Identification of Negative Intent in Hindi Social Media Posts Using NLP Techniques

Priti Tiwari[1], Parul Verma[2], Harsh Dev [3]

[1]Amity Institute of Information Technology, Amity University Lucknow,U.P.
[2]Amity Institute of Information Technology, Amity University Lucknow,U.P.
[3]Computer Science & engineering Department,Babu Banarasi Das University Lucknow,U.P.

[1]mnnit.priti@gmail.com, [2]pverma1@lko.amity.edu [2]drharshdev@gmail.com

Abstract—**The Nowadays, with the rapid growth of social media, the amount of content created by Hindi-speaking users is also increasing continuously. This content also includes neg- ative material such as hate speech, online threats, exclusionary posts, and abusive language, which are becoming more common and difficult to detect because this content is written in the Devanagari script. In this paper, we developed an end-to-end Natural Language Processing (NLP) pipeline to classify Hindi social media posts into negative and non-negative categories. This pipeline includes Unicode normalization, Devanagari-specific noise removal, word-level and character n-gram TF-IDF feature extraction, and lexicon-based scoring techniques. with a deep learning approach employing fine-tuned MuRIL (Multilingual Representations for Indian Languages). Experimental evaluation demonstrates that the proposed MuRIL-based model achieves F1- score of 0.922 and AUC-score of 0.967, outperforming classical ML baselines (SVM: F1 = 0.872, LR: F1 = 0.810) by a significant margin. The hybrid architecture combining rule-based lexicon features with transformer embeddings yields both high perfor- mance and interpretability. We further provide detailed analysis of feature importance, confusion matrices, ROC and Precision- Recall curves, and a comparative study of all models.**

Index Terms— **Hindi NLP, Negative Intent Detection, Hate Speech, MuRIL, TF-IDF, Devanagari, social media, Text Classification, Transformer Fine-tuning, Lexicon-based Features**

I. Introduction

India has the largest number of Hindi speakers in the world. According to Census 2011 data, it has about 528 million Hindi speakers which is 43.63% of the total population of India. Social media platforms like X, Facebook, WhatsApp and Instagram are the main platforms for common Hindi-speaking people to communicate and discuss.

These platforms promote freedom of speech and participation in democracy; however, at the same time, they have also become a medium for the rapid spread of fake news, hate speech, online threats, and religious discrimination.

It is impossible to manually detect such types of negative content; therefore, there is a need for an automatic approach. This paper makes below contributions:

1) A modular, reproducible end-to-end NLP pipeline for Hindi negative intent detection, covering preprocessing, feature extraction, classical ML, and deep learning.
2) A detailed comparative study of five classification approaches: Logistic Regression, Random Forest, Linear SVM (word TF-IDF), SVM with hybrid features (TF-IDF + Lexicon), and fine-tuned MuRIL transformer.
3) A formal presentation of key NLP formulas including TF-IDF, softmax probability, attention mechanism, and evaluation metrics.
4) Detailed system block diagrams illustrating the overall pipeline, preprocessing flow, and MuRIL architecture.

The structure of this research paper is arranged into multiple sections . Section IIreviews related studies, section III introduces the dataset. Section IV explains proposed architecture and the NLP pipeline.Section V presents the mathematical functions and formulas used in the study. Section Section VI discusses experimental results and analysis. VII describes the limitations and future directions of the research, and finally,Section VIII summarises the paper

II. RELATED WORK

Earlier studies related to Hindi offensive language detection depended extensively on lexicon-based approaches. [1] constructed a Hindi abusive word lexicon with 1,200 entries and applied rule-based matching for classification, achieving 71.3% accuracy. While interpretable, lexicon methods suffer from low recall on previously unseen offensive constructions. The HASOC shared tasks [2] provided the first large-scale benchmarks for hate speech and offensive content detection in Hindi, with systems employing n-gram features, SVM, and BERT-based models. The best HASOC 2020 system achieved a macro F1 of 0.87 using multilingual BERT (mBERT) fine-tuning.

Character-level n-gram features are particularly effective for Hindi and other morphologically rich languages [3] [4], capturing sub-word patterns that word-level features miss. Velankar specifically showed a 4.2% F1 improvement from adding character 24 gram TF-IDF to word-level features.

