Hate Speech and Offensive Content Identification in Memes in Bangla, Hindi and Gujarati using Auxiliary Text Supervised Learning

Kongqiang Wang[1,*], Qingli Tan[2]

[1]*School of Information Science and Engineering, Yunnan University, Kunming 650500, Yunnan, China.*
[2]*College of Ecology and Environment, Yunnan University, Kunming 650500, Yunnan, China.*

Abstract

Our group name on the HASOC2025-meme (Gujarati, Hindi and Bangla) competition platform on Kaggle is kongqiang wang. We are very interested in the seventh edition of the HASOC shared task, in which HASOC 2025 offer memes for abuse, sentiment, sarcasm, vulgarity detection. This task involves analyzing multimodal data (image and text) to detect abuse, identify targeted communities, assess vulgarity and sarcasm, and assign sentiment labels. So, the task will be in five parts. We mainly participated in the identification of memes for abuse, sentiment, sarcasm and vulgarity detection. Team rank will be determined based on the Average of all four Macro F1 score. We ranked 10th on HASOC2025-meme(Gujarati) with a score of 0.56253 and 16th on HASOC2025-meme(Hindi) with a score of 0.51985. The above two tracks mainly employ machine learning and deep learning methods. We ranked 13th on HASOC2025-meme(Bangla) with a score of 0.52528. This track mainly employ transformer model. The code sources for the paper are available via GitHub.

Keywords

Multimodal Data Classification, Bert, Machine Learning, Deep Learning

1. Introduction

Social networking platforms such as Twitter and Facebook have become widely popular due to their ease of use and broad accessibility, offering individuals a powerful space to express their thoughts. Users from all age groups actively engage on these platforms, frequently documenting and sharing details of their daily lives, which contributes to an ever-growing volume of user-generated content. Despite the many advantages of social media, it is not without its drawbacks. A significant amount of harmful and offensive content—including hate speech—circulates online, posing serious societal challenges[1].

HASOC[2] provides a forum for developing and testing text classification systems for various languages. It organized a shared task for FIRE 2025[3]. The task is aimed at identifying hateful and offensive language in social media posts[4]. The task is organized for four languages, Hindi, Gujarati, Bangla and Bodo[5], but we only conduct our investigation on the Hindi, Gujarati and Bangla[6] dataset.

The following is a description of our research based on our advanced solutions to the specific problem. Main research fields track description: This task involves analyzing multimodal data (image and text) to detect abuse, assess vulgarity and sarcasm, and assign sentiment labels. So, the task will be in four parts.

Sentiment Detection:

- Positive - The meme conveys a supportive, humorous, or appreciative tone.
- Neutral - The meme is neither overtly positive nor negative in tone.
- Negative - The meme expresses hostility, mockery, or criticism.

Sarcasm Detection:

Forum for Information Retrieval Evaluation, December 17-20, 2025, Varanasi, India

*Corresponding author.

wangkongqiang60@gmail.com (K. Wang); tanqingli@stu.ynu.edu.cn (Q. Tan)
https://github.com/WangKongQiang (K. Wang)

- Sarcastic - The meme presents statements or visuals that imply the opposite of their literal meaning, often to mock or ridicule.
- Non-Sarcastic - The meme directly conveys its message without sarcasm or irony.

Vulgarity Detection:

- Vulgar - The meme contains explicit or offensive words, gestures, or depictions.
- Not Vulgar - The meme does not include any such content.

Abuse Detection:

- Abusive - The meme includes offensive, harmful, or derogatory language, imagery, or implications targeting an individual or a group.
- Non-abusive - The meme does not contain any offensive, harmful, or derogatory content.

All the above tasks are part of HASOC2025-meme (Hindi, Gujarati and Bangla). We did not participate in the identify targeted communities task. There has also been no research on the Bodo language.

2. Related Work

2.1. Dynamics of Online Abuse

Online hostility is a context-dependent notion intended to express hatred and threaten an individual or group based on discriminatory views. Despite the argument that hateful statements ought to be tolerated due to free speech acts, the public expression of hate speech propels the reduction of minority members, and such frequent and repetitive exposure to abusive speech could increase an individual's outgroup prejudice[7]. Real-world violent events could also lead to increased hatred in online space and vice versa[8]. With the rise of online hate, the research community has a massive responsibility to develop solutions to mitigate online hostility.

2.2. Research on abusive speech

The concern of abusive speech has long been studied in the research community. Earlier work on abusive speech attempted to detect abusive users by using lexical, syntactic features extracted from their posts[9]. Over the past few years, research around automated hate speech detection has matured tremendously. Most of the current study consists of diverse but related works. In 2016, Zeerak Waseem and Dirk Hovy[10] contributed a dataset in which thousands of tweets were labeled with racism and sexism markers, and Davidson et al.[11] focused on distinguishing offensive from hate content on Twitter. Using this dataset, the authors examined multiple linguistic features such as character and word n-grams, POStags, emotion lexicon, and tf-idf vectors with several classifiers such as LR, SVM, decision tree, etc. Although multiple datasets were being published, a major problem was the lack of correlation and reusable datasets across hate speech detection tasks[12]. To address this issue, Founta et al.[13] studied these inconsistencies and drew upon a robust labeling mechanism that attempts to circumvent the overlap among various forms of abusive speech.

With the advent of large datasets, most academic research has moved to data-hungry complex models to improve classifier performance, including deep learning[14] and graph embedding techniques[15]. Pitsilis et al.[16], used deep learning models such as LSTMs to identify the abusive tweets in English and noticed that it was pretty effective in this task. Zhang et al.[17] fused convolutional and gated recurrent networks to enhance the classification performance and had remarkable success on 6 out of 7 datasets used. Recently, transformer based language models such as BERT are becoming immensely popular in several downstream tasks and have outperformed several deep learning models such as CNN-GRU, LSTM, etc., for detecting abusive language[18].

