Machine Learning Final Project

Toxic Comment Classification using Bidirectional LSTM and Convolutional layer

Anton Bilchuk / Ivan Prodaiko

Ukrainian Catholic University, May 2019

Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments. [3]

Contents

1 Introduction

Toxic comment is the type of written message containing thread, abuse or harassment.

People like to express themselves in different ways, but none of these ways should abuse others. During internet chatting it is vital to have a safe environment, as well as in society, so using toxic comments is not allowed. Despite most of the public chatting platforms notify their users about the rules of communication, some people break them.

Since October 2017 EU started enforcing social media sites to remove hate speech, fake news, and illegal material in just 72 hours. So, more and more websites require an automatic approach to clean their chats from hate.

We propose a natural language processing model architecture that solves the task with accuracy 95% on a validation dataset.

2 Data collection and preparation

In this competition, We're challenged to build a multi-headed model that's capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate better than Perspective's current models. We'll be using a dataset of comments from Wikipedia's talk page edits. Improvements to the current model will hopefully help online discussion become more productive and respectful. [3]

On the way to apply a model the data require preprocessing step, thus authors tried to improve the quality of row data and to remove some disambiguity from it.

The dataset was prepared by Kaggle and it is a table data with comment texts and it's classification as toxic or not. The first thing we have to start while working with this dataset is to map the comment text column to ready for model state. We have to lowercase words, remove punctuations, and other transformations that are standard in NLP data preparation. In the end, we would transform all words into vectors.

2.1 Preprocessing

Firstly we need to map our data to lower register in order not to differentiate such words as "Bird" and "bird". It is very common that sentences contain

HTML tags, so we need to clear our data from it.

Furthermore, we need to clean our data from punctuations, such as "word?" or "word,". Some words may be written in different forms, such as "centre" and "center", "shoudnt" and "should not", "'cause" and "because". We need to unify these words into one form. We also replace all swear words to the most frequently used one.

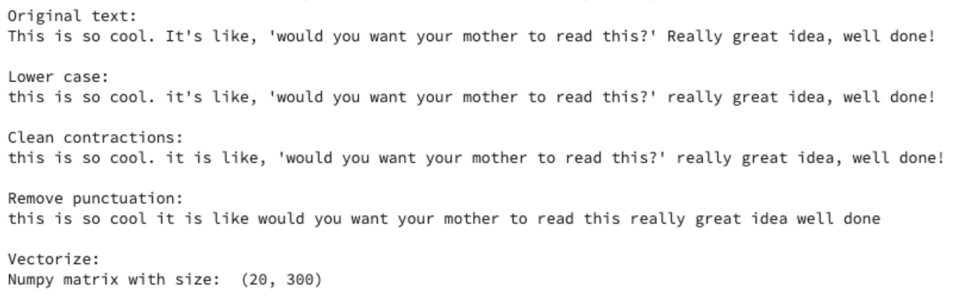
Common approach in NLP stops words removal. But in our case we would use not just frequencies of words, but rather the way words connected to each other, thus removing stop words may affect the overall meaning of the sentence.

We applied tokenization to text in order to obtain the sentences. Tokenization is a process of converting different forms of a word into one, unified. Thus the application of statistical tools is easy to do. If the sentence does not contain enough words (length of the vector equals the longest sentence in the dataset), then we add padding to fulfill the vector.

Than tokens are transformed into vector form, which allows performing computational over vectors rather than the list of strings, providing a convenient way to process the data by the neural network.

Here is an example of how preprocessing looks like:

Figure 1: Preprocessing stages



```
Original text:
This is so cool. It's like, 'would you want your mother to read this?' Really great idea, well done!

Lower case:
this is so cool. it's like, 'would you want your mother to read this?' really great idea, well done!

Clean contractions:
this is so cool. it is like, 'would you want your mother to read this?' really great idea, well done!

Remove punctuation:
this is so cool it is like would you want your mother to read this really great idea well done

Vectorize:
Numpy matrix with size:  (20, 300)
```

3 Modeling

Before choosing a model that is described in sections below we need to talk about its selection. Candidates were:

1. **Rule based solutions**

   Rule-based solutions could help to identify abuse words but it could not give any aid in extracting the sense from sentences that has no rude words, but still could be considered as toxic.

2. **MLP**

   Multilayer Perceptrons, or MLPs for short, are the classical type of neural network.

   They are comprised of one or more layers of neurons. Data is fed to the input layer, there may be one or more hidden layers providing levels of abstraction, and predictions are made on the output layer, also called the visible layer.

