Hate Speech Detection

Alexandru Gavrila Andrei Munteanu

Faculty of Mathematics and Informatics

Bucharest University

Abstract

Hate speech is the act of "public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation" as described in the Cambridge Dictionary. This kind of speech is offensive and/or hateful towards a group of people and is most commonly found in the online medium. The purpose of this project is to detect said speech based on text data using NLP (natural language processing) and machine learning. For this project we combined multiple labeled datasets that target hate speech.

Introduction

Hate speech is a form of discrimination. It is a manner of speaking which is offensive and/or hateful towards a specific group of humans and it is harmful towards the wellbeing of people targeted by it or otherwise. This usually consists but it is not limited to: derogatory nicknames, comparisons based on race/religion/etc., swear words, etc. The most common form of hate speech is specific nicknames addressed to groups of people, followed by threats targeted at a group. This should be easily recognizable by the nature of the phrase. It is usually a unique words which is easily detectable by computers and can be censored/removed (e.g. “nigger”). The second most common form of hate speech is sentences which have words with otherwise no offensive meaning. (e.g. “Homosexuals should just die.”). Even though singular words are not necessarily hateful, the meaning of the sentence is. This is usually harder to detect since it requires some sort of knowledge and a learning algorithm as opposed to a simple string scan. Another problem that arises

is the differentiation of offensive language from hate speech which is fortunately partly solved by the annotation of the datasets we use.

In this project we compared multiple combinations of NLP and NN with the purpose of finding the best hate speech detection algorithm. For NLP we tried a multitude of text processing techniques referenced by On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis(Jose Camacho-Collados ,Mohammad Taher Pilehvar): lowercasing, lemmatizing, stemming and multiword grouping and stop words removal. Each of these techniques help the algorithm generalize and understand the meaning of words and sentences. For machine learning we used Neural Networks. We tried CNN, RNN and simple MLP.

Related work

A lot of work is put towards hate speech detection because it constitutes a serious problem in the online medium where people can hide their real identities behind fake accounts. HateXplain(Binny et.al. 2020) is a dataset geared towards hate speech detection using three labeling annotations for the same entry. One of them is the common annotation used in most hate speech datasets, while the other two are based on target community and the part of the sentence that made the annotator label the text accordingly. Jose Camacho-Collados and Mohammad Taher Pilehvar did a lot of work regarding the text preprocessing for neural networks in the field of sentiment analysis and text categorization. This mostly includes standard approaches for text preprocessing. State of the Art is BERT (Bidirectional Encoder Representations from Transformers) on the benchmark HateXplain using Attention Mechanism.

Dataset

The dataset we used is a combination of https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge and https://data.world/thomasrdavidson/hate-speech-and-offensive-language/workspace/file?filename=labeled_data.csv. The combined dataset has 159.000 + 24.000 samples. The content of the data is composed of mostly

tweets scraped off of Twitter. The labels that were used in the first dataset were “toxic”, “severe toxic”, “obscene”, “threat”, “insult”, “identity hate” and in the second datasets were “offensive”, “hate speech”, “normal/neutral”. These labels were used for training our neural networks.

Preprocessing

The data was loaded in full format and then it was lowercased. This new corpus was then split by whitespaces and tokenized. We also removed punctuation. We applied stemming to this clean dataset and ran our model with it. Different combinations of the steps above were skipped and the model was rerun to check for different approaches and compare the new results.

Features

We used Glove6b100 Word2Vec embedding. We also tried using NLP techniques like: stop words removal, accent striping, lowercasing.

Models

	Layers	Optimizer	Loss
MLP	1 hidden 128 neurons	Adam	Binary Cross Entropy
ConvNet	3 Conv1D 128 neurons 3 kernel size, MaxPooling	RMSprop	Binary Cross Entropy
LSTM	1 LSTM with 15 cells, Global Max Pooling	Adam	Binary Cross Entropy
BiLSTM	1 BiLSTM with 15 cells, Global Max Pooling	Adam	Binary Cross Entropy
Logistic Regression	C=0.01	Solver= liblinear	Penalty=l2

Metrics

We used Accuracy, loss and AUC(Area Under Curve).

Conclusions

We tried to take these data sets and work on them with various preprocessing and machine learning methods and compare the results. We tried different embedding, different models and different preprocessing methods and worked out a table with our values.

	Accuracy	Loss	AUC
MLP +w2v	0.96	0.11	0.96
ConvNet+w2v	0.99	0.10	0.975
LSTM+w2v	0.99	0.05	0.980
BiLSTM+w2v	0.99	0.05	0.985
LogisticRegression + w2v	0.89	-	-

Similar results by using NLP before applying w2v.