2.3. Abusive language detection in Indic languages

In the last few years, several shared tasks, such as Hate-Speech and Offensive Content Identification (HASOC)[19], Dravidian Lang-Tech [20] workshop, TRAC[21], etc., have been organized to develop resources, datasets, and models for abusive speech detection and multiple datasets in Indic languages such as Hindi, Marathi, Tamil, Malayalam, etc. have been made public. The HASOC[22] shared task in Indo-European languages is arguably the most well-known series of competitions. It has been consistently organized from 2019 at the Forum for Information Retrieval (FIRE). The Dravidian Lang-Tech[20] workshop focused on determining the offensive language of the code-mixed dataset in three Dravidian languages, namely, Tamil–English, Malayalam–English, and Kannada–English crawled from social media. In addition, researchers have also developed several datasets for Bengali[23], code-mixed Hindi[24], Urdu[25], etc., for abusive language detection. However, a limited number of studies have been performed on the effect of zero-shot learning, few-shot learning[26], instance transfer, etc. In our work, we try to fill this critical gap by studying various transfer schemes thus opening up new avenues for future research for abuse detection in Indic languages[27].

3. Exploratory Data Analysis

Table 1
Details of Dataset HASOC2025-meme (Hindi, Gujarati, Bangla) Provided by Organizer.

Details	Memes in Train Data	Memes in Test Data
HASOC2025-meme (Hindi)	textbfTotall=1141	
Sentiment detection(Positive)	340	
Sentiment detection(Neutral)	276	
Sentiment detection(Negative)	525	
Sarcasm detection(Sarcastic)	770	769
Sarcasm detection(Non-Sarcastic)	371	
Vulgarity detection(Vulgar)	378	
Vulgarity detection(Not Vulgar)	763	
Abuse detection(Abusive)	308	
Abuse detection(Non-abusive)	833	
HASOC2025-meme (Gujarati)	**Totall=889**	
Sentiment detection(Positive)	404	
Sentiment detection(Neutral)	194	
Sentiment detection(Negative)	291	
Sarcasm detection(Sarcastic)	670	604
Sarcasm detection(Non-Sarcastic)	219	
Vulgarity detection(Vulgar)	297	
Vulgarity detection(Not Vulgar)	592	
Abuse detection(Abusive)	168	
Abuse detection(Non-abusive)	721	
HASOC2025-meme (Bangla)	**Totall=2693**	
Sentiment detection(Positive)	906	
Sentiment detection(Neutral)	311	
Sentiment detection(Negative)	1476	
Sarcasm detection(Sarcastic)	2081	1821
Sarcasm detection(Non-Sarcastic)	612	
Vulgarity detection(Vulgar)	467	
Vulgarity detection(Not Vulgar)	2226	
Abuse detection(Abusive)	739	
Abuse detection(Non-abusive)	1954	

For these tasks, participants are not allowed to use any external resources and datasets. So we only used the training set provided by the official. The test dataset is also provided by the official for the evaluation of models and the submitted result documents. The training and test sets for Hindi, Gujarati and Bangla languages provided by HASOC2025-meme, as well as the labels in the training sets, are respectively presented in Table 1.

4. Graphical Representation

The following images are the word-cloud for various types of memes given in the text dataset. It is a graphical representation of word frequency of different words used in each category of the content in memes on official resources and datasets.

The following is the situation of the word cloud generated by sentiment detection in the training set for the Hindi language provided by HASOC2025-meme (see Figure 1, 2, and 3).

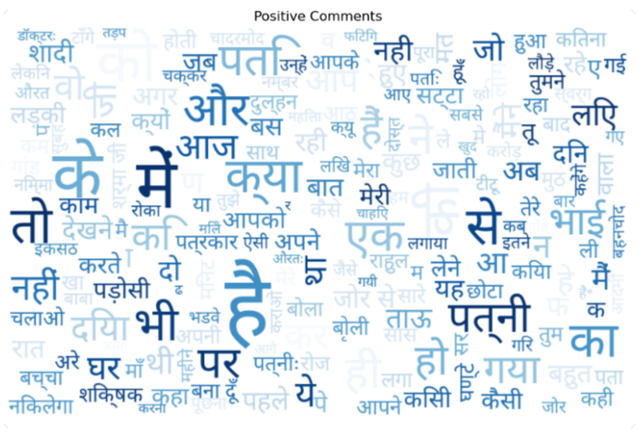


Figure 1: Word Cloud for Positive comments.

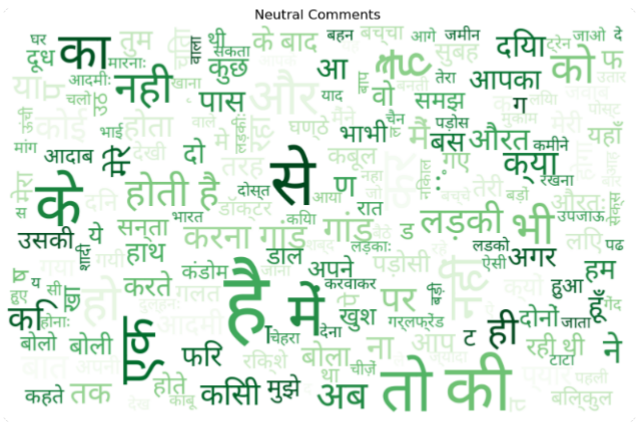


Figure 2: Word Cloud for Neutral comments.

The following is the situation of the word cloud generated by sarcasm detection in the training set for the Hindi language provided by HASOC2025-meme (see Figure 4 and 5).

5. Methodology

5.1. Introduction to Machine Learning

Machine Learning (ML) is an important branch of artificial intelligence (AI). Its core idea is to enable computers to automatically learn patterns through data and experience, rather than completing tasks

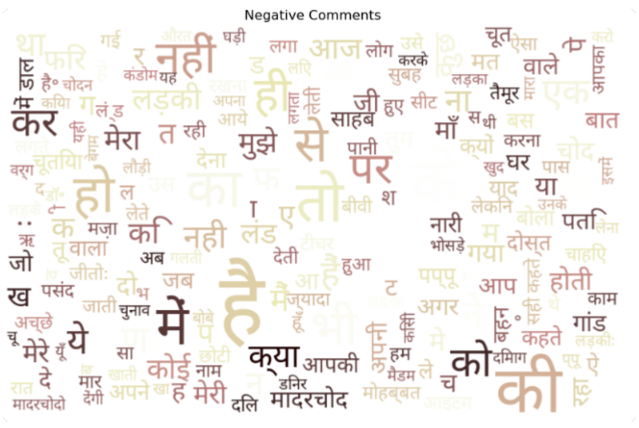


Figure 3: Word Cloud for Negative comments.