   In this particular case, the usage of MLP is not really motivated since we use text data and each word meaning depends on the previous words in the sentence.

3. **RNN**

   Recurrent Neural Networks, or RNNs, were designed to work with sequence prediction problems.

   Sequence prediction problems come in many forms and are best described by the types of inputs and outputs supported.

   The one major drawback RNN has is that it has a threshold for its memory.

4. **LSTM**

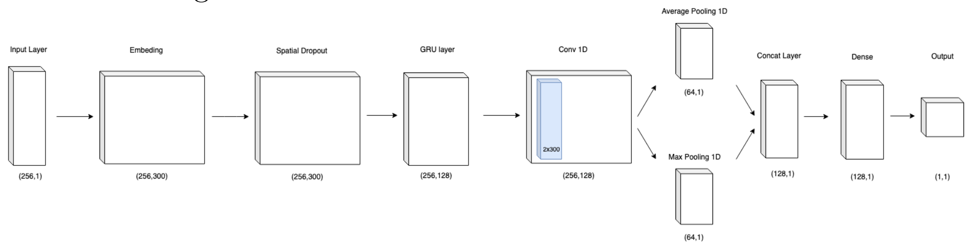
   LSTM is a particular architecture of the recurrent neural networks, which are considered to be the most successful when working with sequences of words and paragraphs, generally called natural language processing.

   The main idea of LSTM is having a so-called "state" or "memory" inside each neuron which has an impact on the next activation result.

But comparing with RNNs the number of previous words that influence the result could be defined.

The architecture of the network is shown in Figure 1. It has embedding, convolutional, spatial dropout, GRU and dense layers presented. We described layers of our neural network architecture below.

Figure 2: An illustration of our neural network



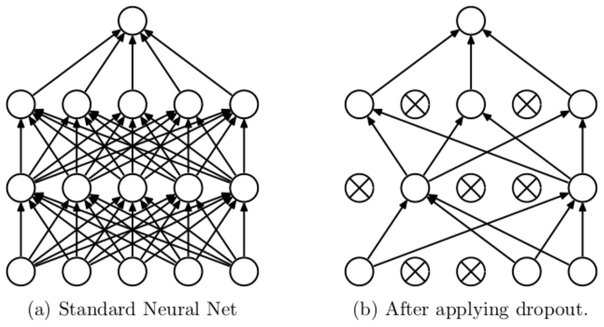
3.1 Embedding layer

An embedding is a mapping from a discrete categorical variable to a vector of continuous numbers.
In our case, we map the unique identifier of a word to 300 dimension vector. This vectors can be used for representation how close are words, what is the difference between them, clustering and so on.
For embedding matrix, we used 1 million word vectors trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset. It is pretty small(2.2 GB) so we can easily use it for neural network and have enough words for our specific task.

3.2 Spatial dropout

Dropout is a method to prevent overfitting [5]. At every training stage, random nodes could be dropped out with probability p. After that only the changed network is trained on the data in that training stage. Removed nodes would be inserted back with unchanged weights. A good note is that it also increases the speed of training.
Spatial dropout 1D performs the same function as dropout, but it would remove all 1d dimension instead of individual elements. It is useful for word processing.

Figure 3: An illustration of dropout

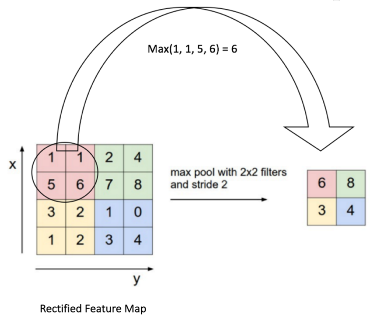


3.3 Convolutional neural networks

CNN technique is commonly used in image processing as it extracts features, patterns from images and remembers them. It is handy for images. But it was found out that is is also very useful in text processing. It can detect different patterns from word vectors as well.
We apply a convolutional layer with 64 filters and kernel size 2. It means that we would find a specific relationship between every two words in order.

3.3.1 Max Pooling

Figure 4: An illustration of max pulling layer



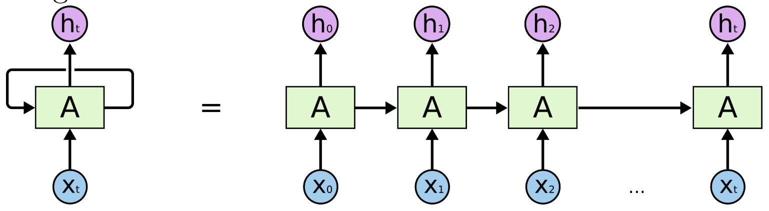
In max pooling layer we find the max value of filter representation out of two words.

