





# Application of Data Augmentation Techniques for Hate Speech Detection with Deep Learning

Lígia Iunes Venturott<sup>(✉)</sup>  and Patrick Marques Ciarelli<sup>(✉)</sup> 

Universidade Federal do Espírito Santo, Vitória, Brazil  
patrick.ciarelli@ufes.br

**Abstract.** In the past decade, there has been a great increase in the usage of social media, and with it also an increase on dissemination of online hate-speech. Some of the most advanced techniques for online hate-speech detection are based on deep learning. Unfortunately, this kind of technique requires a large amount of labeled data, which is not so easy to find. One way of trying to overcome this problem is with the use of data augmentation techniques. The present paper explores data augmentation in order to improve the performance of deep neural networks on the task of hate speech detection on a small dataset in Portuguese.

**Keywords:** Hate-speech · Deep learning · Data augmentation

## 1 Introduction

In the past decade, there has been a great increase in the usage of social media. On 2019, the social network Twitter had approximately 330 millions of active users and 500 millions of tweets posted per day [11]. These platforms make communication possible between people from different cultures, religions and interests. Combined with the false feeling of anonymity, these factors generate a fertile environment for hate speech, such as racism, sexism, xenophobia, homophobia, and others.

Most social networks explicitly forbid the dissemination of this kind of speech. Several countries consider hate speech dissemination illegal, and in some places the platform can be held responsible if the post is not removed. Unfortunately, the great amount of users and posts makes the control of content an almost impossible task [18].

In this context, there is a need for automatic tools capable of detecting hate speech. There are several works on automatic hate speech detection. Several of them use classic machine learning techniques, like Malmasi and Zampieri [10] and Davidson et al. [2], that use SVM (Support Vector Machine) and logistic regression. Recently, there is a number of works that use Deep Learning techniques for NLP (Natural Language Processing) tasks. Even though deep architectures

can perform very well on NLP tasks [17], these models require a large amount of data in order to be trained. Well-constructed datasets are resources which take a lot of time and work to be build and, unfortunately, datasets are specific to each language, so that normally a dataset made of English texts cannot be used to train an algorithm for text classification in other languages.

Although there are several works on hate-speech detection for English, there is less work done in other languages. In case of Portuguese, for example, there are few datasets [7, 12], and they do not contain a large amount of examples. In addition to that, it is not always possible to apply the same methods used in language with extensive hate-speech, because models trained on small datasets are more prone to overfitting. One strategy to try to overcome this problem is using data augmentation.

Data augmentation is common in the field of computer vision, where operations such as cutting or inverting the picture, and adding noise, can substantially increase the size of the dataset. However, these same operations cannot be applied so easily in textual data, since they might change the meaning of the text.

In this paper we explore three data augmentation techniques described in Wei et al. [19] and try to improve the results of deep architectures on a small hate-speech dataset in Portuguese.

The next section presents some previous works related to the subject of automatic hate-speech detection. Section 3 describes the methodology used in this work, such as the dataset used, deep architectures, metrics, training and testing settings. Section 4 contains the results of the experiments carried out and in Sect. 5 we present our conclusions and final considerations.

## 2 Related Work

Some works have already tried to address the problem of hate speech or offensive language detection.

The work of Rosa et al. [14] uses deep neural networks to detect cyberbullying. In Del Vigna et al. [3], in addition to detecting the presence of hate-speech the authors also classify its intensity.

In Badjatiya et al. [1], several methods are explored: Support Vector Machines, Gradient Boosted Decision Trees, Random Forest, Logistic Regression and Deep Neural Networks. They also explore different encoding methods, such as bag-of-words vectors, TF-IDF vectors and GloVe embeddings.

Wei et al. [19] present four techniques of text data augmentation and evaluate them on benchmark text classification tasks. Two of the techniques explored do not require language-specific resources and can be applied to any language.

The majority of existing work is focused on the English language, but in 2017 de Pelle and Moreira [12] created a dataset of offensive comments in Brazilian Portuguese, containing comments taken from a news website. Fortuna et al. [7] also created a dataset for hate-speech in Portuguese. Their dataset is composed of tweets and the texts are classified in 81 hate categories in total.

