*International Journal of Advanced Culture Technology Vol.5 No.4 57-62 (2017)*
*https://doi.org /10.17703/IJACT.2017.5.4.57*

IJACT 17-4-10

# Long Short Term Memory based Political Polarity Analysis in Cyber Public Sphere

Hyeon Kang[+] and Dae-Ki Kang[*†]

[†]*Department of Computer Engineering, Dongseo University, Korea*
*{kang2000h, dkkang}@gmail.com*

***Abstract***

*In this paper, we applied long short term memory(LSTM) for classifying political polarity in cyber public sphere. The data collected from the cyber public sphere is transformed into word corpus data through word embedding. Based on this word corpus data, we train recurrent neural network (RNN) which is connected by LSTM's. Softmax function is applied at the output of the RNN. We conducted our proposed system to obtain experimental results, and we will enhance our proposed system by refining LSTM in our system.*

***Keywords:*** *Long short term memory (LSTM), Word embedding, Recurrent neural network (RNN), Softmax function*

## 1. Introduction

There have been various research on the sentiment analysis and opinion mining in cyber public sphere. Most of these methodologies are concerned in calculating information theoretic measures such as pairwise mutual information (PMI) and application from traditional machine learning methodologies.

In this paper, we performed experimental research on analyzing political polarity from opinions in cyber public sphere, using word embedding and long short term memory (LSTM). For the data analysis, we consider two online public forums, which show completely different political tendency, Today's Humor and Ilgan Best. For this purpose, we collected and analyzed the public opinions of Today's Humor site and Ilgan Best (Ilbe) site, which show opposing political tendencies.

Figure 1 shows the conceptual diagram of the proposed system.

In the experiments, we have acquired a corpus sequence W from a certain cyber public sphere. We can define $W \in R^{Tm}$, where T is the length of sequence and m is the embedding dimension of variance representation. We input the sequence W into an LSTM neural network following temporal order. We use results of the last time T as a feature vector of the document and input the resulting feature vector to a feedforward network to infer the probabilistic distribution on political polarity class labels of W.

## 2. Political Polarity as a Subset of Sentiment Analysis

Sentiment analysis is often referred as opinion mining. In machine learning, sentiment analysis usually means a systematic research on extraction of information units that denote a particular subjective emotion from natural language data. This natural language data is usually review information, but it can be discussion articles from cyber public sphere. In marketing application, customer survey data can be used too.

These sentiments can be opinion, emotion, and attitudes. Note that the sentiments are not a fact, but a subjective impressions. The easiest form of sentiment in sentiment analysis can be regarded as a binary class of classification problem. For example, the sentiment can be classified as for/against, like/dislike, and good/bad. Political polarity analysis is one application of sentiment analysis.

Manuscript received: Demeber 18, 2017 / revised: xxxxxx / Accepted:
Corresponding Author: dkkang@gmail.com
Tel: +82-51-320-1724, Fax: +82-51-327-8955
Department of Computer Engineering, Dongseo University, Korea

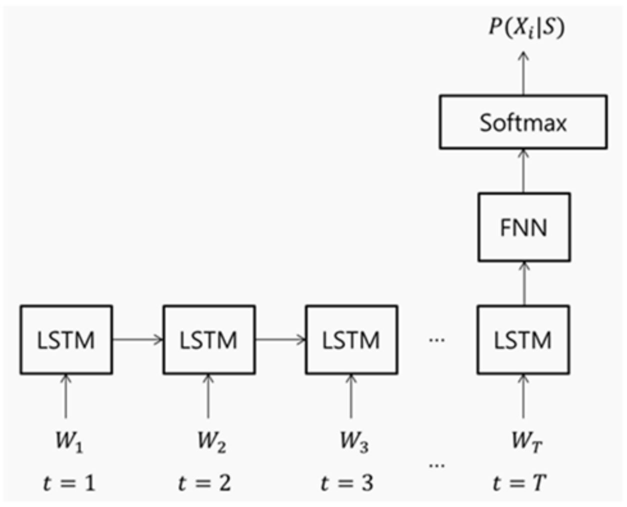


**Figure 1. Block diagram of the overall system**

Some related tasks with sentiment analysis is information extraction, question answering, summarization, “flame” detection, troll detection, identification of child-suitability, bias identification, identification of inappropriate contents in advertisement, etc.

