Ideology Detection with using Deep Neural Network

Lee Hojoon, Jeong Minbyul*

University of Korea
joonleesky90@gmail.com,wjdalsquf@gmail.com

December 17, 2017

Abstract

In the political field, there are many researches trying to detect political ideology inside the text. Traditionally used automated techonology is BoW (Bag of Words) which only counts for frequency ignoring for the sequence. However, recent studies have shown that deep learning is showing great performance in sentiment analysis. Therefore, we apply RNN (Recrusvie Neural Network), LSTM (Long-Term Short-Term Memory), and CNN (Convolutional Neural Network) to detect political ideology in sentence level.

I. Introduction

Living in the era of big data, We are surrounded by the flood of news and media. Even though each press emphasizes for their own fairness and neutrality, lots of experts and journalists claim that there exists political bias in the news (MATTHEW GENTZKOW, 2010). Moreover, as social network service is explosively developing, lot's of unverified news are imprudently consumed. Therefore, in order to critically view the text, knowing political bias in the news is important in prior. One of the ways to detect political bias is by expert with rich knowledge. However, it's so much time consuming and cannot shake off the possibility of expert's subjective perspective.

Past studies have been detecting political bias with automated text analysis. In the paper (MATTHEW GENTZKOW, 2010), they detected news's political bias with BoW (Bag of Words) technique and proposed most of the bias can be detected with the occurrence of certain ideological words. However, in order to effectively apply this scheme, there should be a big bunch of texts. Moreover, they can only detect the overall perspective of the press, not by the sentence or text.

In the area of sentiment analysis, deep learning has shown tremendous success recently. Iyeer (Mohit Iyeer 2013) tried to detect political ideology within a sentence using a Tree structure RNN(Recursive Neural Network) and successfully classified liberal and conservative with the accuracy up to 70 percent. For the longer sentences, however, RNN is suffering from the gradient vanishing problem. LSTM(Long term Short term Memory) is one of the alternatives to solve this issue which is a variant of RNN. Kim (Yoon Kim, 2014) suggested using CNN to sentence classification can show great performance in sentiment analysis.

To overcome these problems and enhance the performances of (Mohit Iyeer, 2013)'s work, we will try to apply LSTM and CNN models to classify liberal and conservative in IBC (Ideological Book Corpus) dataset. Moreover, comparing with the classical RNN model.

II. Model

i. Dataset

We use the modified version of IBC, made by Mohit Iyeer (2013). The Ideological Books Cor-

*A thank you or further information

pus(IBC) was developed by Gross et al. (2013) This is a collection of books and magazine articles from 2008 to 2012 by the authors well-known for political learnings. By the political science experts, each document in the IBC is labeled with left, center, and right, which indicates left wing, center and right wing.

Since, IBC contains more than million sentences with neutral perspective, Mohit Iyeer(2013) filtered the data and left out 11,555 sentences which are politically biased. They surveyed their data from Internet whether they think the sentences are either liberal, neutral or conservative. They cut off the sentences which were highly rated as neutral.They are finally left out 3,412 sentences (1,801 for the liberal and 2,612 for the conservatives).

ii. Preprocessing

Input data consists of 3,412 sentences (1,801 for the liberal and 2,612 for the conservatives). Each sentence has different length of words, vary from 22 to 86 words long.

ii.1 Initialization

The most simple and straight forward way to initialize the input data is representing word in one-hot-vector. This is very simple method in word embedding. However, each word can loses their contextual meanings falling to the curse of dimensionality.

In order to solve the curse of dimensionality, we initialize our input data with 300-dimensional vector provided by the Google Word2Vec. Google word2vec model provides vectors generated by a continuous skip-gram model trained on around 100 billion words from the Google News corpus (Mikolov et al., 2013).

