Text Sentiment Classification with Commitees of Convolutional Deep Neural Networks

Andrey Ignatov

***Abstract*—Regarding the immense traffic within social networks, it is, without using automated evaluations, impossible to get a justified estimate about the opinion of the masses towards certain topics.**
Nevertheless, the task of sentiment analysis appears usefull for various different applications from advertisement to opinion detection for parts of the society.
After embedding the words into a word vectorspace, the obvious way to classify them is to use a classical neural network. In the following we use multiple well known strategies as a baseline. These will be compared to the application of convolutional neural networks. Another step further, we not only use the more advanced strategies, but we also combine multiple models to built committees. A closer look is taken, on whether the performance of the commitee outperformes a single better trained model. (cite Bishop Chapter 14).
The used dataset for out application is a set of about 2,5 million tweets, labeled to whether they are positive or negative. As a consequence from of our observatios it becomes obvious, that a single well trained network outperformes a majority vote over multiple networks.

I. Introduction

Sentiment analysis of texts is a field of wide interest. Especially for the advertisement industry, it is of great value to be capable of analysing the sentiment of tendentially small messages appearing within social networks. In this paper, we apply the technique of convolutional neural networks to the field of sentiment analysis. While convolutional networks are the defacto standard in Computer Vision tasks, their application in natural language processing has been analysed less intensively.
During our analysis we use convolutional neural networks in combination with generated word vectors generated in three different ways. After an evaluation of the precision per network we afterwards run a commitee approach following the majority vote. All our results are compared to classical approaches using BatchOfWords and word average wordvectors per sentence and classical neural networks.
It turns out that considerable improvement can be archieved over naive sentiment analysis approaches.

II. Models and Methods

The strategy for determining the sentiment of statements can be devided into two separate parts. In a first step it is necessary to translate the sentences into a model. This is a word vector space is most of our cases. (Reference for wordembeddings) This word-vector space later gives the opportunity to operate on the words as coordinates in a space, where the position in space is derived from the meaning of certain word. Our approach for this is described in the first part of this section. Another model used within one our baseline algorithms is the BagOfWord model. This simplifying representation only keeps the multiplicity of certain words, but regards neither wordorder nor grammar.
In a second step, the modeled sentences are fed into a form of neural network, which acts as a classifier for the two sentiment classes ”negative” and ”positiv”.

A. Using Wordembeddings

Getting the meaning for a sentences is a overlying task, stacked onto the anaysis of single words. This analysis for single words is the core of what is described as word embeddings. Words are embedded into a vector space what allowes it to look onto the words positions relative to each other. This model itself offers powerfull functionality. There are multiple ways of determining word vectors. Word vectors we use are based upon the following two approaches:

1) Word2Vec: (Reference) This is a... Description following
2) Glove: (Reference) This is a... Description following

To put it all together, in our analysis we compare three dfferent sets of word vectors, that base on the above mentioned generation schemes and have a different dimensionality each:

1) First of all we use own vectors generated by Word2Vec .In contrast to the approach in [1] we use own trained word vectors. We voted for this, since the analysed twitter messages, tendentially use a different vocabular than the one used within every-day language. Using python as a project language the gensim-package (Reference) contains a valuable implementation.
2) Pre processed word embeddings of dimensionality 300 can be obtained from google (Reference). These vectors were trained on everyday langage.
3) Pre processed word embeddings with a dimensionality of 200, which has been especially trained on twitter data. (Reference Standord)

The analysed data is preprocessing in a way proposed by the same team of Stanford (Reference to script). The core of this step is to replace certain widespread word patterns by special tokens. This procedure includes numbers, emoticons

or the appreance of special word styles like repetition of letters in the end of words. The result is a reduced wordset with less sepcial cases.

B. Putting the Word Vectors in Shape

Before we can process the data further, we have to bring it into a form, that can be processed by the neural netwok of choice. Both, the normal and the convolutional neural network handle differrent kinds of inputs, sothat a separate layout of the input data is necessary.
For the common neural network used as a baseline, we only need a single vector as input, sothat each of the vectors dimensionality can be used as a differnt input to the neural network. The obvious strategy to get a single word vector for whole sentences is to build the average out of the avaiable word vectors for the words in the sentence.
In contrast to this, the convolutional neural network is capable to handle the input data as a whole. , it is necessary to think about the layout, in which one wants to provide the data. We stack all the word vectors of one sentence in behind each other. Like this we get a matrix, which is basically the replacesment of our image.

C. Applying a Convolutional Neural Network

The fundamental idea leading to the application of a convolutional neural network, is that one can postulate a inherent locality of the matrix created in the step above.

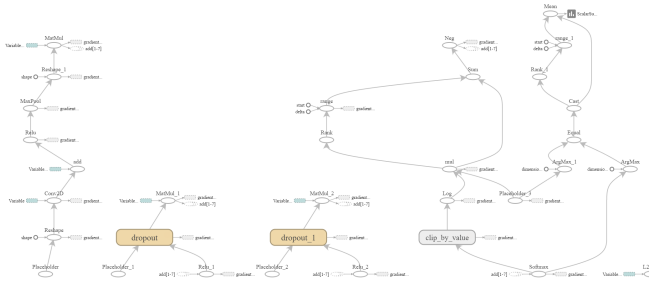

Figure 1. Architecture of the neural network.

There are multiple interesting hyperparameters for the cnn, which might influence the outcome. On the one hand the size of the filters and the their stride can be adapted. Stride means the stepsize of the sliding window, with which it is moved further. This can be used to configure whether multiple filters overlap or not. In the following both approaches have been tested.
For our implementation tensorflow (reference) has been used. As a nice sideeffect, this gave the possibility of easy visualization using tensorboard shown in the text paragraph.