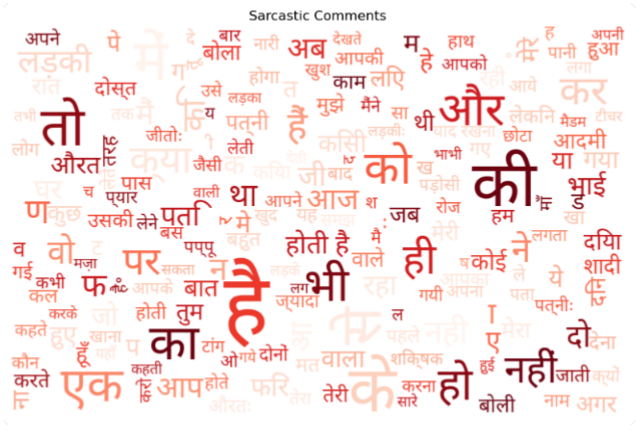


Figure 4: Word Cloud for Sarcastic comments.

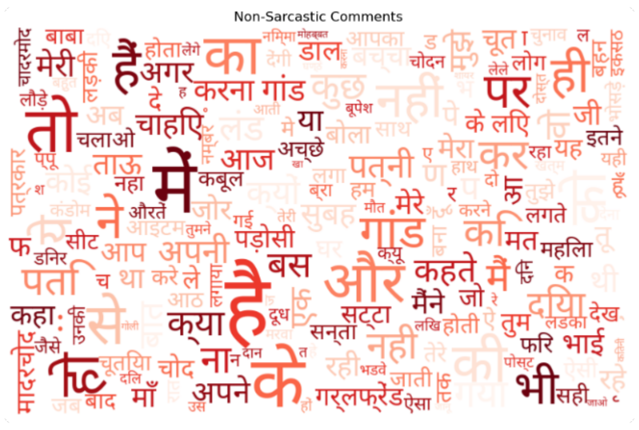


Figure 5: Word Cloud for Non-Sarcastic comments.

through explicit programming. In simple terms, traditional programming is Rules (written by programmers) + data $\longrightarrow$ output. And machine learning is: Data + Output $\longrightarrow$ Learning Algorithm $\longrightarrow$ Rules (Model). In this way, the machine can use the learned ”model” to predict new data.

5.1.1. K-nearest Neighbor Model

The definition of the nearest neighbor algorithm: To determine the category of an unknown sample, all training samples are used as references to calculate the distance between the unknown sample and all training samples, and the category of the nearest neighbor is used as the sole basis for deciding the category of the unknown sample.

The K-nearest neighbor model is based on two important assumptions of machine learning: ① Manifold Assumption: It refers to the fact that examples within a very small local neighborhood have similar properties, and therefore, their notations should also be similar. ② Cluster Assumption: Examples in the same cluster are more likely to have the same label.

The advantages of the KNN algorithm:

- The idea is simple and the theory is mature. It can be used for both classification and regression.
- It can be used for nonlinear classification.
- No assumptions about the data and insensitive to outliers.

The disadvantage of the KNN algorithm:

- Large amount of sorting calculation.
- Sample imbalance issue (that is, the number of samples in some categories is large, while the number of samples in others is small).
- It requires a large amount of memory.

5.1.2. Naive Bayes Model

Logistic regression achieves classification by fitting curves (or learning hyperplanes). Naive Bayes takes a unique approach by predicting classifications by considering feature probabilities.

The theorem derivation for implementing classification using Naive Bayes is mainly as follows:

- Given a training dataset $\{(X_1, y_1), (X_2, y_2), (X_3, y_3), \dots, (X_m, y_m)\}$, where m is the number of samples in the dataset. Each sample contains n features, i.e., $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$. The label set is $\{y_1, y_2, \dots, y_k\}$. Let $p(y = y_i|X = x)$ denote the probability that the output is y_i when the input sample is x.
- Suppose we now have a new sample x and want to determine which class it belongs to. We can solve for the values of $p(y = y_1|x)$, $p(y = y_2|x)$, $p(y = y_3|x)$, ..., $p(y = y_k|x)$. Whichever value is the largest determines the class it belongs to. That is, we solve for the maximum posterior probability $\arg\max p(y|x)$.
- According to Bayes' theorem, we have

$$p(y = y_i|\mathbf{x}) = \frac{p(y_i)p(\mathbf{x}|y_i)}{p(\mathbf{x})} \tag{1}$$

- In general, the naive Bayes method assumes that all features are mutually independent, so the above equation can be rewritten as:

$$p(y = y_i|\mathbf{x}) = \frac{p(y_i)p(\mathbf{x}|y_i)}{p(\mathbf{x})} = \frac{p(y_i)\prod_{j=1}^{n} p(x_j|y_i)}{\prod_{j=1}^{n} p(x_j)} \tag{2}$$

- Since the denominator is the same for each class in the solution, it can be omitted in practice. Finally, the decision rule for the Naive Bayes classifier is:

$$y = \arg\max_{y_i} p(y_i)p(\mathbf{x}|y_i) = \arg\max_{y_i} p(y_i)\prod_{j=1}^{n} p(x_j|y_i) \tag{3}$$

5.1.3. Decision Tree Model

Decision trees make decisions based on tree structures, which is precisely a very natural processing mechanism for humans when facing decision-making problems.

The key question in decision tree construction is how to build the tree, which means how to select the optimal splitting attribute for partitioning. To address this partitioning problem, we introduce the concept of information entropy, denoted as:

$$\mathrm{Ent}(D) = -\sum_{k=1}^{|y|} p_k \log_2 p_k \tag{4}$$

The smaller the value of $\mathrm{Ent}(D)$, the higher the purity of D. Suppose a certain attribute has V possible values $\{a^1, a^2, \ldots, a^V\}$. If we use attribute a to partition the sample set D, V branch nodes will be generated, where the v-th branch node contains all samples in D whose attribute a takes the value a^v, denoted as D^v.