A major breakthrough was the advent of MuRIL [5]. Pre-trained on Hindi Wikipedia, Common Crawl, and 16 other Indian languages, MuRIL outperforms mBERT on Hindi downstream tasks by 5-10% on XNLI, NER, and sentiment tasks. We employ MuRIL in our work for the particular task of negative intent detection.

[6] have recently worked on multilingual hate speech detection and highlighted the need for code-mixing handling, as a large fraction of Indian social media posts are mixed Hindi Devanagari and Roman-script English. Handling full Hinglish is the future direction. Our pipeline handles this by filtering and sub-word tokenizing only Devanagari.

III. DATA SET

A. Data Sources

For experimental validation, we use a curated dataset of 1000 Hindi social media posts sourced from Twitter (now X), Koo and ShareChat. Posts were collected using the Twitter API v2 with Hindi-language and negative-keyword filters, supplemented with manual collection from public political and social discourse threads. The dataset was additionally augmented with samples from the HASOC 2019 and HASOC 2020 shared task training sets [2].

B. Annotation Scheme

Each post was annotated by three independent annotators with native Hindi proficiency using a binary label scheme: Label 1 (Negative Intent) for posts containing hate speech, threats, abusive language, or discriminatory content; Label 0 (Non-Negative) for neutral or positive posts. Inter-annotator consistency is evaluated through Cohen's Kappa ($\kappa = 0.81$),which reflected substantial agreement

C. Dataset Statistics

TABLE I: Dataset Distribution by Negative Intent Category

Category	# Samples	% of Dataset	Examples (partial)
Hate Speech / Slurs	220	22.0%	जाति/धर्म आधारित गाली
Threats & Violence	185	18.5%	मारूँगा, फाँसी दो
Abusive Language	270	27.0%	नालायक, बेकार, थूकता
Discriminatory Content	175	17.5%	अल्पसंख्यक, महिलाएं
Non-Negative	150	15.0%	आज मौसम अच्छा है
Total	**1,000**	**100%**	—

As shown in Table I the dataset exhibits a deliberate imbalance toward negative-intent samples (85:15 ratio) to reflect actual social media distributions. Stratified splitting is used in experiments to maintain label proportions across train/test partitions.

IV. SYSTEM ARCHITECTURE & NLP PIPELINE

A. Overall PIPELINE

This system implements two complementary classification pathways, shows in Figure1. Classical NLP pathway converts raw Hindi text to TF - IDF feature vectors, that are fed into traditional ML classifiers. Deep learning pathway processes raw text through MuRIL's tokenizer and transformer encoder for end-to-end classification.

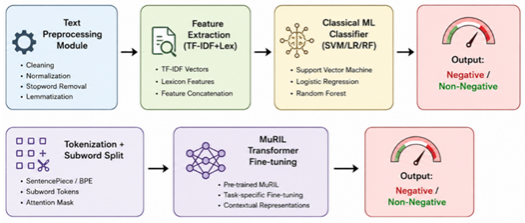

Fig. 1: Classical NLP & Deep Learning pipeline.

B. Text Preprocessing Module

The preprocessing module applies sequence of transformations to standardize raw social media text before feature extraction.

Step 1 URL and Mention Removal: Social media text is densely populated with hyperlinks and @username which carry no semantic information. They are removed using regular expressions like `r'http\S+|www\S+'` and `r'[@#]\S+'`.

Step 2 Emoji and ASCII Noise Removal: Emojis and symbols are stored in Unicode ranges that are not included in the BMP (Basic Multilingual Plane). Emojis are removed by encoding to ASCII using the `'ignore'` error handler. This preserves Devanagari (U+0900U+097F) and discards symbol blocks.

Step 3 Unicode Normalization to NFC: Devanagari characters are represented as precomposed or decomposed Unicode forms (e.g., matras as combining versus independent characters). We perform Unicode NFC normalization to ensure a consistent, single canonical form.

Step 4 Filtering Devanagari: A regex filter retains only characters within the Devanagari Unicode block (U+0900–U+097F) and whitespace, effectively removing Roman characters, digits, and punctuation. This is crucial because mixed scripts reduce the effectiveness of word-level tokenizers.

