University of Jean Monnet

Machine Learning And Data Mining

(MLDM)

Toxic Comment Classification

Authors:
Aleksei Tcysin
Archit Jain
Samaneh Zareian Jahromi

January 12, 2020

Contents

Abstract

The background of this problem starts from the online community and other social medias, wherein people sometimes leave abusive remarks or hatred comments on some sensitive matter, which further on escalates the situation worse. So, it becomes the responsibility of the organisation to find these comments and remove them. We explore solutions based on Convolutional and Recurrent Neural networks and compare their performance with standard Support Vector Machines. The results are consistent with recent research, but further work is needed to push the performance to the state-of-the-art.

1 Introduction

Toxic comment classification has become an active research field with many recently proposed approaches. However, while these approaches address some of the task's challenges others still remain unsolved and directions for further research are needed.

There are some platforms we use to effectively facilitate conversations, however discussing things you care about can be difficult due to toxic comments. Toxic comments are comments that are rude, disrespectful or otherwise likely to make someone leave a discussion. As a result, many people stop expressing themselves and give up on seeking different opinions and many communities limit or completely shut down user comments.

Deep learning models have been used extensively and achieved remarkable results in various fields such as computer vision and speech recognition. Within natural language processing, much of the work with deep learning methods has involved learning word vector representations through neural language models [7].

In this paper we evaluate CNN, RNN and SVM models. The goal is to identify and classify toxicity in online comments.

2 Literature Review

Our approach is based on CNNs, LSTMs and SVMs. We outline some important definitions.

Convolutional Neural Network Model (CNN) is a multistage trainable Neural Networks architectures developed for classification tasks.

Convolutional layer consists of a number of kernel matrices that perform convolution on their input and produce an output matrix of features where a bias value is added. The learning procedures aim to train the kernel weights and biases as shared neuron connection weights.

Pooling layer performs dimensionality reduction of the input features. Pooling layers make a sub-sampling to the output of the convolutional layer matrices combining neighboring elements. The most common pooling function is the max-pooling function, which takes the maximum value of the local neighborhoods.

Embedding layer is a special layer used for text classification. It transforms each word of a text document into a dense vector of fixed size.

Fully-Connected Layer is a classic Feed-Forward Neural Network hidden layer. It can be interpreted as a special case of the convolutional layer with kernel size 1×1. This type of layer belongs to the class of trainable layer weights and it is used in the final stages of CNNs.

Long Short-Term Memory networks (LSTM) are a special kind of Recurrent Neural Network, capable of learning long-term dependencies. Vanilla Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they'll have a hard time carrying information from earlier time steps to later ones. During back propagation, they also suffer from the vanishing gradient problem - earlier layers get progressively smaller gradient update and eventually stop learning. LSTMs are designed to avoid the long-term dependency problem.

The core concept of LSTM's are the cell state, and it's various gates. The

cell state acts as a transport highway that transfers information all the way down the sequence chain.

The *forget* gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

To update the cell state, we have the *input* gate. First, we pass the previous hidden state and current input into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important.

Last we have the *output* gate. The output gate decides what the next hidden state should be. Remember that the hidden state contains information on previous inputs. The hidden state is also used for predictions. First, we pass the previous hidden state and the current input into a sigmoid function. Then we pass the newly modified cell state to the tanh function. We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry.

Often times words have to be encoded before feeding them to NN-based models. For this purpose we use a vocabulary. The vocabulary is constructed as an index containing the words that appear in the set of document texts, mapping each word to an integer between 1 and the vocabulary size. An example of this encoding is illustrated in Figure 1.

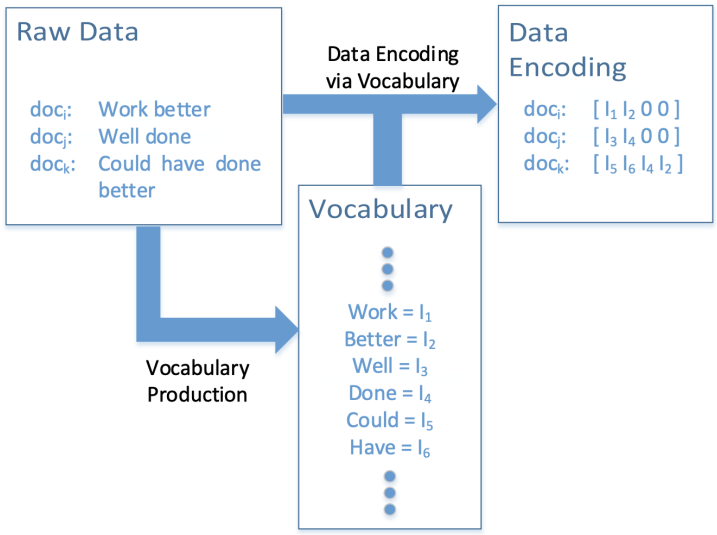

Figure 1: An example of encoding a text using a vocabulary.

On the other hand, CNNs require a constant input dimensionality. As there are documents with variable lengths -number of words in a document- we use the padding technique, filling with zeros the document matrix in order to reach the maximum length amongst all documents in dimensionality.