By calculating the information entropy of D^v using the previous definition, and considering that different branch nodes contain different numbers of samples, we assign weights $|D^v|/|D|$ to the branch nodes, i.e., the more samples a branch node contains, the greater its influence. Thus, we can calculate the "information gain" obtained by partitioning the sample set D using attribute a:

$$\mathrm{Gain}(D, a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|}\mathrm{Ent}(D^v) \tag{5}$$

We can use information gain to perform decision tree attribute selection.

Continuous Values: The simplest method for handling continuous values is to use bi-partition.

Given a sample set D and a continuous attribute a, suppose a has n different values on D. Sort these values from small to large to obtain $\{a^1, a^2, \ldots, a^n\}$. Obviously, for adjacent values a^i and a^{i+1}, the candidate partition points can be selected as:

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} \mid 1 \leq i \leq n-1 \right\} \tag{6}$$

Similarly, calculate the information gain and select the point with maximum gain as the partition point:

$$\mathrm{Gain}(D, a) = \max_{t \in T_a} \mathrm{Gain}(D, a, t) = \max_{t \in T_a} \mathrm{Ent}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_\lambda^t|}{|D|}\mathrm{Ent}(D_\lambda^t) \tag{7}$$

A decision tree divides based on the maximum information gain and makes decisions according to the tree structure.

5.1.4. Random Forest Model

Random Forest is an extension of the Bagging algorithm in ensemble learning applied to decision trees. As the name suggests, it is a forest composed of many decision trees.

Consider a dataset containing m samples. Each time we randomly sample one sample and record its information, then put it back into the dataset. After performing m such samplings, the sample information we record forms a dataset of m samples. Then the probability that a certain sample is not selected in the m samplings is $(1 - 1/m)^m$. Taking the limit:

$$\lim_{m \to \infty} \left(1 - \frac{1}{m}\right)^m \approx \frac{1}{e} = 0.368 \tag{8}$$

That is, approximately 63.2% of the samples will appear in the sampled new dataset.

Based on this, for each sampling we conduct, we train a decision tree using 63.2% of the samples. In this way, by conducting such sampling training n times in total, we can obtain a total of n different decision trees. Ultimately, by using the voting mechanism, if the majority of decision trees agree that it is a positive class, then the final classification result of the random forest is a positive class. Conversely, it is a negative class.

5.1.5. Support Vector Machine Model

Support vector Machine is a common supervised learning algorithm, mainly used for classification and regression tasks. Its core idea is: to find an optimal hyperplane in the feature space, separate data points of different categories, and maximize the classification margin.

In simple terms: Imagine you have two types of data, red dots and blue dots. SVM will find a line (in two dimensions) or a surface (in three dimensions or higher dimensions) and separate them. Moreover, this line should be as far away as possible from the points on both sides, so that the model is more ”robust” to new data.

The advantages of SVM: It remains effective in high-dimensional Spaces (suitable for text classification and genetic data analysis). The classification boundaries are clear and the generalization ability is strong. It only relies on support vectors and is not greatly affected by redundant samples.

The disadvantages of SVM: The training speed for large-scale data is slow (with high computational complexity). Sensitive to the selection of parameters (such as C, kernel function type, γ). It is not very suitable for handling large amounts of noisy data and overlapping samples.

5.2. Ensemble Learning

Ensemble learning is a machine learning method that achieves better performance than a single model by combining the results of multiple models (base learners). Multiple weak models combined together can form a strong model.

There are mainly three common ways of ensemble learning:

Bagging. Idea is perform random sampling with substitution on the data to obtain multiple different training subsets, Train a base learner for each subset, and finally obtain the result by voting/averaging. Features are Reduce variance and enhance stability. Representative algorithm is Random Forest.

Boosting. The idea is that the base learner is trained step by step. The subsequent learner will focus on learning the samples that were wrongly classified by the previous learner. Finally, multiple weak classifiers are weighted and combined to form a strong classifier. Features are reduces bias, improves accuracy, but is prone to overfitting. Representative algorithms are AdaBoost, Gradient Boosting Decision Tree (GBDT), XGBoost, LightGBM, CatBoost.

Stacking. Idea is train multiple base learners (which can be different types of models), and then train a ”meta-learner” to integrate their prediction results. Features: It can combine the advantages of different models and usually achieves the best results, but it is relatively complex. Representative application: It is very common in Kaggle competitions[1].

The advantages and disadvantages of ensemble learning. **Advantages**: It enhances model accuracy and has strong generalization ability. It can reduce overfitting (Bagging) or minimize deviation (Boosting). Suitable for handling high-dimensional and large-scale data. **Disadvantages**: The computational cost is high (multiple models need to be trained). The model has poor interpretability and is not as intuitive as a single decision tree. There are many parameters, and parameter adjustment is rather complicated.

5.3. TfidfVectorizer

TfidfVectorizer is a text feature extraction tool provided by scikit-learn, which is often used in natural language processing (NLP). Its function is to convert a set of text (sentences, documents) into a TF-IDF feature matrix for use by machine learning models. The underlying principle of TfidfVectorizer is based on TF-IDF, that is, term frequency-inverse document frequency.

[1]https://www.kaggle.com/competitions/

Term Frequency (TF). Represents the frequency of a term appearing in a document.

$$TF(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{9}$$

Inverse Document Frequency (IDF). Represents the degree of discrimination a term has in the corpus.

$$IDF(t) = \log \frac{N}{1 + DF(t)} \tag{10}$$

- N: Total number of documents in the corpus
- $DF(t)$: Number of documents containing term t

If a term appears in many documents (such as "the", "is"), its IDF will be low, indicating low discrimination ability.

TF-IDF Formula.

$$TFIDF(t,d) = TF(t,d) \times IDF(t) \tag{11}$$

This way, common words have low weights, while important but infrequent words have high weights.

5.4. FastText

FastText is an efficient word embedding and text classification tool proposed by the Facebook AI Research (FAIR) team in 2016[28]. Its goal is to ensure high-quality word vectors while training faster than Word2Vec and GloVe, and to handle Out-Of-Vocabulary words (OOV) more effectively[29]. The core idea of FastText is different from that of Word2Vec. FastText does not merely view a word as a whole but breaks it down into n-gram characters.