In the business domain, sentiment analysis is very important and widely used. One example is that it is usually very difficult to survey customers who did not buy the company’s laptop. To obtain the opinion from those potential customers, we can search the Web for opinions and reviews for the laptop product and its competitors. Those Web sites for the opinions and reviews include blogs, Epinions, Amazon, Tweets, etc. The sentiment analysis system can create a condensed version of those opinions and reviews.

In the politics domain, there are various applications. Those applications include analysis of trends, identification of ideological bias, targeting messages, estimating reactions, etc., Note that those applications are generally for evaluating the voters’ opinions and summarizing the views of a certain policy.

## 3. Methods

### 3.1 Word embedding

In order for machine to process natural language, the language data has to be represented in numerical data, and data structure decision process has to be performed in advance to extract proper features as natural language from a certain word corpus.

Word embedding is a methodology to represent numerically a certain text data.

Traditional approach to deal with words is one hot representation that constructs vector sphere of V dimension for all words in the corpus where V is the number of words, and denotes the dimension of the target word as one. One corpus is a V dimensional vector, and the value of each index is the frequency of words, i.e. bag of words (BOW) method. This kinds of representation is intuitive, but has drawbacks that the dimension is huge according to V and the resulting vector is sparse. Also, the distance of each word is one, which means the representation does not accommodate similarity concept.

Neural network language model (NNLM) is a language model that, using neural network, predicts joint probability of nth word given a sequence of n-1 words, from the distributional hypothesis that words occurred in a similar context will have the similar meaning. NNLM tries to solve the sparcity problem of semantic space by using internal embedding matrix to transform a V dimensional locally represented word into a m-dimensional feature vector as an input of neural network. From the inner product of V dimensional locally represented word and m-dimensional embedding matrix, we can obtain m dimensional feature vector from a certain word[1]. This distributionally represented word can be mapped into m dimensional real number space to compare the similarity of the particular word with its neighboring words.

Nowadays, there have been various algorithms proposed because of recent popularity of estimating distributed representations of words based on word embedding from neural networks.

Aforementioned method of estimating distributed representations of words based on word embedding from neural networks basically requires much computation for the learning steps of matrices that constitute neural networks to emulate non-linear functions in the hidden layers.

Word2vec algorithm removes non-linearity and significant number of matrices inside neural networks and augment a few training skills for estimating embedding matrices to enhance computational costs from the legacy modus operandi [2].

In our experiment, we use word2vec algorithm from continuous bag of words (CBOW). Table 1 shows two words as an example and their similar words.

**Table 1. Similarity of distributionally represented words**

| Searched words | **Government** | **Democracy** |
|---|---|---|
| **Similar words and their similarities** | Regime / 0.5784 | Justice / 0.5848 |
| | President / 0.5162 | Country / 0.5208 |
| | Blue House / 0.4605 | Our country / 0.5048 |

For example, the table 2 lists three similar words for the word “government”. The most similar word of “government” is “regime” with the similarity value 0.5784. The 2nd similar word for the word “government” is “president” with the similarity value 0.5162. And the 3rd similar word for the word “government” is “blue house” with the similarity value 0.4605. In the table 2, there are three similar words for the word “democracy”. The most similar word of “democracy” is “justice” with the similarity value 0.5848. The 2nd similar word for the word “democracy” is “country” with the similarity value 0.5208. And the 3rd similar word for the word “democracy” is “our country” with the similarity value 0.5048.

### 3.2 Long short term memory (LSTM)

Ordered sequence of distributionally represented words are mapped into multi-dimensional vector space and they have semantic meaning as a part of natural language.

In complicated problems such as speech recognition, sentence classification, etc., there is a tendency that the input sequence is getting longer, and it will lengthen the propagation on a temporal axis. This will cause gradient diminishing or gradient exploding which hinders parameter training. In recurrent neural networks, it is well known that it is difficult to train long-term information with gradient descent technique only [3].

