Data Glacier

Your Deep Learning Partner

Hate Speech detection using Transformers

Group Name: OMAJO
Name:Omar Hamdan
Email: o.hamdaan10@gmail.com
Country: Jordan
College: Princess Sumaya University For Technology
Specialization: NLP

16/04/2024

Agenda

Data Loading and Cleaning

Addressing Class Imbalance

Tokenization and Dataset Preparation

Model Training

Model Evaluation

Confusion Matrix Interpretation

Confusion Matrix Visualization

Data loading And Cleaning

Loading data from CSV file

- The tweet data was loaded into the environment using pandas, a powerful data manipulation library, which allows for easy handling of CSV files.
- Loading the data is the initial step to accessing and manipulating the raw dataset, enabling further analysis and processing.

Cleaning tweets to remove unwanted text (mentions, URLs, special characters)

- Tweets were cleaned to remove usernames, URLs, and special characters. This involves stripping mentions, links, and non-alphanumeric symbols that could interfere with text analysis.
- Tweets were also stripped of hexadecimal characters and hashtags to simplify the text.
- Cleaning the text is crucial for reducing noise and ensuring that the analysis focuses on meaningful words only. It helps in improving the accuracy of the model by eliminating irrelevant data.

Converting text to lowercase for uniformity

After cleaning, all text was converted to lowercase to ensure that the same words are not counted separately due to case differences.

- Normalizing the case of the text helps in maintaining uniformity across the dataset, which is vital for consistent text processing, especially when building predictive models. It ensures that the model treats words like "Hate" and "hate" as the same word, thus improving analysis accuracy.

These steps collectively prepare the data for further analysis and model training by creating a clean, consistent dataset free of extraneous elements. This preparation is critical for effective machine learning applications.

Addressing Class Imbalance

The dataset originally presented a significant imbalance between the majority class (non-hate speech) and the minority class (hate speech).

To correct this, the minority class was upsampled. This involved replicating entries of the minority class to match the number of entries in the majority class.

Specifically, the resample method from Scikit-learn was utilized, which allowed for sampling with replacement. The minority class samples were increased to equal the number of majority class samples, ensuring a balanced dataset.

The resampling process was done using a fixed random state to ensure the reproducibility of the results.

Class imbalance can lead to a model that is biased towards the more frequently occurring class. In such cases, the model might perform well overall by simply predicting the majority class, while failing to accurately identify instances of the minority class.

By balancing the dataset, the model is trained on an equal number of cases for both classes. This promotes better learning and generalization, reducing the likelihood of bias. It ensures the model is equally exposed to both categories of data, hate and non-hate speech, leading to more effective and fair classification.

This approach helps improve the sensitivity of the model towards the minority class, which is crucial for applications like hate speech detection where failing to detect actual instances of hate speech can have serious implications.

Tokenization & Dataset Preperation

Tweets underwent tokenization using the **DistilBERT tokenizer from the Hugging Face Transformers library.** This tokenizer converts the cleaned text into a format that the model can process, specifically transforming text into a sequence of tokens that represent the input in a way the neural network can understand.

The tokenization process involved specific settings such as truncation to manage token sequence length and padding to ensure consistency across all input samples, allowing them to fit the model's fixed input size requirement.

A custom dataset class, HateSpeechDataset, was then created to encapsulate the tokenized data along with their corresponding labels. This class implements the necessary PyTorch dataset methods, providing a structured way to access the tokenized data during the model training and evaluation phases.

Within this class, encodings (tokenized tweets) and labels are stored, and methods for retrieving individual items and the dataset length are defined, ensuring compatibility with PyTorch's data loading utilities.

Why it's important:

Tokenization: This step is crucial because deep learning models like DistilBERT do not understand raw text but rather work with numerical data. Tokenization breaks down text into smaller units (tokens) that can be mapped to embeddings that the model can process. Adjusting parameters like padding and truncation during tokenization ensures that the input is uniform and compatible with the model architecture, which is essential for efficient training.

Custom Dataset Class: Creating a custom dataset class provides a modular and scalable way to manage data handling. This structure is particularly advantageous when training neural networks, as it allows for batch processing of data and integration with data loaders that can automatically handle batching, shuffling, and multithreading. The class ensures that each data point is correctly formatted and paired with its label, facilitating error-free training and evaluation.

