the neural network or transformer to process and analyze the text data.

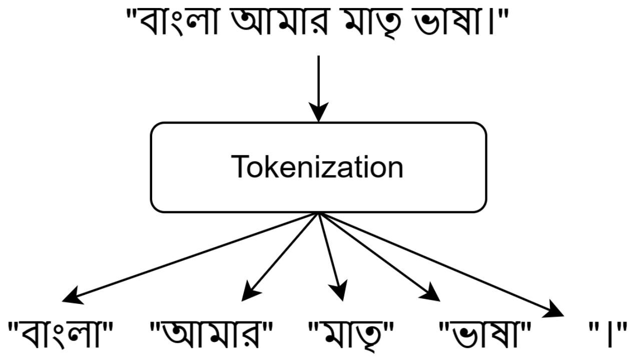

Figure 2: Example of Tokenization

We have used *fill-mask* models from the Hugging Face library [5]. Those models have an in-built tokenizer that can be accessed by the *AutoTokenizer* function from the *transformers* class.

C. Transformer Classifier

Transformers are a type of model architecture in Natural Language Processing (NLP), introduced in 2017 [6]. They use a self-attention mechanism to weigh the importance of words in an input sequence. This includes steps like input embedding, positional encoding, self-attention, feed-forward neural networks, and output generation.

In text classification tasks, Transformers have shown high effectiveness. They capture long-range dependencies in text, making them adept at understanding context and language nuances. This leads to superior performance on tasks like sentiment analysis, topic classification, and spam detection.

Transformer-based models like BERT, GPT, and RoBERTa have achieved state-of-the-art results on a wide range of NLP tasks, including text classification. However, they require significant computational resources and large amounts of training data. Despite this, their ability to understand context and handle complex language structures makes them a strong choice for many text classification tasks.

For text classification in our problem, we have used *fill-mask* transformers to train the models on our data. The *Fill-Mask* Transformer model is a specialized form of language model that is designed to predict missing words or phrases within a given sentence. It leverages the power of Transformer architecture to understand the context of a sentence and accurately predict the most probable word or phrase that should occupy the blank space. This model finds utility in a variety of tasks including but not limited to text completion, text generation, and language understanding.

We used the Hugging Face [5] transformers library to fetch the pre-trained fill-mask models and train them on our data. Hugging Face is an open-source organization that provides a plethora of tools and pre-trained models for Natural Language Processing (NLP), including a wide array of Transformer models. It offers a comprehensive platform for the training, fine-tuning, and deployment of these models. Its library, colloquially known as *transformers*, houses thousands of pre-trained models in over 100 languages. This makes Hugging Face an invaluable resource for a multitude of NLP tasks such as text classification, sentiment analysis, and more. It is noteworthy to mention that the Fill-Mask Transformer model is readily available in the Hugging Face library.

We have used a total of 9 fill-mask transformer models. Some are multi-lingual and some are language-specific models. All the details are discussed below.

i. *BERT-Base-Multilingual-Uncased*

The BERT-base-multilingual-uncased model is a pre-trained variant of the *Bidirectional Encoder Representations from Transformers* (BERT) [7] model. It has been trained on the top 102 languages with the largest Wikipedias by Google. This model does not differentiate between upper and lower case letters, hence the term '*uncased*'.

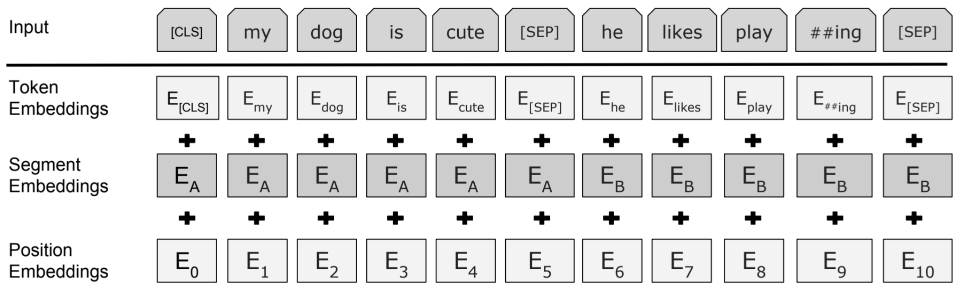
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 3: BERT input representation. [7]

The model is pre-trained using two main objectives:

1. <u>Masked Language Modeling (MLM)</u>: In this, 15% of the words in the input are randomly masked, and the model predicts these masked words based on the context provided by the unmasked words.
2. <u>Next Sentence Prediction (NSP)</u>: The model is trained to predict whether two sentences were consecutive in the original text.

The BERT-base-multilingual-uncased model is designed to be fine-tuned on a downstream task. It excels in tasks that require understanding the full context of a sentence or a paragraph, such as sequence classification, token classification, or question answering.

This model, along with a wide array of other Transformer models, is available in the Hugging Face [5] transformers library. The library is an open-source provider of NLP tools and pre-trained models, making it a valuable resource for many NLP tasks.

ii. *BERT-Base-Multilingual-Cased*

This model is analogous to the *BERT-Base-Multilingual-Uncased* model, trained by Google on a comparable dataset. The sole distinction lies in its case-sensitivity, meaning it differentiates between uppercase and lowercase letters. This case-sensitivity can potentially lead to improved performance in downstream tasks like sentiment analysis or machine translation, particularly when working with languages that rely heavily on case differentiation.

iii. *DistilBERT-Base-Multilingual-Cased*

The *DistilBERT-Base-Multilingual-Cased* model is a distilled [8] version of the BERT base multilingual model [7]. The model is cased, meaning it does differentiate between uppercase and lowercase texts, which means the model is case-sensitive.