Silva and Serapião [15] explore hate-speech detection in Portuguese. They use several methods for this task, including Bayes Naive Classifier, Support Vector Machine, Logistic Regression and Multi Layer Perceptron.

### 3 Methodology

In this section, we describe the applied methodology and the resources used in this work.

#### 3.1 Datasets

In order to evaluate our methodology, we used OffComBR, a dataset of offensive comments on Portuguese [12]. The dataset consists of 1,250 comments found in the Brazilian news website *g1.globo.com*, where each comment was classified by 3 judges into “offensive” and “non-offensive” categories. There are two versions of this dataset, OffComBR-2 and OffComBR-3. In OffComBR-2 the label of each comment is given by the majority of votes. In this version 419 comments (32.5%) were considered offensive by at least 2 judges. OffComBR-3 only includes comments where the 3 judges agreed upon the classification. It contains 1,033 comments, of which 202 (19.5%) are classified as offensive. We chose to use OffComBR-2 because it contains more examples than OffComBR-3.

#### 3.2 Data Pre-processing

Before applying any data augmentation we preprocessed the dataset. First, the words were lemmatized. In the lemmatization process inflexions are removed and the words are returned to their base form or dictionary form. For this, we used the NLPyPort module for Python [6]. The resulting dataset was not as good as expected, and this occurred mainly for two reasons:

- Due to the informal nature of the dataset, we are able to find several slangs and other words specific of internet vocabulary. Also, several words are spelled wrong, which makes recognition harder.
- The module has limitations and cannot recognize some verbal forms of the words.

Despite the suboptimal result, we decided to keep the lemmatized text in order to reduce the number of out-of-vocabulary words. After the lemmatization, we also removed stopwords.

#### 3.3 Data Augmentation

In this paper, we explored the use of data augmentation techniques for texts in order to overcome small dataset problems, such as overfitting. For this, we use 3 different data augmentation techniques summarized by Wei et al. [19] and compare them to the models’ performance on the original dataset.

**Random Deletion.** We randomly choose 1 word from the comment, this word is removed generating a new text entry. This procedure is repeated  $N$  times over the original comment, making sure that the removed word is different each time.

In the case the comment contains a number of words  $L$ , where  $L < N$ , the procedure is repeated  $L$  times, and not  $N$ .

Seeing that  $N$  is generally a very small number and we rarely have  $L < N$ , we could imply that this process generates a dataset of size approximately  $N + 1$  time the original.

We experimented with different values of  $N$ ,  $N = [1, 2, 3]$ .

**Random Swap.** Two words are randomly chosen in the comment, and their positions in the text are swapped. The dataset generated by this method is 2 times the size of the original, containing the original comments and the comments with the swapped words.

**Synonym Replacement.** Like in Random Deletion, we choose a word randomly from the comment. Using a synonym dictionary, we find a synonym and replace the original word. In the case we cannot find a synonym in the dictionary, we select another word, also randomly. This procedure is repeated  $N$  times, making sure not to repeat words.

The synonym dictionary used in this work was compiled by Dias-da-Silva et al. [5] and it can be found on GitHub<sup>1</sup>.

### 3.4 Sentence Encoding

We use pre-trained GloVe embeddings [13] of dimensionality 50. The embeddings were created by Hartmann et al. [8] and are available online<sup>2</sup>.

### 3.5 Architectures

**LSTM.** The Long Short-Term Memory network is a type of recursive neural network (RNN) normally used to analyse sequential data, such as text.

Recursive architectures analyse the information along the temporal axis. When analysing the information of moment  $t$  the network possesses information of the previous analysis done for moment  $t-1$ . In this way, RNNs are able to take order in account.

In this work, we use a simple LSTM model, where we use the last state as input to the Dense Layer. The shown in Fig. 1a.

<sup>1</sup> <https://github.com/stavarengo/portuguese-brazilian-synonyms>.

<sup>2</sup> <http://www.nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc>.

