I Hate You!

Text Mining and Sentiment Analysis Project

Mina Beric

March 2025

UNIVERSITÀ
DEGLI STUDI
DI MILANO

Contents

1. **Introduction** 3
2. **Research question and methodology** 4
3. **Experimental results** 5
   3.1 Dataset 5
   3.2 Data preprocessing 6
   3.3 Exploratory Data Analysis 7
   3.4 BiLSTM 8
   3.5 BERT 9
4. **Evaluation and Concluding remarks** 10
5. **References** 11

1 Introduction

The detection of hate speech on social media is a crucial task. The uncontrolled spread of hate has the potential to gravely damage our society, and severely harm marginalized people or groups. A major arena for spreading hate speech online is social media. This significantly contributes to the difficulty of automatic detection, as social media posts include paralinguistic signals (e.g. emoticons, and hashtags), and their linguistic content contains plenty of poorly written text. Another difficulty is presented by the context-dependent nature of the task, and the lack of consensus on what constitutes as hate speech, which makes the task difficult even for humans. This makes the task of creating large labeled corpora difficult, and resource consuming. The problem posed by ungrammatical text has been largely mitigated by the recent emergence of deep neural network (DNN) architectures that have the capacity to efficiently learn various features. [1]

2 Research question and methodology

Building on recent advances in deep learning, this project aims to investigate the performance of two widely used neural models: BiLSTM and BERT. For this text classification task, the primary objective is to evaluate and compare their effectiveness in classifying hateful content, with particular attention to their ability to recognize and highlight the most relevant hateful terms and expressions.

Hate speech can target various minority groups based on religion, gender, physical appearance, and more. However, this research focuses specifically on racist hate speech, given in the recent years of the global rise of race supremacist movements. To this end, the dataset chosen for this study is the *Hate Speech Dataset from White Supremacist Forums*, which is publicly available on Github.

3 Experimental Results

3.1 Dataset

The dataset consists of textual data collected from Stormfront, a white supremacist online forum. A random selection of posts from various subforums was extracted and subsequently divided into individual sentences. Each sentence was then manually annotated based on specific guidelines, indicating whether it contained hate speech or not.

The original dataset includes several variables, such as `text_id`, `Text`, `file_id`, `user_id`, `subforum_id`, `num_contexts`, and `label`. For the purpose of this analysis, only the `Text` and `label` columns were retained, as the other variables were deemed redundant for the classification task.

For a total of 10,944 annotated sentences, the distribution of the labels is summarized in the table below. The majority of the sentences were labeled as `noHate`, while a smaller portion were marked as `hate`, `relation`, or `idk/skip`.

Label	Count
`noHate`	9507
`hate`	1196
`relation`	168
`idk/skip`	73

Table 1: Distribution of sentence labels in the dataset

3.2 Data preprocessing

No missing values or duplicate entries were found in the dataset. Upon inspecting the `relation` and `idk/skip` labels, it was found that they did not provide meaningful information for the purpose of this analysis. For example, a sentence labeled as `relation` was *"The one part of the video that isn't surprising."*, and one labeled as `idk/skip` was *"S: uomalainen totta kai!"*. Since these categories lack clear relevance to hate speech detection, only sentences labeled as `hate` and `noHate` were retained to avoid introducing noise or confusion into the models during training.

To prepare the data for the text classification task to be normalized and semantically informative, several preprocessing steps were applied to the `Text` column.

1. **Text normalization pipeline**: Implemented through the `normalize_text` function. This function performs several cleaning operations: it converts text to lowercase, expands contractions, removes URLs, digits, punctuation, extra whitespace and removes stopwords.

2. **Lemmatization:** Additionaly, `normalize_text` lemmatizes the text using spaCy, that employs a pre-trained model that assigns POS tags automatically, reducing both complexity and error-proneness compared to NLTK's rule-based approach.

3. **Data Augmentation:** Due to the imbalanced nature of the dataset—containing 9507 instances of `noHate` and only 1196 of `hate`—**data augmentation** was performed. Specifically, **back translation** was applied to the minority class (now with 3588 "Hate" instances) using the `googletrans` API, translating English sentences to German and back to English to generate semantically similar variants. This technique helped increase the number of hate-labeled instances and introduced diversity into the minority class, aiming to improve the robustness of the classification model.