The key difference between DistilBERT and BERT lies in the number of layers, hidden units, and attention heads. DistilBERT has fewer layers, hidden units, and attention heads compared to BERT, making it faster and more efficient, but with slightly lower performance. DistilBERT is significantly faster than BERT, both in terms of training and inference. It has 40% fewer parameters than BERT, which results in faster training times and reduced memory requirements. In terms of inference speed, DistilBERT can be up to 60% faster than BERT, depending on the specific task and hardware used.

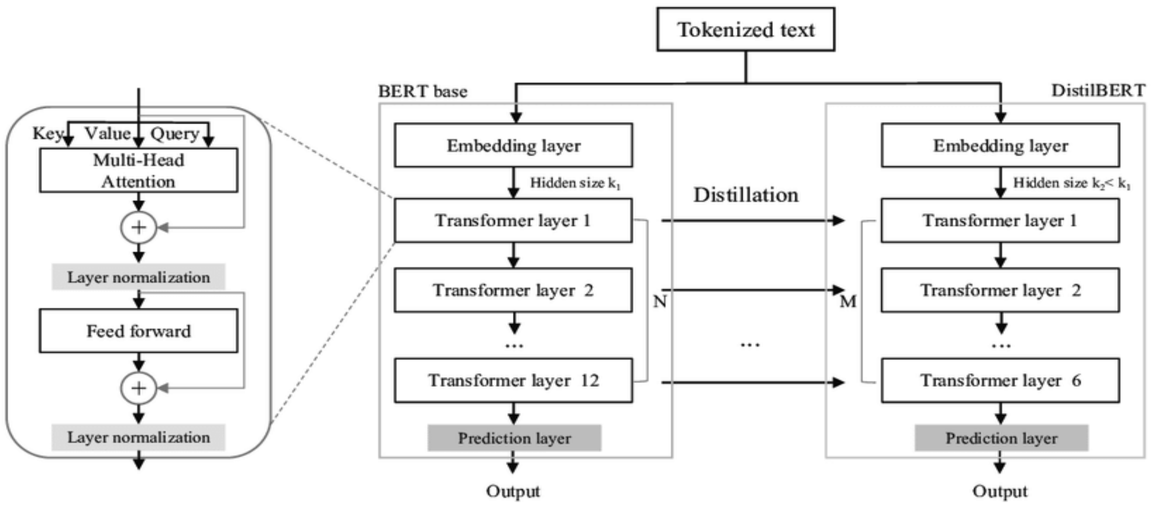

Figure 4: DistilBERT Architecture. [11]

The training process of DistilBERT involves a technique called knowledge distillation. In this process, the outputs from a large, pre-trained BERT model are used to train a smaller, faster, and more compact model. The large pre-trained model acts as a teacher model and the smaller, distilled model acts as a student model. The model was pretrained with the supervision of bert-base-multilingual-cased on the concatenation of Wikipedia in 104 different languages.

iv. *XLM-RoBERTa-Base*

The XLM-RoBERTa-Base model, often referred to as XLM-R, is a multilingual version of the RoBERTa model [9]. It was introduced by Facebook/Meta and is trained on 2.5TB of filtered CommonCrawl data across 100 languages.

XLM-R differs from BERT in several ways. While both models are transformer-based and use a masked language modeling objective, XLM-R employs dynamic masking during training. This means that the masking pattern is generated every time a sequence is fed to the model, allowing for potentially unlimited variations of each sentence. In contrast, BERT uses static masking, where each training sequence is seen with the same mask multiple times during training.

The training protocol for XLM-R leverages a self-supervised pre-training approach on unlabeled text corpora. In this paradigm, human annotation is entirely forgotten. Instead, the model extracts inputs and labels from the text data through an automated process. Subsequently, XLM-R is primarily designed for fine-tuning specific downstream tasks.

v. *TwHIN-BERT-Base*

The TwHIN-BERT-base model [10], developed by Twitter, is a novel multilingual language model trained on 7 billion tweets across over 100 distinct languages. It is designed to model short, noisy, user-generated text.

TwHIN-BERT-base differs from BERT in its training process. While BERT uses text-based self-supervision (Masked Language Modeling), TwHIN-BERT-base incorporates a social objective based on the rich social engagements within a Twitter Heterogeneous Information Network (TwHIN). This allows it to not only outperform similar models in semantic understanding tasks such as text classification but also in social recommendation tasks such as predicting user-to-tweet engagement.

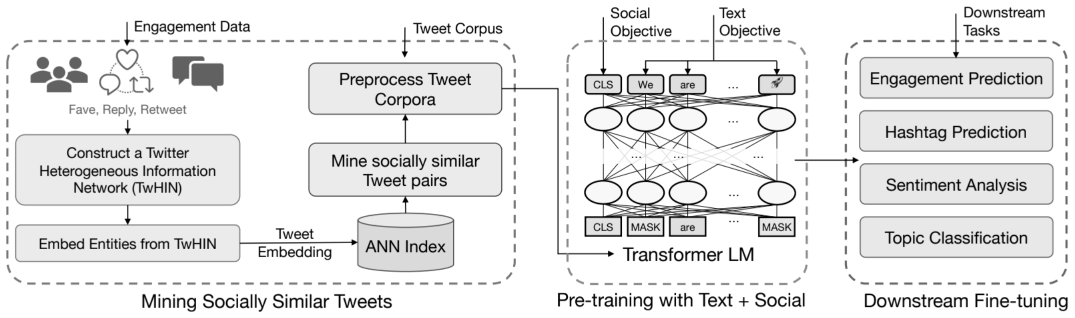

Figure 4: End-to-end TwHIN-BERT process. [10]

The training process involves mining socially similar tweet pairs by embedding a Twitter Heterogeneous Information Network, training TwHIN-BERT using a joint social and MLM objective, and finally fine-tuning TwHIN-BERT on downstream tasks.

vi. *MuRIL-Base-Cased*

The MuRIL-base-cased model [12], developed by Google, is a BERT model pre-trained on 17 Indian languages and their transliterated counterparts. This model is cased, meaning it does differentiate between upper and lower case letters.

