Offensive Language Finding

How To Prevent Abusive Behaviour In Social Media

Research lab "Online Political Polarization" WS 2019/2020

Universität Koblenz-Landau

Shankar, Manjunath
Universität Koblenz-Landau
manjunath@uni-koblenz.de
Mat-Nr.: 218203199

Sharma, Ishan
Universität Koblenz-Landau
ishan.sharma@uni-koblenz.de
Mat-Nr.: 218203352

Geiger, Benedikt
Universität Koblenz-Landau
benegeiger@uni-koblenz.de
Mat-Nr.: 213200397

Rethinakumar, Anurekha
Universität Koblenz-Landau
anurekha@uni-koblenz.de
Mat-Nr.: 219100900

I. Introduction

Twitter policies don't prohibit offensive content, which was our motivation to do research in this area. Our research question was "Are offensive tweets targeted towards individuals more frequently than towards groups?". We categorized tweets into offensive, unoffensive, "targeted at individuals", "targeted at groups" and "untargeted".

II. Data Set

We collected ca. 40,000 tweets from Twitter API with **Tweepy** library and used "Offensive Language Identification Data" (OLID) for training data which had three sets of subtasks.

- 'Offensive and unoffensive'
- 'Offensive targeted and untargeted'
- 'Targeted offensive data with labels for the targets'

We combined all the data frames according to the ratio in the actual Twitter data and used it for training.

III. Background

Bi-LSTM is well suited to process natural language since it preservers sequence information over time.
In our Bi-LSTM, the embedding layer receives sentence tokens and transforms them to word vectors to extract features.

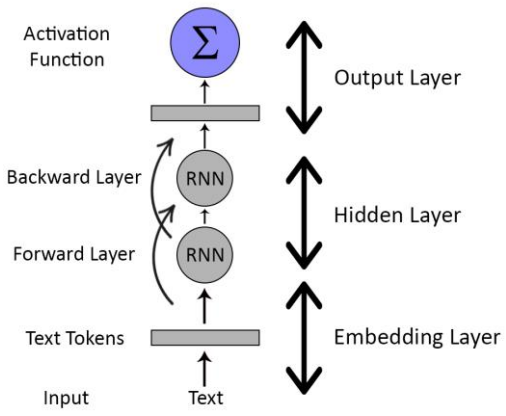

Figure 1: Bi-LSTM architecture

We used **spaCy** library for named entity recognition to identify targets of offensive language.

IV. Methodology

In our approach, we implemented a Neural Network (NN), a LSTM and a Bi-LSTM. Since Bi-LSTM gave best accuracy. For implementation, we used Keras and the TensorFlow framework.

As part of feature engineering, we tagged the data with the relevant entity names using **spaCy** library. As next steps, we used Keras to implement a Neural Network, LSTM and Bi-LSTM subsequently. Thereby, 90% of our data were used for training and 10% for testing. We used softmax as activation function, which turns values from the hidden layers into probabilities. These probabilities sum to one and softmax returns a probability distribution for classifying the input data.

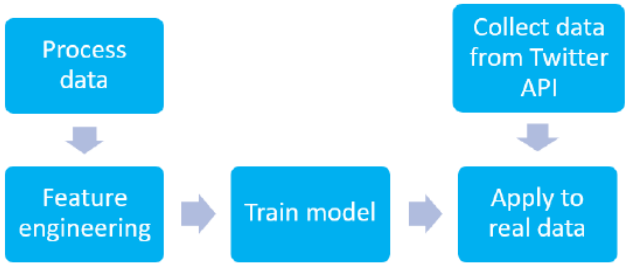

Figure 2: Implementation pipeline

V. Conclusion and Evaluation

Based on existing research, we found that the best classifier to use for the purpose of offensive language finding is a Neural Network with LSTM. With the initial dataset received from olid. Initially, we tried with baseline Neural network without any LSTM layers. We found that it was giving an accuracy of just 43.9%. So, we decided not to invest more effort into improving it. We switched to LSTM. For the same data, we got 71.9% accuracy. We also tried to use Bi-LSTM and we got an accuracy of 72.4 %.

After retraining the Bi-LSTM model multiple times on more training data (around 120,000 rows) from Twitter (Second dataset from Olid), we achieved an accuracy of (86.80+-0.010) % in 10% of the dataset which was not trained. Analysing our data with Bi-LSTM on the twitter data that we got by using twitter API, we found that out of 40,501 tweets, 80.36% were not offensive, 7.94% were offensive and targeted at individuals, 4.62% were offensive and targeted at groups, 6.8% were offensive against other parties and 0.27% were generally offensive.

In our error analysis, we took some tweets and tried to compare the actual label and our prediction. We found that many of the cases were actually ambiguous.

With these results, we can conclude that offensive tweets are more often targeted at individuals than at groups which confirms our hypothesis. Since users tend to insult each other in posts and comments, this result appears plausible and emphasizes the need for offensive language identification in social media.

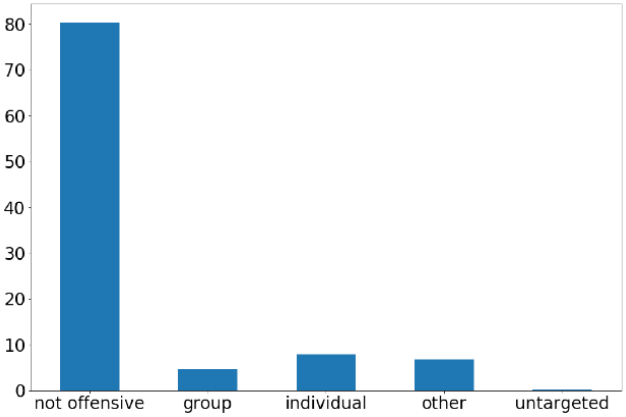

Figure 3: Distribution of tweets regarding offensive categories