HATE SPEECH DETECTION USING MACHINE LEARNING

By- Updesh Kumar

Ayushman Singh

Introduction

- The primary objective of our research is to develop an effective hate speech detection model using machine learning techniques.
- In today's digital age, with the prevalence of written communication on various online platforms, the need for identifying and mitigating hate speech has become crucial for maintaining a safe and inclusive online environment.
- Hate speech detection is the process of identifying and classifying forms of communication (text, audio, etc.) that contain hate or encourage violence against individuals or groups based on attributes like ethnicity, gender, sexual orientation, religion, or age.
- Tried to understand and implement the necessary methodology to eradicate the offensive communication.
- Our research leverages advanced natural language processing (NLP) techniques and biLSTM SNP-based models to extract meaningful features from text data, enabling us to identify patterns indicative of hate speech.

Background

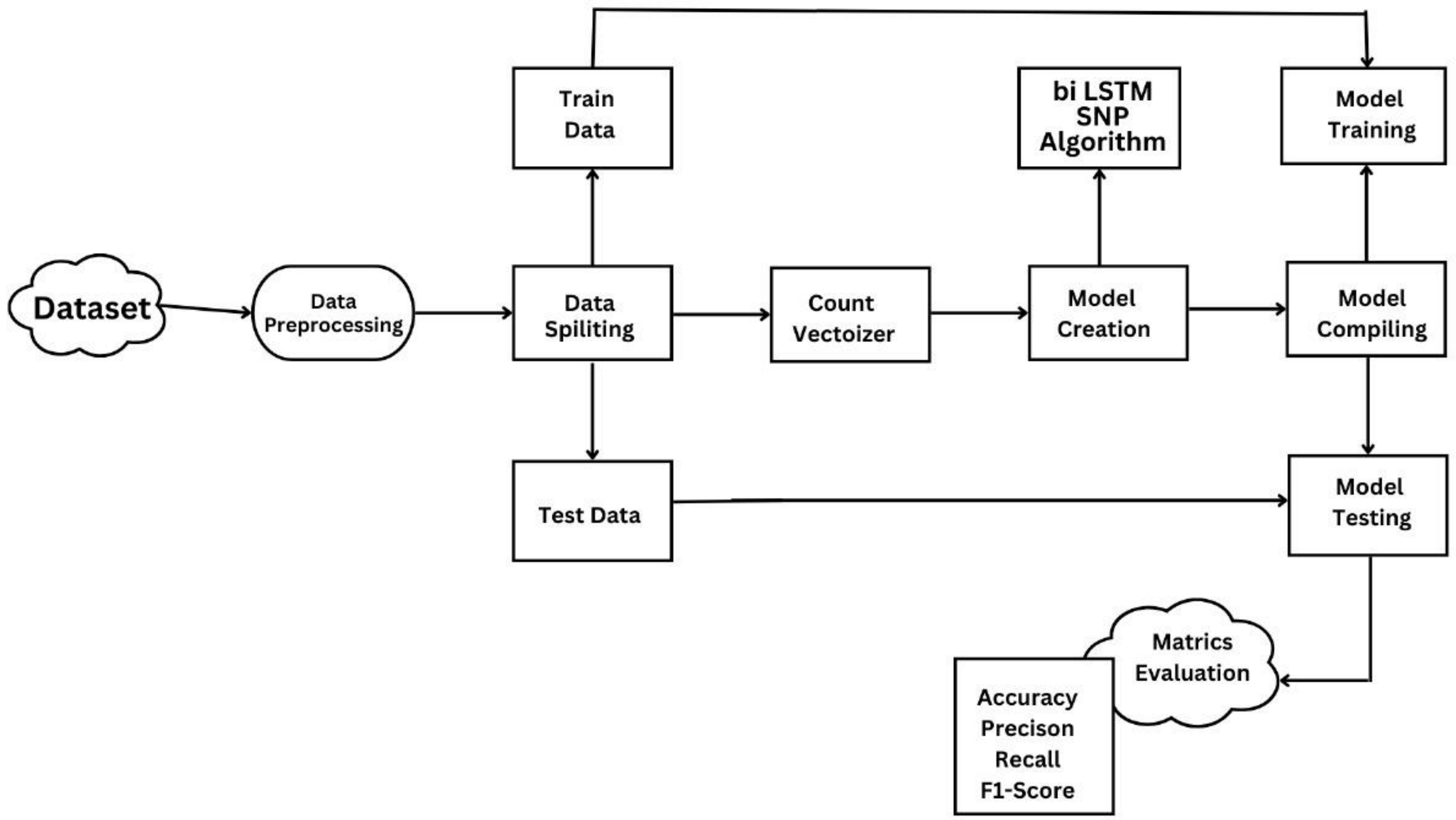
- Research on hate speech detection has explored various methods, with recent emphasis on deep learning approaches.
- Deep learning models are effective when there is a substantial, diverse, and high-quality training dataset available.
- However, in cases where such extensive datasets are lacking, previous research efforts have created smaller datasets with manual labeling, but these are limited in size.
- Numerous publicly available labeled datasets have been utilized by researchers for training and evaluating hate speech detection models.
- To enhance the training data, a large-scale corpus of generated hate and non-hate sequences has been incorporated into these datasets for various classification experiments.
- The section provides an overview of existing approaches to hate speech detection, giving insights into the methods used to classify hate speech.