MuRIL-base-cased differs from BERT in its training process. While BERT uses text-based self-supervision (Masked Language Modeling), MuRIL-base-cased incorporates translation and transliteration segment pairs in training. This allows it to outperform similar models in semantic understanding tasks such as text classification.

The training process involves pre-training the model on monolingual segments as well as parallel segments1. Monolingual data is obtained from publicly available corpora from Wikipedia and Common Crawl for 17 Indian languages. Parallel data includes translated data obtained using the Google NMT pipeline and transliterated data obtained using the IndicTrans library. The model was trained using a self-supervised masked language modeling task.

vii. *Bangla-bert-base*

The *sagorsarker/bangla-bert-base* model [13], developed by Sagor Sarker, is a pre-trained language model for the Bengali language. It employs the masked language modeling technique described in BERT.

The model was trained on two main sources:

1. The Bengali common crawl corpus downloaded from OSCAR [14]
2. The Bengali Wikipedia Dump Dataset[3]

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

The data was preprocessed into BERT format, which is one sentence per line and an extra new line for a new document. The BNLP package was used for training the Bengali sentence-piece model with a vocab size of 1,02,025. The output vocab file was then preprocessed into a BERT format.

The model was trained using the code provided in Google BERT's GitHub repository1. The released model follows the bert-base-uncased model architecture (12-layer, 768-hidden, 12-heads, 110M parameters). The total training steps were 1 million. The model was trained on a single Google Cloud GPU.

viii. *Hindi-bert-v2*

The l3cube-pune/hindi-bert-v2 model [15] [16], also known as HindBERT, is a pre-trained language model for the Hindi language. It was developed by L3Cube and is based on the multilingual BERT (google/muril-base-cased) model.

HindBERT was fine-tuned on publicly available Hindi monolingual datasets. The specifics of these datasets are not explicitly mentioned, but they are likely to include a variety of sources to capture the diverse linguistic characteristics of the Hindi language.

The training process involved fine-tuning the base BERT model on the Hindi datasets. Fine-tuning is a process where a pre-trained model is further trained on a specific task or dataset, allowing it to adapt its knowledge to the specific characteristics of that task or dataset.

ix. *RoBERTa-hindi*

The flax-community/roberta-hindi model, also known as RoBERTa Hindi, is a transformer model pre-trained on a large corpus of Hindi data. This model was developed as part of the Flax/Jax Community Week, organized by Hugging Face.

The model was trained on a combination of several datasets:

1. OSCAR [14]: A large multilingual corpus obtained by language classification and filtering of the Common Crawl corpus.
2. mC4: A colossal, cleaned version of Common Crawl's web crawl corpus.
3. IndicGLUE: A natural language understanding benchmark.
4. Samanantar: A parallel corpora collection for Indic languages.
5. Hindi Text Short and Large Summarization Corpus: A collection of ~180k articles with their headlines and summaries collected from Hindi News Websites.
6. Hindi Text Short Summarization Corpus: A collection of ~330k articles with their headlines collected from Hindi News Websites.
7. Old Newspapers Hindi: A cleaned subset of HC Corpora newspapers.

The training process involved tokenizing the texts using a byte version of Byte-Pair Encoding (BPE) with a vocabulary size of 5,02,651. The inputs of the model take pieces of 512 contiguous tokens that may span over documents. The model was trained on a Google Cloud Engine TPUv3-8 machine1. The details of the masking procedure for each sentence are as follows: 15% of the tokens are masked, and in 80% of the cases, the masked tokens are replaced by <*mask*>.

D. Sigmoid Function

The output predictions generated by the transformer model are subsequently directed to a Sigmoid activation layer. This layer leverages the PyTorch implementation of the Sigmoid function.

The Sigmoid function, alternatively referred to as the logistic function, is a mathematical construct that transforms any real-valued input into a corresponding value constrained within the range of 0 and 1. It finds extensive application in the domain of machine learning, particularly in the context of multi-label classification tasks. The function serves the purpose of converting the model's raw output into a set of probabilities, which can then be interpreted as representing the likelihood of membership in each respective class.

The sigmoid function is defined as follows:

$$f(x) = \frac{1}{1+e^{-x}}$$

We denote the input to the function as x, and e represents the base of the natural logarithm. The sigmoid function produces a real number constrained within the range of 0 and 1. As the input x approaches negative infinity, the output of the sigmoid function asymptotically approaches 0. Conversely, as x approaches positive infinity, the output asymptotically approaches 1. The special case where x equals 0 results in an output of 0.5.

In PyTorch, the sigmoid function is implemented as a tensor operation applied element-wise. This function is particularly useful for transforming the raw output of a model into a probabilistic interpretation. By employing this function, we obtain a list of probabilities corresponding to predefined labels.

E. Label Conversion

The output of the Sigmoid layer is a probability vector containing five elements. These elements represent the likelihood of the input belonging to each of the five possible classes. The Sigmoid function transforms the raw activations from the preceding layers of the model into this probabilistic interpretation.

A threshold value is employed to convert these probabilities into binary labels for classification purposes. Any element in the vector exceeding the threshold is assigned a label of "1", indicating classification into the corresponding class. Conversely, elements falling below the threshold are assigned a label of "0". For our project, we have kept the threshold value at 0.5.

