Università degli Studi di Milano

Data Science for Economics

”I hate you” text mining and sentiment analysis project

Student:
Jing Wang

ACADEMIC YEAR 2023-2024

Contents

1 Introduction

The rise of social networks in recent years has contributed to the spread of undesirable phenomena, including offensive language, hate speech, and sexist remarks, on the Internet. As a result, various efforts have been made to automate the detection and moderation of such abusive content.

2 Research question and methodology

In this project, the objective is to build models for classifying text content as offensive (e.g., racist, sexist, or otherwise) and identify the most relevant terminology for each category while extracting key aspects of hate speech.
Hate speech refers to language that attacks, diminishes, or incites violence or hatred against groups based on specific characteristics, such as physical appearance, religion, descent, national or ethnic origin, sexual orientation, gender identity, or others.
For this project, I have selected a dataset titled **Hate speech dataset from a white supremacist forum**, which is available in the following GitHub repository: Dataset link.
As shown in the next sections, I will perform the RNN algorithm in this text classification task.

3 Experimental results

3.1 Dataset

The dataset consists of files containing text extracted from Stormfront, a white supremacist forum. A random sample of forum posts has been collected from several subforums and split into individual sentences. These sentences have been manually labeled as either containing hate speech or not.
The original dataset also includes columns named 'Filename', 'file_id', 'user_id', 'subforum_id' and 'num_contexts'. These were not considered, as they do not provide meaningful insights. I retained only the 'Text' column, which contains the actual text sentences, and the 'label' column.
There are **10.944** sentences which are classified as shown in the following plot:

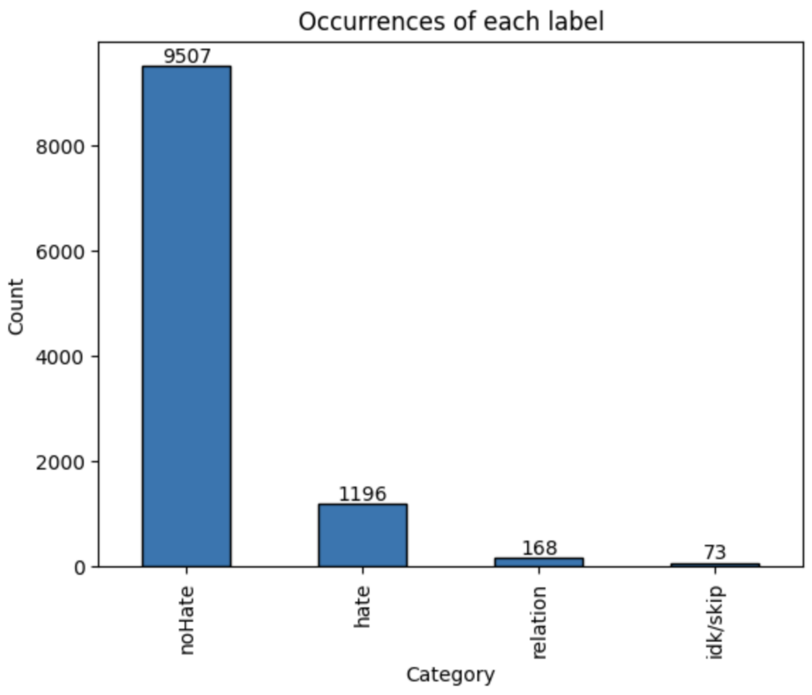

Figure 1: Data distribution

3.2 Data preprocessing

All sentences labeled as 'relation' and 'idk/skip' have been removed, as they do not contain useful information and will not contribute meaningfully to the classification task.
Here is an example of text classified as 'relation':

- Lets be honest here, how many black fathers stick around until the child is a teenager?

And an example for 'idk/skip':

- III .

The dataset is clearly imbalanced, with **9,507** sentences classified as 'noHate' compared to only **1,196** classified as 'hate.' This imbalance will be addressed in the next subsection before applying the algorithm.
Several steps are carried out then:

- Encoding is necessary since our labels are categorical as 'noHate' and 'hate'. In this step, 'noHate' is assigned a value of 0, and 'hate' is assigned a value of 1.
- The **Clear_text** function has been applied, which converts all the terms in the sentences to the lower case, removes all the web links , special characters and digits.
- Lemmatization is applied to the 'Text' column to identify the base form of words, which is crucial for enhancing the accuracy of the classification task.

