TDDE16 Project - Classification of hate speech and offensive language on Twitter

Filip Cornell, filco306

January 8, 2019

Abstract

This article covers the topic of tweet classification, with the intent of classifying different tweets into either being offensive, using offensive language or not being offensive at all. For this, a convolutional neural network is applied.

1 Introduction

The internet's ability to permit everyone to speak their mind and share their knowledge opens up tremendous possibility. The amount of information possible to consume has never been greater for the ordinary person, and it has had a great impact on everyone's lives since it came to be. Social medias such as Facebook, Twitter and Instagram allow each and everyone to live a social life not only in real life, but also online. The internet is thus for most people an enriching experience, and some might even argue that it is the greatest invention of all time [3].

However, not everything on the internet can be considered completely positive. The possibility to speak freely also opens up for the less charming sides of the internet; harassment, hate speech and cyberbullying. This is a rising problem, and the possibility to control this by manually checking harassing posts is not only becoming increasingly difficult, but rather practically infeasible. As of November 2018, more than 500,000 posts were posted each minute on Facebook, yielding a massive amount of information [1]. If only 1 % of these were to be reported as offensive, this would lead to Facebook's employees having to manually check more than 2.6 billion posts each year - a huge cost. This has been a increasingly pressing issue, and the German government even threatened Facebook to fine € 50 million annually if they would not take serious measures tackle the issue of removing hateful and offensive posts [16]. Manual reviews' inefficiency is not only problematic in terms of resources, but also speed, as the posts are left online until removed, causing the damage it was intended for. Detecting and identifying offensive and hateful posts on social media in a quick, automatic and efficient way is thus a pressing issue, and a lot of work has been done previously within the area. In this paper, we investigate how some models previously tried can be combined in a new way and see what results this might bring. The intent is to use a Convolutional Neural Network (CNN) together with the `word2vec` embedding and compare this to the performance of models in previous work. Although several papers before has used CNNs together with `word2vec` to classify tweets, few seem to so far have used `word2vec` as word embeddings (although it has been done for sentences before [9]), but rather GloVe [4, 5, 7]. The paper starts with an introduction to the theory of `word2vec` and CNNs, followed by what previously has been done on tweet classification. This is followed by the method explained, and a more thorough explanation of the specific model used in this task. The results are then presented, followed by a discussion and conclusion.

2 Theory

In this section, the theory behind the models and tools used are presented, as well as previous work within the field.

2.1 Word2vec

In order to train a neural network to be used for natural language processing (NLP), the text must be mapped from text to vectors using a word embedding [12]. One technique to do this is `word2vec`, which uses Skip-gram and Continuous Bag-of-words (CBOW) [12].

The Skip-gram model will during training, given a sequence of words $w_{1:T}$, maximize the average log probability (1). [14]

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t) \tag{1}$$

The word probabilities $p(w_{t+j}|w_t)$ will be computed and trained by a neural network, where the probabilities are calculated with the softmax function (2)

$$p(w_O|w_I) = \frac{exp({\nu'_{w_O}}^T \nu_{w_I})}{\sum_{w=1}^{W} exp({\nu'_w}^T \nu_{w_I})} \tag{2}$$

ν_w and ν_w are the "input" and "output" vectors representing w.

2.2 Convolutional Neural Network (CNN)

A popular approach when analyzing and classifying text data is to use a CNN, originally mainly used in image and sound recognition [4]. Although CNN:s can be formed in many ways, it has a basic, intuitive structure. First, of all, they are not fully connected, but rather has one neurone's output going to only one other neurone in the next layer [2]. The first layer is a layer with convolutional filters. On the matrix inputted, a sliding window, also known as a kernel, feature detector or filter is applied. For each operation the sliding window multiplies each value

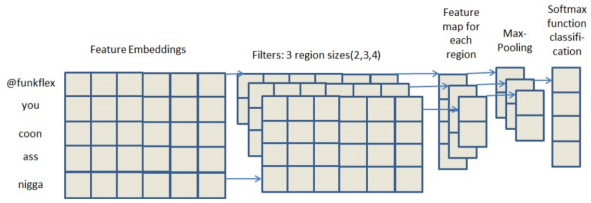

Figure 1: Hate-speech classifier

Figure 1: The layout of the CNN as proposed by Gambäck [7]

.

in the sliding window with the original matrix and sums these values up, creating a new output matrix, also called a feature map [2]. These are sent to a pooling layer, that reduces the size of these outputs by summarizing subregions of the feature map. There are several methods of doing this; one can take the maximum value (max-pooling) or the average value of the sub-region [6]. This basic structure can be used in several layers, along with other types of layers, but these two layers constitute the basic components of a CNN. If the mission of the model is to classify the input, the last layer usually consists of one neurone with the SoftMax function as activation function to classify into the different classes [12]. The SoftMax function can be written as

$$softmax(z)[i] = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \forall k \in K \tag{3}$$

where z_i denotes the output of the second last layer, and k denotes class K.

