Offensive language detection

Yuval Israel Adir (ID. 315476614), Uri Amiel (ID. 313160046)

Submitted as final project report for the NLP course, IDC, 2022

1 Introduction

In the digital world, especially in social media, offensive language attracts a lot of attention. Many users take advantage of anonymity to engage in conduct that they would never risk in real life. By using usernames and anonymous profiles, the user is able to create a new identity and conceal their true identity. This creates an environment without limits, and as a result the amount of offensive language on social media also increases. Therefore, social networks, online communities, and technology companies have already undertaken research on the use and possible prevention of offensive language and irresponsible behavior. A common and effective way is to identify offense, aggression and hate speech in user-generated content like posts, comments and micro-blogs by using computational methods. We used for our project the Offensive Language Identification Dataset, OLID. OLID contains a collection of annotated tweets using an annotation model that encompasses the following three levels:

1. Task A: Offensive Language Detection
2. Task B: Categorization of Offensive Language
3. Task C: Offensive Language Target Identification

In this project we attempted to implement several models that attempt to solve these tasks.

1.1 Related Works

1. Predicting the Type and Target of Offensive Posts in Social Media

2 Solution

2.1 General approach

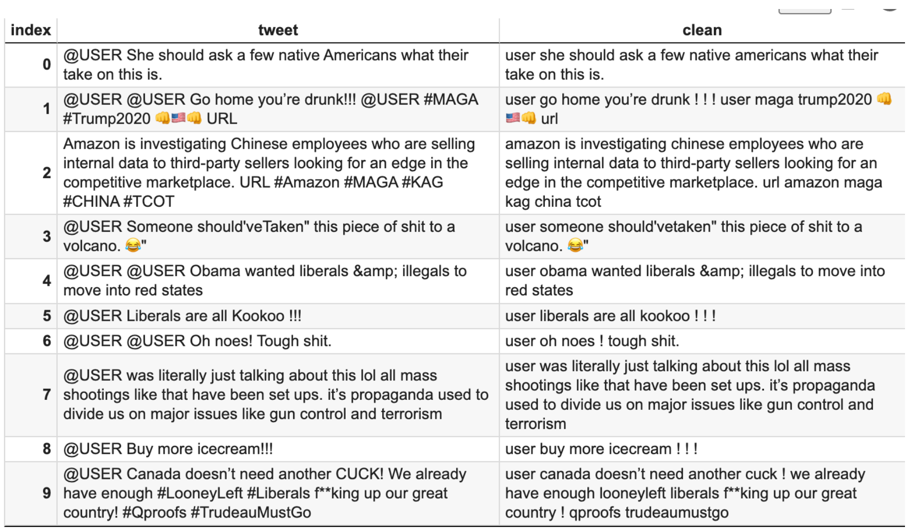
Our solution is divided into 3 main parts. First, we focused on smart preprocessing of the data. Second, we chose a random forest classifier as our base optimization algorithm. Third, we plan to test and experiment with three types of models (including the random forest) and compare them to see which will produce the best results. As mentioned, the first model uses a bag-of-words representation of tweets and a random forest classifier. This is the simplest model out of the three and should provide us with a suitable starting point. The second model is built with Keras and Tensorflow, and implements a CNN (convolutional neural network). For NLP, we are using a 1D convolution, which allows us to capture relationships between nearby words in a sentence. The third model is the most complex one, and uses a combination of a CNN model and a RNN model (recurrent neural network). This allows us to "remember" previous words in the sentence and gives context to the entire phrase.

