Identifying and Categorizing Offensive Language in Social Media with Deep Learning

Vadym Gryshchuk
2gryshch@informatik.uni-hamburg.de

Christina Wiethof
7wiethof@informatik.uni-hamburg.de

Felix Jetschiny
4jetschi@informatik.uni-hamburg.de

Abstract

The ability of a learning system to identify offensive language is of high importance for microblogging and online social networking services. The posts that contain inappropriate or insulting language can be removed even before the users have completed writing them, consequently, providing more pleasant and safe communication between persons by avoiding violations of personal self-esteem and mobbing. This paper corresponds to the development project within the OffensEval 2019 competition. It aims at identifying and categorizing offensive language in social media with Deep Learning. We propose an architecture based on a bidirectional recurrent neural network. The performance of our architecture is evaluated by using F1 score.

1 Introduction

Nowadays offensive language attracts much attention in the digital world particularly in social media. On the strength of anonymity many users take this as a chance to engage in conduct that they would not risk in real life. Usernames as well as self-provided profiles support the creation of a new identity and thus the concealment of the real person. The new obtained courage to behave more impudently increases and with that the amount of offensive language in social media.

Thus, online communities, social media platforms and technology companies have already been carrying out research on the use and the possible prevention of offensive language and irresponsible behavior. Consequently there are many publications dealing with corresponding strategies and methods. One well-known and effective way is to identify offense, aggression and hate speech in user-generated content like posts, comments and microblogs by using computational methods.

In the context of the work at hand, the ”OffensEval 2019” is a competition organized at SemEval 2019 by CodaLab, which appeals many developers to come up with ideas to implement this computational identification and categorization method effectively and efficiently by means of neural networks and Deep Learning. Therefore, the OffensEval is broken down into three sub-tasks:

- Offensive language identification
- Automatic categorization of offense types
- Offense target identification

Pitsilis et al. proposed an architecture for detection of offensive language using an ensemble of reccurent networks (Pitsilis et al., 2018). They achieve high results by evaluating their approach on tweets. In our work we also consider recurrent neural networks, that are de facto standard to learn temporal connections in the input data. This paper deals with the first two sub-tasks.

At first, we present in chapter 2 the embedding model we use and describe the dataset. Then, in 3 we present our method for the identification and classification of offensive language by using recurrent neural networks. Finally, we show the results and discuss the performance of the proposed architecture.

2 Foundations

2.1 Embeddings

Word embeddings are vector representations of words from a large unlabeled text corpus. Word2Vec is the fundamental popular method for the encoding of words

Column name	Values	Description
ID	5 digit number	ID of the tweet
INSTANCE	Text of the tweet	The original text of the tweet
SUBA	NOT or OFF	This column is needed for Sub-task A NOT: Not Offensive - No offense or profanity OFF: Offensive - Post contains offensive language or a targeted offense
SUBB	TIN, UNT or NULL	This column is needed for Sub-task B TIN: Targeted Insult and Threats - Post containing an insult or threat UNT: Untargeted - Post containing non-targeted profanity and swearing
SUBC	IND, GRP, OTH or NULL	This column is needed for Subtask C IND: Individual - The target of the offensive post is an individual GRP: Group - The target of the offensive post is a group of people OTH: Other - The target does not belong to any of the previous two categories

Table 1: Structure of the dataset.

(Mikolov et al., 2013). The Word2Vec approach integrates two different model architectures for learning distributed representations of words. The first model architecture is the Continuous Bag-of-Words Model (CBOW) and it predicts the current word based on a context while the context contains a defined amount of history and future words. The advantage of this model in comparison to the standard bag-of-words model is that it uses continuous distributed representation of the context.

The second model architecture is the Continuous Skip-gram Model which tries to maximize the classification of a word based on another word in the same sentence. This is used to predict a certain range of words before and after the current word. The word vectors could be used to improve many existing NLP (Natural Language Processing) applications due to the semantic relationship (Mikolov et al., 2013).

The presented approach of this paper uses the FastText model (Bojanowski et al., 2017), which is another model for word representations. This model is derived from Mikolovs' Skip-gram Model. In addition to the Skip-gram Model, FastText uses a sub-word model which implements a different scoring function to take the information of the internal structure of a word into account. Therefore, each word is represented as a bag of character n-grams. Due to this approach the model allows sharing of the representations across words, thus allowing to learn reliable representations for rare words. This approach can be also described as morphological word representations (Bojanowski et al., 2017).