Results and Discussions

We trained and tested our dataset in deep learning and transformer models XLM-RoBERTa, DistilBERT, TwHIN-BERT, MuRIL, BERT multilingual base uncased, BERT multilingual base cased with language-specific pre-trained models namely HindBERT, RoBERTa Hindi and BanglaBERT

We conducted our experiments on Kaggle, a free Jupyter Notebook environment that runs in the cloud. This environment facilitates the execution of Python code and provides extensive support for a multitude of machine learning and data science libraries. The specific hardware configurations employed will be outlined in the following section.

- RAM: 29 GB
- CPU: Intel® Xeon® CPU @ 2.20GHz, with 2 virtual cores
- GPU: NVIDIA Tesla P100, with 16 GB of memory
- TPU: Google Cloud TPU v3-8, 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 [17] and Hugging Face [5].

Accuracy in machine learning refers to how often a model makes correct predictions. It's calculated by dividing the number of correct predictions by the total number of predictions. A higher accuracy is generally desirable, signifying a model that performs well.

The F1 score is a metric used for evaluating the performance of classification models. It addresses the limitations of relying solely on accuracy, especially in imbalanced datasets. F1 score combines two key metrics: precision and recall, into a single value between 0 and 1. Precision reflects the proportion of true positives among predicted positives, while recall indicates the ratio of true positives captured by the model. By considering both these aspects, the F1 score provides a more balanced assessment of a model's effectiveness. A high F1 score signifies a good balance between precision and recall.

ROC AUC (Area Under the Receiver Operating Characteristic Curve) is a popular metric used to evaluate the performance of classification models. The ROC curve itself plots the model's ability to distinguish between positive and negative classes across various classification thresholds. AUC summarizes this ROC curve by calculating the total area underneath it. A higher AUC indicates a better model. An AUC of 1 represents a perfect classifier, while an AUC of 0.5 is equivalent to random guessing. AUC is particularly useful because it's independent of the chosen classification threshold, making it a robust metric for assessing a model's overall ability to separate the classes.

In machine learning, Hamming loss is a metric used to evaluate the performance of multi-label classification models. Unlike binary or single-label classification, multi-label problems involve data points belonging to multiple classes simultaneously. It measures the average proportion of labels that are incorrectly predicted by the model. It essentially penalizes models for mistakes in individual labels, providing a more nuanced evaluation compared to metrics like zero-one loss, which considers the entire set of labels for instance.

The Jaccard score, also known as the Jaccard similarity coefficient, is a metric used to assess the similarity between two sets of data. It's particularly useful for evaluating classification tasks where the outputs are represented as sets of labels. The Jaccard score focuses on the proportion of correctly identified items. It achieves this by dividing the number of elements that both the predicted set and the ground truth set have in common (intersection) by the total number of elements in their combined set (union). Scores range from 0 to 1, with 1 indicating perfect similarity and 0 signifying no overlap. The Jaccard score is a valuable tool for tasks like image segmentation, text analysis, and recommendation systems where evaluating set-based predictions is crucial.

The zero-one loss is a fundamental metric for evaluating classification tasks. It provides a simple measure of how many predictions a model makes incorrectly. For each data point, the loss is zero if the predicted class matches the actual class label. Conversely, if there's a mismatch, the loss becomes one. By summing the losses across all data points (and often normalizing by the total number of points), we obtain the overall classification error rate, which complements accuracy (the percentage of correct predictions). While intuitive, the zero-one loss has limitations. It doesn't consider the severity of incorrect predictions and isn't differentiable, hindering its use in certain optimization algorithms. Nevertheless, it remains a valuable tool for understanding a model's basic performance in classification problems.

The formulae used for calculating matrices:

- Accuracy: (TP + TN)/(TP + TN + FP + FN)
- F1-Score: (TP)/(TP + FP)
- TPR: (TP)/(TP + FN)
- FPR: (FP)/(FP + TN)
- ROC - AUC: TPR vs FPR
- Hamming Loss: $(1 / T) * \Sigma (y_i \text{ XOR } \hat{y}_i)$
- Jaccard Score: $|Y \cap \hat{Y}| / |Y \cup \hat{Y}|$
- Zero-One Loss: $L(y, \hat{y}) = 1 - I(y, \hat{y})$

Within this context, True Positive (TP) refers to instances that were genuinely positive and were correctly classified as positive by the model. Conversely, True Negative (TN) signifies instances that were inherently negative and were accurately classified as such by the model. It is also important to consider False Positive (FP) instances, which represent negative examples mistakenly classified as positive by the model. Additionally, False Negative (FN) instances represent positive examples that were incorrectly classified as negative.

The total number of labels or features is denoted by T. The true label set is denoted by Y, where y_i represents the ith element of the true label vector. Similarly, the predicted label set is denoted by $\hat{Y}$, where $\hat{y}_i$ represents the ith element of the predicted label vector. The indicator function, $I(y, \hat{y})$, is employed in this context. This function assigns a value of 1 if the predictions ($\hat{y}$) perfectly match the true labels (y) for all labels within a given data point. Conversely, it assigns a value of 0 if there is any discrepancy between the predicted and true labels.

In the following table, we have shown the performance of the transformer models tested specifically on the low-resource Bengali dataset only. Bengali, the seventh most spoken language globally, presents unique challenges for natural language processing (NLP) tasks due to its complex script and rich morphology. This study aims to evaluate and compare the effectiveness of different models in handling Bengali text data.