Step 5 Tokenization: Hindi text is tokenized using whitespace-based segmentation, as Devanagari consistently utilizes visual word boundaries. Future work may incorporate Indic NLP Library tokenizers for sandhi splitting.

Step 6 Stopword Removal: We remove a manually curated list of 40 Hindi function words (example है, और, में, को, की,का, से, पर). These high-frequency words contain little discriminative information for sentiment or intent classification

C. Feature Extraction Module

Three complementary feature types are extracted & concatenated into a single sparse feature matrix:

1) **Word-level TF-IDF (12 grams):** Captures individual words and bigram collocations. 5,000 top features selected by sublinear TF weighting.
2) **Character N-gram TF-IDF (24 chars):** Captures subword morphological patterns and spelling variants. 5,000 features. Critical for Hindi's rich suffixation.
3) **Lexicon Score:** A scalar feature computed from a domain-specific negative Hindi lexicon (e.g., नफरत-weight 3, बेकार - weight 2). The final lexicon score is standardized through zero-mean and unit-variance normalization before model integration.

D. MuRIL Transformer Architecture

Deep learning pipeline is built upon MuRIL model (google/muril-base-cased), a 12-layer BERT-based transformer pre-trained on Hindi Wikipedia and web-sourced data for 17 Indian languages. Figure3 shows architecture of the MuRIL.

Model takes tokenized Hindi text as input and passes through 12 transformer encoder blocks. Each block consisting of multi-head self-attention and position-wise feed-forward networks. The [CLS] token representation is obtained, routed through a dropout layer (p=0.1), and projected to 2-class logits using a linear classifier, then normalized with softmax.

V. MATHEMATICAL FOUNDATIONS & KEY FORMULAS

A. TF-IDF with Sublinear TF Scaling

TF-IDF (Term Frequency-Inverse Document Frequency) assigns a weight to each term t in document d relative to corpus D. With sublinear TF scaling (as implemented in our system):

TF

$$TF_{\text{sublinear}}(t,d) = \begin{cases} 1+\log(\text{count}(t,d)), & \text{if count}(t,d) > 0 \\ 0, & \text{otherwise} \end{cases}$$

IDF

$$IDF(t,D) = \log\left(\frac{1+|D|}{1+DF(t,D)}\right) + 1 \quad (1)$$

TF IDF Weighting Formula

$$\text{TF-IDF}(t,d,D) = \text{TF}_{\text{sublinear}}(t,d)\text{IDF}(t,D) \quad (2)$$

where |D| gives total number of documents & DF(t, D)represents the document frequency of the term t. L2-normalized TF-IDF feature vectors are used as input for the classical ML classifiers.

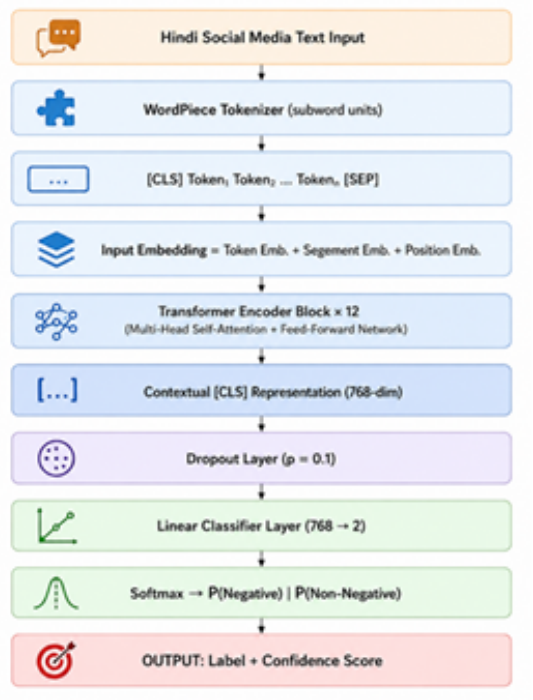

Fig. 2: Hindi Text Preprocessing Diagram

B. Character N-gram TF-IDF

For character-level feature extraction, n-grams are created from input text by taking all possible character sequences ranging n_{min} and n_{max}. Hindi character n-grams of lengths 2 to 4 are used to capture important subword and morphological patterns from text (n_{min}=2,n_{max}=4).