When using a CNN, the input size of the features is fixed. The whole sentence (in the NLP case) is thus inputted as a 2D-matrix into the neural network, with the words as rows, and their features as columns. Since, in practice, sentences can differ in length, one must pad the sentences with values to make them fit into the network. In image classification, the squares sent in usually represent images or compressed such [4] .

Two common activation functions in the layers in-between are the tanh function (equation 4) and Recti-

fied Linear Units (ReLU) function (equation 5). The tanh function, the previous norm, has been shown to have disadvantages compared to the ReLU function in terms of convergence, as the gradient does not saturate as quick. It has been shown that ReLU can converge up to six times faster than tanh [11].

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

$$f(x) = max(0, x) \tag{5}$$

2.3 Related work

Many different approaches have been tried in the field of identifying and classifying hateful speech on the internet before. The first research papers on the topic were published as early as 1997 [15], in which a tree-based classifier managed to reach an accuracy of 88.2 %. The data set was based 720 manually added notes on web page posts from then popular web pages. [15] Over the years, there has been a clear trend of classifying posts on the most popular social medias, such as Twitter and Facebook. In 2016, Waseem and Hovy [17] apply logistic regression combined with character n-gram, achieving an F1-score of 74 %. They are also able to extract the most indicative n-gram features in the set, indicating that features inside "muslim" or "islam" are the most indicative features for racism, while *"bitc"*, *"sex"* or *"xist"* indicate sexual harrassment.

On the same dataset as Waseem and Hovy [17], Gambäck et al. [7] applies a Convolutional Neural Network (CNN) to a set of tweets. To obtain the feature embeddings for the model, `word2vec` and character n-grams were used, while word embeddings were obtained by using `word2vec` and random word vectors. In the neural network, a SoftMax layer calculated the class probability. In between, a pooling layer is utilized to convert each tweet into a vector of fixed length. This pooling layer is followed by a max-pooling layer, used to capture the most important latent semantic factors in each tweet. The model resulted in a precision, at best, of 86 % using random vectors, and an F-score of 78.29 %, obtained through using character n-grams.

A. Gaydhani et al. (2018) [8] uses an n-gram and TF-IDF approach, with n ranging between 1 to 3. TF-IDF is used to be able to reduce the negative influence of tokens that are not really informative and appear very frequently. Each n-gram feature is then weighted in proportion to their TFIDF value. A final result showed results of an accuracy up to 95 % with logistic regression, 93.4 % for Naïve Bayes and 90.1 % for SVMs after tunings of hyperparameters. It must however be noted, that oversampling was performed on the dataset to balance out the dataset, which might create a bias towards the training data, and a possibility to overfit [18]. The results here might thus not be as generalizable as the authors might had desired. Also, the test set is also oversampled, yielding the risk of an unjust accuracy.

Badjatiya P. et al. [5] investigates the possibility to apply three different neural network architectures on classifying tweets; CNN, Long Short-Term Memory (LSTM) and FastText. On these, word embeddings with GloVe- or random beddings are applied. [5] On these, gradient boosting (GBDT) is also applied as learning method. The best result is achieved with LSTM, random embeddings and GBDT, achieving an F-score of 93 %. The GBDT makes a significance difference, seeing that the F-score of LSTM and random embeddings achieves an F-score of 84.8 %.

3 Data

To make this comparable to some previous work done, the same public datasets are used. With this, we can compare this implementation to the performance of logistic regression, SVM:s and a Naïve Bayes implementations [8]. The first two datasets are fetched from `data.world`, where the tweets has been determined as offensive, hate speech or not offensive by different users.[1] [2] The second dataset[3] is a dataset used in three previous works [8, 7, 17], where the data is classified as hateful or not, but rather

[1] `https://data.world/crowdflower/hate-speech-identification`

[2] `https://data.world/ml-research/automated-hate-speech-detection-data`

[3] Available at `https://github.com/ZeerakW/hatespeech`

divided into categories

Investigating the data further, the two data sets have a skewedness towards neither offensive nor hateful and offensive tweets, as seen in Table 1.

Table 1: Distribution of types of tweets in dataset1

Class/Dataset	**1**	**2**	**3**	$\sum$
Normal	7274	4163	11501	22938
Offensive	4836	19190	0	24026
Hate speech	2399	1430	5290	9119
Total	14509	24783	16791	56083

The first dataset contains 20 columns. However, only 2 are significantly interesting for the analysis.

- `does_this_contain_hate_speech` is classified into "The tweet uses offensive language but not hate speech", "The tweet contains hate speech"
- `tweet_text` contains the actual tweet.