Figure 2 shows the memory block of LSTM. LSTM preserves cell state in the hidden layer of recurrent neural network, and manages information by the states of forget gate, input gate, and output gate. Owing to this architecture, LSTM can choose and save pieces of information with long term dependency into memory, and thereby is known to be able to store about 1,000 pieces of temporal information [4,5]. The cell state $C_t$ at the time t is defined from the cell state $\overline{C_t}$ of input $x_t$ at time t, input gate $g_t^I$ and forget date $g_t^F$ from the cell state $C_{t-1}$ at the time t-1. The memory block state $h_t$ at the time t is defined from the output gate $g_t^O$ to decide how much information will be outputted from $C_t$. Each gate is defined from $x_t$ and $h_t$.

\

$$
\begin{aligned}
\mathrm{C_t} &= \mathrm{g_t^F} \times \mathrm{C_t} + \mathrm{g_t^I} \times \overline{\mathrm{C_t}} \\
\overline{\mathrm{C_t}} &= \mathrm{f}(W^I \times \mathrm{x_t} + W \times \mathrm{h_{t-1}}) \\
\mathrm{g_t^F} &= \mathrm{f}(W^F \times \mathrm{x_t} + W \times \mathrm{h_{t-1}}) \\
\mathrm{g_t^I} &= \mathrm{f}(W^I \times \mathrm{x_t} + W \times \mathrm{h_{t-1}}) \\
\mathrm{g_t^O} &= \mathrm{f}(W^O \times \mathrm{x_t} + W \times \mathrm{h_{t-1}}) \\
\mathrm{h_t} &= \mathrm{g_t^O} \times \mathrm{f}(\mathrm{C_t})
\end{aligned}
$$

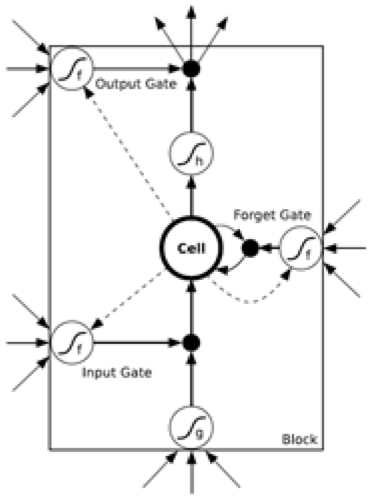
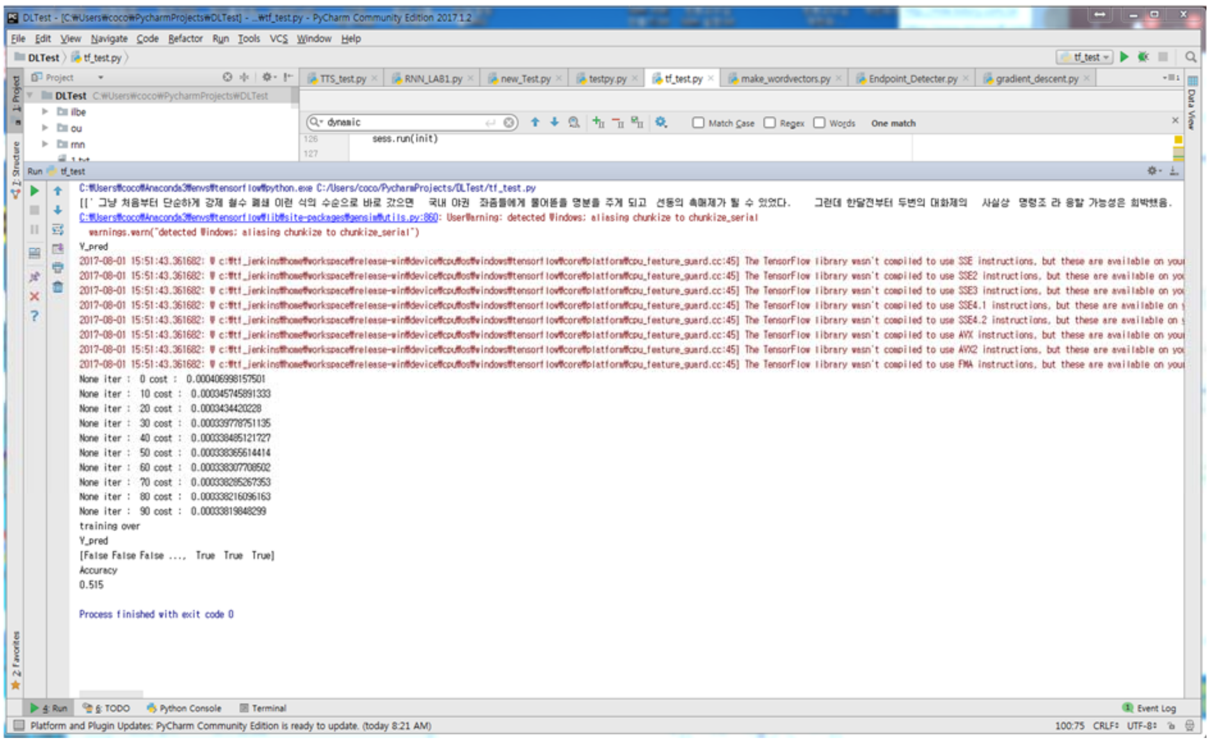


