EmoHate: A Deep Learning approach To Detect Hate Speech using Emojis

Mischa Mez [1] **Guillaume Spahr** [1] **Carlo Opreni** [1]

Abstract

Hate speech detection remains a critical challenge in the field of natural language processing, with significant implications for online safety and social harmony. In this work, we explore the efficiency of an ensemble learning approach leveraging transformer-based models to enhance the accuracy of hate speech classification. Our methodology incorporates three pretrained transformer models, specifically fine-tuned for hate speech detection. The outputs of three of these models are integrated to serve as inputs for a fourth model, designed to provide the final classification of text as either hate speech or non-hate speech. The ensemble learning framework should leverages the strengths of each individual model while addressing and reducing the weaknesses associated with single-model approaches, unfortunately we didn't get this result. Our results show that this ensemble method achieves worse performance when combining each single model.

Keywords: Transformer Models, Ensemble Learning, Online Safety, Hate Speech Detection

1. Introduction

Hate speech detection is a crucial area of research within the field of Deep Learning. It fosters safer and more inclusive online environments. Despite significant advancements in this field, current methodologies often fall short in effectively identifying hate speech that incorporates emojis. Emojis are widely used in digital communication to express emotions, substitute words, or add nuance to text, making them a potent tool for conveying hate speech in a subtler, often more ambiguous manner. The detection of emoji-based hate speech presents unique challenges, as it requires understanding the interplay between text and visual symbols, as well as the contextual meanings that emojis can convey. The primary objective of our work is to address this gap by trying to build a robust system for detecting hate speech in texts that contain emojis. Our contribution to the already present work in this field is including an ensemble learning approach that combines multiple pretrained transformer models, each fine-tuned to recognize various aspects of hate speech, including the subtleties introduced by emojis. Our work builds upon the findings of the Hatemoji paper (Hatemoji: A Test Suite and Adversarially-Generated Dataset for Benchmarking and Detecting Emoji-based Hate), which highlights the complexities and challenges of emoji-based hate speech detection. We hypothesize that by integrating the outputs of three specialized transformer models into a final decision-making model, we can get near or even improve the accuracy and reliability of detecting hate speech based emojis. This work also addresses the current limitations in handling multimodal hate speech.

2. Related Work

Transformer-based models, including BERT, DistilBERT, and ELECTRA, represent the state-of-the-art in natural language processing[1]. Transformer models have shown high performance in hate speech detection tasks. Despite these advancements, several limitations still exist. Most studies rely on a few benchmark datasets such as Davidson, Founta, and TSA, which may not capture the full diversity of hate speech in real-world scenarios. This can lead to poor generalization to other domains. Using emoji-based datasets may help with this struggle. In our work, we use DistilBERT, a faster, cheaper, and lighter version of BERT, that outperforms other models in hate speech Detection [2]. We use the HATEMOJI paper [3] as a baseline for our work. The paper demonstrates how adversarial data generation with emojis can enhance the robustness and accuracy of hate speech detection systems. It uses two datasets, publicly available: HatemojiCheck, HatemojiBuild. The first one is used to test 3,930 short-form statements designed to evaluate the performance of hate detection models specifically on emoji-based hate. The second one is an adversarially-generated dataset of 5,912 examples created using a human-and-model-in-the-loop approach to improve model performance in detecting emoji-based hate. We train our model on the HatemojiBuild dataset and a small part of the HatemojiCheck (25%). We then evaluate and test our pipeline on the rest of the HatemojiCheck and the Dynamically-Generated-Hate-Speech-Dataset from Vidgen et al [4].

[1]Group 26.

EE-559: *Deep Learning*, 2024.

3. Method

We show in the following figure the main Worfklow to train our different models.