Different words with similar meanings share similar vector representation in word2vec model. Which shows higher cosine-similarity in the vector space.

ii.2 Padding

Each sentence in input data has various lengths of words. We initialize each word with 300-dimensional vectors, longest sentence with 86 words will be 86 * 300 matrix, shortest one will be 22 * 300 matrix. Since we have to match the size of the input per sentence, we make a random 300-dimensional vector with a same scale of Google word2vec model padded to all the sentences until they meet up the length of 86. The preprocessed dataset of liberal is shown in Figure 1, same for the conservative dataset.

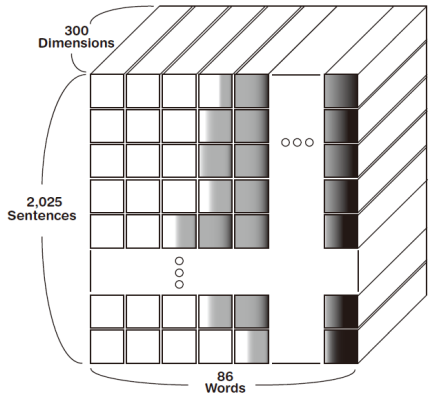

Figure 1: *liberal dataset*

In above picture, *white box* in each sentence indicates that there exists a word. On the other hand, *grey box* indicates that there exists a padded vector.

iii. Models

We have done experiments with 3 models: RNN, LSTM and CNN-multichannel.

iii.1 RNN

Our RNN is standard model of supervised many-to-one RNN model.

Hidden state in each RNN cell consists of 150 dimension vectors. Hidden state is concatenated with a word vector extracted from a sentence sequentially. For a concatenated vector, perform linear operation with weights

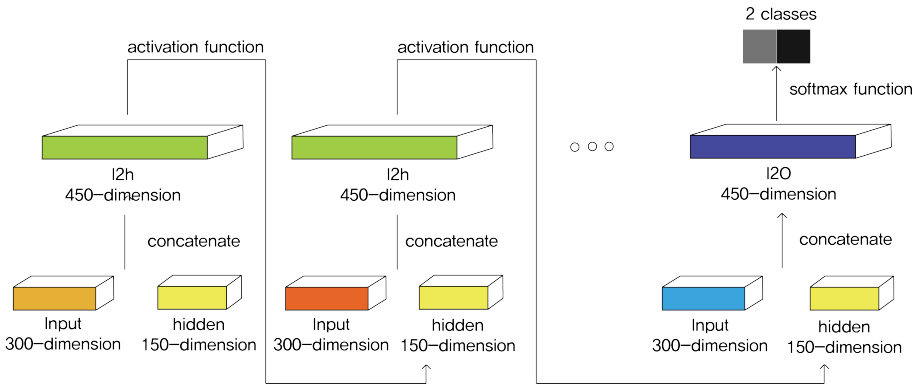

Figure 2: *RNN model*

initialized with *Xavier Initialization*. To make next hidden layer, use *tanh* function as a activation layer. If a word is the last word in sentence, perform linear operation to make 2-dimension vector. Soft-max layer is added to 2-dimension vector, producing a prediction y. Each sentence is labeled with either liberal or conservative. [1, 0] vector for the liberal, [0, 1] vector for the conservatives. Loss is computed with cross-entropy function to prediction y and the label. Perform back propagation and update weight in each cell. Model is described in above Figure 2.

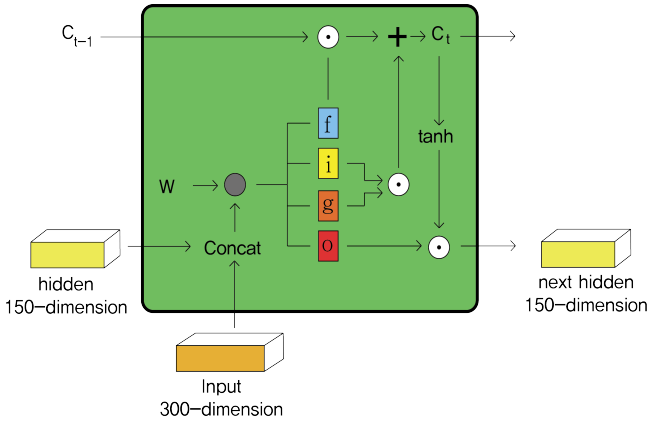

Figure 3: *LSTM model*

iii.2 LSTM

Similar as the RNN, we used supervised many-to-one LSTM model.