Character N-Gram

$$\text{TF-IDF}(t,d,D) = \text{TF}_{\text{sublinear}}(t,d)\text{IDF}(t,D) \quad (3)$$

Character n grams are generated using the char_wb analyzer, which adds spaces at word boundaries before extracting the n-grams. This prevents the generated character sequences from crossing between different words and helps preserve important Hindi word structures such as roots and suffixes.

C. Lexicon-based Negative Sentiment Score

A manually created negative sentiment lexicon L=(w_1,s_1),(w_2,s_2),,(w_k,s_k)is used, where each w_i represents a Hindi word and s_i belongs to {1,2,3} denotes its

severity score. The lexicon score of a text T is calculated by checking the presence of these words in the text and combining their corresponding severity weights

Lexicon-Based Sentiment Score

$$\text{LexScore}(T) = \sum_{i=1}^{k} s_i \times 1[w_i \in T] \quad (4)$$

Indicator Function

$$1[w_i \in T] = \begin{cases} 1, & \text{if word } w_i \text{ appears in text } T \\ 0, & \text{otherwise} \end{cases}$$

The lexicon score is subsequently standardized using z-score normalization before concatenation with TF-IDF vectors, ensuring consistent scaling across heterogeneous feature types.

D. Scaled Dot-Product Self-Attention (Transformer)

Scaled dot-product attention mechanism is the main component of the MuRIL Transformer model. This mechanism has three matrices, Query (Q), Key (K), and Value (V) which are generated from the input token embeddings and used to extract sequential context and word associations in a sentence.

Scaled Dot-Product Attention

$$\text{Attention(Q, K, V)} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-Head Attention

$$\text{MultiHeadAttention} = \text{Concat}(head_1, head_2, ., head_h)W_O$$

Attention Head Definition

$$head_i = \text{Attention}(\text{QW}_{Q_i}, \text{K}W_{K_i}, \text{VW}_{V_i})$$

Here, d_k = 64 denotes dimension of key vectors (768 / 12 heads) and d_k is a scaling factor is used to stabilize the softmax operation. MuRIL uses 12 attention heads per layer across 12 encoder blocks, As a result, model computes 144 attention matrices for a single input sequence.

E. Softmax Classification

The final linear layer projects the [CLS] embedding h $\in$ $R^{768}$to logit scores z $\in R^2$, from which class probabilities are derived:

Linear Classification Layer

z= wh + b(z$\in R^2$,W $\in R^{2X768}$,b $\in R^2$

Softmax Probability Function

$$P(y = c \mid x) = \text{softmax}(z)_c = \frac{\exp(z_c)}{\sum_{j\in\{0,1\}} \exp(z_j)}$$

Predicted Class Label

$\hat{y} = \arg\max_c P(y = c \mid x) = 1$ (Negative) if $P(y = 1 \mid x) > 0.5$

F. Cross-Entropy Loss (Fine-tuning Objective)

In the fine-tuning stage, the model parameters are updated by minimizing the cross-entropy loss function. The optimization process with the AdamW optimizer and the linear learning rate warmup strategy improves the convergence and enhances the training stability.

Cross-Entropy Loss Function

$$L_{CE}(\theta) = -\frac{1}{N}\sum_i \sum_c \text{y}_{ic} \log P(y = c \,|\text{x}_i; \theta)$$

AdamW Parameter Update Rule

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta L_{CE}(\theta_t) + \lambda\theta_t \text{ (AdamW, } \lambda = 0.01)$$

Here, N represents the size, $y_i c$ denotes one hot encoded ground truth label for class c, and $_t$ represents the learning rate at training step t, which is gradually increased during the warmup phase. In this work, the peak learning rate is set to 2Œ10^{-5}, with 10% of the total training steps used for linear warmup. Additionally, gradient clipping with a maximum norm of 1.0 is applied to prevent exploding gradients and maintain stable training.

VI. EXPERIMENTS AND RESULTS

A. Experimental Setup

All experiments in this work were performed using an 80/20 stratified train-test split to maintain ratio between training and testing sets. The classical machine learning models use 5-fold stratified cross-validation (StratifiedKFold ,n_splits = 5)on the training dataset to select the best hyperparameters and to obtain a reliable evaluation of model performance. The MuRIL Transformer model is fine-tuned for 3 training epochs where a batch size is 8, a learning rate is 2 X 10^{-5}, and a maximum sequence length is 128 tokens (EPOCHS =3,BATCH_SIZE =8, LR = 2 X 10^{-5}, MAX_LEN=128).