Subword Representation, for example, for the word ”learning”, suppose n=3 (3-gram), then the subwords include: The word vectors of lea, ear, arn, rni, nin, and ing. FastText are equal to the sum of these sub-word vectors. In this way, even when encountering new words (such as ”learningssss”), the model can generate reasonable vectors through its subwords, solving the OOV problem.

The main functions of FastText: Word vector training, similar to Word2Vec, supports two training methods: Skip-gram and CBOW. However, due to the introduction of sub-word information, the vector representation is richer. FastText is equipped with an efficient text classifier that supports multi-class and multi-label classification[30]. Improve the training speed through Hierarchical Softmax.

The advantages of FastText: Fast training speed (based on efficient C++ implementation). It can handle unlogged words (represented by sub-words). It performs well in low-resource languages and small corpora. It can be used for two major tasks: word vector training and text classification.

The disadvantages of FastText: Compared with Transformer models (such as BERT), its semantic understanding ability is limited. It is impossible to capture long-distance dependencies (after all, it is still based on the bag-of-words idea).

To improve convergence, we apply Cyclic Learning Rate (CLR) scheduling, as proposed by (Smith, 2017)[31]. Specifically, we use the exp range policy to periodically vary the learning rate between 0.001 and 0.006 using a base-2 exponential decay factor (gamma=0.99994). This prevents premature con- vergence and encourages the model to escape local minima during training. The model is trained using the Adam optimizer with default hyperparameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1e - 8$). The loss function used is categorical_crossentropy, and additional evaluation metrics include accuracy and a custom-defined F1-score metric implemented using Keras backend operations. In the two language tasks, i.e. HASOC2025-meme (Hindi) and HASOC2025-meme (Gujarati), the word vectors we use are respectively cc.hi.300.vec and cc.gu.300.vec. They can be download from the fasttext official website [2].

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

5.5. Deep Learning

5.5.1. Introduction to Deep Learning

Deep Learning (DL) is a branch of machine learning, which is based on Artificial Neural networks (ANN), especially deep neural networks (DNN) with multi-layer structures. Common understanding: **Machine learning**. It requires manual design of features (feature engineering) and then learning with models. **Deep learning**. Through multi-layer neural networks, it automatically learns features from data, reducing manual intervention. Therefore, deep learning performs outstandingly in tasks such as image recognition, speech recognition, and natural language processing.

The inspiration for deep learning comes from the way neurons in the human brain work: a neuron receives input $\longrightarrow$ weighted summation $\longrightarrow$ activation function $\longrightarrow$ output result. Multiple neurons combine to form layers, and multiple layers combine to form a deep neural network.

5.5.2. Deep Learning Architecture

In this study, a text classification model based on deep learning was constructed. This model is built using the Keras Sequential API. The overall structure includes the embedding layer, the recurrent neural network layer and the fully connected classification layer. The specific design is as follows:

Firstly, the discrete word index is mapped to a low-dimensional continuous dense vector by using the word embedding layer, thereby capturing the semantic relationships between words. The input length set in this experiment is 2500, and the embedding dimension is 256.

Secondly, based on the embedded representation, the Long Short-Term Memory (LSTM) layer is introduced to model the context dependency and long-distance features of the text. This layer adopts a dropout rate of 0.2 and a recurrent dropout rate during the training process to alleviate the overfitting problem and enhance the generalization ability of the model.

Finally, a fully connected Layer (Dense Layer) is used for classification. This layer contains two neurons, and the softmax function is adopted to map the network output to a probability distribution, thereby achieving the binary or multiple classification task.

During the model training process, the Adam optimizer is adopted for parameter update. The categorical cross-entropy is selected as the loss function, and the Accuracy is used as the evaluation index. All three models are based on this overall architecture. The detailed model situation is shown in Figure 6.

5.6. TransformerTrainer

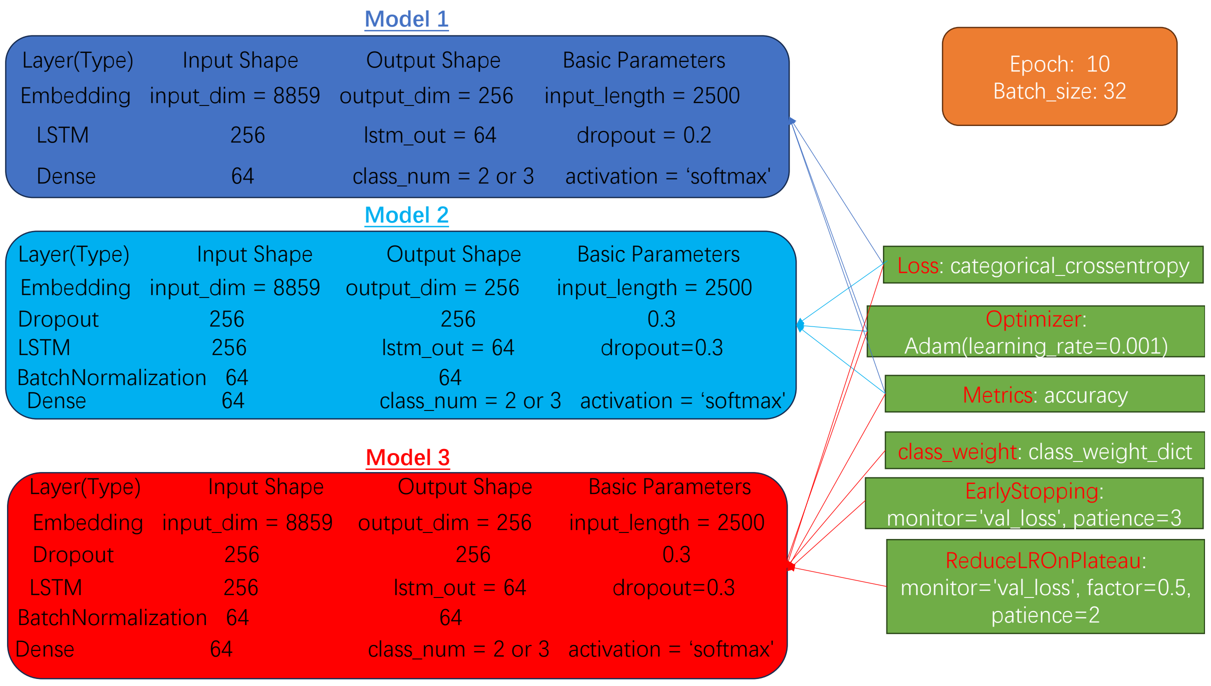
Trainer is a training manager provided by Hugging Face[3] to simplify the training/evaluation/inference process of Transformer models. It helps you encapsulate many underlying details, such as: Training loops (forward, loss calculation, backward, optimizer update), validation loops, gradient accumulation, learning rate scheduling, mixed-precision training (FP16/bf16), distributed training (multi-GPU, TPU, DeepSpeed), log and model saving.