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Each cell in LSTM consists of input word, hidden state, cell state and 4 gates, each computed as above equation. Weight in each cell is initialized with Xavier initialization. Each gate and state is computed as below Figure 3.

iii.3 CNN-multichannel

Yoon kim (Yoon Kim, 2014) proposed to perform sentence classification task with CNN. Even though CNN is generally used to image-related task, it can be also applied to texts. With each sentence represented 86 * 300 vector, apply convolution layer with 6 different filters.

We have two 2 * 300 filters, two 3 * 300 filters, and two 4 * 300 filters. Filter with size 2 * 300 works similarly as detecting bi-gram, so as 3 * 300 filter works as detecting for tri-gram.

Perform convolution operation for each filter. Perform 1-dimension max-pooling for each filter outcome and concatenate all to make 6-dimension vector. Add fully connected layer to make 2- dimension vector. Soft-max layer is added to final 2-dimension vector, producing

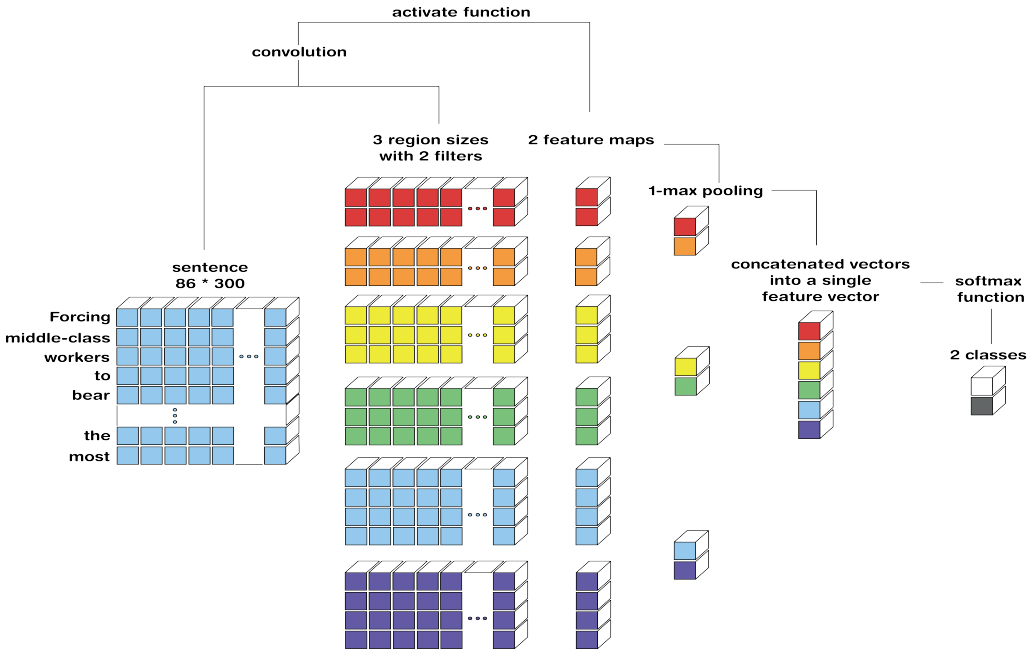

Figure 4: *CNN model*

a prediction y. Loss is computed with cross-entropy function to prediction y and the label. Perform back propagation and update weight of each filter.

Procedure is shown above Figure 4.