Proposed Methodology

- Bidirectional LSTM (BILSTM) networks are an extension of LSTM networks that enhance their ability to capture contextual information by processing input data in both forward and backward directions simultaneously.
- BLSTM networks consist of two LSTM layers: one that processes the input sequence in the forward direction and another that processes it in the backward direction.
- The forward layer captures information from the past to the future, while the backward layer captures information from the future to the past.
- By combining the outputs of both layers, the BILSTM network can effectively capture dependencies and context from both preceding and succeeding elements in the input sequence.
- The BILSTM network follows a similar sequence processing approach as LSTM networks. It incorporates input gates, forget gates, memory cells, and output gates to process and store information over time.
- The input gate determines which information is relevant for the forward and backward layers, while the forget gate discards unnecessary information.
- BILSTM models find applications in various domains where capturing contextual information and dependencies is crucial for accurate analysis and prediction.

Methodology

- For the purpose of detecting hate speech, the offered code carries out a number of operations relating to data preprocessing, model training, and evaluation. Below is the outline figure of the work:



Methodology

Import Libraries:

- The Python libraries used in this section include NumPy and pandas for data manipulation, NLTK for natural language processing, Matplotlib and Seaborn for data visualization, Scikit-Learn for machine learning, and Keras for building neural network models.
- These libraries provide essential tools for tasks such as data handling, text preprocessing, model construction, and performance evaluation, enabling efficient machine learning and data analysis workflows.
- These libraries provide tools for data manipulation, NLP, data visualization, machine learning, and model building.

Data Collection:

- An augmented dataset is created by merging two or more existing datasets based on a shared identifier or key column.
- Merging datasets allows for the addition of new data or variables to an existing dataset.
- In this case, the 'df_twitter' and 'df_offensive' datasets, related to Twitter sentiment analysis and hate speech/offensive language, are merged after preprocessing to create a consolidated dataset named 'df.'

Preprocessing:

- Apply data preprocessing to clean and prepare text data.
- Preprocessing involves preparing and modifying input data before feeding it into a machine learning model for training or inference.
- In this section, data preprocessing is performed on text data by applying the 'clean_text' function. This function applies various steps such as lowercasing, removing URLs, HTML tags, punctuation, and stopwords, and applying stemming to standardize the text data in the 'tweet' column of the 'df' DataFrame.

Methodology

Analyze Data:

- The distribution of class labels in the combined dataset is visualized using Seaborn's countplot function. This allows us to understand the balance between hate speech and non-hate speech samples in the data.
- The shape of the DataFrame is printed to provide an overview of the dataset's size, indicating the number of rows and columns. This information helps us understand the scale and dimensions of the dataset.

Splitting the Data:

- Split the dataset into training and testing sets using Scikit-Learn's train_test_split function.
- Executed data splitting using Scikit-Learn's train_test_split for supervised machine learning.
- 'tweet' column designated as 'x' for input features; 'label' column as 'y' for target labels.
- Introduced strategic randomness to enhance model adaptability and mitigate overfitting risks.
- Ensured a balanced and representative partition for fair model evaluation.

Adding Custom SNP layer.

- Custom layer for linear transformations in neural networks.
- Initialization sets output_dim for desired output dimensionality.
- Build method establishes a trainable weight matrix (kernel) with uniform initialization.
- Call method computes the dot product for a linear transformation.