3.3 PMI

Analyzing the entire dataset reveals that certain terminologies have a direct impact on sentiment classification. These words are identified using Pointwise Mutual Information (PMI), which is defined as

$$\text{PMI}(x; y) \equiv \log_2 \frac{p(x, y)}{p(x)p(y)}$$

where x is a specific terminology and y is the hateful or not hateful context.
They are shown in the following table with the their connected Hate Score(HS), which is computed as

$$HS(w) = PMI(w, HATE) - PMI(w, NOHATE)$$

The HS is a meaningful way to capture the relative association of a word w with hateful contexts versus non-hateful ones.

- Positive HS: the word w is relatively more frequent in hateful contexts compared to non-hateful ones.
- Negative HS: the word w is relatively more frequent in non-hateful contexts compared to hateful ones.

Table 1: Terminologies with Hate Rate

Words	**Hate Rate**
filth	3.5725
groid	3.5035
jungle	3.5035
libtard	3.5035
filthy	3.3494
mongrel	3.3494
ape	3.3212
sicken	3.1671
invade	3.1671
nigs	3.1671

(a) Negative terminologies

Words	**Hate Rate**
link	-2.2150
pm	-2.2489
happy	-2.2489
type	-2.2709
university	-2.3543
water	-2.3741
band	-2.3935
email	-2.3935
idea	-2.5854
thanks	-2.86115

(b) Positive terminologies

3.4 RNN

Recurrent Neural Network(RNN) is one of the most important algorithms in deep learning, used across a wide range of problems, including text classification, language detection, translation tasks and author identification.
RNN is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.

In this project, various RNN models were implemented to determine the most effective model for text classification.
The dataset was imbalanced, which posed a challenge for the algorithm's learning process. To address this issue, all sentences labeled as 'hate' were selected, and 2,500 sentences labeled as 'noHate' were randomly chosen. These sentences were then combined, shuffled, and split into training and testing sets for model evaluation.

The first model is a simple RNN composed of

- embedding layer, which serves as a learnable way to map input tokens (e.g., words or sub-words) into dense vector representations. These representations capture semantic meaning and relationships between tokens in a continuous vector space, enabling the model to process textual data effectively.
- SimpleRNN layer that is designed to process sequential data by maintaining a "memory" of previous inputs. It iteratively processes each time step in a sequence, combining the current input with information from past inputs. Here tanh activation function is selected in order to introduce nonlinearity and regulate the range of values within the hidden state.
- Dropout rate, which determines the fraction of neurons to be randomly "dropped" during training to prevent overfitting.
- Dense layer, also known as a fully connected layer, is a layer where every neuron is connected to all neurons in the preceding layer. In this case, the sigmoid activation function is used because the labels are binary, representing two distinct classes.

This model exhibits significant overfitting, as evident in the following graph:

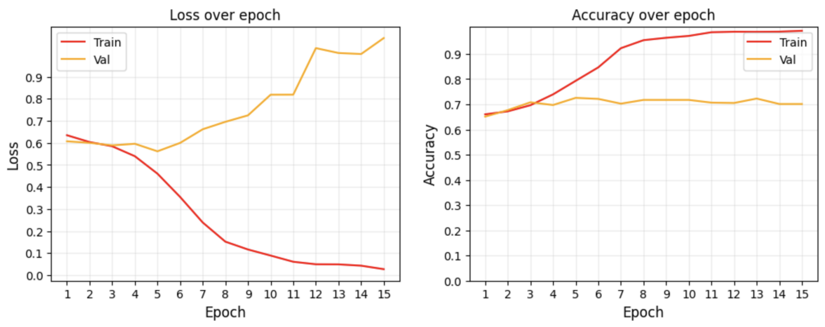

Figure 2: Confusion matrix

The overfitting issue could be attributed to the model's simplicity, which restricts its capacity to represent the data's complexity effectively. To solve this problem, a more complex model was implemented. The following elements are added:

- Bidirectional SimpleRNN layer which is a variation of Simple RNN layer that processes input sequences in both forward and backward directions. This approach enhances the model's ability to capture information from the entire context of the sequence, making it particularly useful for tasks where the relationship between elements depends on both past and future context.

- BatchNormalization layer, which is applied to normalize the inputs or hidden states to improve convergence and performance.
- L2 regularizer with $\lambda = 0.01$ was used to reduce overfitting.
- Adam optimizer with learning rate $= 0.0005$, which can help with training stability, dealing with complex data.