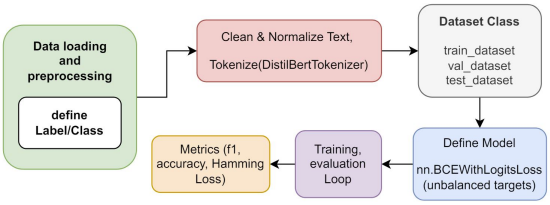

Figure 1. Training the model

From the start, the main problem that we encountered was the emojis themselves. The emojis can't be used like strings. The second problem is their utilisation. An emoji-based sentence can have a structure way different from a text-only sentence. To face these challenges, we have implemented three base models and one model for the ensembling. Here is a scheme of the whole structure:

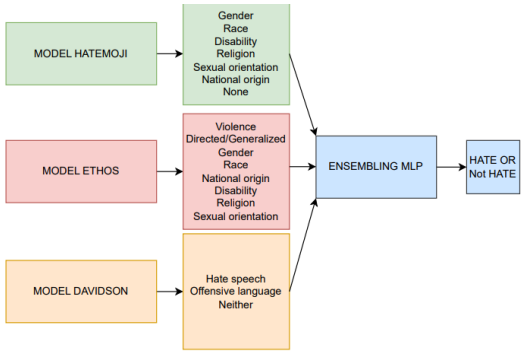

Figure 2. Structure of the model

The three base models were trained on three different datasets: Ethos, Davidson, and HatemojiBuild. The first two datasets contain text-only sentences, while the last one includes sentences with emojis. All datasets were sampled from online content and labelled by experts. We chose to train three models on different text structures, anticipating that this approach would improve generalization.

The Ethos and Davidson datasets differ in their labelling. Ethos is a binary and multi-label dataset, meaning each text can be classified into multiple categories simultaneously. In contrast, Davidson is a multi-class dataset, meaning each sentence can be classified into only one category at a time. As we will discuss in the next section, both models have achieved excellent results.

For the HatemojiBuild dataset, we use a multilabel classification method. Beforehand we need to preprocess our datasets by categorizing the initial target labels from the dataset into broader categories such as gender, race, disability, religion, sexual orientation, national origin, and none. We achieve this by defining a dictionary of category labels, where each key represents a broad category and the corresponding values are sets of specific labels. Subsequently, we clean the datasets by dropping irrelevant columns, retaining only those necessary for subsequent analysis and modeling.

Each base model uses transformer architectures. We chose the DistilBERT as already mentioned because of if effectiveness in this field [2]. Each DistilBERT model is followed by one or more classifier layers, enabling fine-tuning depending on the problem they are addressing.

Before passing the data to the models, we had to preprocess the Davidson and HatemojiBuild datasets. For the Davidson dataset, which contains posts from X (formerly Twitter), we applied several operations to remove unnecessary elements like hashtags, URLs, usernames, etc. To process the emojis, we used a library named "emojis" that converts emojis into text. We then converted some of these new texts to avoid having numbers in the words by converting them into letters. There was also the necessity to preprocess the labels. A function, **add_category_columns()**, is implemented to add binary indicator columns to the datasets, indicating the presence of each broad category in the target labels. These binary columns are then converted to integers (1 for presence, 0 for absence). We apply this function to our training, validation, and test datasets, enriching them with these new features. One further step for the Emoji based dataset involves downsampling the majority class ('none') to balance the dataset with the minority classes, ensuring equitable class representation for improved model training and evaluation. One other method is to compute the loss function with weights. This weights are computed following this process: 1. Count the occurrences of each class, 2. Calculate the class frequencies by dividing the number of occurrences of each class by the total number of occurrences 3. Compute the inverse of each class frequency and then normalize these values to obtain the final weights for each class.

After that, we tokenized the sentences using the pretrained DistilBERT tokenizer, making the datasets ready for training.

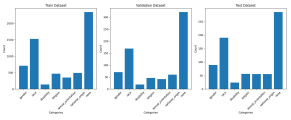

Figure 3. Structure of the MLP model

After obtaining the fine-tuned models, we implemented ensembling. We chose to use an MLP for this task. To save time and make it computationally efficient, we passed our sentences through the models and kept their outputs in

memory. We then passed these outputs into the ensembling model to train it and obtain the final results. For the training process of the MLP, we decided to vary to number of layers and the hidden dimensions that we kept constant throughout the model. As for the training set, we used a combination of HatemojiCheck for emojis and D.G.H.D. for text.

In multi-label classification, each instance can belong to multiple classes simultaneously. We therefore use a new metrics compared to the other usual ones, the Hamming loss. It treats each label prediction independently and averages the error. This is crucial because an instance may have multiple labels, and getting even one wrong affects the overall performance. It also isn't influenced by class imbalance as much as other metrics since it considers each label independently.