Its core components include: **TrainingArguments**: Configure training-related parameters (number of epochs, batch size, learning rate, save policy, etc.). **Trainer**: Encapsulates the training logic, manages models, datasets, and optimizers. **compute_metrics**: Allows for custom evaluation metrics such as precision and F1 score. **Callback**: You can add custom logic, such as EarlyStopping.

5.7. BERT Model

BERT (Bidirectional Encoder Representations from Transformers) Proposed by Google AI (2018)[32], it is a pre-trained language model based on the transformer encoder.

[3]URL: https://huggingface.co/


Model 1

Layer(Type)	Input Shape	Output Shape	Basic Parameters
Embedding	input_dim = 8859	output_dim = 256	input_length = 2500
LSTM	256	lstm_out = 64	dropout = 0.2
Dense	64	class_num = 2 or 3	activation = 'softmax'

Model 2

Layer(Type)	Input Shape	Output Shape	Basic Parameters
Embedding	input_dim = 8859	output_dim = 256	input_length = 2500
Dropout	256	256	0.3
LSTM	256	lstm_out = 64	dropout=0.3
BatchNormalization	64	64	
Dense	64	class_num = 2 or 3	activation = 'softmax'

Model 3

Layer(Type)	Input Shape	Output Shape	Basic Parameters
Embedding	input_dim = 8859	output_dim = 256	input_length = 2500
Dropout	256	256	0.3
LSTM	256	lstm_out = 64	dropout=0.3
BatchNormalization	64	64	
Dense	64	class_num = 2 or 3	activation = 'softmax'

Epoch: 10
Batch_size: 32

Loss: categorical_crossentropy

Optimizer: Adam(learning_rate=0.001)

Metrics: accuracy

class_weight: class_weight_dict

EarlyStopping: monitor='val_loss', patience=3

ReduceLROnPlateau: monitor='val_loss', factor=0.5, patience=2

Figure 6: The three deep learning model frameworks and their basic parameters used in our experiment.

Its core idea is to conduct pre-training through large-scale unsupervised corpora and then fine-tune with a small number of supervised tasks, thereby significantly improving performance on various natural language processing tasks, such as text classification, named entity recognition, question-answering systems, etc.

The model structure mainly includes the following parts:

- Transformer Encoder Stacking: BERT uses multiple layers of Transformer Encoders as its basic structure, and each layer contains: multi-Head self-attention mechanism, feedforward neural network, residual connection and layer normalization.
- Bidirectional context modeling: Unlike traditional language models (which Model only from left to right or from right to left), BERT uses the masked language model (MLM) technique, enabling the model to obtain context information from both left and right directions simultaneously.
- Special symbols: **[CLS]** : Added at the beginning of the sentence during classification tasks, and the hidden vector at the end can be used for classification. **[SEP]** : Delimiter used to distinguish sentences for tasks such as natural language reasoning.

The BERT pre-training task is trained through two unsupervised tasks:

- Masked Language Model (MLM). Randomly covering 15% of the words, the model needs to predict these words. Let the model learn deep context representations.
- Next Sentence Prediction (NSP). Given two sentences, the model determines whether the second sentence is a true follow-up to the first one. Let the model learn semantic relations at the sentence level.

There are two situations regarding the scale of pre-trained models:

- BERT Base: 12-layer transformer encoder, 768 hidden dimensions, 12 attention heads, and approximately 110 million total parameters.
- BERT Large: 24-layer transformer encoder, 1024 hidden dimensions, 16 attention heads, total parameters approximately 340 million.

Table 2
Machine Learning Model in HASOC2025-meme(Hindi) Macro F1 score Validation Results.

Model	Sentiment detection	Sarcasm detection	Vulgarity detection	Abuse detection
GaussianNB	0.32	0.46	0.52	0.49
LogisticRegression	0.37	0.41	0.51	0.54
KNeighborsClassifier	0.25	0.41	0.41	0.48
SVC	0.37	0.42	0.50	0.53
DecisionTreeClassifier	0.40	0.44	0.52	0.53
LinearSVC	0.43	0.50	0.51	0.53
RandomForestClassifier	0.42	0.49	0.55	0.59

Table 3
Machine Learning Model in HASOC2025-meme(Gujarati) Macro F1 score Validation Results.

Model	Sentiment detection	Sarcasm detection	Vulgarity detection	Abuse detection
GaussianNB	0.38	0.46	0.45	0.40
LogisticRegression	0.37	0.56	0.63	0.50
KNeighborsClassifier	0.32	0.47	0.53	0.50
SVC	0.34	0.53	0.60	0.47
DecisionTreeClassifier	0.32	0.47	0.52	0.51
LinearSVC	0.32	0.56	0.61	0.61
RandomForestClassifier	0.35	0.55	0.62	0.72

Fine-tuning BERT pre-training model: The core advantage of BERT lies in the fact that the same pre-trained model can be adapted to different downstream tasks. fine-tuning, simply add a task-specific output layer (such as a classification layer) to the [CLS] vector and then train on the downstream dataset. Application scenarios include text classification (sentiment analysis, hate speech detection, etc.), sequence labeling (named entity recognition, word segmentation), question-answering system (SQuAD), natural language reasoning (NLI), information retrieval (semantic matching).

The advantages of the Bert pre-trained model include **Bidirectional representation**: Considering the information on both sides of the context simultaneously, its effect is superior to that of the unidirectional model. **Versatility**: One model is adapted to multiple downstream tasks. **Powerful performance**: When BERT was proposed, it refreshed the SOTA (state of the art) results on 11 NLP tasks.