The FastText library provides word vectors for 157 languages trained on Wikipedia and Crawl[1]. For our approach we used the *crawl-300d-2M.vec* (Mikolov et al., 2018) pre-trained vector file, which contains 2 million word vectors trained on Common Crawl[2].

2.2 Dataset

The dataset for this task is provided by CodaLab[3]. There is one common dataset for all of the three sub-tasks. The file contains 13,240 tweets, which were annotated using crowd sourcing. Three annotators were taken into contribution for the assignment of the gold labels and no correction has been carried out on the crowd sourcing annotations. All of the user mentions contained in tweets were replaced by a general "@USER" annotation and also the URLs have been replaced by the term "URL". The dataset contains five columns, described in detail in Table 1. Deriving from this structure a tweet can have one of the following label combinations:

- [NOT, NULL, NULL]
- [OFF, UNT, NULL]
- [OFF, TIN, (IND | GRP | OTH)]

3 Approach

3.1 Data Preprocessing

Tweets contain a lot of information that is irrelevant or abundant, which can have a negative impact on the

[1]https://github.com/facebookresearch/fastText
[2]https://fasttext.cc/docs/en/english-vectors.html
[3]https://competitions.codalab.org/competitions/20011

learning of the model. Therefore, we cleanse the input data. Furthermore, some input data must be transformed.

We replace classification labels by unique numbers. If some category is underrepresented, upsampling is utilized, where the same input data is multiplied by a constant to achieve approximately the same distribution of samples per each category. Emoticons, stop words, punctuation, digits, hidden generalized information due to personal data protection guidelines like URL or $@USER$ are removed and verbs are lemmatized. Subsequently the tokenization is applied to remaining words. A tweet is now a list of numbers, where a number points to the corresponding word.

The length of a tweet is reduced to twenty words. If a tweet exceeds this number, the other words are not considered, otherwise zeros will be appended. Zero means that no word exists. The preprocessed data is then arranged into an embedding matrix. Each word is encoded by a vector of the size 300. We use the pretrained embeddings provided by the FastText model that contains 2 million word vectors. The resulting embedding matrix has two dimensions, where the first dimension represents the vocabulary size and the second dimension the vector size. The index of the first dimension corresponds to the index of a word in a mapping.

3.2 Model

The model receives a two-dimensional input, where the first dimension is a batch size and the second dimension is a tweet's length. An entry in this input represents a tweet. Tweets must be replaced by embedding vectors. Therefore, for each tweet and for each word in a tweet a vector representation is looked up in an embedding matrix. Consequently, the shape of the input data has now three dimensions, where the first dimension represents the batch size, the second dimension the tweet's length and the third the size of a an embedding vector.

To model the recurrent connections with memory and forget capabilities we utilize LSTM cells that are arranged into 3 layers. The number of hidden units in each cell is 256. We apply a dropout function to inputs and outputs of each LSTM cell with the probability of 0.5. Furthermore, a bidirectional RNN is used to encompass LSTM layers. Outputs from forward and backward paths from a bidirectional RNN are added together and forwarded into a fully-connected layer. The dense layer uses a ReLU activation function and has 128 neurons. The last layer of the model is the classification layer, which uses a softmax activation function and the number of neurons is set to the number of distinct classification labels. An RMSProp optimizer is used. To achieve better generalization L2 norm regularization loss is used. In Fig. 1 the proposed architecture is visually depicted.

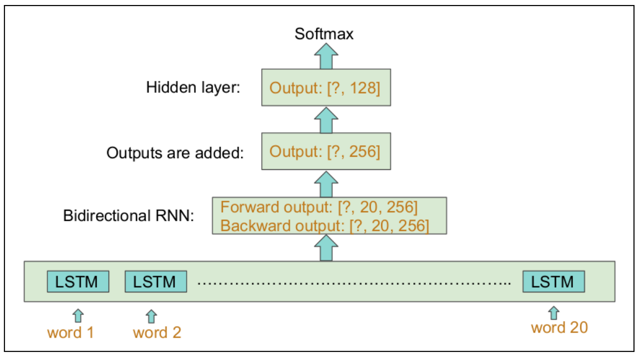

Figure 1: The architecture of the proposed model. LSTMs are arranged into three layers and encompassed as a bidirectional RNN. Outputs are added and forwarded into a fully-connected layer. Afterwards a classification layer with the softmax activation function is utilized.