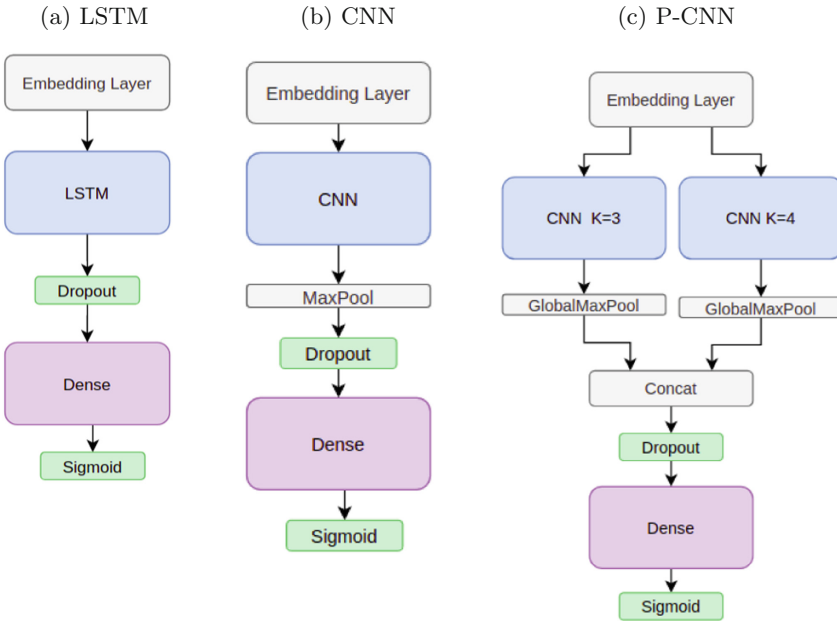
**CNN.** The Convolutional Neural Network (CNN) is a deep architecture that uses filters to extract information from the data. Although it was originally created to be applied to images in Computer Vision (CV), this architecture has shown good efficiency in Natural Language Processing tasks (NLP) [9].

We use the 1 dimensional CNN, where the kernel goes through the data along the temporal axis. Considering that the data has the shape  $T \times N$  where  $T$  is the number of words in the sentence and  $N$  is the size of the representation of each word, a kernel of size  $k$  will be of shape  $k \times N$ . A layer of the CNN is composed of  $D$  kernels in parallel. These  $D$  kernels process the same data, and their outputs are concatenated.

We used a simple CNN, with a ReLU activation function and Max Pooling, with pooling size  $p = 3$ . The output of the Max Pooling is flattened and dropout is applied. The resulting vector goes through a Dense Layer.

**Parallel CNN.** We also test a CNN architecture based on the one described in Badjatiya et al. [1]. In this architecture, we use 2 parallel branches, each containing a CNN with kernels of different sizes. For this architecture we fixed the sizes  $k = [3, 4]$ .

The output of each CNN goes through a ReLU activation function and a GlobalMaxPooling and then they are concatenated. We apply dropout to the resulting vector and then it goes through a Dense Layer.



**Fig. 1.** Architectures used in experiments

### 3.6 Experimental Setup

**Hyperparameters Optimization.** In order to optimize the hyperparameters for each model, we used the Bayesian Optimization algorithm [16]. We fixed the Adam optimizer for all the architectures and we used the Bayesian Optimization to find the best learning rate. The best hyperparameters can be found in Table 1.

**Table 1.** Hyperparameters used for LSTM, CNN and P-CNN

CNN		P-CNN		LSTM	
Kernel size	3				
Filters	45	Filters	45	LSTM size	25
Learning rate	0.01	Learning rate	0.01	Learning rate	0.03
Dropout rate	0.1	Dropout rate	0.1	Dropout rate	0.1

**Training-Testing Settings.** We used the “best model” approach, where the model is trained for a high number of epochs and on each epoch the validation accuracy is compared to the best validation accuracy so far. The model that obtains the best validation accuracy is stored and used for testing. In our experiment the number of epochs was set to 20, but we observed that, for every architecture, the best results were achieved on the first 10 epochs.

**Statistical Significance.** In order to make sure our results had statistical significance, we used the Wilcoxon signed-rank test [4]. The Wilcoxon test is a non-parametric statistical hypothesis test. We chose to use this test because it does not presume any distribution, such as the normal distribution. We use the k-fold cross-validation in order to use all the dataset for training and testing, with  $k = 5$ . We run this experiment 8 times for each architecture. The Wilcoxon test is applied over these samples. The numbers shown in the results are the average of the results.

