Project Report

Aman Khan, Manav Saini, Mudit Balooja, Navidha Jain

1 Problem definition

We had chosen the H1 task. Our problem was: Given a tweet, we had to determine whether it is offensive or not. Here offensive is used as an umbrella term to determine any form of profane, abusive, hateful, abusive language being used.

Due to the prevalence of offensive content in social media, governmental agencies, online communities, and social media platforms are gravely concerned. One of the most popular approaches to the issue is to develop software that can identify offensive information and flag it for the deletion or manual moderation. Several studies have been done in the last few years on the use of computational methods to solve this issue.

The dataset given to us distinguishes between: Not Offensive (NOT): Posts that do not contain offense or profanity.

Offensive (OFF): Posts containing any form of non-acceptable language (profanity) or a targeted offense, which can be veiled or direct. This includes insults, threats, and posts containing profane language or swear words.

We have to train a model or use a pre-trained model to identify and classify texts into the above classes.

2 Related Works

Prior work has studied offensive language in Twitter (Xu et al., 2012; Burnap and Williams, 2015; Davidson et al., 2017; Wiegand et al., 2018), Wikipedia comments [1], and Facebook posts (Kumar et al., 2018). Previous studies have looked into different aspects of offensive language such as the use of abusive language (Nobata et al., 2016; Mubarak et al., 2017), (cyber-)aggression (Kumar et al., 2018), (cyber-)bullying (Xu et al., 2012; Dadvar et al., 2013), toxic comments [2], hate speech (Kwok and Wang, 2013; Djuric et al., 2015; Burnap and Williams, 2015; Davidson et al., 2017; Malmasi and Zampieri, 2017, 2018), and offensive language (Wiegand et al., 2018). A typology that dis-

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[2] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

tinguishes between whether the (abusive) language is directed towards a specific person or entity or towards a generalised group, as well as whether the abusive content is explicit or implicit, is needed, according to Waseem, who recently analysed the similarities between various approaches proposed in previous work. Wiegand expanded on this concept by using tweets in German. They experimented with a task on detecting offensive vs. nonoffensive tweets, and also with a second task on further subclassifying the offensive tweets as profanity, insult, or abuse. [3]

3 Methodology

3.1 Preprocessing

We have performed a few preprocessing functions on the data.

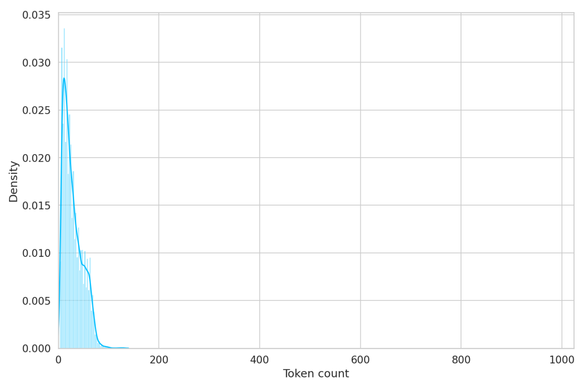
- Converting all the fields into string in order to avoid the nan fields that might be present.

- Lowercasing all the word in the dataset.

- Removing all mentions of @user, url from the dataset

- Removing any extra whitespaces present

- Removing all the tokens that are that are not alphabets since punctuation and numbers do not play a role in Sentiment Analysis.



Further the data was tokenized, padded and truncated to 100 tokens. The “OFF” label was converted to a 0 and the “NOT” label was converted to a 1.

4 Models Implemented

4.1 BERT

A stack of encoders in the BERT design read the text inputs. It processes a word list that is sent as input up the encoder stack. Each layer performs self-attention before transmitting its findings to the following encoder via a feed-forward network.

BERT employs Transformer, an attention mechanism that determines the contextual relationships among words or subwords in a text, i.e., determines the relative importance of each word in the phrase by examining its precise placement inside the sentence.

While the attention mechanism pays attention to one in-

[3] https://aclanthology.org/N19-1144.pdf

dividual word in a sentence at a time, the Transformer examines entire sequences of tokens at once and ignores directionality.

A Decoder makes predictions based on the text inputs. Since BERT uses MLM (Masked Language Modelling) which processes each masked token in the Encoder , it does not need a Decoder. BERT model only uses the Encoder part of the Transformer. [4] [5]

In our implementation, we use a pre-trained BERT Model. Using the Trainer class of Transformers, we train the model on the pre-processed data. The model is trained for 4 epochs and fine-tuned by setting the learning rate to 5e-6 and weight decay to 0.01. We see that after each epoch, the model converges very fast as both training and validation loss reduces. On the test dataset, we achieve a macro F1 score of 0.81.

