AI-Based Hate Speech Detection System Using LSTM

Mahima S
Department of Information Technology
College Of Engineering Guindy,
Anna University,
Tamil Nadu, India
Email: mahimasj5868@gmail.com

Dhanalakshmi G
Department of Information Technology
College Of Engineering Guindy,
Anna University,
Tamil Nadu, India
Email: dhanalakshmig03092004@gmail.com

Abstract—**The increasing usage of social media platforms like Twitter has led to a significant rise in hate speech and offensive content. Manual moderation techniques are insufficient given the vast volume of user-generated data. This project focuses on building an automated AI system that uses a Deep Learning model to detect and classify hate speech. Specifically, a Long Short-Term Memory (LSTM) neural network was trained on the Twitter Hate Speech Dataset to classify textual inputs into three categories: Hate Speech, Offensive Language, and No Hate/Offense. A user-friendly web application was developed using Streamlit to enable both single-text and batch-text predictions, making the model easily accessible and scalable for real-world use. The solution contributes toward creating safer digital environments by empowering proactive moderation.**

Index Terms—**Hate Speech Detection, Deep Learning, LSTM, Streamlit, NLP, Offensive Language, Content Moderation, Text Classification**

I. Introduction

The proliferation of social media has significantly amplified the spread of hate speech, creating a toxic digital environment. Manual monitoring methods are no longer sufficient due to the scale and velocity of content generation. The increasing demand for automated tools that detect and moderate such harmful content has led to the exploration of advanced NLP-based models. This paper introduces a Deep Learning model based on LSTM that can effectively detect hate speech in real time. We also present a web interface for easy user access and operational deployment.

Hate speech, offensive content, and toxic behavior can lead to psychological harm, misinformation, and social unrest. By enabling automated detection systems, this project addresses a vital societal need and also contributes to the field of ethical AI.

The importance of such systems is also reinforced by governmental and institutional calls for improved online content regulation. This project aligns with broader efforts to ensure internet safety through AI-powered solutions.

II. Objectives

- Develop a Deep Learning-based hate speech detection model using LSTM.
- Classify textual data into Hate Speech, Offensive Language, and No Hate/Offense.
- Preprocess text data effectively to improve model performance.
- Deploy a user-friendly web application using Streamlit.
- Enable both single-text prediction and batch-file prediction options.
- Support exportable results and visual output insights.
- Promote ethical AI practices for moderation.
- Improve user trust through transparency and explainable results.
- Allow extensibility for multilingual datasets in the future.

III. Methodology

A. Dataset Collection and Preprocessing

Dataset Used: The experiment utilizes the *Twitter Hate Speech Dataset* developed by Davidson et al. [1]. The dataset comprises over 24,000 labeled tweets categorized into three classes: **Hate Speech**, **Offensive Language**, and **Neither**. These tweets represent a diverse sample of real-world social media language, making the dataset suitable for training and evaluating hate speech detection systems.

Preprocessing Pipeline: To enhance model performance and ensure linguistic uniformity, the following preprocessing steps were applied:

- **Text Cleaning:** All tweets underwent cleaning to remove noise, including punctuation, special symbols, emojis, hyperlinks (URLs), hashtags, mentions, and numeric digits. All characters were also converted to lowercase to maintain consistency.
- **Tokenization:** Cleaned tweets were tokenized into individual words (tokens) using the NLTK library, enabling word-level analysis.
- **Stopword Removal & Lemmatization:** Common stopwords (e.g., ”is”, ”the”, ”and”) that do not contribute significant meaning were removed. Each token was then lemmatized to its root form (e.g., ”running” to ”run”) using spaCy, reducing inflectional variance in the corpus.
- **Sequence Transformation:** Tokenized texts were converted into numerical sequences using Keras Tokenizer, creating an index-based vocabulary suitable for neural network inputs.

- **Padding and Truncation:** To ensure uniform input lengths, all sequences were padded or truncated to a fixed length of 100 tokens, enabling consistent input dimensions for the LSTM model.
- **Train-Test Splitting:** The preprocessed dataset was partitioned into training and testing sets using an 80:20 ratio. This ensures sufficient data for model training while preserving a holdout set for unbiased evaluation.

