BRAC UNIVERSITY

Inspiring Excellence

Final Project

# Offensive Language Detection

Name: Asim Baidya
ID: 20301239
Course Code: CSE440

## Introduction

Offensive content has become a major part of the social media. Due to the nature of social media, for example, spreading rates of offensive language, it is crucial to detect any harsh words that can create conflict between community. In this task, I worked on a Twitter dataset called(OLID) that contained around 14000 tweets that are labeled under 3 categories. In this task, we experimented with two different types of deep learning models to get desirable results.

## Data

As a part of the competition, the OLID dataset was previously provided. The dataset was scrapped from Twitter using Twitter's API. As the nature of Twitter, most of the tweets collected contain contractions and short sentences that may not form full grammar but are well enough to hurt someone's feelings or mock a certain person or community. After collecting, the creator of the dataset used various techniques to make a balance between offensive and non-offensive texts. Primarily, Twitter is a global platform thus not half or even quarters of the tweets are offensive. it's a tiny number of tweets compared to the overall uses of Twitter and tweets. To resolve this, by choosing to exclude unrequited scrapped tada, they were able to produce 14k entries where 8840 tweets are non-offensive and 4400 tweets are classified as offensive. Now, apart from only classifying offensiveness, they hierarchically created two more categories. In the OLID dataset, they created 3 different types of annotation on a given tweet. for example, a tweet "@USER He is so generous with his offers" can be classified as offensive or offensive, further, they leveled whether the tweet is targeted toward the audience or it was just containing offensive words or a generalized mockery. In the dataset, lastly, they also annotated what type of audience, like if a tweet only offended an individual or a group or community. However, in this task, I only worked with the first two annotations, Here is a quick summary of how offensive language detection and categorization of offensive language are represented.

### Offensive Language Detection

In this annotation, a tweet is annotated in two categories, one is offensive and one is non-offensive.

#### Offensive:

In the data set, column tweet represents the tweet scraped, then manually all the tweets that can be considered as offensive are represented as OFF in the subtask_a column. The categorization process was, to check if the tweet contained any words or messages that were not accessible. For example, it can be an insult to a person or a threat. Also, many tweets that contain offensive or inappropriate words are labeled as offensive an example is swear words. A tweet containing threats, mockery or swear words is labeled as OFF which stands for the offensive.

#### Non-offensive:

This category represents a general tweet that does not contain any characteristics that are included in the offensive list. Like, any tweet that does not have any threat, satirical meaning, or swear words is listed in this category, and NOT labeled used in the subtask_a column to represent non-offensive words.

### Categorization of offensive language

Apart from only identifying offensive tweets, all those offensive tweets are classified as either targeted insults or untargeted insults.

#### Targeted Insult:

These are those tweets that are posted for a particular group or individual. in the subtask_b, these tweets are annotated with TIN

#### Untargeted:

These are tweets that are offensive but it was not posted to target any individual or community. This list includes most of the tweets that are classified as swear words or words that are not acceptable as naive tweets you can say.

Table 1: Example Tweets from OLID Dataset

| **ID** | **Tweet** | **Subtask A** | **Subtask B** | **Subtask C** |
|---|---|---|---|---|
| 86426 | ”@USER She should ask a few native Americans what their take on this is.” | OFF | UNT | NULL |
| 90194 | ”@USER @USER Go home you’re drunk!!! @USER #MAGA #Trump2020 URL” | OFF | TIN | IND |
| 16820 | ”Amazon is investigating Chinese employees who are selling internal data to third-party sellers looking for an edge in the competitive marketplace. URL #Amazon #MAGA #KAG #CHINA #TCOT” | NOT | NULL | NULL |
| 62688 | ”@USER Someone should’ve taken¨this piece of shit to a volcano. ¨” | OFF | UNT | NULL |
| 43605 | ”@USER @USER Obama wanted liberals & illegals to move into red states” | NOT | NULL | NULL |