### 3.7 Measures

We chose to use the F1 score to evaluate our models. Due to the imbalanced datasets, we believe F1 score would be a better measure than accuracy.

## 4 Results

The first experiments were made to see if the data augmentation techniques would bring any improvement. For the Random Deletion and Synonym Replacement techniques, we experimented different values of N to see which would present better results.

In order to guarantee statistical significance, we used the Wilcoxon test and calculated  $p$ . We only present the results of tests where  $p < 0.05$ .

First, we show the results for each architecture without applying any data augmentation technique. The results are shown in Table 2. Then, we evaluate the results for each architecture individually. In Tables 3, 4 and 5, the terms [del, swap, syn] refer to the augmentation techniques Random Deletion, Random Swap and Synonym Replacement, respectively. The number after the term refers to the value of  $N$  in the case of Random Deletion and Synonym Replacement.

**Table 2.** Results without applying any data augmentation technique

Arch	ACC	F1
CNN	0.69133	0.64673
P-CNN	0.78078	0.77379
LSTM	0.77989	0.77117

#### 4.1 Results with CNN

The simple CNN had worse results than other architectures. But, it also presented better relative results with the data augmentation techniques. All the augmentation methods applied presented  $p < 0.05$ . The F1 values and their increase are exposed in Table 3. Random Deletion with  $N = 2$  presented the greatest increase.

**Table 3.** Results of data augmentation with CNN architecture

CNN	F1	F1 increase (%)
del1	0.67983	3.310
del2	0.69068	4.395
del3	0.68636	3.963
swap	0.68916	4.243
syn1	0.68245	3.572
syn2	0.68703	4.030
syn3	0.68676	4.002

#### 4.2 Results with Parallel CNN

The Parallel CNN, or P-CNN, had much better results than the simple CNN. Most data augmentation methods applied showed some improvement with this

architecture. Only the Synonym Replacement with  $N = [1, 2]$  had  $p > 0.05$ , which leads to the question if a larger value of  $N$  could present more improvement. The results are shown in Table 4. Even though the Synonym Replacement method did not bring any significant boost in the results with  $N = [1, 2]$ , when  $N = 3$  it has the best results, with almost 2.3% gain.

**Table 4.** Results of data augmentation with P-CNN architecture

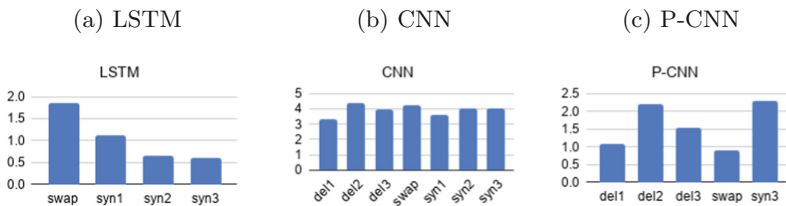
P-CNN	F1	F1 increase
del1	0.78475	1.096
del2	0.79570	2.191
del3	0.78927	1.548
swap	0.78270	0.890
syn3	0.79678	2.298

### 4.3 Results with LSTM

For the LSTM architecture, the only augmentation strategies that resulted in  $p < 0.05$  on the Wilcoxon test were Random Swap and Synonym Replacement with  $N = 1$ . Any other value for  $N$  in the Synonym Replacement generated increases too small to be considered relevant. We can see in Table 5 the results for the LSTM architecture. We chose to show the other results for Synonym Replacement in order to better observe the gradual decrease in gain with the increase of  $N$ . We can observe in Fig. 2 the increase for each technique.

**Table 5.** Results of data augmentation with LSTM architecture

LSTM	F1	F1 increase (%)
swap	0.78972	1.855
syn1	0.78246	1.129



**Fig. 2.** F1 increase with data augmentation for each architecture (%)



## 5 Conclusions

In this paper, we addressed the problem of small datasets on hate-speech for Portuguese and we proposed a method to try to handle the problem. We evaluated some data augmentation techniques on a small dataset and the effects when applied to common deep neural networks. It can be observed that these techniques provide some improvement on the results.