3.3.2 Average Pooling

In average pooling layer, we find the average value of filter representation out of two words.

3.4 RNN

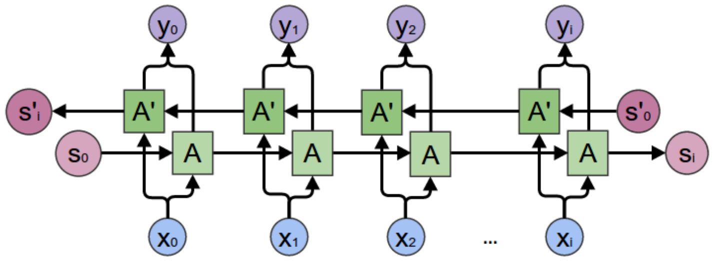
Figure 5: An illustration of recurrent neural network



There are some issues when we try to use classic neural networks with text classification. For example, it does not share features learned across the different position of texts that can cause errors.
RNNs share features between each training step of input data that allows it to learn sequence patterns better. It learns some pattern from the first word and then pass this knowledge to the second word and so on.

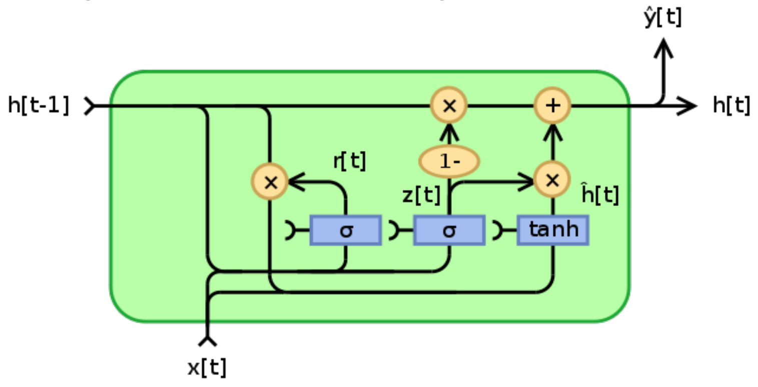
3.4.1 Bidirectional RNN

Figure 6: An illustration of bidirectional recurrent neural network



The difference between classic RNN and bidirectional is that it use features not only from early in the sequence but also later in the sequence.

Figure 7: An illustration of gated recurrent unit



3.4.2 GRU

Gated Recurrent Unit is a modification of RNN architecture [2]. It is not passing features of prev step, but instead, it uses specific cell where patterns are stored. And the gate which decides whether to gate pass previous feature to next Cell or not. Forget gate is nothing but additional Mathematical Operations with a new set of Weights.
For example, it can remember the subject of a sentence and do not overwrite this information till the end or when we subject would appear.

4 Evaluation

To evaluate our model accuracy, we test it on a validation set. We split train dataset into 90% for training and 10% for validation.

For model coding, we used python 3.6, Keras as a frontend for neural network and Tensorflow as a backend.

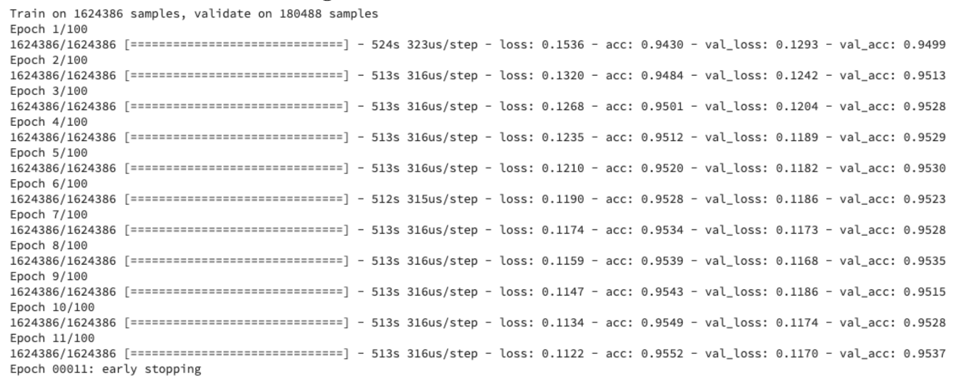
The model was run on Google Cloud machine. It's configuration:

- GPU: Tesla K80
- CPU: 8 kernels, Unknown CPU Platform
- RAM: 30GB

For pertained embedding matrix we used 1 million word vectors trained on Wikipedia. [1]

4.1 Result

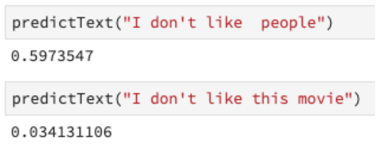
Figure 8: Model train results



```
Train on 1624386 samples, validate on 180488 samples
Epoch 1/100
1624386/1624386 [==============================] - 524s 323us/step - loss: 0.1536 - acc: 0.9430 - val_loss: 0.1293 - val_acc: 0.9499
Epoch 2/100
1624386/1624386 [==============================] - 513s 316us/step - loss: 0.1320 - acc: 0.9484 - val_loss: 0.1242 - val_acc: 0.9513
Epoch 3/100
1624386/1624386 [==============================] - 513s 316us/step - loss: 0.1268 - acc: 0.9501 - val_loss: 0.1204 - val_acc: 0.9528
Epoch 4/100
1624386/1624386 [==============================] - 513s 316us/step - loss: 0.1235 - acc: 0.9512 - val_loss: 0.1189 - val_acc: 0.9529
Epoch 5/100
1624386/1624386 [==============================] - 513s 316us/step - loss: 0.1210 - acc: 0.9520 - val_loss: 0.1182 - val_acc: 0.9530
Epoch 6/100
1624386/1624386 [==============================] - 512s 315us/step - loss: 0.1190 - acc: 0.9528 - val_loss: 0.1186 - val_acc: 0.9523
Epoch 7/100
1624386/1624386 [==============================] - 513s 316us/step - loss: 0.1174 - acc: 0.9534 - val_loss: 0.1173 - val_acc: 0.9528
Epoch 8/100
1624386/1624386 [==============================] - 513s 316us/step - loss: 0.1159 - acc: 0.9539 - val_loss: 0.1168 - val_acc: 0.9535
Epoch 9/100
1624386/1624386 [==============================] - 513s 316us/step - loss: 0.1147 - acc: 0.9543 - val_loss: 0.1186 - val_acc: 0.9515
Epoch 10/100
1624386/1624386 [==============================] - 513s 316us/step - loss: 0.1134 - acc: 0.9549 - val_loss: 0.1174 - val_acc: 0.9528
Epoch 11/100
1624386/1624386 [==============================] - 513s 316us/step - loss: 0.1122 - acc: 0.9552 - val_loss: 0.1170 - val_acc: 0.9537
Epoch 00011: early stopping
```

We trained our model with the dataset of 1,624,388 samples. The model was configured to stop after there would be no improvements during three epochs, in the other word when the model starts overfitting. The model training was finished on 11 epoch with validation accuracy to be equal to **95.37%**

4.2 Examples of model prediction



```
predictText("I don't like  people")
0.5973547
predictText("I don't like this movie")
0.034131106
```

As we can see from the above result, the model does not only react on some trigger words like "don't like" but also makes attention to the context.

4.3 Code

The train and test evaluation is in GitHub repository: https://github.com/IvanProdaiko94/machine-learning-final-project-UCU

5 Conclusions

Our model performed well on the validation set and showed 95% accuracy. And it does not only react on some trigger words like swears or insults but also understands the context of the sentence and reacts respectively.

References

[1] 1 million word vectors trained on wikipedia 2017. https://fasttext.cc/docs/en/english-vectors.html.

[2] On the properties of neural machine translation: Encoder-decoder approaches arxiv:1409.1259.

[3] Jigsaw/Conversation AI. Toxic comment classification challenge, 2018.

[4] Maria Dobko. Nlp text data cleaning and preprocessing, 2018.

[5] A. Krizhevsky I. Sutskever G. E. Hinton, N. Srivastava and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2017.

[6] Roopal Garg. Deep learning for natural language processing : Part 2 - rnn, 2018.

[7] Bishop H.M. *Pattern Recognition and Machine Learning.* Springer, 2006.

[8] Ujjwal Karn. An intuitive explanation of convolutional neural networks, 2016.

[9] Ceshine Lee. Understanding bidirectional rnn in pytorch, 2017.

[10] Alex Krizhevsky Ilya Sutskever Ruslan Salakhutdinov Nitish Srivastava, Geoffrey Hinton. Dropout: A simple way to prevent neural networks from overfitting. 2014.