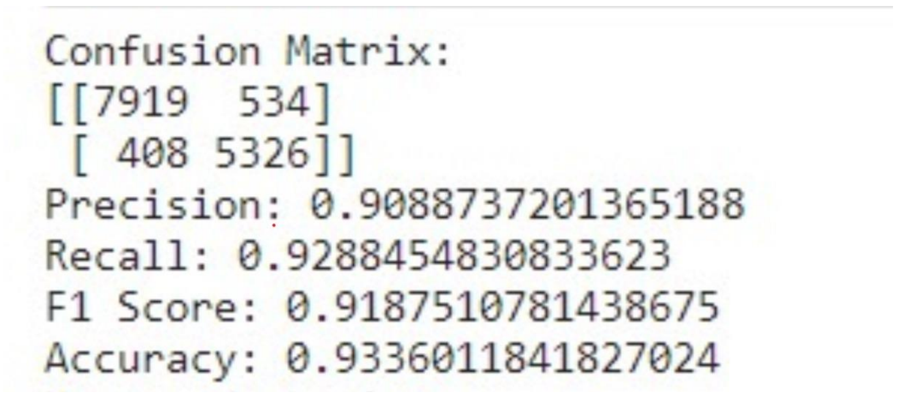
Methodology

Building Model and Prediction:

- This code segment constructs and compiles a Bidirectional Long Short-Term Memory (BiLSTM) model for binary classification. Training data is tokenized with a specified max_words and padded to ensure a uniform input size.
- The model architecture includes layers such as embedding, spatial dropout for regularization, Simple Neural Processor, Bidirectional LSTM, and a dense layer with sigmoid activation for binary classification.
- Compilation utilizes accuracy, RMSprop optimizer, and binary crossentropy loss as evaluation metrics.
- The model's architecture summary, displaying layers, output shapes, and parameters, is printed for reference.
- To enhance training efficiency and generalization, EarlyStopping and ModelCheckpoint callbacks are defined to track validation accuracy and store optimal model weights.

Evaluating the Result:

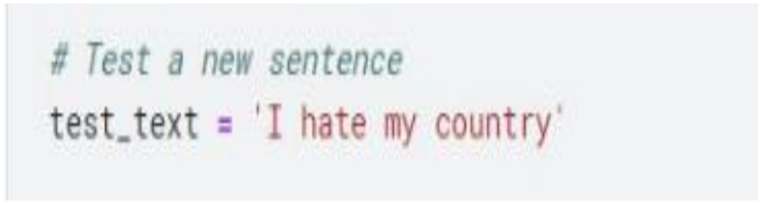
- Model performance is assessed using key metrics: precision, recall, F1 score, and overall accuracy, along with a comprehensive confusion matrix.
- Following evaluation, the Bi-LSTM model and its associated text Tokenizer are preserved for future use.
- The model is stored in an HDF5 file, while the Tokenizer is serialized using the pickle module.
- The script outlines the process for reloading the saved model and Tokenizer.
- Emphasis is placed on the potential incorporation of a custom neural processor named 'Simple Neural Processor' during the model loading phase.
- Thorough documentation and preservation are crucial for ongoing analysis, deployment, or retraining of the model.



```
Confusion Matrix:
[[7919  534]
 [ 408 5326]]
Precision: 0.9088737201365188
Recall: 0.9288454830833623
F1 Score: 0.9187510781438675
Accuracy: 0.9336011841827024
```

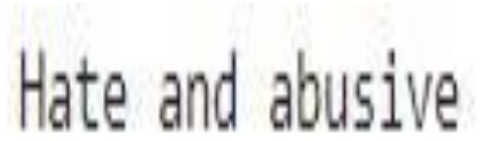
Test a New Sentence with the Loaded Model:

- Demonstrates using the trained model for predictions on new text data.
- Utilizes a sample sentence ('test_text') and preprocesses it with the 'clean_text' function.
- After tokenization and padding, a prediction is made using the loaded model.
- The result is printed, and if the prediction is below 0.5, it's classified as 'No hate'; otherwise, as 'Hate and abusive.'
- Illustrates the practical application of the trained model on real-world text inputs.
- Input:



```
# Test a new sentence
test_text = 'I hate my country'
```

- Output:



```
Hate and abusive
```

Discussion

- Comparing our analyses with machine learning algorithm.

Models	Accuracy
Proposed Model (BiLSTM SNP)	0.9336
Naive Bayes(unigram)	0.8521
SVM	0.4002
Decision Tree	0.8137
Word2vec model using LSTM	0.7815
GloVe Model using LSTM	0.829

The suggested model outperforms multiple benchmark models with a noteworthy accuracy of 93.36% .

Conclusion

Dataset Challenges:

- New hate speech datasets have size and reliability issues, with variations in language and region, making model comparisons difficult.

Definition Consensus:

- A lack of consensus in defining hate speech poses challenges for detection.

Dataset Improvement:

- There's a need to enhance existing datasets to minimize bias.

Research Contribution:

- The suggested model outperforms multiple benchmark models with a noteworthy accuracy of 93.36% .
- These models can be used for automated content moderation to combat online toxicity.

Practical Application:

- Results can improve content filtering on online platforms, fostering a safer digital environment.

Thank You