## Models

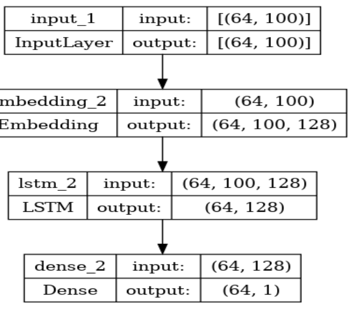
Now to train on these dataset, I have used two model. One is based on Transformer and another is LSTM. The transformer model is very difficult to train on local PC, therefore I used that BertForSequenceClassification only for subtask_a.

### BertForSequenceClassification:

The BertForSequenceClassification model is a powerful tool in the realm of natural language processing. It’s like having a language understanding guru at your disposal. This model, based on BERT (Bidirectional Encoder Representations from Transformers), can take a piece of text and classify it into predefined categories or labels, making it great for tasks like sentiment analysis, spam detection, or topic categorization. It learns to recognize contextual nuances and relationships within the text, making it particularly adept at understanding the subtleties of language. However, I probably due ot limitation of hardware resources and my ignoronance, I could not utilize the full potential of bard model. Here is model parameters that I used to train in this task,

#### LSTM-based Text Classification Model

The model I have used is an LSTM (Long Short-Term Memory) based text classification model. LSTMs are a type of recurrent neural network (RNN) that are particularly effective at modeling sequences, making them well-suited for natural language processing tasks. In this architecture, I started start with an input layer, followed by an embedding layer that converts words into dense vectors. The LSTM layer with 128 units is added to capture sequence information and account for dependencies in the text. A dropout layer is included to prevent overfitting, and finally, a dense layer with a sigmoid activation function is used for binary classification. This model is trained to classify text into two categories based on its learned features and context. The model takes sequences of text data as input, processes them through the embedding and LSTM layers, and produces a binary classification output using a sigmoid activation function.



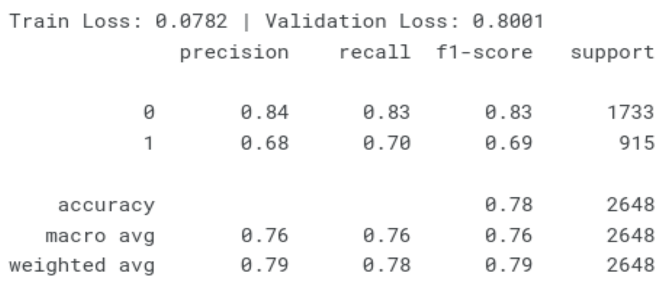
input_1 | input: | [(64, 100)]
InputLayer | output: | [(64, 100)]
embedding_2 | input: | (64, 100)
Embedding | output: | (64, 100, 128)
lstm_2 | input: | (64, 100, 128)
LSTM | output: | (64, 128)
dense_2 | input: | (64, 128)
Dense | output: | (64, 1)

Figure 1: LSTM parameters

## Results

After running the model, there were the classification report I got.

For BertForSequenceClassification, this is the classification report,



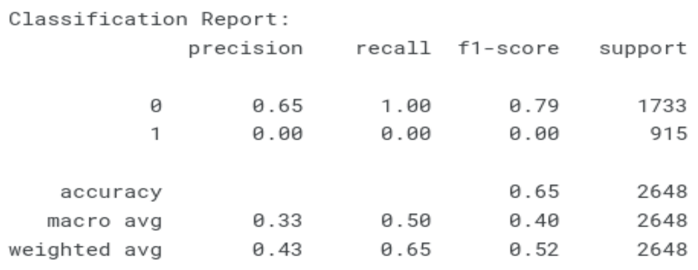
```
Train Loss: 0.0782 | Validation Loss: 0.8001
              precision    recall  f1-score   support

           0       0.84      0.83      0.83      1733
           1       0.68      0.70      0.69       915

    accuracy                           0.78      2648
   macro avg       0.76      0.76      0.76      2648
weighted avg       0.79      0.78      0.79      2648
```