Table 5: Metrics for different transformer models for the Bengali language

Model	Accuracy	F1-Score	ROC AUC	Hamming Loss	Jaccard Score	Zero-One Loss
twhin-bert-base	**74.712**	**81.889**	**0.882**	**0.084**	**0.693**	**0.253**
bert-base-multiling	71.259	79.266	0.864	0.096	0.656	0.287
distilbert-base-multiling	69.953	79.225	0.862	0.095	0.655	0.301
xlm-roberta-base	73.282	81.357	0.878	0.086	0.686	0.267
muril-base-cased	51.291	68.789	0.775	0.123	0.524	0.487
bangla-bert-base	70.264	79.344	0.866	0.096	0.657	0.297

In this table, we have shown the performance of the transformer models tested specifically on low-resource Hindi datasets only. We will delve into the model's effectiveness in handling the unique characteristics of the Hindi language. Hindi, spoken by over 600 million people globally, presents unique challenges for NLP models due to its complex morphology and rich grammatical structures. This work investigates the performance of a Hindi dataset specifically designed for the task of multilabel classification for toxic comments. We aim to identify the most effective model for this task by comparing their performance metrics on the Hindi dataset. This evaluation will not only provide insights into the strengths and weaknesses of different models but also contribute to the development of robust NLP tools for the Hindi language.

Table 6: Metrics for different transformer models for the Hindi language

Model	Accuracy	F1-Score	ROC AUC	Hamming Loss	Jaccard Score	Zero-One Loss
twhin-bert-base	**75.136**	**78.641**	**0.863**	**0.075**	**0.647**	**0.249**
bert-base-multiling	74.289	78.156	0.846	0.072	0.641	0.257
distilbert-base-multiling	71.808	76.158	0.831	0.078	0.615	0.282
xlm-roberta-base	73.382	77.674	0.847	0.076	0.635	0.266
muril-base-cased	67.393	70.671	0.791	0.091	0.564	0.326
hindi-bert	67.635	72.756	0.799	0.083	0.572	0.323
roberta-hindi	73.079	77.126	0.844	0.077	0.627	0.269

This table analyzes the performance of various machine learning models on a merged dataset encompassing Hindi and Bengali languages. The motivation behind this study stems from the inherent similarities between these two languages, both belonging to the Indo-Aryan language family and sharing a significant lexical base. By combining datasets from both languages, we aim to leverage the strengths of each language. The merged dataset offers a richer pool of data, potentially leading to improved model generalizability and performance and to address data scarcity issues for certain tasks or languages, limited data availability can hinder model training. Merging datasets can mitigate this issue, particularly for tasks requiring large amounts of data and facilitate cross-lingual learning as the model's exposure to both languages might enable it to learn transferable features beneficial for tasks in either language.

Table 7: Metrics for different transformer models for the Merged Dataset (Hindi + Bengali)

Model	**Accuracy**	**F1-Score**	**ROC AUC**	**Hamming Loss**	**Jaccard Score**	**Zero-One Loss**
twhin-bert-base	**74.856**	**80.976**	**0.877**	**0.081**	**0.681**	**0.251**
bert-base-multiling (u)	72.288	78.969	0.859	0.088	0.652	0.277
bert-base-multiling (c)	72.288	78.969	0.859	0.088	0.652	0.277
distilbert-base-multiling	70.583	78.406	0.855	0.089	0.645	0.294
xlm-roberta-base	73.315	80.351	0.871	0.082	0.671	0.267
muril-base-cased	56.758	69.331	0.779	0.112	0.531	0.432

In this table, we have shown the performance of the transformer models tested specifically on low-resource Bengali and Hindi datasets specifically. This work investigates the performance of a custom dataset designed for the task of multilabel classification for toxic comments. Furthermore, it accentuates the necessity of evaluating the models on unobserved data to ascertain their applicability in real-world scenarios.

Table 8: Metrics for different transformer models for the Custom Dataset tested individually (Bengali + Hindi)

Model	**Bengali Accuracy**	**Bengali F1-Score**	**Hindi Accuracy**	**Hindi F1-Score**
twhin-bert-base	**63.512**	**64.431**	**61.128**	**62.302**
bert-base-multiling	58.661	59.526	57.492	58.427
distilbert-base-multiling	57.831	58.441	58.079	59.515
xlm-roberta-base	55.583	57.429	59.432	63.835
muril-base-cased	54.516	57.834	52.806	55.329

i. *BERT-Base-Multilingual-Uncased*

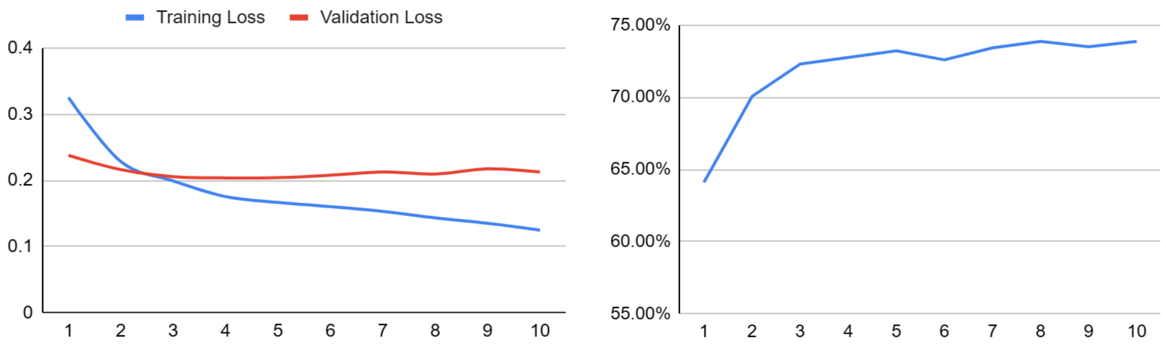

Figure 5: Training Log for BERT-Base-Multilingual-Uncased trained on Bengali and Hindi