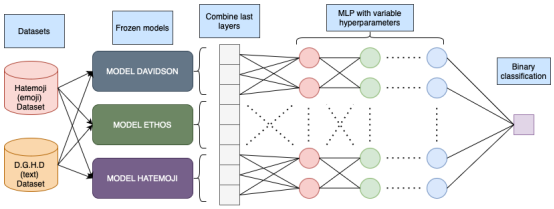

Figure 4. Structure of the MLP model

4. Validation

For the training of the Emoji and Ethos model, we employed the Hugging Face Transformers library to fine-tune our multi-label classification model. We used the **TrainingArguments()** class to define specific training parameters, including a batch size of 8 per device for both training and evaluation, and a total of 5 epochs. We retain the best-performing model based on evaluation metrics. We used an early stopping callback to halt training if no improvement in evaluation metrics was observed after one evaluation step. For the ensemble model, we used the same batch size with 10 epoch. Then, the best performing hyper-parameters of the model were chosen based on its accuracy. In conclusion, as table 2 shows, the individual models achieve a reasonable accuracy on a new dataset. However when we connect their last layer to an MLP, and test it on a combination of text and emoji, we end up with a slightly better accuracy then the individual models, showing that the MLP provides a fair generalisation but not accurate enough to be usable in real scenarios.

Dataset	F1 Score	Accuracy	ROC AUC	Hamming Loss
ETHOS	0.82	-	0.88	0.09
DAVIDSON	0.91	0.92	-	-
HatemojiBuild	0.77	-	0.87	0.07

Table 1. Performance metrics of different datasets.

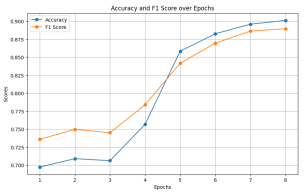

Figure 5. Evolution of the performance through the epochs for Davidson

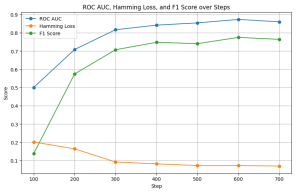

Figure 6. Evolution of the Emoji model performance for k timesteps

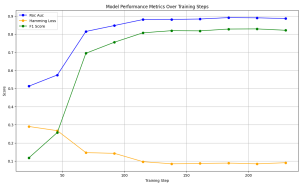

Figure 7. Evolution of Ethos model performance for k timesteps

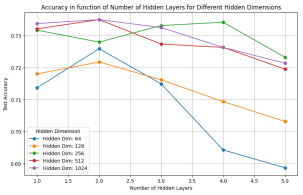

Figure 8. Evolution of the accuracy in function of the MLP's hyperparameters

Models	text-based	emoji-based	emoji text -based
DAVIDSON	0.59	0.56	0.58
Hatemoji binary	0.66	0.7	0.68
emsemble MLP	0.74	0.74	0.73

Table 2. Accuracy for testing the Ensemble model

5. Conclusion

In conclusion, we achieved excellent results with models trained on datasets based on a specific social media. However, when we tried to ensemble all models to be able to overcome the intrinsic features present in each model, we came to the conclusion that an ensemble model that uses an MLP in the last step was not sufficient to achieve to same accuracy as for individual models. Their can be many reasons for that, for example the features learned by the individual models might not be compatible or complementary when combined. This can result in the MLP struggling to find a coherent representation that captures the strengths of each model. Also, we can hypothesis that the mlp model struggles to get good results, because it works with a dataset that is quite different in it's structure then the previously used datasets for the single models.

References

[1] J. S. Malik, H. Qiao, G. Pang, and A. v. d. Hengel, "Deep learning for hate speech detection: a comparative study," *arXiv preprint arXiv:2202.09517*, 2022.

[2] R. T. Mutanga, N. Naicker, and O. O. Olugbara, "Hate speech detection in twitter using transformer methods," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 9, 2020.

[3] H. R. Kirk, B. Vidgen, P. Röttger, T. Thrush, and S. A. Hale, "Hatemoji: A test suite and adversarially-generated dataset for benchmarking and detecting emoji-based hate," *arXiv preprint arXiv:2108.05921*, 2021.

[4] B. Vidgen, T. Thrush, Z. Waseem, and D. Kiela, "Learning from the worst: Dynamically generated datasets to improve online hate detection," *arXiv preprint arXiv:2012.15761*, 2020.