Figure 2: TaskA - BertForSequenceClassification

Now, in LSTM based classifer this is the classfication report for task-a, For BertForSequenceClassification, this is the classification report,



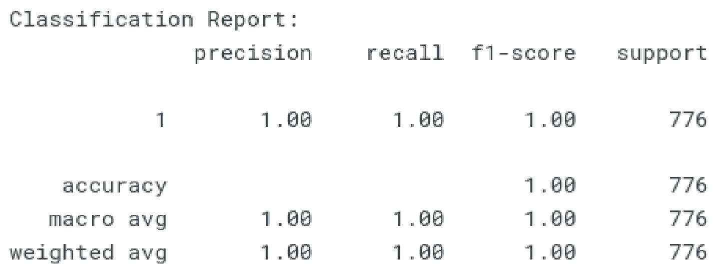
```
Classification Report:
              precision    recall  f1-score   support

           0       0.65      1.00      0.79      1733
           1       0.00      0.00      0.00       915

    accuracy                           0.65      2648
   macro avg       0.33      0.50      0.40      2648
weighted avg       0.43      0.65      0.52      2648
```

Figure 3: TaskA - LSTM-based Text Classification Model

Finally, this was the classification report from LSTM based model for task 2,



```
Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00       776

    accuracy                           1.00       776
   macro avg       1.00      1.00      1.00       776
weighted avg       1.00      1.00      1.00       776
```

Figure 4: TaskA - LSTM-based Text Classification Model

## Limitation

Every model has some limitations that can vary depending on the type of data. Basied on OLID dataset, there are the limitation of used models.

BertForSequenceClassification Model: While the BertForSequenceClassification model is a powerful tool for natural language processing tasks, it comes with certain limitations. One primary limitation is its computational complexity and resource requirements. Training and fine-tuning a large-scale BERT model can be computationally expensive and may not be feasible for smaller research or production environments with limited computing resources. For example, for this project, I had to use the free quota on Keggle to run the model because it's not feasible to run the model on local users' PCs without high-quality graphics and CUDA support. Additionally, fine-tuning BERT models requires a substantial amount of labeled data, which may not always be available for specific domains or languages. Furthermore, BERT models may struggle with handling long documents due to their token limitations, potentially leading to information loss when processing lengthy texts.

LSTM-based Text Classification Model:

The LSTM-based text classification model that I have used may offer a robust solution for various natural language processing tasks. however, it also has its limitations. One key limitation is its sensitivity to the choice of hyperparameters, including the number of LSTM units, dropout rates, and sequence length. Fine-tuning these hyperparameters can be a time-consuming process and may require domain-specific knowledge. Additionally, LSTMs, like other recurrent neural networks, can suffer from vanishing gradient problems, which may limit their ability to capture long-range dependencies in text data effectively. Moreover, this model may not fully capture contextual information present in more advanced models like BERT, which excel in understanding complex language nuances. Lastly, the performance of LSTM-based models can be affected by imbalanced datasets, requiring additional techniques such as class weighting or oversampling to address class distribution issues effectively.

on Data:

apart from the limitation of model, using a good dataset and appropriate preprocessing is also important. In the dataset, swear words are also categorized as offensive tweets, but in general, one may use swear words to express their feelings, so I believe using more advanced methods to preprocess like using a word2vec and then training on that dataset so it understanding the context like when words used, does it really offends anyone or one just using that to express his feelings can produce a better result for real-world application.

## Conclusion

In conclusion, both the BertForSequenceClassification model and the LSTM-based text classification model offer valuable solutions for analyzing text data in the OLID dataset. However, as offense identification requires a good understanding of context, using more fine-grained training & context analysis could have improved the the overall result

## References

[Zampieri et al., 2019] Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. (2019). Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86.

[Zampieri et al., 2019]