ii. *BERT-Base-Multilingual-Cased*

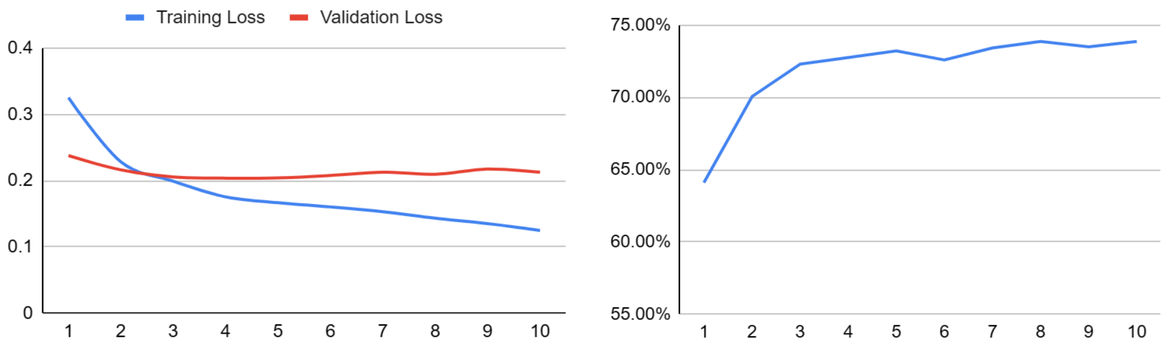

Figure 6: Training Log for BERT-Base-Multilingual-Cased trained on Bengali and Hindi

iii. *DistilBERT-Base-Multilingual-Cased*

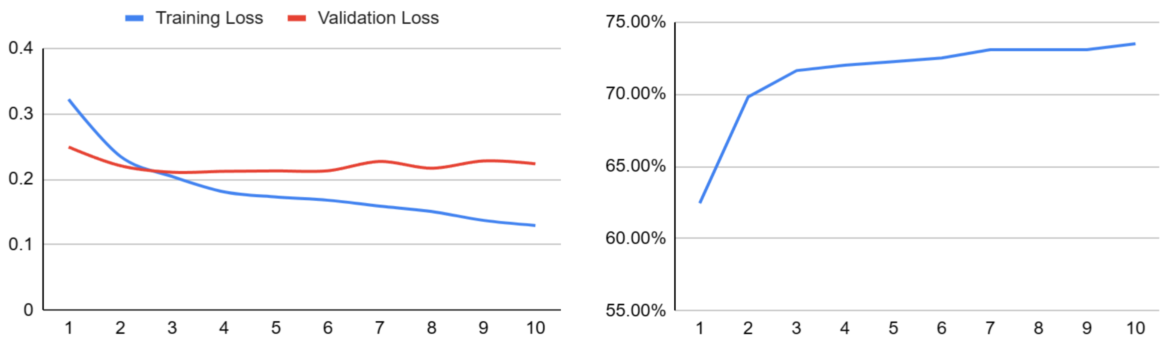

Figure 7: Training Log for DistilBERT-Base-Multilingual-Cased trained on Bengali and Hindi