4 Results

We have evaluated the proposed method for sub-tasks A and B. The dataset was divided into train, validation and test sets. The test set did not include the output labels and the model results were provided by the organizers of the OffensEval competition. The F1 score was used as an evaluation metric, where F1 score was calculated for each label and the unweighted mean was then calculated on individual values. This kind of F1 score is also known as macro F1 score[4].

We have achieved 0.789 F1 score on the sub-task A using test data. Fig. 1 presents a confusion matrix for the sub-task A. Our model achieves good results for identification of not offensive tweets. Only a small part of posts were categorized offensive, though they were not. The identification of offensive language is not very successful. A lot of tweets were misclassified.

For the sub-task B the model has achieved 0.687 F1 score using test data. The behavior of the model tends to be same as in sub-task A, where the model can classify correctly most of the tweets that belong to one cat-

[4]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

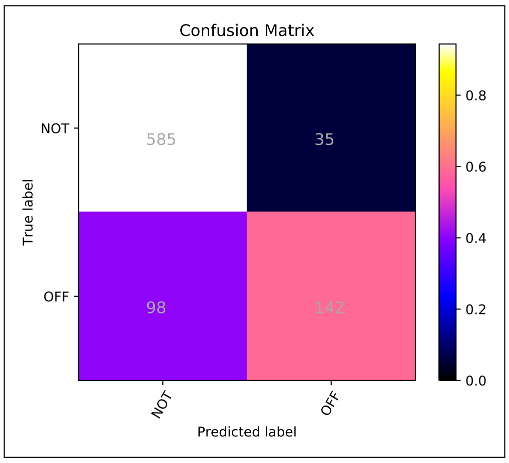

Figure 2: Confusion matrix for the sub-task A.

egory, but is unable to appropriately classify samples of the other category. In Fig. 3 we can see that the model identifies most of the targeted tweets correctly, but the performance neglects extensively the untargeted tweets.

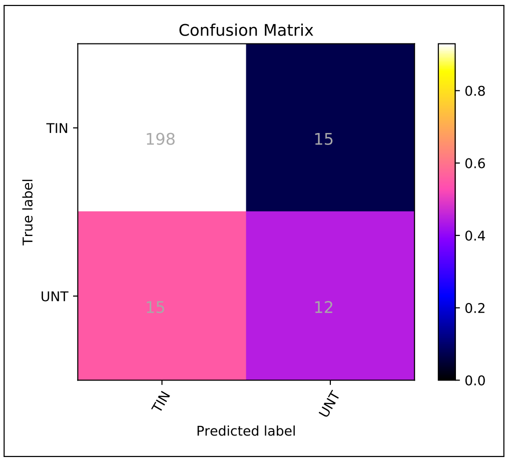

Figure 3: Confusion matrix for the sub-task B.

5 Discussion

The conducted experiments show that our approach works primarily in one direction, where it can identify correctly most of the tweets that belong to one category. For example, for the sub-task A it is the identification of non-offensive posts and of targeted insults for the sub-task B.

During the training process we have evaluated also the model on the validation set. The achieved validation accuracies were 0.74 and 0.91 for the sub-task A and sub-task B respectively. The resulting values are averaged over 5 iterations for better approximation. It can be seen, that the test and validation results for the sub-task A do not deviate significantly and the model has learned good representations and was not overtrained. For the sub-task B the validation and test results deviate significantly. It can lead to the conclusion that the model was overfitted on the train data and the validation set is very related to train set.

During the OffensEval competition we were able to get the 13th place out of 103 for the sub-task A and the 10th place out of 75 for the sub-task B. The best achieved macro F1 scores during the competition by other participants were 0.829 and 0.755 for the sub-tasks A and B respectively. To achieve better results ensemble learning can be used, where different architectures are utilized to get better results.

6 Conclusion

In this paper we have proposed an architecture for identification of offensive language in tweets as a part of the OffensEval competition. The proposed method utilizes a bidirectional RNN and pretrained word embeddings from the FastText model. The achieved results show that the proposed approach can be used to identify a non-offensive tweet or if a tweet is a targeted insult. For successful application of the method in real life scenarios further improvements and extensions are needed.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting Offensive Language in Tweets Using Deep Learning. *arXiv e-prints*, page arXiv:1801.04433.