H1: Offensive Language Identification

Ankit Kuswah
MT21010
ankit21010@iiitd.ac.in

Hanzalah Firdausi
MT21027
hanzalah21027@iiitd.ac.in

Mahvash Fatima
MT21126
mahvash21126@iiitd.ac.in

Rameez Raja
MT21067
rameez21067@iiitd.ac.in

Abstract

The report summarizes our work for the class project on Natural Language Processing entitled 'H1: Offensive Language Identification', which deals with solutions to identify the offensive language in social media content based on deep learning techniques and classical machine learning algorithms such as Ensembling and SVMs.

1 Problem definition

The massive amount of data on social media on a daily basis with almost no filters gives rise to the mixing of informative and malicious content. The rate at which this data is being generated makes manual reviewing impossible. However, with all the negativity, ,anger and rage reflected by users' posts,cyber bullying being another consequence of the virtual freedom of speech provided by social networking sites,filtering of content,before it influences or in a lot of cases, provokes millions of people makes automatic classification of hate speech and offensive language a very significant problem. Therefore,Machine learning methods to handle this are a necessity to make the internet a better place.Offensive language detection can be considered a classification problem where tweets or posts are fed as input to the system, which outputs whether the post is offensive or not. As shown in (Chatzakou et al., 2017), on Twitter, each tweet provides a limited context. Many times, the offensive words may be hidden, or the content may contain irony or sarcasm and can be offensive without appearing so. Despite several attempts by researchers to solve the problem, the task of offensive language detection does not have a well-defined strategy and may be complicated. This work is our attempt to build a solution to detect offensive language using deep learning techniques effectively.

2 Related works

Many researchers have carried out work in the field of offensive language detection. Some of the notable ones are mentioned below.

Unsupervised learning methods applying NLP (Chen et al., 2012) or AI-based solutions (Warner and Hirschberg, 2012) has been used to detect offensive messages in text. AI-based solutions have proven to be less effective. (Waseem and Hovy, 2016) put another unsupervised learning-based solution defining a set of criteria to classify a tweet as offensive. They demonstrated that the detection performance is only little impacted by variations in user geography.

(Pérez et al., 2020) proposed a single multilingual offensive language detection system, a fine-tuned version of Multilingual BERT (Devlin et al., 2018). The authors use datasets for five proposed languages: Arabic, Danish, English, Greek, and Turkish, as described in (Zampieri et al., 2020). They use a fine-tuned version of Multilingual BERT, which contains a transformer stack (Vaswani et al., 2017) pre-trained in masked language model and NSP. They used a pre-trained multilingual BERT-base model consisting of 12 transformer blocks, 12 self-attention heads, and a hidden layer size of 768 with a linear layer on top and a sigmoid activation function to the outputs. In monolingual training, they trained the model on each dataset to test specific languages and obtain the reference values. In multilingual training, they trained the model on all the training sets concatenated. The single model proposed had a performance matching top-performing systems. The multilingual BERT version had comparable results to the monolingual versions showing that adding languages doesn't contribute to performance. Further zero-shot experiments showed no transference learning proving that a classifier trained in one language doesn't perform well when

tested in a different one. Also, few-shot experiments had better results.

(Yao et al., 2020) built several pre-trained models for multilingual offensive language recognition. The problem was divided into 3 subtasks, namely, offensive language recognition, automatic classification of attack types, and attack target recognition. They used OffensEval2020 dataset containing 13240 tweets from the OLID dataset and around 10 million tweets on offensive language identification marked by the machine. The authors filtered the extended dataset based on a threshold value of confidence 0.88 to overcome any issues caused by the imbalance of the dataset. They performed preprocessing steps of replacement of user tags and URLs by standard ones, emoji substitution, lemmatization, lowercasing, etc. The authors used BERT and RoBERTa as their base models. BERT uses the multi-head transformer structure (Vaswani et al., 2017) and considers the context in a bidirectional manner, therefore, developing a deeper understanding and saving long-distance dependencies. It comprises NSP(Next sentence Prediction), making it fit for relationship recognition between sentences. RoBERTa, compared to BERT, uses more data to train and considers NSP's judging criteria to be very simple, therefore, removes it. Due to its large amount, the data was biased in understanding shallow features, so it was used to fine-tune RoBERTa. It was then used for the offensive language recognition task. The transformer had 12 layers with a size of 768 and 12 self-attention heads with softmax classification heads on top of the pre-trained language model. The pre-processed training dataset was used as input to fine-tune the classification model. Some of the hyper-parameters used in fine-tuning training were sentence length, batch size, the learning rate, weight decay, and the number of epochs. The authors compare BERT and RoBERTa on the dataset and conclude the RoBERTa model can achieve better results on large-scale datasets. RoBERTa performed well and was found to have a macro-F1 score of 0.9128.