The combination of these methods optimizes the preprocessing pipeline and maximizes the efficiency and effectiveness of model training, crucial for handling large datasets typically involved in machine learning tasks.

Model Training

The DistilBERT model, a lighter version of BERT designed for faster processing while retaining most of its accuracy, was configured for sequence classification. This involves setting the model to handle two output labels—hate speech and non-hate speech.

Training parameters were carefully set to optimize the learning process. This included defining the number of training epochs (3), the batch size for training (16) and evaluation (64), along with other parameters like warm up steps (500) and weight decay (0.01). These parameters help control the training dynamics and prevent overfitting.

The model was trained using the Hugging Face's Trainer API, which provides a streamlined way to train transformers models efficiently. The Trainer handles the details of setting up the training loop, managing device placement, and evaluating the model periodically.

Why it's important:

Model Configuration: Configuring DistilBERT for sequence classification ensures the model can interpret the input data as sequences of text and apply the appropriate weights to classify them into predefined categories. This step tailors the general capabilities of DistilBERT to the specific needs of hate speech detection.

Training Parameter Adjustment: Setting appropriate training parameters is critical as they directly influence how well the model learns from the dataset. Parameters like the number of epochs and batch size balance the training speed and model performance. The warm up steps and weight decay help in stabilizing the learning process, reducing the chance of overfitting, and ensuring the model generalizes well to new, unseen data.

Effective Training: Proper training empowers the model to extract meaningful patterns from the data, crucial for achieving high accuracy in predictions. The use of the Trainer API simplifies this process, making it more accessible and reproducible.

Data Glacier
Your Deep Learning Partner

Model Evaluation

The model’s performance was thoroughly evaluated using key classification metrics: accuracy, F1 score, precision, and recall. These metrics were computed to reflect the model's ability to correctly identify and classify tweets into 'hate speech' and 'non-hate speech'.

Results from the evaluation show:

Accuracy: Reached 99.65% by the final epoch, indicating a high overall rate of correct predictions.

F1 Score: Achieved 99.65%, demonstrating the balance between precision and recall, particularly in detecting the positive class (hate speech).

Precision: At 99.30%, this score reflects the model's ability to label as hate speech only those tweets that are actually hate speech.

Recall: Perfect score of 100%, meaning the model successfully identified all actual instances of hate speech in the validation data.

Why it's important:

Comprehensive Metrics: Using a range of metrics provides a holistic view of the model's performance across different aspects. Accuracy alone can be misleading, especially in imbalanced datasets, but when combined with precision, recall, and the F1 score, a clearer picture of model effectiveness emerges.

Highlighting Strengths and Weaknesses: These metrics are critical for understanding the strengths and weaknesses of the model. For example, while a high recall indicates no missed instances of hate speech, the precision score is crucial to ensure that non-hate speech is not wrongly classified, thereby avoiding unnecessary censorship.

G**uiding Further Improvements:** The detailed results guide further refinements. For instance, if precision were lower, it would suggest a need to reduce false positives, perhaps by adjusting classification thresholds or further tuning the model.

Confusion Matrix Interpretation

A confusion matrix was generated to visually assess how well the model distinguishes between 'hate speech' and 'non-hate speech'. This matrix provides a summary of prediction results on a validation dataset and shows how predictions correspond to the true labels.

The confusion matrix was then normalized to show the proportion of predictions in each category, allowing for a clear comparison of the model's performance across different classes regardless of their frequency.

The matrix visualization was created using Seaborn, a Python visualization library, which rendered the matrix as a heatmap. This heatmap provides an intuitive display of the model’s performance, highlighting true positives, false positives, true negatives, and false negatives with varying intensities of color.

Results from the Confusion Matrix:

High True Positive Rate: The model is highly effective at correctly identifying hate speech. This indicates strong sensitivity, meaning the model is reliable at detecting instances of hate speech when they occur.