The other columns display information such as the confidence of the labelling, based on the judgements from the manual labellers who has labelled the data, as some might have labelled some tweets differently. These would be interesting to incorporate in future work, but is left in this paper for simplification. Instead, the column `does_this_contain_hate_speech`

The second dataset contains only 7 columns, with two columns used for actual analysis.

- `class` - the class as judged by majority of users.
- `tweet` - a string containing the actual tweet to be analyzed.

The data has been labelled by users on CrowdFlower, who has made judgments on whether the tweets are offensive, contains hate speech or neither. Other columns includes information such as number of votes on each class, tweet id and such.

The third dataset only contains two columns; `class` and `id`, where class refers to "racism" or "sexism", indicating hate speech, or "neither?, indicating a normal tweet. Just like Gaydhani A. et al [8], we consider both "racism" and "sexism" to be hatespeech, and "neither" to be a normal tweet. It is important to note that not all tweets from this dataset were possible to be fetched. This was due to two reasons. First of all, some tweets did not seem to remain on Twitter, as they were not fetched. Secondly, some tweets were labelled as both "neither" and either "sexism" or racism in the dataset, making it impractical to use these as neither. These were thus dropped, and out of the 16907 original tweets, only 11155 were retrieved and used. For example, out of the 1970 tweets marked as racist, only 12 remained fetchable. It is probable that these have been removed due to its racist nature.

3.1 Preprocessing

First, the datasets are merged. The following preprocessings are then made (in given order).

1. All characters are set to lowercase.
2. All escaped characters, such as "&gt" and "&lt" are removed.
3. All url:s are removed.
4. The words are stemmed using the Porter Stemmer from the package `nltk`.
5. Some common words are changed, such as "im" to "i'm", "lil" to "little" and so on.
6. All stopwords are removed.
7. All non alpha-numeric character are removed.

Note that step 4, the stemming, will not be performed when using pre-trained word-embeddings.

4 Method

To briefly describe the method for the neural network, each tweet is converted to be represented as a matrix in numerical form using the word embedding `word2vec` using the package `gensim`. If a tweet is shorter than the most amount of words found in the set after preprocessing, padding is added to have the

same dimensions of every tweet. The data is then split up using different seeds (a total of 10 seeds will be used) and then CNN is trained using 80 % of the data in X epochs, and is then tested on the remaining 20 % (the test set). This is repeated on a number of seeds. The metrics used to measure the performance are then calculated based on the confusion matrix retrieved from the test sets, and the average of each metrics will be reported.

In order to perform a comparison with A. Gaydhani et al. [8] and ensure that the comparison is fair, we also create their best performing model; logistic regression with a regularizer C set to 100, solved with `liblinear`, using the package `sklearn`. This is to be able to validate the comparison and ensure that the comparison actually is fair. Since A. Gaydhani et al. used oversampling, the logistic regression model and the CNN-model will be run both with balanced and imbalanced data and compared. We will then also be able to conclude whether the oversampling introduces a bias.

It is also worth mentioning that the oversampling made by A. Gaydhani et al. many times includes the same tweets in both the training and the test set. Thus, we will also try out both methods when reducing the test set to only unique values, to yield a balanced training set but imbalanced test set, thus testing the oversampled model on a dataset similar to real life. This resulted however in only 4596 tweets in the test set and 35982 in the training set, which is a partition of roughly 89%/11%.

4.1 Word embedding

The word embedding `word2vec` is applied to the whole dataset. We use two embeddings: One with pre-trained word vectors, trained on more than 2B tweets[4], and one which we train on the entire dataset.

The pre-trained vectors have a dimension of X. When these are used, words will not be stemmed to match the words in the pre-trained vectors. Also, we will try both removing and keeping stopwords to see whether semantics are lost.

[4]Available at `https://nlp.stanford.edu/projects/glove/`

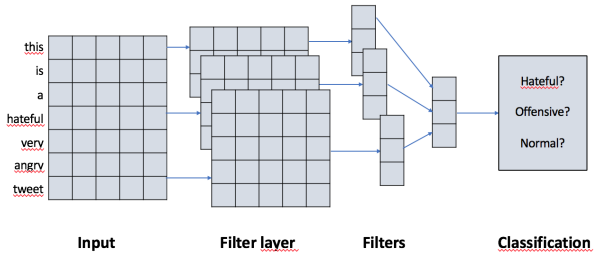

Figure 2: The network architecture used in this paper.

We test dimension of 25, 50, 100 and 200 on the word vectors and set a minimum count of 2 for a word to be included; in other words, if a word does not exist at least twice, we will not include it as a vector, and it will be considered non-informative and the corresponding vector will be set to a 0-vector. This will yield loss of information, and one might argue that `char2vec` might be useful for tweets as the number of characters is limited in a tweet to 140 characters (although it is discussed whether it is about to be doubled soon [13]).