iv. *XLM-RoBERTa-Base*

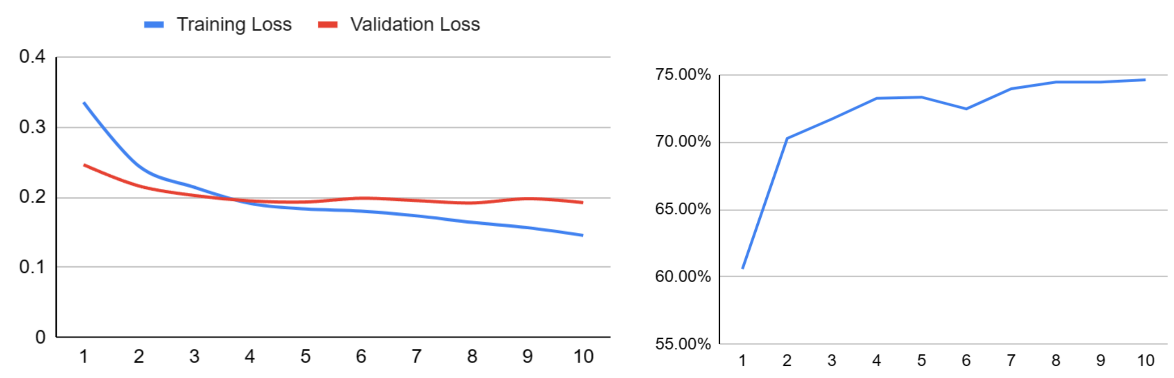

Figure 8: Training Log for XLM-RoBERTa-Base trained on Bengali and Hindi

v. *TwHIN-BERT-Base*

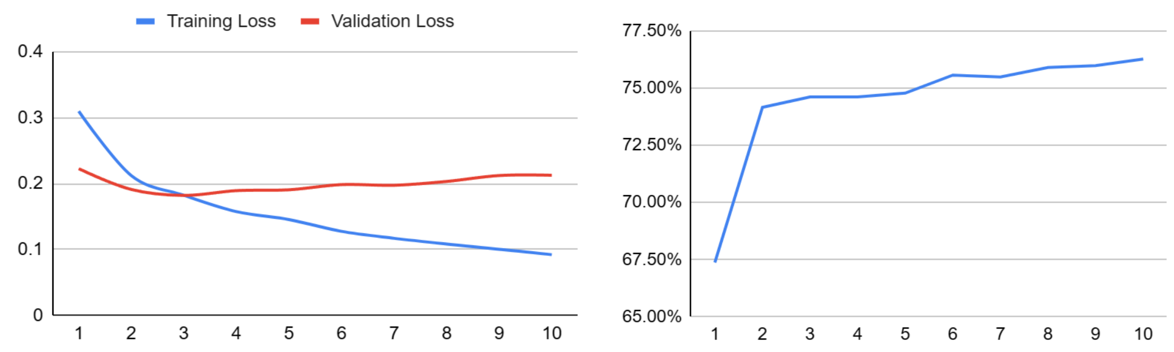

Figure 9: Training Log for TwHIN-BERT-Base trained on Bengali and Hindi

vi. *Muril-Base-Cased*

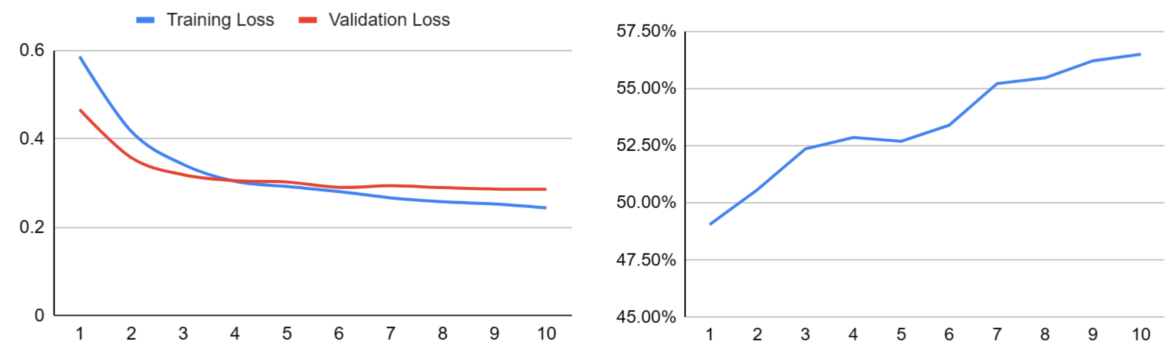

Figure 10: Training Log for Muril-Base-Cased trained on Bengali and Hindi

vii. *Bangla-BERT-Base*

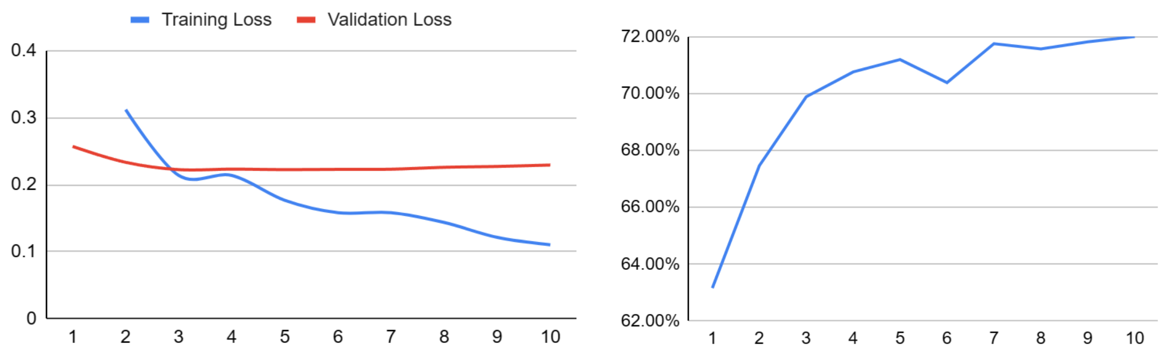

Figure 11: Training Log for Bangla-BERT-Base trained on Bengali

viii. *Hindi-BERT-v2*

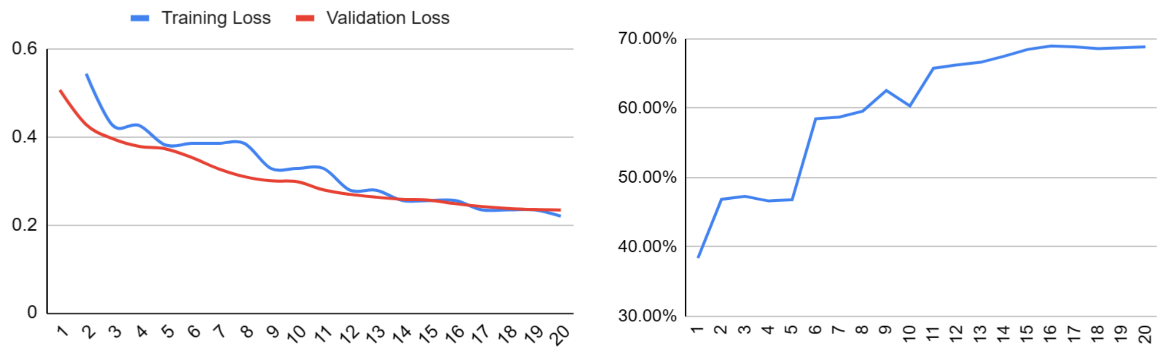

Figure 12: Training Log for Hindi-BERT-v2trained on Hindi

ix. *RoBERTa-Hindi*

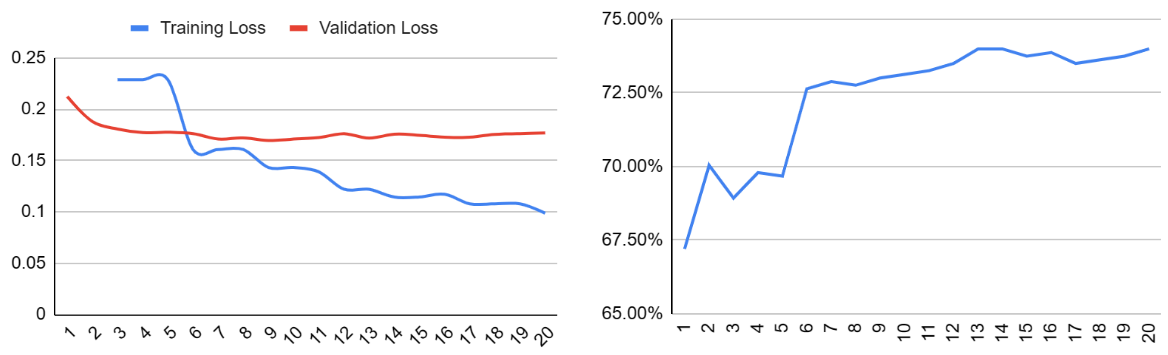

Figure 13: Training Log for RoBERTa-Hindi trained on Hindi

In this table, we have shown the performance of the transformer models tested specifically on low-resource Hindi datasets only. We will delve into the model's effectiveness in handling the unique characteristics of the Hindi language. Hindi, spoken by over 600 million people globally, presents unique challenges for NLP models due to its complex morphology and rich grammatical structures. This work investigates the performance of a Hindi dataset specifically designed for the task of multilabel classification for toxic comments. We aim to identify the most effective model for this task by comparing their performance metrics on the Hindi dataset. This evaluation will not only provide insights into the strengths and weaknesses of different models but also contribute to the development of robust NLP tools for the Hindi language.

Conclusion

This research project explores the use of transformer models for classifying text with multiple labels in both Bengali and Hindi. We implemented a comprehensive set of nine such models to address the crucial issue of comment filtering on social media platforms, ultimately aiming to create a safer online environment for users. The surge in Bengali and Hindi speakers on social media has fueled research efforts dedicated to filtering harmful content in these languages. Deep learning-based models have surpassed traditional machine learning techniques for toxicity detection, particularly when dealing with large datasets. Our approach utilizes a novel methodology that combines dataset modification, model implementation, and a comparative analysis of the nine transformer models. We built separate datasets for Bengali and Hindi before merging them after modifying the labels. This merged dataset was then used to fine-tune the transformer models. The models' performance was evaluated using a diverse set of metrics, including accuracy, F1-score, ROC AUC, Hamming Loss, Jaccard Score, and Zero-One Loss. The model fine-tuned on TwHIN-BERT-Base emerged as the most effective, achieving an accuracy of 74.85% and an F1-score of 80.98% on the merged dataset. This model consistently outperformed others across all metrics. It's important to note that relying solely on accuracy and F1-score for multi-label classification can be misleading, hence the importance of incorporating a broader range of metrics for a more robust assessment. Ultimately, these models have the potential to identify and flag harmful content within online conversations, fostering a safe and respectful online environment for Bengali and Hindi speakers. Our project significantly contributes to the ongoing research on abusive language detection in languages with limited resources, offering a valuable resource for future studies.