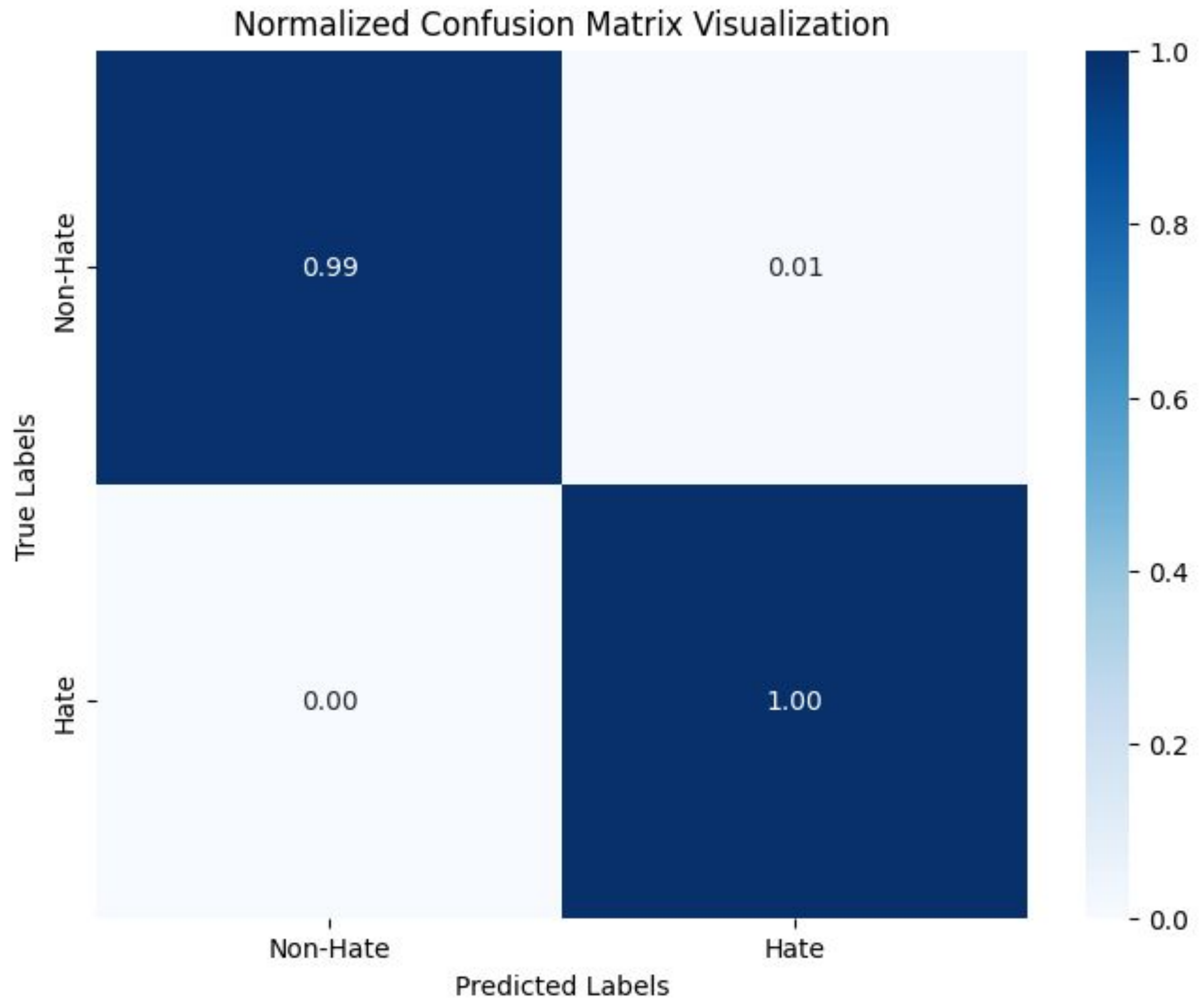
High True Negative Rate: The model also performs well in correctly identifying non-hate speech. This shows good specificity, ensuring that non-hate speech is seldom misclassified as hate speech, which is important for avoiding censorship of benign content.

Low False Positives: The minimal occurrence of false positives, where non-hate speech is incorrectly labeled as hate speech, suggests that the model is conservative in its classification of hate speech. This minimizes the risk of mislabeling innocent content as harmful, which is crucial for practical applications where such errors can have significant social consequences.

Very Low False Negatives: The near absence of false negatives, where hate speech is misclassified as non-hate speech, is particularly critical. It demonstrates that the model almost always catches hate speech, which is essential for effective moderation tools intended to create safer online environments.

Data Glacier
Your Deep Learning Partner

Confusion Matrix Visualization



Data Glacier
Your Deep Learning Partner

Thank You

Data Glacier

Your Deep Learning Partner