Hyperparameter tuning was then performed to identify the optimal hyperparameters for the RNN (model2). Using these optimal hyperparameters, the model's performance is presented as follows:

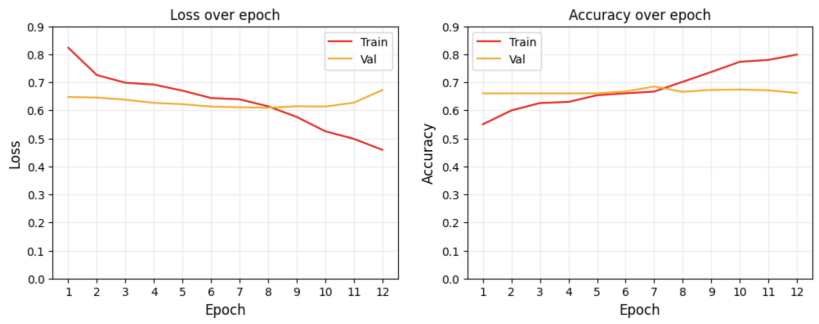

Figure 3: Confusion matrix

The model is more stable than the previous ones; however, it does not achieve high accuracy.
The last model is a Long Short Term Memory(LSTM), which is a specialized type of Recurrent Neural Network (RNN) designed to capture long-range dependencies in sequential data. Unlike traditional RNNs, which struggle to maintain information over long sequences, LSTMs are capable of retaining information for longer periods, making them highly effective for tasks like speech recognition.

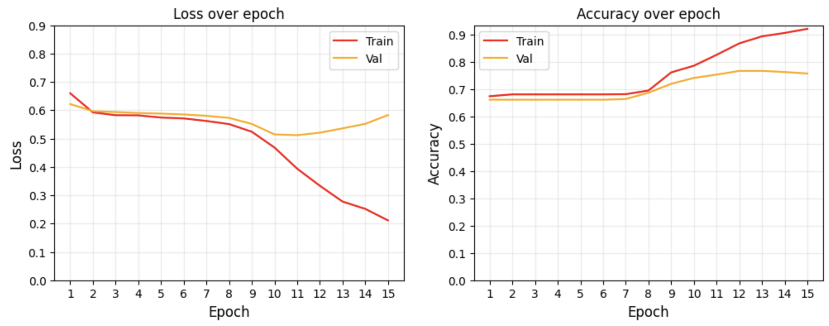

Figure 4: Confusion matrix

The performance plot clearly shows that while the overfitting issue persists, the model achieves better accuracy compared to the RNN models.
The following confusion matrix, derived from the last model discussed, identifies 92 as false positives and 88 as false negatives, which means that 24% of the sentences in the test set have been missclassified.

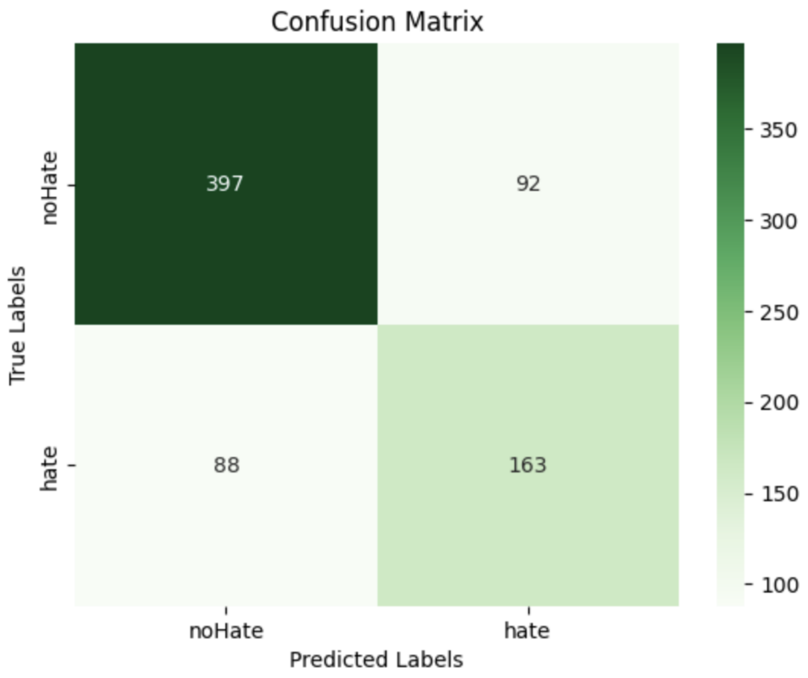

Figure 5: Confusion matrix

4 Concluding remarks

This analysis provides a solid foundation for further discussion and research on the topic of hate speech detection. For future work, it would be interesting to investigate methods for incorporating world knowledge or the context of an online conversation. By understanding the context in which a sentence is expressed, models could achieve more robust and accurate classification, reducing errors due to misinterpretation of meaning.
Additionally, future studies could explore the impact of sentences labeled as 'relation' on classification performance. These sentences were excluded from the current analysis.

5 References

1. PAULA FORTUNA,SÉRGIO NUNES,A Survey on Automatic Detection of Hate Speech in Text
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.