It is important to remark that these techniques are easy to implement, with no extra cost in training besides the enlargement of the dataset itself. Therefore, they can be easily used at most applications. Synonym Replacement is the only technique that requires extra resources, a language specific synonym dictionary. The other two techniques can be applied regardless of the language.

In summary, the methods presented provide some improvement, but they leave room for more exploration. Future work might include more complex methods of data augmentation and experimenting how these methods would affect the performance of other deep architectures.

**Acknowledgements.** We thank the Postgraduate Program in Electrical Engineering at Universidade Federal do Espírito Santo, and FAPES for the financial support.

## References

1. Badjatiya, P., Gupta, S., Gupta, M., Varma, V.: Deep learning for hate speech detection in tweets. In: *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 759–760. International World Wide Web Conferences Steering Committee (2017)
2. Davidson, T., Warmsley, D., Macy, M., Weber, I.: Automated hate speech detection and the problem of offensive language. In: *Eleventh International AAAI Conference on Web and Social Media* (2017)
3. Del Vigna, F., Cimino, A., Dell’Orletta, F., Petrocchi, M., Tesconi, M.: Hate me, hate me not: Hate speech detection on facebook (2017)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**, 1–30 (2006)
5. Dias-Da-Silva, B.C., Moraes, H.R.d.: A construção de um thesaurus eletrônico para o português do brasil. *ALFA: Revista de Linguística* (2003)
6. Ferreira, J., Gonçalves Oliveira, H., Rodrigues, R.: Improving NLTK for processing Portuguese. In: *Symposium on Languages, Applications and Technologies (SLATE 2019)* (June 2019) (in press)
7. Fortuna, P., da Silva, J.R., Wanner, L., Nunes, S., et al.: A hierarchically-labeled Portuguese hate speech dataset. In: *Proceedings of the Third Workshop on Abusive Language Online*, pp. 94–104 (2019)
8. Hartmann, N., Fonseca, E., Shulby, C., Treviso, M., Rodrigues, J., Aluisio, S.: Portuguese word embeddings: Evaluating on word analogies and natural language tasks. *arXiv preprint [arXiv:1708.06025](https://arxiv.org/abs/1708.06025)* (2017)
9. Kim, Y.: Convolutional neural networks for sentence classification. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751. Association for Computational Linguistics, Doha, Qatar, October 2014. <https://doi.org/10.3115/v1/D14-1181>, <https://www.aclweb.org/anthology/D14-1181>

10. Malmasi, S., Zampieri, M.: Detecting hate speech in social media. arXiv preprint [arXiv:1712.06427](https://arxiv.org/abs/1712.06427) (2017)
11. Omnicore: Omnicore. <https://www.omnicoreagency.com/twitter-statistics/> (2020). Accessed 15 Mar 2020
12. de Pelle, R.P., Moreira, V.P.: Offensive comments in the Brazilian web: a dataset and baseline results. In: Anais do VI Brazilian Workshop on Social Network Analysis and Mining. SBC (2017)
13. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014). <http://www.aclweb.org/anthology/D14-1162>
14. Rosa, H., Matos, D., Ribeiro, R., Coheur, L., Carvalho, J.P.: A “deeper” look at detecting cyberbullying in social networks. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2018)
15. Silva, A., Roman, N.: Hate speech detection in portuguese with naïve bayes, svm, mlp and logistic regression. In: Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional. pp. 1–12. SBC (2020)
16. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. arXiv preprint [arXiv:1206.2944](https://arxiv.org/abs/1206.2944) (2012)
17. Socher, R., Bengio, Y., Manning, C.D.: Deep learning for nlp (without magic). In: Tutorial Abstracts of ACL 2012, ACL 2012, p. 5. Association for Computational Linguistics, USA (2012)
18. Watanabe, H., Bouazizi, M., Ohtsuki, T.: Hate speech on Twitter: a pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. IEEE Access **6**, 13825–13835 (2018)
19. Wei, J., Zou, K.: Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint [arXiv:1901.11196](https://arxiv.org/abs/1901.11196) (2019)