(Hakimov and Ewerth, 2021) analyze the effects of combining multiple textual features to detect the hateful, offensive language in the tweet text. They use three Identification in English and Indo-Aryan Languages (HASOC): HASOC-2019, HASOC-2020, and HASOC-2021 for the evaluation. Datasets include two large classes: Hate and Offensive (HOF) and Not Hate or Offensive (NOT). Since all the datasets were not balanced, they randomly picked the data point from the minority class and duplicated them to equalize them with the majority class. Their model architecture is built on top of three textual features. The first one is the BERT Encoder, in which the author analyzes two pre-trained BERT models - a BERT-Base[Dlevin], and a HateBert[by Caselli]. Both models give the output a sequence of 768-dimensional vectors. The second one is the Character Encoder, in which tokens are converted into the one-hot encoding of characters in English (a-z). The third one is the Hate Words Encoder, in which they collected the list of 1493 hate terms provided by (Gomez et al., 2020) and extracted them manually. This encodes the output in a 1493-dimensional vector. In the beginning, to extract feature vector representations, the tokens are fed into BERT, Character, and Hate Words encoders. Now extracted features are fed into separate components(RNNs) to obtain a one-dimensional vector. This vector fed three different blocks (containing a Linear layer, normalization, and a ReLu) to get probabilities.

(Saha et al., 2021) performed two tasks. First, they investigate the current performance of multilingual models in these Dravidian languages - Tamil, Kannada, and Malayalam. The second task is how they can use ensembling methods to improve classification performance. Dataset is available for Tamil (Chakravarthi et al., 2020b), Kannada (Hande et al., 2020), and Malayalam (Chakravarthi et al., 2020a). The Dataset contains row text and corresponding labels-not-offensive, offensive-untargeted, offensive-targeted-individual, offensive-targeted-group, offensive-targeted-other, or not-in-indented-language. The Malayalam split of the Dataset does not contain a row with the 'Offensive-targeted-other' label. So, they used five classes only instead of six for classification. The Dataset contains 85.5, 84.2, and 83.0 percent misspelled words in Tamil, Malayalam, and Kannada, respectively, in the context of the Dakshina Dataset. Initially, the authors used machine learning-based algorithms such as random forests and logistic regression and trained them with TF-IDF vectors. The best result they obtained with ExtraTrees Classifier (Geurts et al., 2006) with 0.70, 0.63,

and 0.95 weighted F1-scores on Tamil, Kannada, and Malayalam, respectively. Secondly, they fine-tuned different state-of-the-art multilingual BERT models on the given Dataset to improve the performance. That includes XLM-RoBERTa (Conneau et al., 2019), multilingual-BERT (Devlin et al., 2018), Indic BERT, and MuRIL9. We also pre-train XLM Roberta-Base. All the models were also fine-tuned using unweighted and weighted cross-entropy loss functions. They trained using HuggingFace with PyTorch. Further, they propose a new BERT-CNN fusion classifier where they train a single classification head on the concatenated embedding from different BERT and CNN models. Among the individual transformer models, the best performance was obtained using XLM-RoBERTa-large (XLMR-large) in the Tamil dataset and Custom XLM-RoBERTa-base (XLMRC) in the Kannada dataset.

3 Methodology

We have used four methods to train our datasets. Each of the methods is described below:

3.1 TF-IDF model

TF-IDF or Term Frequency - Inverse Document Frequency algorithm uses the frequency of words to determine their relevancy to a given document. The process to classify words as offensive or not offensive using TF-IDF was carried out as follows:

1. Performed pre-processing of data
2. Tokenized words and count frequencies
3. Found TF and IDF for words
4. Vectorized vocabulary
5. Passed vectors to a classifier such as SVM or Logistic Regression to get the labels

3.2 BERT-based models

BERT, or Bidirectional Representation for Transformers, was proposed by researchers at Google AI in 2018. BERT has state-of-the-art accuracy in performing various NLP tasks, such as classification or question-answer tasks.
The following tasks were performed while evaluating BERT-based models:

1. Performed pre-processing of data
2. Imported various BERT models from Hugging Face
3. Fined-tuned BERT models with classification head
4. Performed hyperparameter tuning of models
5. Tried different BERT models to evaluate results
6. Trained different datasets containing hate speech and offensive text
7. Tried stacked ensembling using 5 different BERT classifiers trained on various datasets containing hate speech and offensive text