2.2 Design

2.2.1 Data processing methodology

Cleaning the data

Machine learning models, especially NLP models, require data cleaning as a first step. Datasets that are not cleaned are often clusters of words that cannot be understood by a computer. This is why we decided to clean the data first. The first step of clearing the data was to delete the unlabeled tweets from each dataset. Afterwards, we pre-processed the data. All tweets were converted to lowercase, and unnecessary punctuation was deleted, so that the model could easily compare them. It required removing hashtags and '@', removing repetitive '@USER', and adding space between words and punctuation. Before cleaning, tweets typically look like this: '@USER @USER @USER It's not my fault you support gun control'. Our cleaning process will result in a tweet as follows : 'user it's not my fault you support gun control'.



index	tweet	clean
0	@USER She should ask a few native Americans what their take on this is.	user she should ask a few native americans what their take on this is.
1	@USER @USER Go home you're drunk!!! @USER #MAGA #Trump2020 👊🇺🇸👊 URL	user go home you're drunk ! ! ! user maga trump2020 👊🇺🇸👊 url
2	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	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
3	@USER Someone should'veTaken" this piece of shit to a volcano. 😂"	user someone should'vetaken" this piece of shit to a volcano. 😂"
4	@USER @USER Obama wanted liberals &amp; illegals to move into red states	user obama wanted liberals &amp; illegals to move into red states
5	@USER Liberals are all Kookoo !!!	user liberals are all kookoo ! ! !
6	@USER @USER Oh noes! Tough shit.	user oh noes ! tough shit.
7	@USER was literally just talking about this lol all mass shootings like that have been set ups. it's propaganda used to divide us on major issues like gun control and terrorism	user was literally just talking about this lol all mass shootings like that have been set ups. it's propaganda used to divide us on major issues like gun control and terrorism
8	@USER Buy more icecream!!!	user buy more icecream ! ! !
9	@USER Canada doesn't need another CUCK! We already have enough #LooneyLeft #Liberals f**king up our great country! #Qproofs #TrudeauMustGo	user canada doesn't need another cuck ! we already have enough looneyleft liberals f**king up our great country ! qproofs trudeaumustgo

Figure 1: The original and cleaned tweets

Balancing the data in each dataset

To avoid our model being biased towards one class, we altered the amount of offensive tweets to be close to the number of non offensive tweets. The outcome is a more reliable model that is not biased towards a specific class. The initial datasets were highly imbalanced, and initially the number of non-offensive tweets was double than the offensive examples. A combination of over-sampling and under-sampling was used to equalize the weights of the classes. Under-sampling is performed by taking an input representing the percentage of under-sampling desired. This number represents a trade-off between deleting potentially relevant information from a majority class and producing too many duplicate examples for a minority class. To select it, we ran various cross validations on the random forest model.