III. Results

In following we give the results of our concrete implementation. First we argue about the time, that was needed

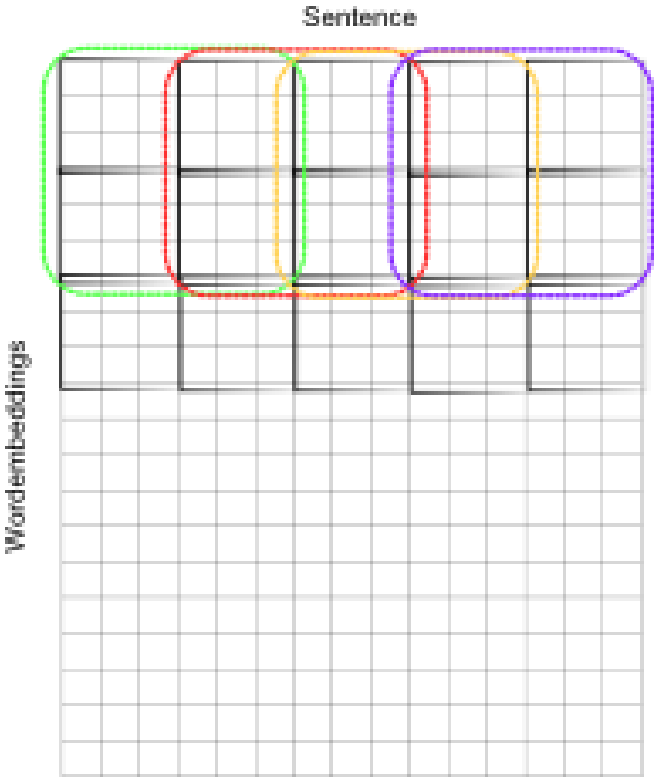

Figure 2. Application of the neural network on the sentences of word embeddings.

to train the different networks to the same level. Afterall 50k learning steps have been performed. The difference in dimenisonality of the word embeddings led to significant differences in the training performance.
In a second step we compare the results of the various approaches. The Baseline algorithms, the results of the CNNs and the result of the commiteee are described each in a separate paragraph.

A. Training the Neural Networks

Using word vectors of dsifferent dimensionality and different background, differences in performance can be expected from the beginning. Fig In the following we give the results of our approach. We compare the results of our convolutional neural network to the plain results, achieved by only using the wordembeddings without further processing.

For the plain approaches using the wordembeddings, there is of course no training necessary. Although the training has been done on a cluster only an extended amount of memory has been used. Due to issues about the parallelization, the scaling beyond 3 processor cores did not lead to improved performance (Parallel perfromance determined by using the benchmark provided (reference auf den Tensorflow thread)). Therefore the training took a long time.

Word Vectors	Dimensionality	Time Consumption
Common neural network averaged	100	???
Self computed enbeddings	100	6:00h
Standford twitter embeddings general	200	0:50h
Google general enbeddings	300	16h

Table I
TIME CONSUMPTION FOR TRAINING THE NEURAL NETWORK.

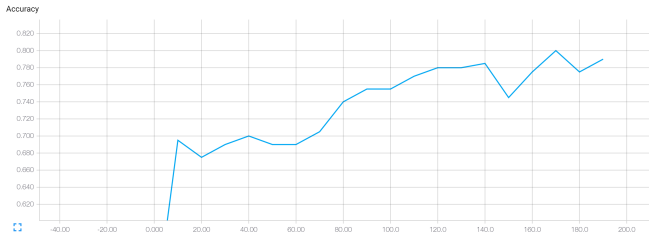

Figure 3. Increasing accuracy during learning.

Another difference between the 3 different sets of word vectors, was the amount of words which have been skipped during the analysis. Beeing trained on the given dataset, our own word embedding did not have exceptions of unknown words. In contrast to this the google dataset faced an overall amount of ??? (Insert after training ran through) words, that could not be found in the word index. Using the standford pretrained dataset this effect was less drastic, since a preprocessing step was included and the words vectors have been originally trained on twitter data either.

B. Baseline

The baseline approach using a neural network on a bag of words model and the averaged word vector per tweet already gave usable results. While the Bag of words model yielded a accuracy of. 80.40%, the averaged word embeddings lead to a final precision of 80,32%.

C. Results of the Convolutional Neural Network

One can expect better results from the CNN already from the fact, that it uses more informaton from the beginning. This expectation was validated during the test runs. Using the three different word embeddings, described above, the precision lay always above 85%. A comparison of the gradual increase in accuracy between different models during the training can be seen in reffig:accuracy. (Maybe say one or two sentences about the outcome).

D. Results of the Commitee

In a last step, the predictions of the three different concolutional neural networks have been combined to form a mazority vote. Using this prediction instead of the prediction of a single neural network improved the results again slightly. (Give the real results)

IV. Discussion

Since it can be expected the learning has not been pushed to the limit, an extended learning procedure should lead to improved results. At this, it would sbe of course beneficial to train the neural network on a graphics card, to profit from a considerable increase in performance. The good results of our approach give space for assumptions, that more sophisticated architectures for deep learning nets might lead to even better results.

Observing convolutional neural networks only in this approach, it could be interesting to look onto whole different deep learning networks like recurrent neural networks (reference) or recursive neural tensor nets (reference).

V. Summary

In the end, the application of convolutional neural networks yielded good result. The baselines, using plain neural network classification on top of an average word vector per sentence, have been clearly outperformed by already a single convolutional neural network. The combination of multiple trained networks, tried in a second step, was capable to even improve the results.

References

[1] C. N. dos Santos and M. Gatti, “Deep convolutional neural networks for sentiment analysis of short texts.” in *COLING*, 2014, pp. 69–78.