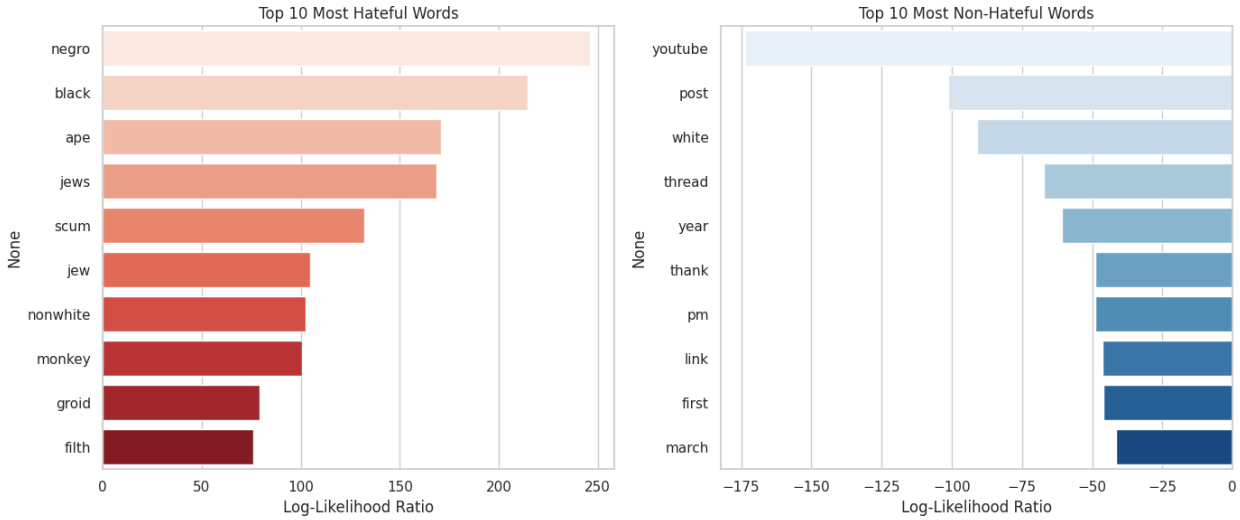
3.3 Exploratory Data Analysis

Features such as text length, word count, sentiment polarity, and subjectivity were calculated. While polarity revealed that hateful comments tend to lean more negative, there was considerable overlap between `hate` and `noHate` classes in all feature distributions. This suggests that basic statistical features alone are insufficient to clearly distinguish hate speech.

To better understand lexical distinctions between classes the **log-likelihood ratio (LLR)** was applied to word frequencies. This statistical measure quantifies how strongly a word is associated with one class over another, offering insight into class-specific vocabulary and providing an initial lexical insights that can inform downstream modeling. The LLR is computed as:

$$G^2 = 2\left[h \log\left(\frac{h}{E_h}\right) + n \log\left(\frac{n}{E_n}\right)\right]$$

where h and n are the observed counts of a word in the `hate` and `noHate` classes respectively, and E_h, E_n are the expected counts under the null hypothesis of independence. Positive values indicate stronger association with hate speech, and negative values with non-hateful speech.



3.4 BiLSTM

BiLSTM (Bidirectional Long Short-Term Memory) networks extend the standard LSTM by adding a second LSTM layer that reads the input sequence in reverse order. This allows the model to capture both past (left context) and future (right context) information for each word in the sequence. This is particularly useful in hate speech detection, where context is crucial — a word may be offensive in one context but harmless in another. The pipeline consists of the following steps:

1. Data Preparation

The dataset was split into training and test sets using stratified sampling to preserve the distribution of the label.

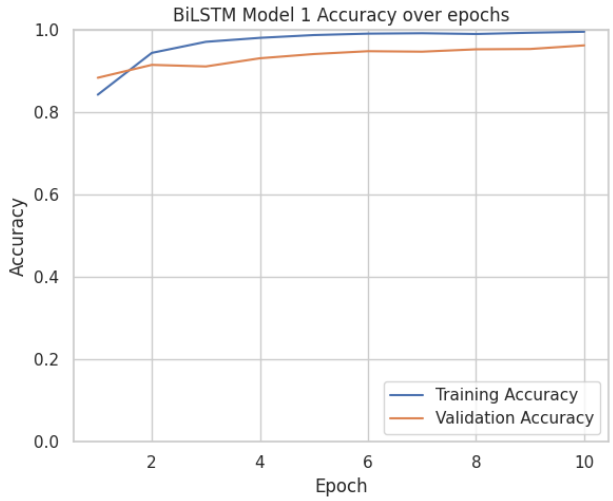
2. Tokenization and Padding

Text preprocessing was conducted using Keras' `Tokenizer`, with the vocabulary size chosen to cover 95% of the most frequent terms in the corpus. The resulting sequences were padded to a maximum length defined by the 95th percentile of training sequence lengths to reduce truncation and unnecessary padding.