4.2 CNN

The CNN used in this paper is quite simple and similar to the one used in Gambäck et al. [7], although slight modifications have been made. As input, we will have an NxM matrix with the word embeddings, where N is the maximum amount of words contained in a tweet, and M is the number of dimensions for the embedded vectors. The vectors will pass into three convolutional 2-dimensional layers with convolutions of 3, 4 and 5 respectively. The output of these ones, are passed on to its respective max-pooling layer, which then passes it on the last classification layer with only one neurone, the Softmax layer. In figure 2, the architecture for the network can be seen.

During training, gradient descent with the ADAM algorithm is used, which has shown to be computationally efficient and suitable for problems with large datasets [10]. It combines the advantages of the

4.3 Metrics

To make this comparable to previous works [7, 5, 8], the same metrics used in those will be used here. That is, the precision, recall and the F1-score, three measures commonly used in classification problems. As we have three classes here, it should be clarified that the measures are calculated as follows.

$$P_k = \frac{M_{ii}}{\sum_j M_{j,i}} \tag{6}$$

$$R_k = \frac{M_{ii}}{\sum_j M_{i,j}} \tag{7}$$

$$F1_k = \frac{2 \cdot P_k \cdot R_k}{P_k + R_k} \tag{8}$$

The average scores are then calculated by multiplying the classes' scores with their corresponding percentual representation in the test set.

5 Results

The confusion matrix can be seen in table 3. Table 2

Table 2: Explanation of classes

Type of tweet	**Class**
Hate speech	0
Offensive	1
Neither	2

Table 3: Confusion matrix of result on test set.

Class	**0**	**1**	**2**
0	**0.811**	0.052	0.065
1	0.092	**0.865**	0.137
2	0.097	0.082	**0.800**

Model	**Acc**	**Prec**	**Recall**	**F-score**
Log. regr. [8]	0.95	0.96	0.96	0.96
Log. regr. remade*	0.83	0.77	0.75	0.76
CNN pretrained vectors				
CNN, not pretrained				

6 Discussion

Although the overall accuracy achieved is comparable to previous works, it does not seem to match state-of-the-art. On overall accuracy, it does perform better. The TF-IDF approach tried out

As we can see in the accuracy matrix, few hateful or offensive tweets are classified as normal, but rather as offensive or hateful. However, the distinguishing between hateful and offensive seems a difficult task, and an accuracy of only 60 % is achieved for tweets containing hate speech. This might be due to a few reasons. First of all, the data set is unbalanced, with significantly fewer data points for hate speech tweets than the other two classes. Secondly, the manual labelling cannot be considered completely accurate. Some loss of accuracy might also be caused by the preprocessing stages causing loss of semantics. The preprocessing is also slightly different from previous works. However, trying to reproduce the results of the work of A. Gaydhani's et al. [8] did not yield the same results, but rather gave worse results than my model.

7 Conclusion

8 References

References

[1] Facebook statistics. https://zephoria.com/top-15-valuable-facebook-statistics/. Accessed: 2018-12-28.

[2] Introduction and explanation of. http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/. Accessed: 2018-12-28.

[3] List of the greatest inventions of all time. http://shortsleeveandtieclub.com/the-top-10-inventions-of-all-time/. Accessed: 2018-12-28.

[4] Hate speech detection using natural language processing techniques. 2018.

[5] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. *CoRR*, abs/1706.00188, 2017.

[6] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2016. cite arxiv:1603.07285.

[7] Björn Gambäck and Utpal Kumar Sikdar. Using convolutional neural networks to classify hate-speech. 2017.

[8] Aditya Gaydhani, Vikrant Doma, Shrikant Kendre, and Laxmi Bhagwat. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and TFIDF based approach. *CoRR*, abs/1809.08651, 2018.

[9] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, 2014.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[12] Marco Kuhlmann. Lecture, text mining course tdde16, 2018.

[13] Selena Larsson. Twitter doubles tweet length for some users. `https://money.cnn.com/2017/09/26/technology/business/twitter-tweet-280-characters/index.html`, September 27, 2018. Accessed: 2018-12-28.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[15] Ellen Spertus. Smokey: Automatic recognition of hostile messages. pages 1058–1065, July 1997.

[16] Emma Thomasson. German cabinet agrees to fine social media over hate speech. `https://uk.reuters.com/article/uk-germany-hatecrime-facebook/german-cabinet-agrees-to-fine-social-media-over-hate-` April 5, 2017. Accessed: 2018-12-28.

[17] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California, June 2016. Association for Computational Linguistics.

[18] Gary M. Weiss, Kate McCarthy, and Bibi Zabar. Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs? In *DMIN*, 2007.