B. Model Development

Neural Network Architecture:

A deep learning architecture based on a Long Short-Term Memory (LSTM) network was implemented for effective sequence modeling and context understanding:

- **Embedding Layer:** The first layer transforms each word index into a dense 100-dimensional vector representation, capturing semantic similarities between words.
- **LSTM Layer:** A single-layer LSTM with 128 units captures long-range dependencies and contextual information across the word sequences. Dropout regularization was applied to mitigate overfitting.
- **Dense Output Layer:** A fully connected layer with Softmax activation produces a probability distribution across the three target classes: Hate Speech, Offensive Language, and Neither.

Training Configuration: The model was trained using the following hyperparameters and evaluation protocols:

- **Loss Function:** `Categorical Crossentropy`, suitable for multi-class classification tasks.
- **Optimizer:** `Adam` optimizer, selected for its adaptive learning rate and convergence efficiency.
- **Epochs:** 5 training iterations were conducted, with potential for tuning based on validation results.
- **Batch Size:** 32 samples per batch, balancing memory efficiency and convergence stability.
- **Evaluation Metrics:** Model performance was assessed using Accuracy, Precision, Recall, and F1-score to provide a comprehensive view of classification effectiveness, especially for imbalanced class scenarios.

IV. Deployment

Frontend: The system is deployed using the **Streamlit** web framework, which provides a simple yet powerful interface for building and deploying machine learning applications. Streamlit enables rapid prototyping with interactive UI components and is well-suited for handling both textual and file-based input.

Features Overview:

- **Single Text Input:** Users can manually enter a single line or paragraph of text into a designated input box. The model processes the input and instantly provides class prediction and confidence scores.
- **Batch Input via File Upload:** For handling multiple entries, users can upload `.txt` or `.csv` files containing bulk text data. The application reads and processes each row, generating predictions in batch mode.
- **Class Probability Visualization:** The output includes a dynamic bar chart displaying the model's probability distribution across all potential classes, aiding in better interpretation of the results.
- **Real-time Feedback:** Once the input is submitted, the prediction process is triggered immediately and results are displayed in real-time, providing a seamless user experience.
- **Exportable Results:** Users can download the prediction results along with corresponding confidence scores in `.csv` format for record-keeping or further analysis.
- **Output Preview Interface:** A preview table is shown within the interface for reviewing outputs before downloading, which includes inputs, predicted classes, and their confidence scores.
- **Model Retraining Capability:** The system includes an option for retraining the model with new labeled data, allowing it to evolve over time and improve accuracy. This retraining functionality can be triggered from the interface, using uploaded datasets.

A. Usage Flow

- **Step 1: Input Submission** — Users either manually enter a text snippet or upload a file (TXT or CSV) containing multiple inputs.
- **Step 2: Prediction Triggered** — Upon submission, the model processes each input and generates class predictions along with corresponding confidence probabilities.
- **Step 3: Result Display** — Predictions are shown on-screen, and confidence scores are visualized using a color-coded bar chart to enhance interpretability.
- **Step 4: Export and Download** — Users can download the entire prediction output in a structured CSV format for external use, such as reporting or academic evaluation.
- **Step 5: Graphical Summary** — The interface includes summary plots (e.g., class distribution histograms) to give users an intuitive understanding of the classification results.

V. Tools and Technologies

- **Programming Language:** Python — used for backend model implementation, text preprocessing, and integration with the Streamlit frontend.
- **Libraries and Frameworks:**
  - **TensorFlow & Keras:** Deep learning libraries used for building, training, and deploying the classification model.
  - **NLTK & spaCy:** Natural Language Processing (NLP) libraries used for tokenization, lemmatization, and preprocessing of text data.
  - **Pandas:** For efficient handling of structured data and manipulation of uploaded CSV files.
  - **Matplotlib & Seaborn:** Utilized for data visualization, including bar charts, histograms, and other graphical outputs.
- **Visualization Tools:**