In summary, BERT is a pre-trained language model based on the Transformer Encoder. It uses two tasks, MLM and NSP, to learn deep bidirectional context representations and is widely applied in various natural language processing tasks through fine-tuning.

6. Result

6.1. Machine Learning Model Experimental Results

We used different machine learning methods to train the models for the HASOC2025-meme(Hindi) and HASOC2025-meme(Gujarati) tasks. We extracted 15% of the data from the training set as the validation set. The performance of these models in the two languages is shown in Table 2 and Table 3.

The text is processed by TfidfVectorizer into digital matrices and then fed to the machine learning model respectively. The final performance on the test set is shown in Table 4 and Table 5. Among them, the voting mechanism was used to conduct ensemble learning on the machine learning KNeighborsClassifier, LinearSVC, and RandomForestClassifier, and it achieved good results in the overall situation.

Table 4
Machine Learning Model in HASOC2025-meme(Hindi) to the Average of all four Macro F1 score Test Results.

Model	the Average of all four Macro F1 score
KNeighborsClassifier	0.41461
LinearSVC	0.51985
RandomForestClassifier	0.51120
overall (rfc, knc, svc)	0.49283

Table 5
Machine Learning Model in HASOC2025-meme(Gujarati) to the Average of all four Macro F1 score Test Results.

Model	the Average of all four Macro F1 score
KNeighborsClassifier	0.45552
LinearSVC	0.52184
RandomForestClassifier	0.54716
overall (rfc, knc, svc)	0.52823

Table 6
The fasttext model combines a cyclic learning strategy in HASOC2025-meme(Hindi) and HASOC2025-meme(Gujarati) to the Average of all four Macro F1 score Test Results.

Language Tasks	the Average of all four Macro F1 score
HASOC2025-meme(Hindi)	0.45552
HASOC2025-meme(Gujarati)	0.37507

Table 7
The deep learning model in HASOC2025-meme(Hindi) and HASOC2025-meme(Gujarati) to the Average of all four Macro F1 score Test Results.

Model ID	the Average of all four Macro F1 score	
	HASOC2025-meme(Hindi)	HASOC2025-meme(Gujarati)
Model 1	0.46059	0.47087
Model 2	0.48101	0.52650
Model 3	0.51648	0.56253

6.2. FastText Model Experimental Results

This word vector-based model performed averagely in these two language tasks, i.e. HASOC2025-meme (Hindi) and HASOC2025-meme (Gujarati), their final results are shown in Table 6.

6.3. Deep Learning Model Experimental Results

In our experiment, we used the LSTM model in Model 1 as the baseline model. Due to the overfitting problem of the model, the performance was not satisfactory. Therefore, we added Dropout and BatchNormalization parts to Model 1. The performance of the model has been improved to a certain extent, solving the overfitting effect of the training set. We call this model Model 2. To further enhance the performance of the model, we have added class_weight to address the issue of label imbalance in the dataset. The use of EarlyStopping and ReduceLROnPlateau strategies has further enhanced the model, which we call Model 3. The performance of these models on the HASOC2025-meme(Hindi) and HASOC2025-meme(Gujarati) tasks is shown in Table 7.

Table 8
The Bert model in HASOC2025-meme(Bangla) to the Average of all four Macro F1 score Test Results.

Model	the Average of all four Macro F1 score
HASOC2025-meme(Bangla)	0.52528

6.4. BERT Model Experimental Results

In our task, we used the l3cube-pune/bengali-bert[4] model combined with TransformerTrainer and achieved good results on the HASOC2025-meme (Bangla) dataset. See Table 8. BengaliBERT is a Bengali BERT model trained on publicly available Bengali monolingual datasets. Preliminary details on the dataset, models, and baseline results can be found in their paper[33].

7. Conclusion

In this paper, several machine learning and deep learning approaches have been used to detect hate speech and offensive language content, and the models have been compared. Several techniques have been employed to increase accuracy. Our proposed Bert model achieved good results compared to its simplicity. We believe with proper feature extraction and data augmentation techniques, these will be able to improve our proposed model.

Acknowledgments

We are very grateful to the organizers of the Shared Task on Hate Speech and Offensive Content Identification in Memes in Bangla, Hindi, Gujarati and Bodo @ HASOC-meme (2025) and the School of Information of Yunnan University for providing the environment and equipment.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. Bhandari, S. B. Shah, S. Thapa, U. Naseem, M. Nasim, Crisishatemm: Multimodal analysis of directed and undirected hate speech in text-embedded images from russia-ukraine conflict, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 1994–2003.

[2] Koyel Ghosh and Mithun Das and Mwnthai Narzary and Saptarshi Saha and Shubhankar Barman and Animesh Mukherjee and Sandip Modha and Debasis Ganguly and Utpal Garain and Sylvia Jaki and Thomas Mandl, Overview of the HASOC Track at FIRE 2025: Abusive Meme Identification — Shadows Behind the Laughter, in: K. Ghosh, T. Mandl, S. Pal, S. Majumdar, A. Chakraborty (Eds.), Forum for Information Retrieval Evaluation (Working Notes) (FIRE 2025) December 17-20, Varanasi , India, CEUR-WS.org, 2025.

[3] Koyel Ghosh and Mithun Das and Sumukh Patel and Nilotpal Bhandary and Alloy Das and Animesh Mukherjee and Sandip Modha and Debasis Ganguly and Utpal Garain and Sylvia Jaki and Thomas Mandl, Overview of the HASOC Track at FIRE 2025: Abusive Meme Identification — Shadows Behind the Laughter, in: FIRE '25: Proceedings of the 17th Annual Meeting of the Forum for Information Retrieval Evaluation. December 17-20, Varanasi , India, Association for Computing Machinery (ACM), New York, NY, USA, 2025.

[4] HuggingFace: https://huggingface.co/l3cube-pune/bengali-bert

[4] K. Ghosh, N. K. Singh, J. Mahapatra, et al., Safespeech: a three-module pipeline for hate intensity mitigation of social media texts in indic languages, Social Network Analysis and Mining 14 (2024). URL: https://doi.org/10.1007/s13278-024-01393-9. doi:10.1007/s13278-024-01393-9.