Future Scope

The problem statement of detecting toxic comments in Bengali and Hindi language 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 both Bengali and Hindi, as the current existing models have some limitations in tokenizing them. For instance, the models 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 [7] based model that is specifically trained on toxic words of Bengali and Hindi 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, and also different dialects of Hindi that are present in India, which could reduce the confusion for the transformer [6] [7] based models. The current dataset contains different styles of Bengali and Hindi, 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 different dialects of Bengali and Hindi could help the models learn the specific features and patterns of each dialect and improve their accuracy.

Finally, better-labeled data can reduce confusion for the models. It is noticed that the current dataset has multiple comments that may be classified to some label of toxic comment, but due to limitations of labels, it is classified as non-hate. This creates confusion for the model and makes it difficult for the models to understand the patterns of the data. So proper labeling needs to be done to make the data less confusing.

References

[1] R. Karim, S. Dey, and B. Chakravarthi, *DeepHateExplainer: Explainable Hate Speech Detection in Under-resourced Bengali Language*. 2020.

[2] M. F. Ahmed, Z. Mahmud, Z. T. Biash, A. A. N. Ryen, A. Hossain, and F. B. Ashraf, “Bangla Online Comments Dataset,” vol. 1, Jan. 2021, doi: 10.17632/9xjx8twk8p.1.

[3] M. Bhardwaj, M. S. Akhtar, A. Ekbal, A. Das, and T. Chakraborty, “Hostility Detection Dataset in Hindi.” arXiv, Nov. 06, 2020. Accessed: May 12, 2024. [Online]. Available: http://arxiv.org/abs/2011.03588

[4] 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.

[5] 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

[6] A. Vaswani *et al.*, “Attention Is All You Need,” 2017, doi: 10.48550/ARXIV.1706.03762.

[7] 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

[8] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” 2019, doi: 10.48550/ARXIV.1910.01108.

[9] A. Conneau *et al.*, “Unsupervised Cross-lingual Representation Learning at Scale,” 2019, doi: 10.48550/ARXIV.1911.02116.

[10] X. Zhang *et al.*, “TwHIN-BERT: A Socially-Enriched Pre-trained Language Model for Multilingual Tweet Representations at Twitter,” 2022, doi: 10.48550/ARXIV.2209.07562.

[11] H. Adel *et al.*, “Improving Crisis Events Detection Using DistilBERT with Hunger Games Search Algorithm,” *Mathematics*, vol. 10, no. 3, p. 447, Jan. 2022, doi: 10.3390/math10030447.

[12] S. Khanuja *et al.*, “MuRIL: Multilingual Representations for Indian Languages,” 2021, doi: 10.48550/ARXIV.2103.10730.

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

[14] J. Abadji, P. O. Suarez, L. Romary, and B. Sagot, “Towards a Cleaner Document-Oriented Multilingual Crawled Corpus,” 2022, doi: 10.48550/ARXIV.2201.06642.

[15] R. Joshi, “L3Cube-MahaNLP: Marathi Natural Language Processing Datasets, Models, and Library,” 2022, doi: 10.48550/ARXIV.2205.14728.

[16] R. Joshi, “L3Cube-MahaCorpus and MahaBERT: Marathi Monolingual Corpus, Marathi BERT Language Models, and Resources,” 2022, doi: 10.48550/ARXIV.2202.01159.

[17] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems”.