3.3 Stacking Ensemble

Stacking, also known as Stacked Generalization, is an ensemble machine-learning technique. It combines the prediction of multiple machine-learning models on the same dataset. This method addresses the two questions:

1. Do the given machine learning models fit a problem?
2. How do we choose which model to use?

To find the solutions to the above questions, we use other machine learning models that learn when to use each model in ensemble. A stacking model contains two or more base models, also known as level-0 models, and a meta-model that combines the predictions of the base model, also known as level-1 model.
Level-0 Models : Level-0 Models: These models fit on the training data and whose predictions are compiled.
Level-1 Model: These models learn how to best combine the predictions of level-0.
The predictions of base models on sample data are used to train the meta-model. Now, the dataset that is not used to base-model, that data is given to the base model to get predictions; these predictions are pair of input and output and use to fit the meta-model. The outputs from the base model may be real value, probability, or class labels for the classification. The best way to prepare the training dataset for the level-1 model is k-fold cross-validation of the base model. After preparing and training the dataset for the meta-model, the meta-model can be trained individually on this

Table 1: Experimental Results for various models

Model Type	Macro F1
TF-IDF	0.40
Ensembling	0.765
RoBERTa-based-Offensive-finetuned	0.843
RoBERTa-based-hate-finetuned	0.793
Hate-BERT	0.765
English-Ebusive-MuRIL	0.768
DistilRoBERTa-tweets-hatespeech-finetuned	0.775
Neural-Architecture Search	0.794

dataset, and the base model can be trained on the entire original training dataset.
Base models are generally complex and can be linear models, decision trees, support vector machines, neural networks, and more.
Meta-models are often simple such as linear models.

3.4 Neural Architecture Search

The neural architecture search is used to discover the best neural network architecture for a specific need. NAS automates the human effort to find the best neural architecture. This domain represents a set of tools and techniques that will test and evaluate a large number of architectures across search space using search strategy and get the best fit model for that problem, maximizing the fitness function. NAS is very expensive, and we are still determining how it will be on real-world data.

4 Experimental Results

The Experimental Results consisting primarily of Macro F1 scores, along with the model types, have been summarised in the Table 1

It can be noted that the TF-IDF model was the worst performing while Stacked Ensembling had a significant improvement over it. Among the BERT-based models, the RoBERTa-based-offensive-finetuned was the best performing, and HateBERT was the lowest performing model. In the end, the Neural Architecture Search model had a macro F1 score of 0.794. Neural Architecture Search could not run completely on our systems due to limited computation resources.

5 Analysis

The dataset is a bit noisy and there are a lot of out-of-vocabulary words, which hampers the word token vectors. Although we achieved a validation f1 of 0.834, it would have been better if the texts were less noisy and contained fewer out-of-vocabulary words with fewer spelling mistakes and fewer grammatical incorrectness.

6 Contribution of each member

While working on BERT-based classifiers, each member performed fine-tuning and hyperparameter tuning of one of the BERT models. Mahvash worked on the TF-IDF classification model, Rameez worked on the Stacked Ensembling model and Ankit and Hanzalah worked together on Neural Architecture Search. All group members together worked on documentation to prepare report and presentation slides.

References

Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Mean birds: Detecting aggression and bullying on twitter. In *Proceedings of the 2017 ACM on web science conference*, pages 13–22.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 71–80. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep

bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Raul Gomez, Jaume Gibert, Lluis Gomez, and Dimosthenis Karatzas. 2020. Exploring hate speech detection in multimodal publications. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1470–1478.

Sherzod Hakimov and Ralph Ewerth. 2021. Combining textual features for the detection of hateful and offensive language. *arXiv preprint arXiv:2112.04803*.

Juan Manuel Pérez, Aymé Arango, and Franco Luque. 2020. Andes at semeval-2020 task 12: A jointly-trained bert multilingual model for offensive language detection. *arXiv preprint arXiv:2008.06408*.

Debjoy Saha, Naman Paharia, Debajit Chakraborty, Punyajoy Saha, and Animesh Mukherjee. 2021. Hate-alert@ dravidianlangtech-eacl2021: Ensembling strategies for transformer-based offensive language detection. *arXiv preprint arXiv:2102.10084*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the second workshop on language in social media*, pages 19–26.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Yinnan Yao, Nan Su, and Kun Ma. 2020. Ujnlp at semeval-2020 task 12: Detecting offensive language using bidirectional transformers. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2203–2208.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020). *arXiv preprint arXiv:2006.07235*.