**Figure 2. The Memory block of LSTM**


DLTest - [C:₩Users₩coco₩PycharmProjects₩DLTest] - ...₩tf_test.py - PyCharm Community Edition 2017.1.2
File Edit View Navigate Code Refactor Run Tools VCS Window Help
DLTest tf_test.py
Project
DLTest C:₩Users₩coco₩PycharmProjects₩DLTest
ilbe
ou
rnn
TTS_test.py
RNN_LAB1.py
new_Test.py
testpy.py
tf_test.py
make_wordvectors.py
Endpoint_Detecter.py
gradient_descent.py
dynamic
Match Case
Regex
Words
One match
Run tf_test
4: Run
6: TODO
Python Console
Terminal
Event Log
Platform and Plugin Updates: PyCharm Community Edition is ready to update. (today 8:21 AM)
100:75 CRLF UTF-8

```
sess.run(init)
```

```
C:₩Users₩coco₩Anaconda3₩envs₩tensorflow₩python.exe C:/Users/coco/PycharmProjects/DLTest/tf_test.py
[[' 그냥 처음부터 단순하게 강제 철수 폐쇄 이런 식의 수순으로 바로 갔으면   국내 야권  좌좀들에게 물어뜯을 명분을 주게 되고  선동의 촉매제가 될 수 있었다.    그런데 한달전부터 두번의 대화제의   사실상  명령조 라 응할 가능성은 희박했음.
C:₩Users₩coco₩Anaconda3₩envs₩tensorflow₩lib₩site-packages₩gensim₩utils.py:860: UserWarning: detected Windows: aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows: aliasing chunkize to chunkize_serial")
Y_pred
2017-08-01 15:51:43.361682: W c:₩tf_jenkins₩home₩workspace₩release-win₩device₩cpu₩os₩windows₩tensorflow₩core₩platform₩cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE instructions, but these are available on your
2017-08-01 15:51:43.361682: W c:₩tf_jenkins₩home₩workspace₩release-win₩device₩cpu₩os₩windows₩tensorflow₩core₩platform₩cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE2 instructions, but these are available on yo
2017-08-01 15:51:43.361682: W c:₩tf_jenkins₩home₩workspace₩release-win₩device₩cpu₩os₩windows₩tensorflow₩core₩platform₩cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE3 instructions, but these are available on yo
2017-08-01 15:51:43.361682: W c:₩tf_jenkins₩home₩workspace₩release-win₩device₩cpu₩os₩windows₩tensorflow₩core₩platform₩cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these are available on
2017-08-01 15:51:43.361682: W c:₩tf_jenkins₩home₩workspace₩release-win₩device₩cpu₩os₩windows₩tensorflow₩core₩platform₩cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but these are available on
2017-08-01 15:51:43.361682: W c:₩tf_jenkins₩home₩workspace₩release-win₩device₩cpu₩os₩windows₩tensorflow₩core₩platform₩cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but these are available on your
2017-08-01 15:51:43.361682: W c:₩tf_jenkins₩home₩workspace₩release-win₩device₩cpu₩os₩windows₩tensorflow₩core₩platform₩cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 instructions, but these are available on yo
2017-08-01 15:51:43.361682: W c:₩tf_jenkins₩home₩workspace₩release-win₩device₩cpu₩os₩windows₩tensorflow₩core₩platform₩cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use FMA instructions, but these are available on your
None iter :  0 cost :  0.000406998157501
None iter :  10 cost :  0.000345745891333
None iter :  20 cost :  0.0003434420228
None iter :  30 cost :  0.000339778751135
None iter :  40 cost :  0.000338485121727
None iter :  50 cost :  0.000338365614414
None iter :  60 cost :  0.000338307708502
None iter :  70 cost :  0.000338285267353
None iter :  80 cost :  0.000338216096163
None iter :  90 cost :  0.00033819848299
training over
Y_pred
[False False False ....  True  True  True]
Accuracy
0.515

Process finished with exit code 0
```