3. Baseline BiLSTM Model

The initial BiLSTM model was structured as follows:

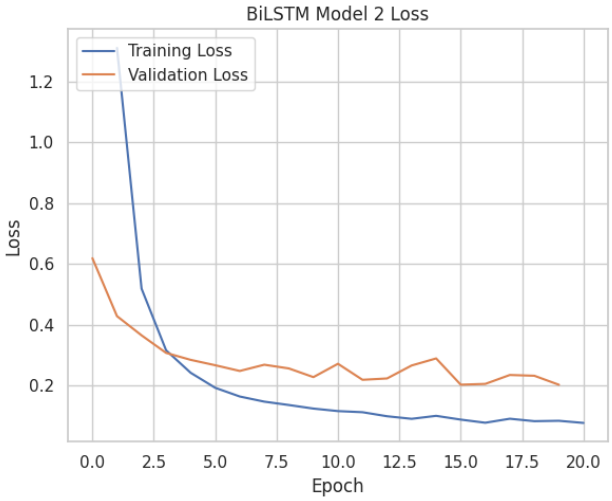
- **Embedding Layer:** Initialized randomly and set as trainable to learn word representations during training.
- **Bidirectional LSTM Layer:** 64 LSTM units to process the sequence in both forward and backward directions.
- **Dropout Layers:** Added to prevent overfitting by randomly deactivating units during training.
- **Dense Output Layer:** A single neuron with sigmoid activation for binary classification.
- To address class imbalance, computed class weights were applied during training.

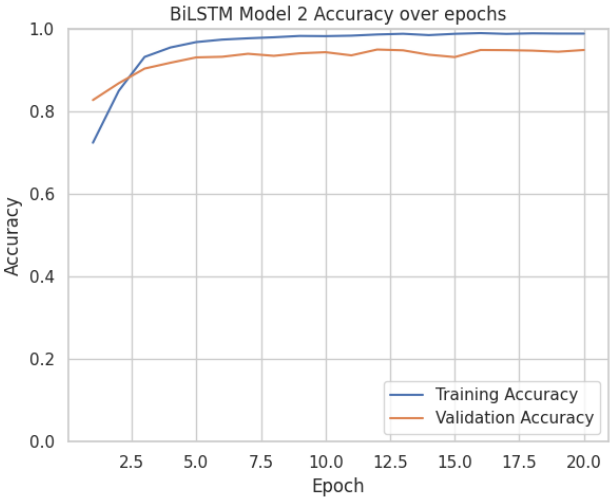


4. Regularized BiLSTM Model

To improve generalization and reduce overfitting, a regularized variant of the BiLSTM model was developed.

- **Reduced LSTM Units:** Fewer units to limit model capacity.
- **L2 Regularization:** Applied to the dense layer weights.
- **Increased Dropout:** Higher dropout rates on both embedding and LSTM outputs.
- **Early Stopping:** Based on validation loss with patience to prevent unnecessary training epochs.





5. Embedding Variants

To evaluate the influence of word representation, three types of embeddings were integrated:

- **Word2Vec:** Trained on the task corpus using the skip-gram method (vector size = 100), allowing the model to learn contextual relationships specific to the dataset.
- **GloVe:** Loaded pretrained `glove-twitter-100` embeddings via `gensim`. These vectors incorporate global co-occurrence statistics, improving semantic coherence.

- **FastText:** Trained on the corpus. Its character n-gram representation enables robustness against out-of-vocabulary words and improves performance on morphologically rich languages.

For all pretrained embeddings, the embedding layer was set as non-trainable to retain the semantic information captured during their original training.