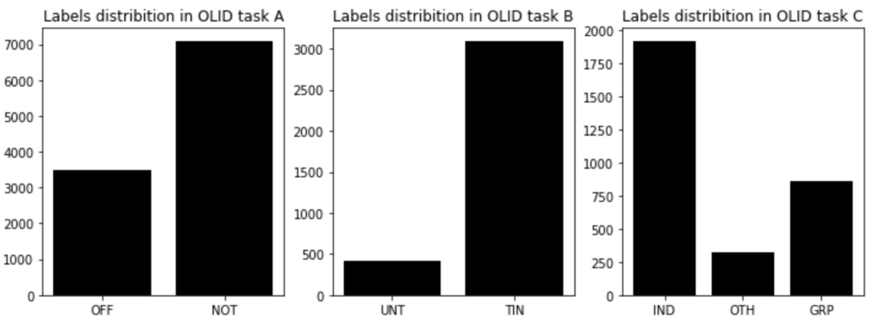


Figure 2: The original data distribition

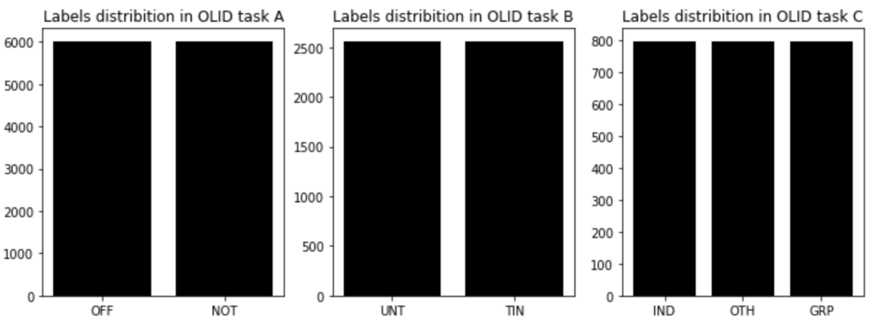


Figure 3: Data distribution after balancing

2.2.2 Models

Random Forest with bag of words representation

At first, we used a simple bag-of-words representation. In the bag-of-words (BOW) model, the text is transformed into fixed-length vectors by counting how many times each word appears. This representation served as a baseline for our desired results. To begin, we convert our tweets into a document-term matrix, where each cell represents the number of times a word appears. As a simple yet powerful model, we have chosen a random forest classifier for the matrix. The advantage of working with this model is that since training is tremendously fast, it allows us to test different variations and fine tune parameters, while saving us time. We used the model throughout the project as the 'validation model' for tuning data processing and hyper-parameters.

Word embeddings with fastText

Word embeddings are learned representations for text where words that have the same meaning are represented similarly. We used it to improve our results in the more complex models. We used fastText which is a popular word vector model in NLP. FastText is an extension of the word2vec model for embedding words. Instead of learning vectors for words directly, fastText represents each word as an n-gram of characters. In this way, short words can be captured and suffixes and prefixes can be understood by embeddings. To learn the embeddings, we trained a skip-gram model with character n-grams. We trained our own embeddings from the data using fastText. FastText is also known for its ability to handle rare words effectively. Even if a word wasn't seen during training, its embeddings can be derived from n-grams. This is well suited for the task, because tweets tend to showcase a very creative use of the English language.

CNN

Our second model is a convolutional neural network. To begin, we started with a layer of our trained word embeddings. To overcome over-fitting, Dropout layers of 0.5 were applied. The model is made simpler by randomly dropping 50 percent of its nodes and preventing over-fitting. The dropout is followed by a 1D convolution, with an added activation function to add non-linearity to the network. Relu is chosen as the activation function. Following the convolutional layer is the max pooling layer. In order to reduce computation costs, this layer reduces the size of the convolved feature map. A ReLU activation function is also added to the max pooling. Before the output layer, a fully-connected network of 10 hidden neurons with a sigmoid activation function is applied.

CNN and RNN combination

Our third model combines a bi-directional RNN with LSTM cells, a CNN, and a Feed Forward Neural Network. A bidirectional RNN is followed by a convolution. Afterwards, we perform a max pooling and an average pooling separately, and then concatenate the results. These are then fed into a 2 layer fully-connected network with 10 hidden neurons. As for activation functions, we used the standard Tanh in the LSTM cells, Relu for all the hidden layers in the Feed Forward Network, and then softmax for the output layer.

- **Hyper parameters tuning**
  Bayesian optimization was used to search for hyper parameters in our model. Using Bayesian optimization, we ran 10 epochs on a wide range of possible values and arrived at the following optimal parameters: learning rate = 0.001 and weight decay = 6e-10. It is probably due to the fact that we are already utilizing the Dropout technique (with a rate of 0.5) to prevent over-fitting that weight decay is unnecessary.