**Figure 3. Screenshot of Our Proposed System**

## 4. Experimental Results

Figure 3 shows a screenshot of our proposed system. Our system is developed in Python language. For the development, we use PyCharm as an integrated development environment (IDE). For the deep learning library, we use TensorFlow 1.0 on Windows machine. The TensorFlow library we have used is a binary version for Windows. Since we have not compiled the TensorFlow library using Bazel under our Windows system, you can see some warning messages in the beginning of the training process, however the warning messages do not affect the training process.

Table 2 shows the information on the word corpuses of cyber public spheres. For each cyber public sphere, we assigned 1,000 instances, which is a total of 2,000 instances. The 1,000 instances is divided into 700 training instances and 300 test instances. Therefore there is a total of 1,400 instances for training and 600 instances for testing our proposed system.

We have collected the data from two online public forums, which show completely different political tendency. The one is Today's Humor site, also known as Ou. Ou is famous for its political polarity to Deobooleoh Democratic Party, also known as Together with Democratic Party. Most opinions prefer the political opinions and ideas of Deobooleoh Democratic Party. And most sentences in Ou are basically polite words. When a user uses rude words or casual words, she/he could be criticized and banned.

The other one is Ilgan Best, also known as Ilbe. In contrast, Ilbe is famous for its political polarity to Liberty Korea Party (LKP). Ilbe shows opposing political tendency to Ou. Most opinions prefer the political opinions and ideas of Liberty Korea Party. Ilbe is notorious for its rude sentences in their articles and reviews. When a user uses polite words, she/he could be criticized as a hypocrite.

Table 3 shows the results for our proposed system. We applied various parameters to the system to estimate the performance for the experimental data and compare the performance results.

**Table 2. Word corpuses of cyber public spheres**

| | Cyber public sphere A | Cyber public sphere B | Total |
|---|---|---|---|
| **Training Data** | 700 | 700 | 1,400 |
| **Test Data** | 300 | 300 | 600 |

**Table 3. Performance comparison by parameters**

| | Learning rate | # of LSTM Cells | Accuracy |
|---|---|---|---|
| **LSTM 1** | 0.01 | 5 | 52.00% |
| **LSTM 2** | 0.001 | 5 | 57.00% |
| **LSTM 3** | 0.0001 | 5 | 55.00% |
| **LSTM 4** | 0.0003 | 5 | 54.50% |

For the experiment, we fix the number of LSTM cells to be 5. From the table 3, it can be seen that we have the best results 57.00 accuracy when the learning rate is 0.001. The results is not very encouraging at this time, which will be the future work for improving the experimental results.

## 5. Conclusion

In this paper, we focus on the classification of political polarity in cyber public sphere. For this purpose, we apply two well-known deep learning methodologies, word embedding and long short term memory (LSTM). The discussion data in the cyber public sphere is transformed into word corpus through word embedding for deep learning processing. From the transformed word corpus data, we train a recurrent neural network (RNN) system with softmax function. The resulting accuracy is not very remarkable at this time, and we will improve the accuracy by refining the parameter of the LSTM and application of related work.

## Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (no. NRF-2015R1D1A1A01061328).

## References

[1] Y. Bengio, "A Neural Probabilistic Language Model", Journal of Machine Learning Research, Vol. 3, pp. 1137-1155, March 2003.
[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," arXiv preprint, arXiv:1301.3781, 2013.
[3] Y. Bengio, P. Simard, and P. Frasconi "Learning Long-Term Dependencies with Gradient Descent is Difficult," IEEE Transactions on Neural Networks, Vol. 51, No. 2, pp. 157-166, March 1994.
[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, Vol. 9, No. 8, pp. 1735-1780, 1997.
[5] A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks," Textbook, Studies in Computational Intelligence, Springer, 2012.