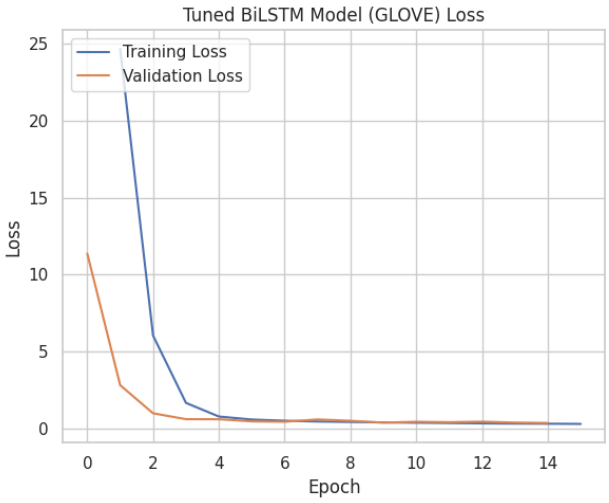
6. Hyperparameter Tuning and Final Model

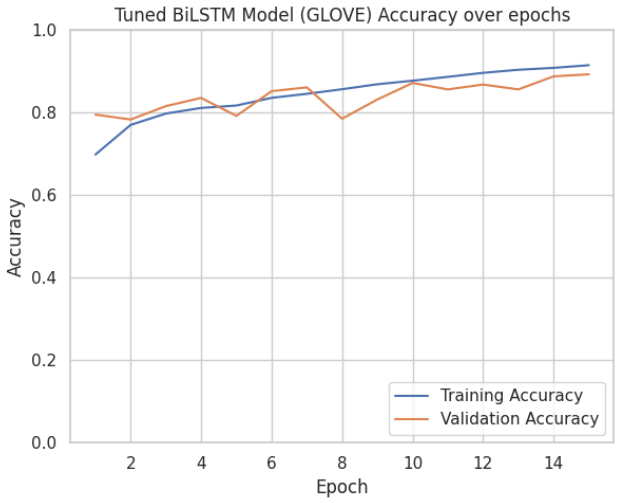
Hyperparameter tuning was performed using `Keras Tuner` with Bayesian Optimization. The search space included:

- Number of layers,
- Embedding type (Word2Vec, GloVe, FastText),
- Number of BiLSTM units,
- Dropout rates,
- L2 regularization coefficients,
- Batch size and learning rate.

The best configuration identified through tuning incorporated:

- **GloVe embeddings**,
- **3-layer BiLSTM with 64, 64, and 32 units**,
- **Dropout rate of 0.3**,
- **L2 regularization: 0.05 (LSTM), 0.02 (Dense)**,
- **Learning rate of 0.00019**.





4 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a powerful language model developed by Google, widely used in natural language processing tasks. Unlike traditional models that process text in a single direction, BERT reads the entire sequence of words at once, in both directions, allowing it to capture the full context of a sentence. This bidirectional approach makes BERT particularly effective for tasks like hate speech detection.

Built on the transformer architecture, BERT captures relationships between words regardless of their position in the sentence. It first converts the input text into tokens and processes them through multiple layers to generate contextual word embeddings.

For text classification tasks, BERT introduces a special classification token at the beginning of the input. The final representation of this token is used to make predictions, passed through a simple output layer to provide a classification score, such as distinguishing hate speech from non-hate speech.

BERT is pre-trained on large text corpora through language modeling tasks to develop a broad understanding of language. It is then fine-tuned on specific tasks, like text classification, using labeled data, allowing the model to adapt to the particular context of the task.

For the project, I've decided to use the pre-trained `bert-base-uncased` model from *Hugging Face*'s `transformers` library modified for binary classification task, with truncation and padding to a maximum lenght of 128 tokens. This version contains 12 transformer blocks, 768 hidden units, and 110M parameters.

The fine-tuning pipeline included:

- **Tokenizer:** BERT's own WordPiece tokenizer. This tokenizer splits words into subwords, enabling it to handle out-of-vocabulary words effectively. Additionally, it ensures uniform input size by padding and truncating input sequences to 128 tokens, which is essential for processing batches efficiently, allowing for parallel processing by the transformer model (e.g. [CLS], [SEP])
- **Input Formatting:** Each sentence was tokenized and converted into input IDs, attention masks, and token type IDs.
- **Model Head:** A classification head (a fully connected layer with sigmoid activation) was added on top of the [CLS] token representation to output the probability of the `hate` class.