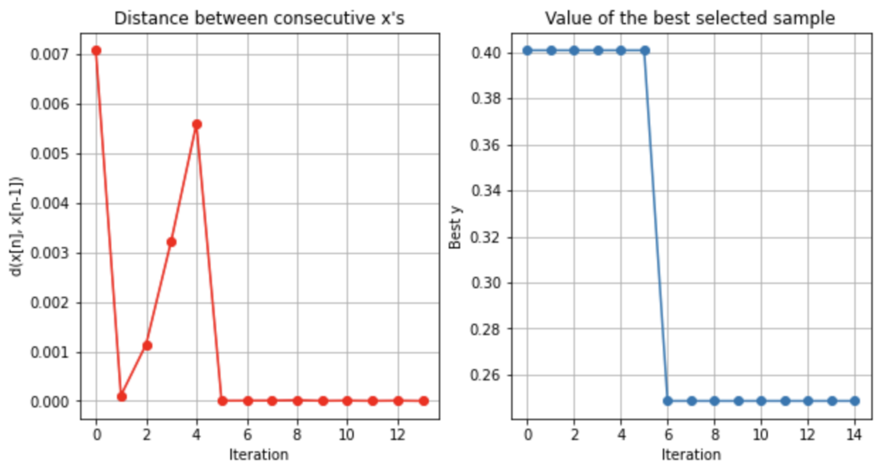


Figure 4: Bayesian optimization results

- **Transfer learning**
  We used a technique called transfer learning to improve the performance. Transfer learning helps to store the knowledge gained while solving task A and applying it to the different but related tasks B and C. For each task, we add a new feed forward neural network at the end of our model, reusing the initial and middle layers. In this case, this technique is particularly useful when there are not many labels for the subsequent tasks.

3 Experimental results

Our model was evaluated by splitting the dataset into train and validation, with 20 percent of the total dataset assigned to validation. As for the first two models, the Random Forest on the bag of words and the CNN on the word embeddings, we would expect the CNN to perform better than the Random Forest. In fact, the random forest performed better in this case. Because of the high speed of computation of the random forest, it was used throughout the project to tune data processing and various shared hyper-parameters. Our data preparation was more suited to the random forest than the CNN, so the latter appears to have a lower performance. As for the third model, its performance was also challenged by the random forest baseline model. During our experiments we tried to use various techniques mentioned above to improve the model performance (word embeddings, transfer learning and Bayesian optimization). Despite our efforts to improve it we were not able to significantly improve upon the baseline results.

F1 macro average score			
OLID Task	**Random Forest**	**CNN**	**CNN and RNN combination**
Task A	0.72	0.60	0.70
Task B	0.58	0.55	0.57
Task C	0.55	0.30	0.47

F1 is computed for each label, and returns the average without considering the proportion for each label in the data.

F1 weighted average score			
OLID Task	**Random Forest**	**CNN**	**CNN and RNN combination**
Task A	0.76	0.62	0.72
Task B	0.72	0.74	0.78
Task C	0.70	0.44	0.65

F1 is computed for each label, and returns the average considering the proportion for each label in the data.

4 Discussion

To achieve better results, It is possible to conduct several experiments on our trained models. First, making the embeddings more accurate can improve the second and third models performances. In order to accomplish this, more tweets can be added to the FastText. Second, we used the random forest to tune data processing and various shared hyper-parameters. The third model performed better, but we did not necessarily find the optimal parameters as a result. It would be better to use our third model for tuning in order to improve its own performance. Furthermore, our loss function for the neural network was binary cross-entropy. In this way, the neural network maximizes accuracy rather than

F1. To improve the model in the future, we could design a loss function that maximizes the F1 score. In addition, there are other NLP techniques that can be used instead (such as Bert as an example).
There were several difficulties encountered throughout the course of the project. The imbalance in the datasets is one of the main challenges of this project. In order to balance the data, we over-sampled minority classes, under-sampled majority classes, and penalized the loss of majority classes. We saw some improvement in our results as a result of these solutions, but building a model that effectively classifies such imbalanced samples remains very difficult.
When it came to task C, we had difficulty improving our results. The results didn't improve much after several experiments on tuning the model and it's parameters. Considering the small amount of data that is associated with the task, we thought transfer learning could help, but it preformed better only for task B.

5 Code

Final project NLP git

References

1. Predicting the Type and Target of Offensive Posts in Social Media
2. A Practitioners' Guide to Transfer Learning for Text Classification using Convolutional Neural Networks
3. Racial Bias in Hate Speech and Abusive Language Detection Datasets
4. Understanding Abuse: A Typology of Abusive Language Detection Subtasks
5. Automated Hate Speech Detection and the Problem of Offensive Language