III. Experiments

i. RNN , LSTM

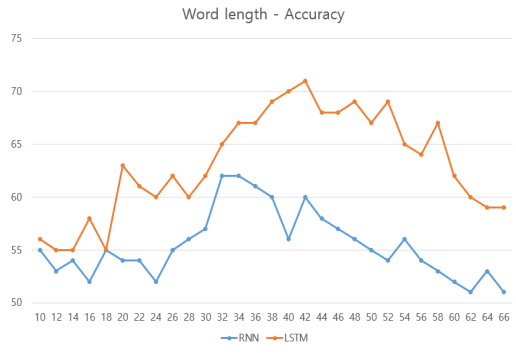

Figure 5: *accuracy per length of the sentence*

Above Figure 5 is the accuracy of RNN and LSTM model in regard of the length of the sentence. Even though full length of the sentence consists of 86 words, length of 32 words shows the best performance for the RNN model and 42 for the LSTM model. Since RNN frequently confronts with gradient vanishing problem, it suffers from seeking through the whole sentences. LSTM works better than RNN but still suffers for the same problem as well.

For Regularization, Dropout is one of the most common and effective regularization measure in deep learning. In this case, however, because modified IBC dataset contains only 4000 sentences, it tends to be under-fit when using dropout, resulting worse performances. Therefore, we did not use dropout and finished our training just after few epochs, before it over-fits.

ii. CNN

With batch normalization, CNN does not suffer from vanishing gradients. Also, dropout works very well for regularizations.

IV. Results

Our baseline of the project was the performance of Tree-RNN(Mohit, 2013) with *69* percent accuracy.

Since we randomly assigned test set at the beginning, there might be different results under different test sets.

Accuracy for each model is shown below Table 1.

Table 1: *Accuracy Table*

Model	Accuracy
Random	50
Tree-RNN (Mohit, 2013)	69
RNN	62
CNN	65
LSTM	71

LSTM model seems to be showing the best performance, achieving the task to enhance our baseline model Tree-RNN (Mohit, 2013). Though CNN-multichannel seems to show great performances in many other sentiment analysis tasks (Yoon Kim, 2014), it's not a tempting option to try in this task.

V. Conclusion

In this paper we apply RNN, LSTM, multichannel CNN model to political ideology detection. Previous ideology detection problems heavily relies in bag-of-words models, not a good model in detecting short texts. Though, RNN model is a rising star in ideology detection problems it still suffers from long-term dependency problems. We show using LSTM model solves this problem partially, detecting bias more accurately than previous work.

VI. Further Work

We fail to solve for the long-term dependency problem for LSTM and RNN models. While Batch normalization is the most successful method to solve this problem in CNN based models, it's not a successful method in RNN based models. Recently, Layer normalization proposed by (Jimmy Lei Ba,2016) shows up to be the good alternative to Batch normalization in LSTM model. This can help through solving the long term dependency, using the full contextual meaning in the sentence.

Sentiment analysis with LSTM is one of the active research area. Recently, combining LSTM with attention models(Y Ma, 2018) shows great performance in sentiment analysis. Applying these models will be the great try to enhance the performance.

References

[Mohit Iyyer, 2013] Mohit Iyyer, Peter Enns, Jordan Boyd-Graber and, Philip Resnik. (2013). Political Ideology Detection Using Recursive Neural Networks. *2014 Annual Meeting of the Association of Computational Linguistics*

[Getnzkow, 2010] Matthew Gentzkow and Jesse M Shapiro (2010). What drives media slant? evidence from us daily newspapers. *Econometrica*, 78(1):35âĂŞ71.

[Mikolov, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). Efficient estimation of word representations in vector space. *arXiv preprint*, arXiv:1301.378.

[Yoon Kim, 2014] Yoon Kim (2014). Convolutional Neural Networks for Sentence Classification. *https://arxiv.org/abs/1408.5882*

[Jimmy Lei Ba,2016] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton (2016). Layer Normalization. *https://arxiv.org/abs/1607.06450*

[Y Ma, 2018] Y Ma (2018). Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. *2018 AAAI*