[5] K. Ghosh, S. Saha, T. Mandl, S. Modha, Findings from shared tasks on hate speech detection: Performance patterns for low-resource languages, Pattern Recognition Letters (2025). URL: https://www.sciencedirect.com/science/article/pii/S0167865525003150. doi:https://doi.org/10.1016/j.patrec.2025.09.004.

[6] M. Das, A. Mukherjee, Banglaabusememe: A dataset for bengali abusive meme classification, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023, pp. 15498–15512.

[7] W. Soral, M. Bilewicz, M. Winiewski, Exposure to hate speech increases prejudice through desensitization, in: Aggressive behavior 44, 2, 2018, pp. 136–146.

[8] A. Olteanu, C. Castillo, J. Boy, K. Varshney, The effect of extremist violence on hateful speech online, in: Proceedings of the International AAAI Conference on Web and Social Media, volume 12, 2018.

[9] Y. Chen, Y. Zhou, S. Zhu, H. Xu, Detecting offensive language in social media to protect adolescent online safety, in: 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing. IEEE, 2012, pp. 71–80.

[10] Z. Waseem, D. Hovy, Hateful symbols or hateful people? predictive features for hate speech detection on twitter, in: Proceedings of the NAACL student research workshop, 2016, pp. 88–93.

[11] T. Davidson, D. Warmsley, M. Macy, I. Weber, Automated hate speech detection and the problem of offensive language, in: Proceedings of the International AAAI Conference on Web and Social Media, volume 11, 2017.

[12] R. Kumar, A. K. Ojha, S. Malmasi, M. Zampieri, Benchmarking aggression identification in social media, in: Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), 2018, pp. 1–11.

[13] A. M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, N. Kourtellis, Large scale crowdsourcing and characterization of twitter abusive behavior, in: Twelfth International AAAI Conference on Web and Social Media, 2018.

[14] P. Badjatiya, S. Gupta, M. Gupta, V. Varma, Deep learning for hate speech detection in tweets, in: Proceedings of the 26th international conference on World Wide Web companion, 2017, pp. 759–760.

[15] M. Das, P. Saha, R. Dutt, P. Goyal, A. Mukherjee, B. Mathew, You too brutus! trapping hateful users in social media: Challenges, solutions & insights, in: Proceedings of the 32nd ACM Conference on Hypertext and Social Media, 2021, pp. 78–89.

[16] G. K. Pitsilis, H. Ramampiaro, H. Langseth, Detecting offensive language in tweets using deep learning, in: ArXiv abs/1801.04433, 2018.

[17] Z. Zhang, D. Robinson, J. Tepper, Detecting hate speech on twitter using a convolution-gru based deep neural network, in: European semantic web conference. Springer, 2018, pp. 745–760.

[18] S. Banerjee, M. Sarkar, N. Agrawal, P. Saha, M. Das, Exploring transformer based models to identify hate speech and offensive content in english and indo-aryan languages, in: arXiv preprint arXiv:2111.13974, 2021.

[19] T. Mandl, S. Modha, G. K. Shahi, H. Madhu, S. Satapara, P. Majumder, J. Schaefer, T. Ranasinghe, M. Zampieri, D. Nandini, Overview of the hasoc subtrack at fire 2021: Hate speech and offensive content identification in english and indo-aryan languages, in: arXiv preprint arXiv:2112.09301, 2021.

[20] B. R. Chakravarthi, R. Priyadharshini, N. Jose, T. Mandl, P. K. Kumaresan, R. Ponnusamy, R. Hariharan, J. P. McCrae, E. Sherly, Findings of the shared task on offensive language identification in tamil, malayalam, and kannada, in: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, 2021, pp. 133–145.

[21] R. Kumar, A. K. Ojha, M. Zampieri, S. Malmasi, Proceedings of the first workshop on trolling, aggression and cyberbullying (trac-2018), in: Proceedings of the First Workshop on Trolling,

Aggression and Cyberbullying (TRAC-2018), 2018.

[22] T. Mandl, S. Modha, P. Majumder, D. Patel, M. Dave, C. Mandlia, A. Patel, Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages, in: Proceedings of the 11th Annual Meeting of the Forum for Information Retrieval Evaluation, FIRE '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 14–17. URL: https://doi.org/10.1145/3368567.3368584. doi:10.1145/3368567.3368584.

[23] N. Romim, M. Ahmed, H. Talukder, M. S. Islam, Hate speech detection in the bengali language: A dataset and its baseline evaluation, in: Proceedings of International Joint Conference on Advances in Computational Intelligence. Springer, 2021, pp. 457–468.

[24] A. Bohra, D. Vijay, V. Singh, S. S. Akhtar, M. Shrivastava, A dataset of hindi-english code-mixed social media text for hate speech detection, in: Proceedings of the second workshop on computational modeling of people's opinions, personality, and emotions in social media, 2018, pp. 36–41.

[25] M. P. Akhter, Z. Jiangbin, I. R. Naqvi, M. Abdelmajeed, M. T. Sadiq, Automatic detection of offensive language for urdu and roman urdu, in: IEEE Access 8 (2020), 2020, pp. 91213–91226.

[26] T. Ranasinghe, M. Zampieri, Multilingual offensive language identification with cross-lingual embeddings, in: arXiv preprint arXiv:2010.05324, 2020.

[27] M. Das, S. Banerjee, A. Mukherjee, Data bootstrapping approaches to improve low resource abusive language detection for indic languages, in: Proceedings of the 33rd ACM conference on hypertext and social media, 2022, pp. 32–42.

[28] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, M. Douze, H. Jégou, Fasttext.zip: Compressing text classification models, arXiv preprint arXiv:1612.03651 (2016).

[29] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146.

[30] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, 2017, pp. 427–431.

[31] L. N. Smith, Cyclical learning rates for training neural networks, in: Computer Vision and Pattern Recognition, arxiv.org, 2017. URL: https://arxiv.org/abs/1506.01186. doi:10.48550/arXiv.1506.01186.

[32] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[33] R. Joshi, L3cube-hindbert and devbert: Pre-trained bert transformer models for devanagari based hindi and marathi languages, arXiv preprint arXiv:2211.11418 (2022).