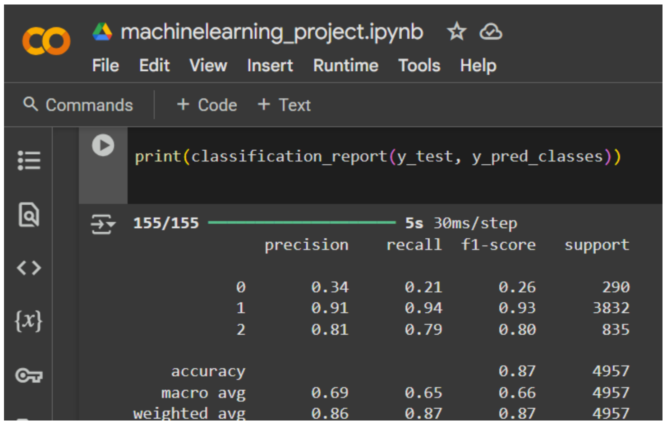
  - **Matplotlib:** Used for generating custom plots of model output such as confidence bar charts and result distribution.
  - **Seaborn:** For aesthetically pleasing and informative statistical plots, often used in performance summaries.

- **Web Framework: Streamlit** — Provides the main interface for user interaction, input processing, and result display in a browser-based environment.
- **Integrated Development Environment: Google Colab** — Used for model development, training, experimentation, and initial deployment before integration into the Streamlit app.
- **File Formats:**
  - **.h5:** The trained deep learning model is saved in this format for efficient loading and deployment.
  - **.pkl:** The tokenizer used for preprocessing is stored as a serialized Python object for reuse during inference.
  - **.csv:** Used for both input (batch data) and output (predictions), facilitating easy integration with spreadsheets or downstream applications.

VI. Evaluation Metrics

TABLE I
CLASSIFICATION REPORT

Label	Precision	Recall	F1-Score	Support
Hate Speech (0)	0.34	0.21	0.26	290
Offensive (1)	0.91	0.94	0.93	3832
Neutral (2)	0.81	0.79	0.80	835
Accuracy		0.87		
Macro Avg	0.69	0.65	0.67	4957
Weighted Avg	0.86	0.87	0.87	4957



```
print(classification_report(y_test, y_pred_classes))
```

```
155/155 ━━━━━━━━━━━━━━━━━━━━ 5s 30ms/step
              precision    recall  f1-score   support

           0       0.34      0.21      0.26       290
           1       0.91      0.94      0.93      3832
           2       0.81      0.79      0.80       835

    accuracy                           0.87      4957
   macro avg       0.69      0.65      0.66      4957
weighted avg       0.86      0.87      0.87      4957
```

