Document Classification and Topic Detection with CNNs

Semih Akbayrak
Boğaziçi University
semih.akbayrak@boun.edu.tr

Abstract

This study presents a Convolutional Neural Network model for document classification task. This model works as topic detector, which detects all the topics in a document. A simple model with limited number of documents works good in classification task but can't exceed the performance of Naive Bayes. But this model can be used for many different tasks which the other classifiers not good at in, like topic detection, short text classification, text summarization, etc.

1. Introduction

Most of the time, words are assumed that they are independent from each other for document classification or modeling tasks. This is the bag of words representation, and with this model we accept that word frequencies supply enough information about the document. On the other hand, we all know that adjacent words create a semantic meaning, and if we can use these meanings somehow, it would help us to represent documents better.

Convolutional Neural Networks are being used in computer vision applications mostly, because of their power to detect patterns independent of their location on the images, or video scenes.

In recent years, CNNs have been used for NLP tasks as well. Kalchbrenner et al. used CNN to model sentences. Then Kim utilized from CNNs to classify sentences. Denis et al. used CNNs for classification and summarization of documents.

All the studies above however were used for sentiment like tasks. Even Denis et al. used IMDB's movie review dataset which contains positive and negative critics about movies, in their study for document classification.

But document classification is a more general task, and it would also need much more time for training part which is already long enough according to other papers.

Therefore a simpler model with ability to distinguish K classes from each other, should be proposed.

The model proposed in this study is simple enough and need small amount of documents for training. It uses different filter banks for classification task with more than 2 classes.

2. Model

Important semantic meaning created by adjacent words can be anywhere in the document. Therefore a good classifier which consider semantic meaning should be able to find these patterns in the document. The other important issue is sentence and document lengths vary, so a good model should also handle this issue.

The reason to use CNNs in this model is to handle these issues while we take semantic meaning into account. Weight sharing feature of CNNs solve the first issue, and max-pooling is the solution of the second one.

In the first layer, we need word embeddings. This can be one-hot word vectors but in order to utilize full semantic meaning, vectors obtained by Mikolov's Skip-Gram word2vec method

were used. word2vec is a revolutionary unsupervised method to create word vectors. It places semantically similar words close to each other. After obtaining word vectors, we can represent sentences as matrices by combining these vectors.

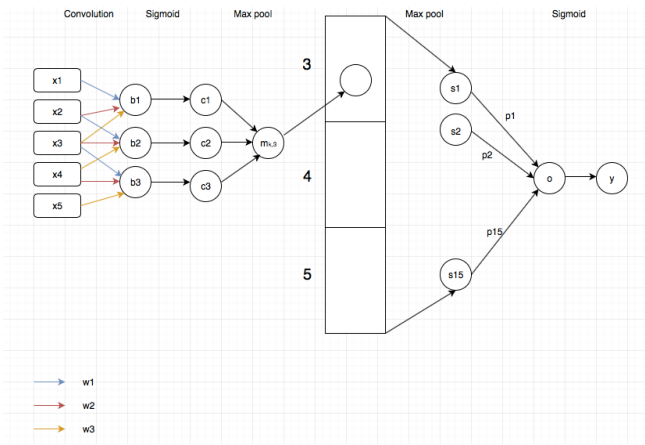

Figure1: An example model with 3-window sized filter

The model contains K filter banks for K classes which means every filter bank specializes to different classes and they learn the patterns belong to classes. Every filter bank formed by 3 or 4 filters with different window sizes (3,4,5,6).
In order to look at the model more detailly, let's assume we learned weights. Convolutional layer, search for a pattern in the sentence and if it finds, dot product gives a large value. Then we use non-linear function sigmoid and it guarantees us to give a value between 0 and 1 always. Max-pooling layer take the max value among them. It basically captures the pattern independent of its position in the sentence. So, if there is an important semantic meaning in the sentence, this model captures it. Then, to represent the document, we take the max 5 values among max sentence values and sort them. We repeat this process for all the filters in the filter bank and we create 15 dimensional vector which represents the document under given topic(this is for filter banks with 3 filters, if we use 4 filters, this time dimension will be 20).
If we leave the process here, this will be the model of document under given topic. But we go one step further and use a single perceptron to find the posterior probability of document belongs to given category. This last step, by the way, essential for back propagation and learninig the weights. But after training the model and finding weights, the layer before the last perceptron can be used for modeling and clustering documents.

$$b_i = \overrightarrow{x_i}.\overrightarrow{w_1}^T + \overrightarrow{x_{i+1}}.\overrightarrow{w_2}^T + \overrightarrow{x_{i+2}}.\overrightarrow{w_3}^T + w_0$$

$$c_i = f(b_i) \quad f : sigmoid \quad f(t) = 1/(1+exp(-t))$$

$$m_{k,3} = max(c_1, c_2, \ldots, c_n) \quad k : k^{th} sentence\ of\ doc$$

$$o = \left(\sum_{h=1}^{15} p_h . s_h\right) + p_0 \quad y = f(o) \quad f : sigmoid$$