To accelerate training while retaining the knowledge acquired by BERT during its pre-training phase, the model's lower layers were frozen. Specifically, the embedding layer and the bottom eight transformer layers were frozen, leaving only the upper layers trainable.

The model was fine-tuned using the AdamW optimizer with weight decay, a learning rate of 5×10^{-5}, and a batch size of 16. The training was carried out for 10 epochs using a stratified 80/20 train-test split. A linear learning rate scheduler was implemented with a 10% warmup period. This approach gradually increases the learning rate during the initial training phase before linearly decaying it. The warmup phase helps stabilize training in the early stages, while the subsequent decay helps the model converge to better optima as training progresses. To prevent overfitting, an early stopping mechanism was employed with a patience of 2 epochs.

To incorporate class weights into training, a custom trainer called WeightedTrainer was implemented. This trainer modifies the loss function to account for class imbalance by applying the computed class weights. As a result, the model places greater emphasis on correctly predicting the minority class, penalizing its misclassification more heavily and encouraging more balanced predictions.

5 Evaluation and Concluding remarks

As evaluation metrics accuracy, recall, precision, F1 were used.

Model	Accuracy	Precision	Recall	F1-Score
BiLSTM Model 1 (Baseline)	0.9630	0.9449	0.9658	0.9546
BiLSTM Model 2 (Regularized)	0.9496	0.9299	0.9471	0.9379
Tuned BiLSTM Model (GLOVE)	0.8931	0.8618	0.8757	0.8683
BERT	0.9300	0.91	0.92	0.92

Table 2: Comparison of Classification Metrics for Different Models

Surprisingly, the BiLSTM Model 1 (Baseline) achieved impressive results. However, despite these strong metrics, the training and validation loss trajectories suggest potential overfitting and indicating a growing gap between training and generalization performance.

In contrast, the BiLSTM Model 2 (Regularized), which includes regularization techniques to mitigate overfitting, showed a slight decrease in accuracy (94.96%) and F1-score (93.79%) but performed more consistently on the validation set, suggesting improved generalization.

The Tuned BiLSTM Model (GLOVE), although adjusted for optimal performance using GloVe embeddings, demonstrated a drop in metrics, particularly a decrease in accuracy (89.31%) and F1-score (86.83%), indicating suboptimal performance compared to the other BiLSTM models.

Despite the regularization of BiLSTM, the BERT model achieved superior results, with an accuracy of 93.00%, precision of 91%, recall of 92%, and F1-score of 92%, combined with the absence of overfitting and demonstrating its superiority w.r.t. the other implementations.

6 Conclusion

BERT significantly outperformed the BiLSTM models in both accuracy and generalization. While regularization and fine-tuning improved the BiLSTM models, they still exhibited some overfitting, making BERT the superior model for this task, particularly in classifying difficult examples that required a deep understanding of the text's context and longer dependencies. This underscores the importance of context in hate speech detection, where understanding the relationships between words and their meanings is crucial.

One of the major challenges that both models faced was the imbalanced nature of the dataset, where the majority of sentences were labeled as `noHate`. While data augmentation methods helped to mitigate this issue to some extent, it remains an area for improvement. Further techniques, such as synthetic data generation or class balancing methods, could be explored to enhance the model's performance on imbalanced datasets.

More broadly, this study confirms that deep learning models—especially contextual architectures like BERT—are well suited for hate speech detection. They provide a robust foundation for inspecting and interpreting complex textual patterns, offering promising results even in nuanced and context-dependent cases.

Looking forward, several directions could further enhance the hate speech detection task. These include extending models to multilingual and cross-lingual settings using architectures like XLM-R, and applying domain-specific fine-tuning to better capture the language dynamics of particular online platforms. Incorporating sociolinguistic features such as conversation context or user metadata may help improve classification in ambiguous scenarios. Together, these advancements could increase the predictibility of the models implemented and make the automated hate speech detection more accurate, adaptable, and socially aware.

References

[1] Fortuna, P., Soler, J., Wanner, L., & Rosso, P. (2021). Challenges of hate speech detection in social media: Data scarcity and leveraging external resources. *SN Computer Science*, 2(2), 95. https://doi.org/10.1007/s42979-021-00457-3

[2] O.de Gibert, N. Perez, A. García-Pablos, M. Cuadros. Hate Speech Dataset from a White Supremacy Forum. In: Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), pp. 11-20, 2018.