

Text Augmentation for Neural Networks

Anna V. Mosolova¹, Vadim V. Fomin², and Ivan Yu. Bondarenko¹

Novosibirsk State University¹,
a.mosolova@g.nsu.ru

i.yu.bondarenko@gmail.com

National Research University Higher School of Economics²
wadimiusz@gmail.com

Abstract. This study considers the problem of using small text datasets for learning of neural networks. We explore the method used for image and sound datasets that augments data in order to increase the performance of models trained on it. We propose a method for augmenting that is based on synonymy.

Keywords: NLP, neural networks, small datasets, synonymy

1 Introduction

Natural language processing is an actively developing area today. Machine learning develops in this direction, and developers need for their approaches a lot of labeled data, but it costs a lot of hours of human's work. So, there is a need for increasing amount of data which was labeled earlier. These methods already exist in other parts of machine learning such as image classification, speech, and sound recognition, but all technologies that can be used for images and sounds are not suitable for text because of the danger of losing the sense of a sentence. These methods are named data augmentation and they are a common way to increase the performance of the model, avoid overfitting and improve the model's robustness. In this paper, we suggest a method for text augmentation that can improve the performance, does not very computationally cost and allows not losing the sense of the sentence. The paper consists of Related Work where we present some augmentation technologies, Methods where the model is described, Dataset provides the information about data for experiments and Experiments and Results represent the results of our work. In Conclusion part, some future goals are outlined.

2 Related Work

Data augmentation is a common problem in many areas of machine learning because it enables models to generalize better. This is crucial for fields where a good generalization is a challenging task, e. g. those where one must rely on small datasets. The problem of data augmentation has a certain history of research

in certain tasks, such as picture recognition or speech recognition. For instance, [1] suggests methods of image augmentation based on cropping, rotating, and flipping input images. They also suggest using GANs to generate images of different styles, and a new method that allows a neural net to choose which additions will improve the classifier. An approach to sound augmentation is suggested in [2], which proposed a method that consists in changing the speed of an audio signal, producing 3 versions of the original with speed factors of 0.9, 1.0 and 1.1. A Python library for sound data augmentation has been suggested in [3]. All of these methods allow increasing the performance of a model without using any additional data. However, there are no such methods for text augmentation, so our proposal will be one of the first open-source solutions.

3 Methods

3.1 Algorithm

We propose an approach to text augmentation which consists in using synonymous words instead of original ones without losing of sentence's sense. The process of augmentation runs in several steps which we will describe below. The first step implies excluding all words that do not require replacing like pronouns, conjunctions, prepositions, and articles so that they remain intact. The necessity to leave such words intact is explained by the fact that such words tend to have no synonyms and by its function which is to define the relationships of other words rather than to introduce a meaning. The second step is to find the synonyms for some words in the sentence. The words to be substituted are picked randomly depending on the initial settings (specifically, the percentage of replaced words). For example, if a sentence consists of 10 words and augmentation was set for 25%, the algorithm will substitute 2 words in the sentence. The choice of the synonyms is also random. During the next step, new words are put in the places of changed words and the algorithm returns the resulting sentence. The algorithm has a parameter that allows increasing the number of sentences in the corpora n-tuply. For example, Figure 1 shows a 7-times augmentation. Also, there is one additional option that saves writing in capital letters (This option is presented in Figure 2).

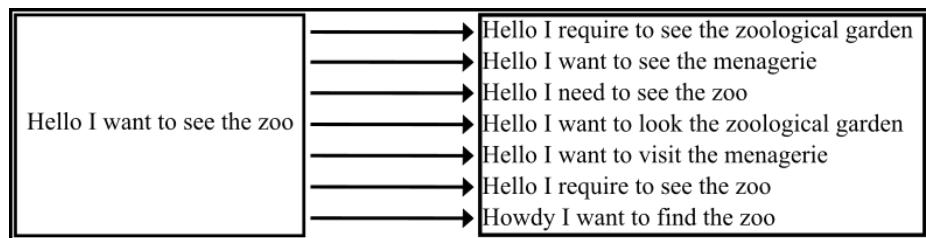


Fig. 1. An example of a 7-times augmentation with 25% changes

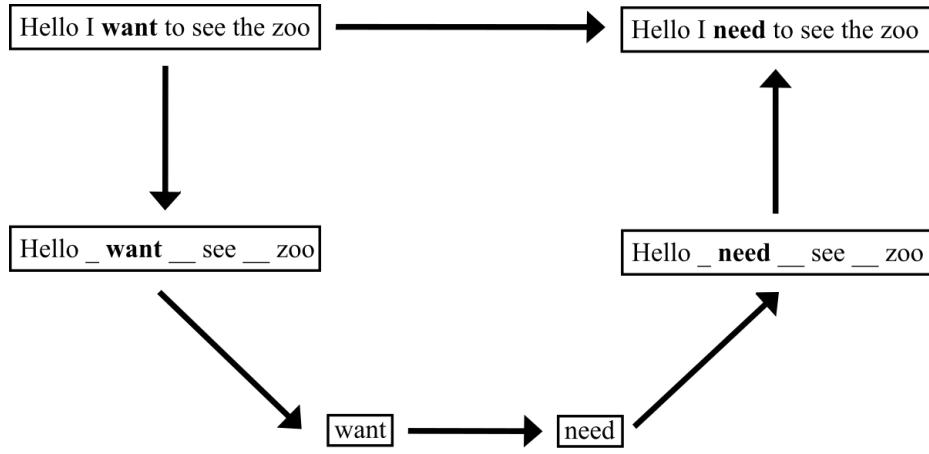


Fig. 2. The alrogithm of augmentation

3.2 Synonymy

The procedure of replacing the words with synonyms described in the paragraph 2.1 was realized by means of WordNet [4]. WordNet contains sets of synonymous words and represents a base of words which are related in some other ways. We used WordNet as one of the modules in NLTK [5]. Also, we used POS-tagging from NLTK for disambiguation of part-of-speech in the sentence. It caused some problems because POS-tags in NLTK differ from POS-tags in WordNet, so we added a module for changing it to the desired form.