The formulas above are the mathematical equivalence of the model's description. Because x values are vectoral representations of the words, w weights are also vectors. Vector values represented with arrows in the formulas and they were taken as row vectors.
Back propagation, of course, is the fundamental algorithm we use to learn the model parameters and update equations with stochastic gradient descent are derived below.

$$E^t = -(r^t.log(y^t) + (1-r^t).log(1-y^t))$$

$$\triangle p_h = -\eta\frac{\partial E}{\partial p_h} = -\eta\frac{\partial E}{\partial y}\frac{\partial y}{\partial o}\frac{\partial o}{\partial p_h} = \eta(r^t - y^t)s_h^t$$

$$\triangle w_l = -\eta\frac{\partial E}{\partial w_l} = -\eta\frac{\partial E}{\partial y}\frac{\partial y}{\partial o}\sum_h\frac{\partial o}{\partial s_h}\sum_k\frac{\partial s_h}{\partial m_{k,3}}\sum_i\frac{\partial m_{k,3}}{\partial c_i}\frac{\partial c_i}{\partial b_i}\frac{\partial b_i}{\partial w_l}$$

$$-\eta\frac{\partial E}{\partial y}\frac{\partial y}{\partial o}=\eta(r^t-y^t) \qquad \frac{\partial o}{\partial s_h}=p_h$$

$$\frac{\partial s_h}{\partial m_{k,3}}=\begin{cases}1\ if\ s_h=m_{k,3}\\0 \qquad\qquad o.w.\end{cases}$$

$$\frac{\partial m_{k,3}}{\partial c_i}=\begin{cases}1\ if\ m_{k,3}=c_i\\0 \qquad\qquad o.w.\end{cases}$$

$$\frac{\partial c_i}{\partial b_i}=c_i.(1-c_i) \qquad \frac{\partial b_i}{\partial w_l}=x_{i+l-1}$$

3. Dataset and Results

Yıldız Technical University's 42000 Turkish news in 13 categories dataset is used in this study. Two versions of it used actually, the first one is full dataset with 13 categories(dunya, ekonomi, genel, guncel, kultur-sanat, magazin, planet, saglik, siyaset, spor, teknoloji, turkiye, yasam) and second one is reduced dataset with 7 categories(dunya, ekonomi, kultur-sanat, saglik, siyaset, spor, teknoloji). For comparison, Naive Bayes classifier used and here are the results.

Model	Classification Accuracy	Topic Detection Accuracy
CNN 3-filter	79.91	83.13
CNN 4-filter	67.06	86.34
Naive Bayes	93.27	–

Table1: Percentage of accuracies for reduced dataset

Model	Classification Accuracy	Topic Detection Accuracy
CNN 3-filter	47.62	77.19
CNN 4-filter	34.10	82.26
Naive Bayes	57.14	–

Table2: Percentage of accuracies for full dataset

As a result, the CNN model doesn't give better results than Naive Bayes in classification task, nevertheless results are not bad and give us some clues for feature studies. It should be remembered that 42000 news dataset is used to get word vectors and this is a small dataset, with a larger corpus the results may be improved and it is not too hard to find a larger corpus because word2vec is an unsupervised method and we don't need classified documents for this task.

Another important issues is, this model works as topic detector. So, it is able to make multi-class classification opposite to Naive Bayes classifier. Furthermore, Naive Bayes determines a document's class if the probability of belonging to that class is higher than the others, but this actually doesn't mean that document contains that class really. So CNN model is able to handle this problem as well.

Lastly, by looking at the results for long documents, we can estimate the CNN-model's performance on short document classification. We see that, word frequencies is enough most of the time to classify long documents but this is not the case for short documents like tweets, or documents with two-three sentences. But CNN-model doesn't consider word frequencies and it directly consider semantic meaning. So one version of this model can be a good option for the cases which Naive Bayes is useless for.

4. Conclusion

A simple CNN-model used in this study for long document classification task. With small modifications this model can be used for different tasks like short document classification where Naive Bayes gives poor results, multi-class document classification, document modeling, sentiment analysis, document summarization by marking the word sequences which causes large results in convolutional layer part, etc.

References

N.Kalchbrenner, E.Grefenstette, P.Blunsom. A Convolutional Neural Network for Modeling Sentences. 2014.

Y.Kim. Convolutional Neural Networks for Sentence Classification. 2014.

M.Denil, A.Demiraj, N.Kalchbrenner, P.Blunsom, N.Freitas. Modeling, Visualising, and Summarising Documents with a Single Convolutional Neural Networks. 2014.

J.Gao, P.Pantel, M. Gamon, X.He, L.Deng. Modeling Interestingness with Deep Neural Networks. 2014.

T.Mikolov, K.Chen, G.Corrado, J.Dean. Efficient Estimation of Word Representations in Vector Space. 2013.

J.Xu, P.Wang, G.Tian, B.Xu, J.Zhao, F.Wang, H. Hao. Short Text Clustering via Convolutional Neural Networks. 2015.

R.Johnson, T.Zhang. Effective Use of Word Order for Text Classification with Convolutional Neural Networks. 2015.

G.Song, Y.Ye, X.Du, X.Huang, S.Bie. Short Text Classification: A Survey. 2014.