CSE440 Project Report on Offensive Language Identification

Anika Ahmed, ID: 21101029
Nafis Chowdhury, ID: 21101034
Moinul Haque, ID: 21101186
Mohammed Adib Ahnaf Hamim, ID: 21101054

November 11, 2024

Abstract

As the use of social media increased, the number of offensive posts and comments increased simultaneously. In this project, we tried to build models which are capable of analyzing tweets and concluding whether the tweets are offensive or not. Also, if the tweets are offensive, the model can also decide whether the offense was targeted or not. Here, we used a public dataset available online, called the Offensive Language Identification Dataset (OLID). We used two types of Recurrent Neural Network (RNN) Models to do our task, which include both Unidirectional and Bidirectional Long Short-Term Memory (LSTM) and Bidirectional Gated Recurrent Unit (GRU). The accuracy, precision, recall, F1 and F2 scores of our models are discussed in this report, along with the process in which these models work.

1 Introduction

In social media offensive writing is seen very regularly, in order to control or detect the culprits we need to first detect the sentences that are offensive and their type. Because, without this information the offensive posts or comments can not be controlled. In cases of post that are offensive, we will classify whether they were intended to anyone or they were untargeted.

2 Data Collection

We have used the dataset from olid-training-v1.0 that contains the features and outcome. We trained our models on the basis of olid-training-v1.0. In order to test our model, we used testset-levela tsv file as input to predict the outcomes for offensive or not and used labels-levela csv file to compare those outcomes. For predicting whether the offensive post were targeted or untargeted we used testset-levelb tsv file as input to predict the outcomes and used labels-levelb csv file to compare those outcomes.

2.1 Data analysis

We begin the analysis of data in the training data points and determine how many posts of the training data are offensive and how many are not offensive. From the offensive posts, we determine how many of them are targeted and how many are untargeted. In the testing dataset, we performed the same process to determine offensive and not offensive posts and from the offensive posts we determine how many of them are targeted and how many are untargeted. The maximum length of a sentence in our training data was 416, the minimum length was 4 and the average length was 74. The most common word is 'liber' and least common word is 'vetaken' in the traning dataset. The maximum length of a sentence in our testing data for predicting whether the post was offensive was 236, the minimum length was 7 and the average length was 88. The maximum length of a sentence in our testing data for detecting whether the offensive posts are targeted or untargeted was 220, the minimum length was 7 and the average length was 84. For both the testing datasets, the most common word is 'liber' and least common word is 'whoisq'.

Total data	Offensive	Not Offensive
13240	4400	8840

fig: Distribution of Offensive and Not offensive posts in the training data set

Offensive data	Targeted	Untargeted
4400	3876	524

fig: Distribution of Targeted and Untargeted posts in the training dataset

Total data	Offensive	Not Offensive
860	240	620

fig: Distribution of Offensive and Not Offensive posts in the testing dataset

Offensive posts	Targeted	Untargeted
240	213	27

fig: Distribution of Targeted and Untargeted posts in the testing data set

3 How does LSTM work?

Long Short-Time Memory (LSTM) is a Recurrent Neural Network (RNN) algorithm which uses ‘gates’ which use sigmoid function to understand how much of an input it needs to remember and how much history it needs to forget. LSTM uses three different gates: forget gate, input gate and output gate. Let’s have a look at these gates. We used both Unidirectional and Bidirectional LSTM models. In Unidirectional LSTM, the input goes into the model starting from the beginning of the sentence to the end to give us an output. In Bidirectional LSTM, a similar task is done, along with running the algorithm another time where the inputs are given starting from the end of the sentence to the beginning. The Bidirectional model then gives us an output after comparing the two results.

3.1 Forget gate

When an input is received, it is passed along with the history through a sigmoid gate which does some calculations to generate a value. If the value is close to 0, a small amount of information is passed through the gate. If the generated value is closer to 1 than 0, more of the information is allowed through the gate.

3.1.1 Equation of forget gate

$f(t)=\sigma(W(f).[h(t\text{-}1),x(t)]+b(f)])$

3.2 Input gate

The input along with the history is then passed through another sigmoid gate with similar functionalities and through a tanh function, which outputs a value between 1 and -1. The output from this sigmoid gate and tanh function is then multiplied and the resultant value is passed forward. This is how the input gate works.

3.2.1 Equation of input gate

i(t)=σ(W(i).[h(t-1),x(t)]+b(i)])

‘C(t)=tanh((W(C).[h(t-1),x(t)]+b(C)])

3.3 Cell update

The new value of our memory cell is then passed through another tanh function. Also, the input and history is passed through another sigmoid gate which is the output gate. The output from these two functions is then multiplied and then passed forward as the history to the next network.

3.3.1 Equation of cell update

C(t)=f(t)$*$C(t-1)+i(t)$*$‘C(t)

3.4 Output gate

The new value of our memory cell is then passed through another tanh function. Also, the input and history is passed through another sigmoid gate which is the output gate. The output from these two functions is then multiplied and then passed forward as the history to the next network.