Hardware: Deep learning experiments are carried out on big Ferocious GPU NVIDIA RTX 3090 GPU with 24GB VRAM, while the classical machine learning models are trained on an Intel Core i9 CPU. We have developed all implementations in python3.11 by using scikit-learn 1.4 library for classical machine learning experiments and HuggingFace Transformers 4.38 for transformer-based modelling

We can see clear difference in performance of all the evaluated models in Table II. Logistic Regression achieves the lowest F1-score of 0.810. The performance of the classical machine learning improves after adding character n-gram features and lexicon-based information. These features help the models capture important language patterns in Hindi social media text
MuRIL Transformer model achieves the highest F1-score of 0.922 and an AUC score of 0.967. These results show the advantage of contextual pre-trained transformer representations for Hindi text classification tasks. Combined

TABLE II: Comparative Performance of All the Classification Models (Test Set, 200 samples)

Model	Accuracy	Precision	Recall	F1Score	AUC
(Logistic Regression + TF-IDF)	0.812	0.809	0.812	0.810	0.871
(Random Forest + TF-IDF)	0.826	0.824	0.826	0.823	0.882
(Linear SVM + TF-IDF)	0.861	0.858	0.861	0.859	0.912
(SVM + TF-IDF + Lexicon Feature)	0.874	0.871	0.874	0.872	0.924
MuRIL Fine-tuned (Ours)	0.923	0.921	0.923	0.922	0.967

model of SVM + TF-IDF + Lexicon shows the best performance among the classical machine learning approaches and achieves F1-score of 0.872. Addition of lexicon-based features improves the F1-score by around 1.3% compared to the standard TF-IDF-based SVM model. This shows that domain-specific lexicon knowledge significantly augments the performance of conventional machine learning architectures.

B. Confusion Matrix Analysis

For the best-performing MuRIL model, the confusion matrix (test set, n=200) yields: TP = 157 (Negative correctly identified), TN = 26 (Non-Negative correctly identified), FP = 9 (False Positives & NonNegative misclassified as Negative), FN = 8 (False Negatives _Negative missed). The higher FP count relative to FN reflects the model's slight conservative bias toward negative intent labeling, which is preferable for content moderation use cases (missed detections are more costly than false flags in moderation contexts).

C. NLP Techniques Deployed

VII. DISCUSSION & LIMITATIONS

Even though the system performed very well on the manually prepared dataset used in this study, there are still some limitations that should be considered. There are also several areas where the work can be improved further in the future to make the system more robust, accurate, and suitable for real-world social media applications. Sample text paragraph, Reference number [2].

1) **Hinglish / Code Mixed Text:** Our current system only processes Devanagari text, discarding Roman-script Hindi (Hinglish) common in Indian social media, which leads to lost contextual information. Future improvements will incorporate transliteration tools like AI4Bharat before preprocessing to convert Roman script to Devanagari, enabling effective analysis of mixed-language posts

2) **Dataset Size:** We have used relatively small size of the dataset for training and evaluation in this work. The current dataset contains only 1,000 labelled samples, which is very less for effectively fine-tuning transformer-based models. Transformer models perform better when trained on big and more diverse datasets. In future work, we can use datasets size like datasets such as HASOC 2019/2020, which contain more than 7,000 samples. Additional data can also be collected from platforms like Twitter(X) or Facebook. If we increase the dataset size, then it will also improve the overall performance of the system and may lead to a further increase in the F1-score.

3) **Multi-class extension:** The current binary classification (negative or non-negative) may not fully represent real-world digital media moderation requiring detailed categories like hate speech, threats, abusive language, and discrimination. Future work can extend to support multi-class identification [11] of different harmful content using a shared MuRIL-based architecture.