3.3 Realization

We used for the realization of augmentation Python 3.6 and NLTK library, because it provides access to WordNet's base.

4 Dataset

It is the Toxic Comment Classification Challenge, a competition launched by Kaggle, that inspired us to test text data augmentation as an approach. The aim of the competition was to classify comments written by Wikipedia users against 6 binary classifications, each binary classification representing a certain type of toxicity. Thus, we used the dataset from this competition for our experiments and evaluation. The dataset is available at <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data>. The train set consisted of 159 571 samples, each of which was assigned 6 class labels, according to the 6 classification tasks. The test set consisted of 153 164 samples. The 6 binary classifications are related to the classes of toxic, severe toxic, obscene comments, threats, insults and identity hate. Each class is illustrated by examples 1–6 correspondingly:

- (1) Bye! Don't look, come or think of comming back! Tosser.
- (2) SHUT UP, YOU FAT POOP, OR I WILL KICK YOUR ASS!!!
- (3) A pair of jew-hating weiner nazi schmucks.
- (4) Hi! I am back again! Last warning! Stop undoing my edits or die!
- (5) Hey, you freaking hermaphrodite. Please unprotect your user page; I would like to move it to a more suitable title or three.
- (6) Bla bla bla....suck it Irishguy =)

The number of samples and their percentage for each class in the dataset is presented in Table 1.

Type	Samples	Percentage
Toxic	15249	9,6%
Severe toxic	1959	1,2%
Obscene	8449	5,3%
Threat	478	0,3%
Insults	7877	4,9%
Identity hate	1405	0,9%
Overall	159571	100%

Table 1. Dataset structure

Every comment that is marked as severe toxic is also labeled toxic. This is not the case with other classes. It is obvious in the table above that the classes are not quite balanced, e. g. the number of samples in the class 'toxic' is drastically larger than the number of comments that contain threats. It is also evident in the examples above that, while some classes are largely dependent on the presence of specific words in comments, others depend on the meaning of the sentence on the whole. For instance, classes 'identity hate' and 'obscene' rely on the presence of words that signify words that either insult a nation, a political view etc. or are obscene.

5 Experiments and results

5.1 Model

To solve the problem suggested in the competition, we used a convolutional neural network with 128 feature maps, 6 convolutional layers, 6 pooling layers and 2 dense layers and a dropout of 0.5. The feature representation of a sentence that was used as an input for the neural network consisted of vector representations of each word in a sentence. As a source of vector representations, we tried two embeddings trained upon the training set described in section 4: a word2vec model as a word embedding and a word2vec trained to predict character ngrams as a character embedding. The result is presented in Table 2.

5.2 Metrics

As a metric it was used ROC-AUC in this competition. The score of an algorithm is the average of the individual AUCs of each predicted type of toxicity.

Model	Public score	Private Score
CNN with character embeddings	0.9065	0.8933
CNN with character embeddings and with a 6 times augmentation for 25% of all words	0.9436	0.9446
CNN with word embeddings	0.9752	0.9742
CNN with word embeddings and with a 6 times augmentation for 25% of all words	0.9743	0.9721

Table 2. Results of the Kaggle competition

6 Analysis

It is obvious from the evaluation presented above that text data augmentation appeared capable of making character embeddings more relevant for classification but did not affect the usefulness word embeddings in any way. This is because vector representations of synonymous words in word embeddings are very close. As a result, the artificial samples from the augmented training set are very close to the existing ones, which is not the case for character embeddings.

7 Conclusion and Future Work

Data augmentation has been shown to produce promising ways to increase the accuracy of classification tasks. In this paper, we proposed an algorithm that worked well in the competition from Kaggle and it can be used by researchers as it free distributed on gitlab. We are going to develop our augmentation model and add the possibility of augmenting Russian texts using synonyms from Wiktionary.

References

1. Wang, J., Perez, L.: The effectiveness of data augmentation in image classification using deep learning (No. 300). Technical report (2017)
2. Ko, T., Peddinti, V., Povey, D., Khudanpur, S. Audio augmentation for speech recognition. In Sixteenth Annual Conference of the International Speech Communication Association (2015)

3. Salamon, J., MacConnell, D., Cartwright, M., Li, P., Bello, J. P.: Scaper: A library for soundscape synthesis and augmentation. In Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017 IEEE Workshop on (pp. 344- 348). IEEE. (2017)
4. Miller, G. A.: WordNet: a lexical database for English. Communications of the ACM, 38(11), 39-41 (1995)
5. Bird, S., Loper, E.: NLTK: the natural language toolkit. In Proceedings of the ACL 2004 on Interactive poster and demonstration sessions (p. 31). Association for Computational Linguistics (2004)