2.1 Related Work

Toxic comment detection is similar to standard text classification tasks. Related research [1] includes investigation of hate speech, online harassment, abusive language, cyber-bullying and offensive language. Each uses different definitions for their classification, still similar methods can often be applied to different tasks.

Besides traditional binary classification tasks, related work considers different aspects of toxic language, such as *racism* and *sexism*, or the *severity* of toxicity. These tasks are framed as multi-class problems, where each sample is labeled with exactly one class out of a set of multiple classes. The great majority of related research considers only multi class problems. This

is remarkable, considering that in real-world scenarios toxic comment classification often be seen as a multi-label problem, with user comments fulfilling different predefined criteria at the same time [1].

Shallow classification and neural networks. Toxic comment identification is a supervised classification task and approached by either methods including manual feature engineering or the use of(deep) neural networks. While in the first case manually selected features are combined into input vectors and directly used for classification, neural network approaches are supposed to automatically learn abstract features above these input features. Neural network approaches appear to be more effective for learning, while feature-based approaches preserve some sort of explainability. [1]

3 Methodology

3.1 Data Processing

Before developing any method to distinguish toxic comments from non-toxic ones, there are a few steps to apply on the data itself before doing anything else onto it. We label comment as toxic if it contains any form of misbehavior, non-toxic otherwise. Further, we balance the data by randomly sampling non-toxic samples to match the count of toxic ones. We use 80% of the data for training and remaining 20% for testing.

For CNN, words are encoded using vocabulary in the following manner: we keep 10000 most frequent words and encode them using integers. The comments have different lengths, so we use post-pad short comments with zeros and cut longer comments to achieve common length of 40 words per comment.

For LSTM, we keep all found words and pad sentences to maximum length of 400.

For SVM, we employ bag-of-words encoding with TF-IDF weighting scheme.

We keep top 300 words ordered by term frequency across the corpus.

3.2 Convolutional Neural Network

For Convolutional Neural Network, we employ architecture described in [6] and [3].

The input is passed through trainable embedding layer, where encoded words are mapped to high-dimensional vectors of length 300. The weights are initialized randomly and tuned during the training. Embedding layer produces a matrix, where rows are words and columns are embedded vectors. Further, we use three different convolutional layers simultaneously, with 128 filters each. Filter width is equal to embedding dimension, heights are 3, 4 and 5 respectively. After each convolutional layer, max-over-time pooling [2] is employed. We concatenate output of pooling layers and pass it to the final fully-connected layer with *sigmoid* activation function. For regularization, we apply dropout (0.5) before the last layer with a constraint on l2-norms of the weight vectors [4]. The model is trained using Adam algorithm with learning rate 0.005 and mini-batches of size 64.

3.3 Long Short-Term Memory Network

Model was created using functional API Model(), which takes the input and output. Input was created using the embedding of Glove vector, and output was 1D after applying the sigmoid function.

Lets look at architecture, first we create an input layer with the size 60 and pass and embedding layer to it. Create a max pool layer which would reshape the 3D tensor (created in previous layer) to 2D. Drop 10% of the nodes. Create a dense network with 50 nodes, pass a Relu activation. Again drop out 10% of the nodes. In the end and squeeze the layer into 1 dimensional output by applying the sigmoid function. We use Adam optimiser with default parameters and train for 2 epochs.

Model	F1-score
LSTM	0.90
CNN	0.89
SVM	0.80

Table 1: F1-score scores across all the experiments.

4 Results and Discussion

The results are presented in Table 1. Generally, they are consistent with findings of previous research [3], but still far from top results of Toxic Comment Classification Challenge [5]. We tried to experiment with vocabulary size, maximum sentence size and embedding dimensions. However, changing mentioned hyperparameters does not improve the performance.

Going further, in order to improve the performance we can try a number of things. One would be to work with the data - add more samples for toxic comments, use better cleaning procedure to remove / correct misspelled words. The other is to use NLP-related techniques (lemmatizing, POS tagging, sentiment classifiers, etc.). Yet another idea is to use transfer learning by initializing embedding layer with weights from pre-trained words embeddings. Finally, one might experiment with model architecture.

References

[1] Betty van Aken et al. "Challenges for toxic comment classification: An in-depth error analysis". In: *arXiv preprint arXiv:1809.07572* (2018).

[2] Ronan Collobert et al. "Natural language processing (almost) from scratch". In: *Journal of machine learning research* 12.Aug (2011), pp. 2493–2537.

[3] Spiros V Georgakopoulos et al. "Convolutional neural networks for toxic comment classification". In: *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*. ACM. 2018, p. 35.

[4] Geoffrey E Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[5] Kaggle. *Review of Deep Learning Algorithms for Image Classification*. 2017. URL: `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview`.

[6] Yoon Kim. "Convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1408.5882* (2014).

[7] P. Vincent Y. Bengio R. Ducharme. "Neural Probabilitistic Language Model". In: *Journal of Machine Learning Research 3:1137–1155* (2003).