Fig. 1. Classification Report Output from Google Colab

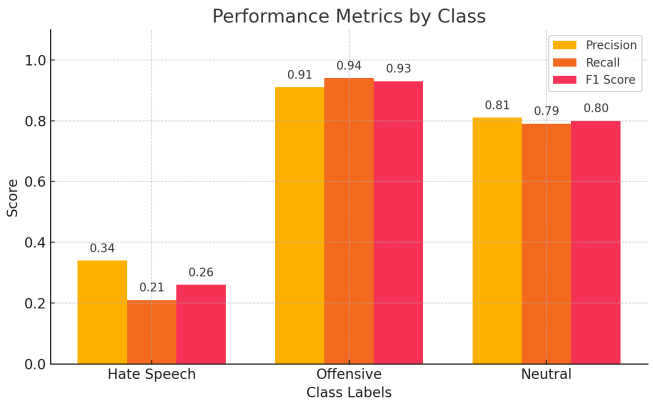


Fig. 2. Bar Chart of Precision, Recall, and F1 Score for Each Class

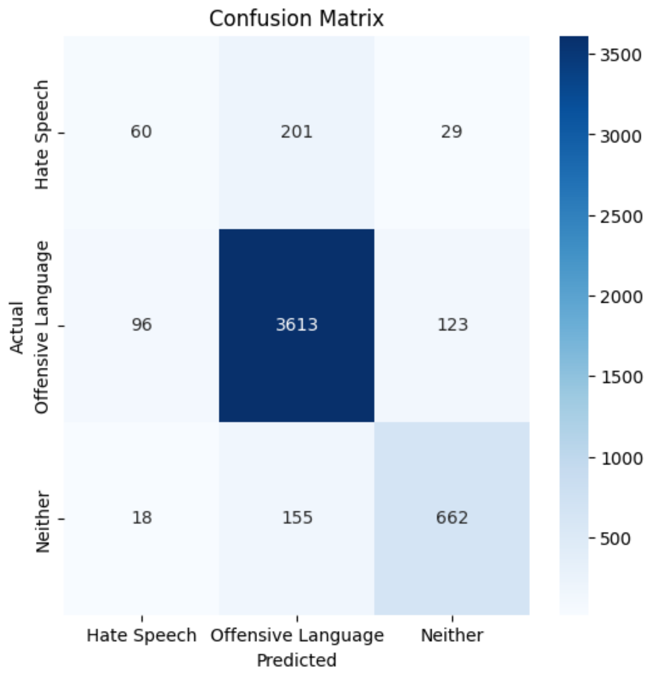


Fig. 3. Confusion Matrix (Insert your confusion matrix plot here)

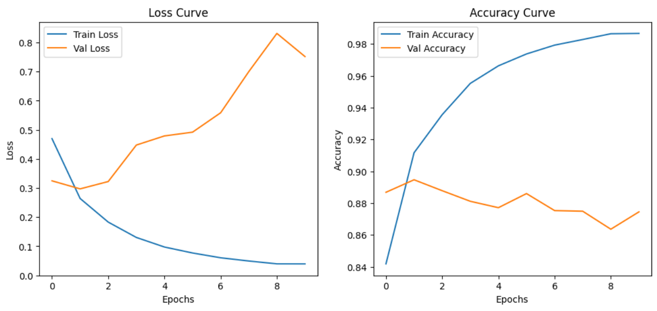


Fig. 4. Training vs Validation Accuracy and Loss (Insert plot here)

VII. Expected Outcomes

The model is expected to classify hate speech, offensive, and neutral content with high accuracy. The system provides instant prediction and supports real-world use cases for content moderation. The accessible interface ensures usability for both technical and non-technical users.

- Helps reduce hate speech propagation
- Improves platform safety and trustworthiness
- Allows flexible deployment in moderation tools
- Enables batch and single predictions
- Offers downloadable outputs and metrics for analysis
- Supports future extensibility for languages and platforms

VIII. Conclusion

The project demonstrates a successful implementation of an LSTM-based model for hate speech detection. Using real-world Twitter data, the system effectively identifies different categories of text. The integrated Streamlit app enhances the reach of the model by offering an intuitive interface. This solution has real potential in automating moderation and promoting safer digital spaces. Future enhancements can include attention mechanisms, integration with multilingual datasets, and deployment as an API for enterprise use cases.

Acknowledgments

We would like to thank our project supervisor and our institution for their support and guidance throughout this project. We also extend gratitude to the open-source community for tools, resources, and collaborative platforms.

References

[1] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated Hate Speech Detection and the Problem of Offensive Language," in *Proc. Int. Conf. Weblogs and Social Media (ICWSM)*, 2017.

[2] Z. Zhang and L. Luo, "Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter," *Semantic Web*, vol. 10, no. 5, pp. 925–945, 2018.

[3] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep Learning for Hate Speech Detection in Tweets," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 759–760.

[4] P. Mishra, M. Del Tredici, and G. Boleda, "Tackling the Class Imbalance Problem in Hate Speech Detection Using Ensemble Learning," in *Proc. 23rd Conf. Comput. Natural Lang. Learn. (CoNLL)*, 2019.

[5] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention is All You Need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 5998–6008.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol. (NAACL-HLT)*, 2019, pp. 4171–4186.

[7] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *Proc. 2014 Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1746–1751.

[8] R. Socher, A. Perelygin, J. Wu, et al., "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," in *Proc. 2013 Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 1631–1642.

[9] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive Language Detection in Online User Content," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 145–153.