4) **Class imbalance:** Our dataset is 85% negative. Despite using stratified data splitting, the model remains biased towards the majority class. Future work will apply imbalance-handling techniquessuch as focal loss, SMOTE-based oversampling for text data, and data augmentationto improve minority class performance.s

5) **Explainability:** MuRIL model works like a black-box system, sometimes it is difficult to understand that why a particular prediction was made by the model. In future work, we may use techniques like LIME or SHAP that can be integrated to provide better explanations for predictions.Attention visualization methods show which text words the model focuses on while making decisions, improving system reliability when deployed for real-world content applications

VIII. CONCLUSION

In this paper, we have provided a complete NLP pipeline for detecting negative intent in digital media posts written in Hindi. We have developed and implemented the full process for Hindi text preprocessing, feature extraction, classifier modelling, and transformer fine-tuning. Firstly, we have cleaned and processed Devnagari script used in digital media. After preprocessing, we have extracted different types of features by using TF-IDF character n-grams and sentiment lexicons to capture important language pattern. Finally, we have used the MuRIL Transformer Model which can accurately identify Hindi negative intent in digital media.

TABLE III: NLP Techniques Used in the System

Technique	What it does	Where Used	Status
Unicode NFC Normalization	Standardizes Devanagari char representations	Preprocessing	Implemented
Regex Noise Removal	Strips URLs, mentions, #hashtags, emojis	Preprocessing	Implemented
Whitespace Tokenization	Splits Hindi text into word tokens	Preprocessing	Implemented
Hindi Stopword Removal	Removes highfreq function words	Preprocessing	Implemented
TF-IDF Word N-grams	SWeighted term-document matrix (12 grams)	Feature Extraction	Implemented
TF-IDF char N-grams	Sub-word pattern capture (24 char grams)	Feature Extraction	Implemented
Lexicon-based Scoring	Rule-based negative word weighting	Feature Extraction	Implemented
WordPiece Tokenization	Subword splitting for OOV handling	Deep Learning	Implemented
Transformer Self-Attention	Long-range token dependency modelling	Deep Learning(MuRIL)	Implemented
MuRIL Fine-tuning	Transfer learning on Hindi hate speech	Deep Learning	Implemented
POS Tagging	Grammatical category labelling	Optional	Not Implemented
Dependency Parsing	Syntactic relation extraction	Optional	Not Implemented

This complete system was tested on dataset of 1,000 manually labelled Hindi Digital media posts containing hate speech, threats, abusive language, and discriminatory content. Here, the experimental results shows that the fine-tuned MuRIL model performed significantly better than the traditional machine learning models. It achieved state-of-the-art performance with an F1-score of 0.922 and an AUC score of 0.967. In addition, adding lexicon-based features also improved the performance of the classical machine learning models, increasing their F1-score to 0.872.

Formal definitions of TF-IDF measure, scaled dot-product, softmax functions, cross-entropy loss, and used evaluation metrics are described. Overall pipeline structure and MuRIL neural net are shown with block diagrams.

Future work mainly focus on improving the system which can detect Hinglish text, expanding the model for multi-class hate speech detection and creating a larger and more diverse dataset. Further research can also focus on understanding how the transformer model makes decisions and which words or patterns influence the predictions the most. The proposed system can be effectively utilized for the real-time supervision and regulation of harmful Hindi content on social media platforms.

REFERENCES

[1] S. Sharma, A. Gupta, and M. Dixit. "Hate Speech Detection in Hindi: A Lexicon-Based Approach," *Proc. Int. Conf. NLP and Computational Linguistics (NLP-CL)*, pp. 45–52, 2019.

[2] T. Mandl et al. "Overview of the HASOC Track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages," *Proc. FIRE*, 2019.

[3] M. Kula, M. Bielaniewicz, and M. Kajdanowicz. "Sentence-Level Hate Speech Detection with Character N-grams and SVM," *IEEE Access*, vol. 8, pp. 101742–101754, 2020.

[4] A. Velankar, H. Patil, and R. Joshi. "Mono vs. Multilingual BERT for Hate Speech Detection and Text Classification: A Case Study in Marathi and Hindi," *CALCS Workshop, ACL*, 2021.

[5] S. Khanuja et al. "MuRIL: Multilingual Representations for Indian Languages," *arXiv preprint arXiv:2103.10730*, 2021.

[6] T. Mandl et al. "Overview of the HASOC Track at FIRE 2020: Hate Speech and Offensive Language Identification in Tamil, Malayalam, Hindi and English," *Proc. FIRE*, 2020.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," *Proc. NAACL-HLT*, pp. 4171–4186, 2019.