4.2 XLNET

XLNET is a generalized autoregressive model where the next token is dependent on all previous tokens. XLNET is “generalized” because it captures bi-directional context by means of a mechanism called “permutation language modeling”. It integrates the idea of auto-regressive models and bi-directional context modeling, yet overcomes the disadvantages of BERT.

The Autoregressive and Autoencoding LM techniques were combined to create a new pre-training strategy. In essence, sampling is carried out from n! permutations of an n-token lengthy sentence, and given the original positional encoding of that sentence, they attempt to predict each token along the permutation, given the original positional order. XLNET uses two-stream self-attention to do this, with a context representation and a query representation.
[6]

The first step we took while training XLNET was to preprocess the data and divide the data into train set and the dev set. We used the function train_test_split to split the data into train and val set. The following hyperparameters were used:

- EPOCHS = 3
- no_decay variable was declared with bias, layernormbias and layernorm.weight to replicate the original optimizer
- Total_steps were declared by calculating the length of the train_data_loader and multiplying it with number of EPOCHS

[4] https://medium.com/mlearning-ai/twitter-sentiment-analysis-with-deep-learning-using-bert-and-hugging-face-830005bcdbbf

[5] https://medium.com/analytics-vidhya/introduction-to-bert-and-its-application-in-sentiment-analysis-9c593e955560

[6] https://towardsdatascience.com/xlnet-explained-in-simple-terms-255b9fb2c97c https://medium.com/swlh/using-xlnet-for-sentiment-classification-cfa948e65e85

- Scheduler was declared with steps and optimizer.

Two main functions were declared to train_epoch (to train the model) and eval_model (to evaluate the model) . train_model function trained the model and calculate the accuracy using the input_ids, attention_mask, targets, outputs and logits. eval_model function evaluated the current model using the val_data_loader. In the end if the accuracy returned by the eval_model function was greater than previous one then the model was saved. In the end we declared and used predict_sentiment function and transformed the text to the form that can be fed to the trained model and returned 1 or 0 according to the probabilities of being offensive text or not.

4.6 ROBERTA

Roberta, A Robustly Optimized BERT Pretraining Approach., initially presented by researchers at Facebook and Washington University. It is based on Google's BERT model developed in 2018. Both models follow similar architecture, but changes were made during the pre-training and training phase. Key hyperparameters were modified, removing the next-sentence pretraining objective and training with much larger mini-batches and learning rates. [7]

In our implementation, we used a pre-trained case-sensitive RoBERTa model which uses the MLM (Masked Language Modelling) objective. Since the model was pre-trained on raw text, we have kept initial pre-processing to a minimal. The model was trained using Trainer class of Transformers. For pre-processing, the data was tokenized, padded and truncated to 100 tokens. Learning rate and weight decay was fine-tuned to 5e-6 and 0.01 respectively. The batch size was set to 32 and the model was trained for 4 epochs. At the end of each epoch, training loss was computed and the model was evaluated against the validation data. It achieved a macro F1 score of 0.83512 on the limited test dataset on Kaggle; the best of our models.

[7] https://arxiv.org/abs/1907.11692

5 Contributions

All members equally contributed to the project and the work was all mixed up. There was no one person working on a specific thing.

4.3 BERT

Epoch	Training Loss	Validation loss	Accuracy
1	0.519	0.458	0.793
2	0.414	0.459	0.796
3	0.364	0.472	0.802
4	0.335	0.485	0.803

Table 1: We can see that as the number of epochs increase, the accuracy increases and the training loss decreases so we can say that the model converges fast.

4.4 XLNET

Epoch	Training Loss	Validation loss	Accuracy
1	0.616	0.556	0.765
2	0.597	0.828	0.772
3	0.518	0.918	0.785

Table 2: We can see that as the number of epochs increase, the accuracy increases and the training loss decreases so we can say that the model converges fast.

4.5 ROBERTA

Epoch	Training Loss	Validation loss	Accuracy
1	0.499	0.433	0.814
2	0.417	0.434	0.820
3	0.385	0.441	0.808
4	0.361	0.448	0.808

Table 3: We can see that as the number of epochs increase, the trainig loss decreases but the accuracy decreases.