3.4.1 Equation of output gate

o(t)=σ(W(o).[h(t-1),x(t)]+b(o)])

h(t)=o(t)$*$tanh(C(t))

4 Why did we choose LSTM?

We know, when we are working with Natural Language, we have to keep track of context in order to understand the following sentences. A computer does not know which information is important. So, it also does not understand which information to remember and which ones to forget. We need to understand this so that we do not lose useful information, and so that our memory does not get filled up with useless information. Also, we cannot remember every information as we do not have unlimited memory. As LSTM understands how much information it needs to forget and how much it needs to remember, we can use this for our benefit.

5 How does GRU work?

Gated Recurrent Unit (GRU) is similar to LSTM, but uses fewer parameters. It has an update gate and a reset gate, which consist of sigmoid functions. We used Bidirectional GRU Model which works by running the algorithm from the start of the sentence to the end (left to right) and running a similar algorithm from the end of the sentence to the beginning (right to left), and comparing their outputs.

5.1 Equation of update,reset gates, and updating hidden layer

z(t)=σ(W(z).[h(t-1),x(t)])

r(t)=σ(W(r).[h(t-1),x(t)])

‘h(t)=tanh(W.[r(t)$*$h(t-1),x(t)]

h(t)=(1-z(t))$*$h(t-1)+z(t)$*$‘h(t)

6 Why did we choose GRU?

The reason behind using GRU is very similar to that behind using LSTM, except the fact that as GRU uses fewer parameters, it is way faster than

LSTM, which is good for us. Fewer parameters mean fewer weight updates, less memory requirements and faster processing. As the outputs given by LSTM and GRU are pretty similar, and in most of the cases GRU works as well or better than LSTM, GRU is more feasible for us.

7 Model performances for subtask(a)

As we can recall from data analysis, in our training data set 33.23% of the corpus were offensive and 66.77% of the corpus were not offensive. Similarly in the testing dataset, 27.9% of the corpus was offensive and 72.1% of the dataset was not offensive. So our label is not evenly spread out. Therefore, we will calculate f1 score, f2 score, precision, and recall based on the confusion matrix.

7.1 Confusion matrix

The tables consisting values from our confusion matrices are given below.

Distribution	Predicted Negative	Predicted Positive
True Negative	378	242
True Positive	75	165

fig: Confusion Matrix for Unidirectional LSTM

Distribution	Predicted Negative	Predicted Positive
True Negative	176	444
True Positive	24	216

fig: Confusion Matrix for Bidirectional LSTM

Distribution	Predicted Negative	Predicted Positive
True Negative	147	473
True Positive	52	188

fig: Confusion Matrix for Bidirectional GRU

7.2 f1 score, f2 score, Precision and Recall

Model	f1 score	f2 score	Precision	Recall
Unidirectional LSTM	0.5100	0.6035	0.4054	0.6875
Bidirectional LSTM	0.4800	0.6667	0.327	0.9
Bidirectional GRU	0.4173	0.5799	0.2844	0.783

fig: f1 score, f2 score, Precision and Recall for all Models

8 Model performances for subtask(b)

Similarly from data analysis, in our training data set 88.09% of the offensive posts were targeted and 11.91% were untargeted. In the testing dataset, 88.75% were targeted and 11.25% of the dataset were not targeted. So our label is not evenly spread out. Therefore, we will calculate f1 score, f2 score, precision, and recall based on the confusion matrix.

8.1 Confusion matrix

The tables consisting values from our confusion matrices are given below.

Distribution	Predicted Negative	Predicted Positive
True Negative	168	45
True Positive	18	9

fig: Confusion Matrix for Unidirectional LSTM

Distribution	Predicted Negative	Predicted Positive
True Negative	85	128
True Positive	5	22

fig: Confusion Matrix for Bidirectional LSTM

Distribution	Predicted Negative	Predicted Positive
True Negative	48	165
True Positive	4	23

fig: Confusion Matrix for Bidirectional GRU

8.2 f1 score, f2 score, Precision and Recall

Model	f1 score	f2 score	Precision	Recall
Unidirectional LSTM	0.2222	0.2778	0.166	0.333
Bidirectional LSTM	0.2486	0.4264	0.1467	0.815
Bidirectional GRU	0.2140	0.3885	0.1223	0.852

fig: f1 score, f2 score, Precision and Recall for all Models

Hence, Unidirectional LSTM worked better than Bidirectional LSTM and bidirectional GRU for this dataset.

9 Challenges

There were multiple challenges that we had to face during creating our models. One of the challenges was that our dataset was not evenly spread, which means that the number of datapoints in our dataset that gave an output 'Not Offensive' (for part a) was much larger than the number of datapoints that gave us the other output. Similar problem was faced for part b. This was leading to overfitting of our data and even though we got a decent accuracy, our precision and recall scores were both 0. We overcame this problem by mapping our outputs using a sigmoid function. Then, we rounded off the result given by the sigmoid function and compared the value to our actual test labels to calculate precision, accuracy and recall. Even though our accuracy decreased a lot comparatively, our precision and recall scores are better than before.

10 Conclusion

In conclusion, the prevalence of offensive writing on social media platforms is a pressing issue that demands effective control and detection mechanisms. To effectively combat this problem, it is imperative to first identify and classify offensive sentences based on their type and intent. This crucial step is essential for ensuring a safer and more inclusive online environment. By distinguishing between targeted and untargeted offensive content, we can tailor our response strategies and work towards a more respectful and constructive digital community.