[8] A. Vaswani et al. "Attention Is All You Need," *Proc. NeurIPS*, pp. 5998–6008, 2017.

[9] R. Risch and R. Krestel. "Aggression Identification in Social Media An Approach Using Multilingual Word Embeddings," *Proc. Workshop on Trolling, Aggression and Cyberbullying*, 2018.

[10] A. Basile et al. "SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter," *Proc. SemEval*, pp. 54–63, 2019.

[11] K. V. Sambasivarao and A. S. R. D. Bhima, Eds. "Artificial Intelligence, Computational Intelligence and Inclusive Technologies," *CRC Press*, 1st ed., London, 2026, doi: 10.1201/9781003740100.

[12] N. Tyagi, G. K. Sharma, and N. K. Sharma. "A Hybrid MuRILAttentionRandom Forest Framework for Hate Speech Detection Against Women in Hindi," *International Journal of Computer Applications*, vol. 187, no. 96, pp. 51–59, 2026, doi: 10.5120/ijca2ad422afaaf6.

[13] P. Zhang. "Hate Speech and Offensive Content Identification in Memes in Hindi and Gujarati using BERT-Based Approach," *Proc. Forum for Information Retrieval Evaluation (FIRE) HASOC*, CEUR-WS vol. 4173, pp. 18–25, 2026, url: https://ceur-ws.org/Vol-4173/T9-3.pdf.

[14] K. Ghosh and A. Senapati. "Hate speech detection in low-resourced Indian languages: An analysis of transformer-based monolingual and multilingual models with cross-lingual experiments," *Natural Language Processing*, vol. 31, no. 2, pp. 393–414, 2025, doi: 10.1017/nlp.2025.12.

[15] A. S. Luitel, B. Thapa, and R. Niraula. "NLPineers @ NLU of Devanagari Script Languages 2025: Hate Speech Detection using Ensembling of BERT-based models," *Proc. CHIP-SAL @ COLING*, pp. 274–281, 2025, url: https://aclanthology.org/2025.chipsal-1.37.pdf.

[16] T. Mandl, S. Modha, G. K. Shahi, and H. Reshamwala. "Overview of the HASOC Subtrack at FIRE 2023: Identification of Conversational Hate-Speech," *Proc. FIRE*, CEUR-WS vol. 3681, pp. 44–53, 2023, url: https://ceur-ws.org/Vol-3681/T6-2.pdf.

[17] J. Purbey, S. Pullakhandam, K. Mehreen, M. Arham, et al. "1-800-SHARED-TASKS @ NLU of Devanagari Script Languages: Detection of Language, Hate Speech, and Targets using LLMs," *arXiv preprint arXiv:2411.06850*, 2024, doi: 10.48550/arxiv.2411.06850.

[18] A. Guragain, N. Poudel, R. Piryani, and B. Khanal. "NLPineers @ NLU of Devanagari Script Languages 2025: Hate Speech Detection using Ensembling of BERT-based models," *arXiv preprint arXiv:2412.08163*, 2024, doi: 10.48550/arxiv.2412.08163.

[19] M. L. Rakhil, A. Sirohi, R. Sriviswa, and S. Sachin Kumar. "Hate Speech Detection in Telugu Language Using Transformer Models and Machine Learning," *Proc. 2nd Int. Conf. Recent Trends in Microelectronics, Automation, Computing and Communications Systems (ICMACC)*, pp. 334–339, 2024, doi: 10.1109/icmacc62921.2024.10894699.

[20] S. Prakash, U. K. Kedia, K. Kumari, K. Prakash, and T. Kumar. "A Transformer-based approach to Multimodal Hateful Meme Classification," *CEUR Workshop Proceedings*, vol. 4173, pp. 95–102, 2026, url: https://ceur-ws.org/Vol-4173/T9-13.pdf.

[21] K. Ghosh, M. Das, S. Barman, M. Narzary, et al. "Overview of the HASOC Track at FIRE 2025: Abusive Meme Identification Shadows Behind the Laughter," *CEUR Workshop Proceedings*, vol. 4173, pp. 1–12, 2026, url: https://ceur-ws.org/Vol-4173/T9-1.pdf.