International Core Journal of Engineering Volume 7 Issue 12, 2021
ISSN: 2414-1895 DOI: 10.6919/ICJE.202112_7(12).0042

Natural language sentiment analysis system based on self-designed deep learning framework

Ziyang Liu[1,*], Xiangyi Yin[2, a]

[1] College of Information Science and Technology, Northwest University, Xi'an, Shaanxi, China

[2] SWJTU-Leeds joint School, Southwest Jiaotong University, Chengdu, Sichuan, China

[a]1571421634@qq.com

*Corresponding author: 2519357947@qq.com

These authors contributed equally to this work

Abstract

With the rapid development of the entertainment industry, movie, which is one of the most important, essential and popular forms of art, is getting more and more commercial. The investment in this area needs reliable index and indicator to justify whether a movie is acclaimed or not. A self-designed deep learning framework is built in order to sentiment analyzing in comments of movie, which is thought that can accurately reflect the audiences' attitudes. The system is designed based on the traditional deep learning framework, including two sets of convolutional layer and pooling layer, and fully connected layer. The system uses multiple losses to optimize the framework, Adam algorithm is used to optimize the learning rate, and the dropout algorithm is used to improve the Accuracy. After the system had been built, data that contains a mass of comments is used to test the system and the result is relatively precise. 82% average accuracy is reached after many parameters' combinations are experimented. It also shows a potential uses in public opinion and many other areas.

Keywords

Deep Learning(DL), Convolutional Neural Network, Sentiment Analysis, Natural Language Processing, Movie Comment.

1. Introduction

Movie, which has much capital, manpower and attention behind, is becoming one of the most indispensable art forms in human's daily routine.The filmdom and many investors deeply concern a lot about the quality of every film when considering the investment plan, while the comment of movie is taken as one of the main point when justifying it. Audiences always give these comments to movies with diverse sentiment like elated, resentful and despondent. These emotions can be roughly split to two types: positive and negative. It is well known that larger percentage of positive comments means better and more acclaimed movie. The movie market has the demand to have a programme which can analyse the integral emotion among comments and tell user the precise, specific and timely percentage and number about data. To address this requirement, we considered that if we can randomly collect comments from a movie and find out the percentage of positive and negative movies, we can learn the overall impression of audiences to the movie. But it is unpractical to achieve this by hands, since we can not list thousands of comments or even more.

To deal with this problem, an sentiment analysis based on Deep Learning is developed. Deep learning is a new region in artificial intelligence. It focus on giving computers the ability to study like human

ISSN: 2414-1895 DOI: 10.6919/ICJE.202112_7(12).0042

beings and can help with the process which is unattainable by hand. Since the aim of project is analyzing the emotion of movie comments by computer itself, convolutional neural networks, which is involved in deep learning, is used to compute, collect and process data through iterative algorithm. Convolutional Neural Network is a new technology in artificial intelligence area, while the old method needs human's help to extract object's feature, which is more about artificial not intelligence. Convolustional neural network's advantage is that it takes the feature extraction automatically instead of hand. Besides, since the project refers to comments that is presented as language, the natural language processing technology, which refers to transform language strings into computerized data and process the data, is used in this paper.

2. Model build

This part mainly describes the deep learning framework model used in this paper. And this experiment was carried out under the Windows operating system, using the TensorFlow 1.15.0 framework to construct a network model, the python version was Python 3.7, and the integrated development environment was carried out on Pycharm 2021.2.

2.1 Deep learning framework model

The model we used is self-designed deep learning framework, which includes convolutional layer, pooling layer and fully connected layer. The framework is :

INPUT —> [[CONV]*N —> POOL]*M ——> [FC]*K

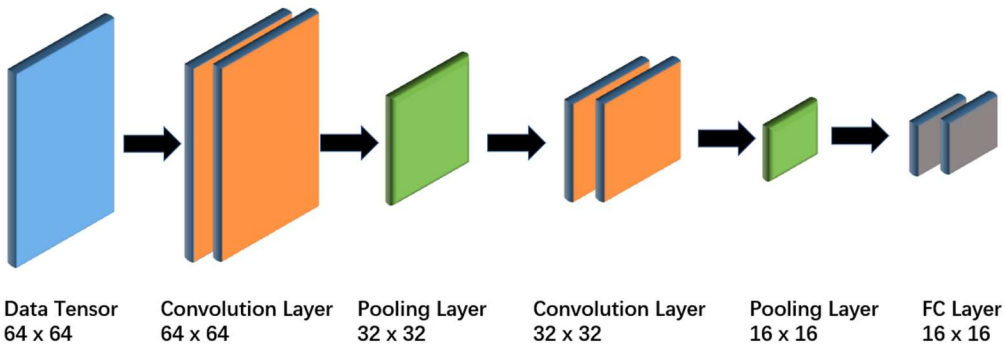


Figure 1. Framework sketch

Data tensor is the input, which then goes through M systems consisting of N convolutional layers and a pooling layer. Superimpose N convolutional layers, then (optionally) superimpose a Pooling layer, repeat this for M times, and finally superimpose K fully connected layers. This model chooses N = 2, M = 2 and K = 2. The figure shows that two convolutional layers and a pooling layer are combined, while there are two such combinations before fully connected layer, which has also two layers superimposed.

Convolutional neural network, which refers to feature extraction technology, is a feedforward neural network involving convolution and deep structure. It is one of the representative algorithms in deep learning [3] and has the ability to express learning, and provides translation-invariant classification for input information according to the class structure, so it is called translation-invariant artificial neural network (SIANN) [4].

Firstly, tf.nn.conv2d is used for data convolution. SAME's padding method is used to fill the matrix to facilitate convolution. "SAME" tries to pad evenly left and right, but if the amount of columns to be added is odd, it will add the extra column to the right (the same logic applies vertically: there may be an extra row of zeros at the bottom) [5].

We number each element of the array, use $a_{i,j}$ to represent the element in the i row and j column of the array, and number each wright of the filter, use $w_{m,n}$ to represent the weight in m row and n

ISSN: 2414-1895 DOI: 10.6919/ICJE.202112_7(12).0042

column, and use w_b to represent the bias term of the filter; number each element in feature map, use $R_{i,j}$ to represent the element of feature map in i row and j column; use f to represent activation function(We choose Relu function f(x) = max(0,x) to be the activation function, which reduces the problem of vanishing gradients, is faster and uses less memory). F is the width of filter, D is the depth of the image before convolution, and is also the depth of the corresponding filter. The calculation process is as follows:

The value of corresponding element in each filter:

$$w_{d,m,n}a_{d,i+m,j+n} + w_b \tag{1}$$

The value corresponding to each feature map

$$R_{i,j} = f(\sum_{d=0}^{D-1}\sum_{m=0}^{F-1}\sum_{n=0}^{F-1} w_{d,m,n}a_{d,i+m,j+n} + w_b) \tag{2}$$

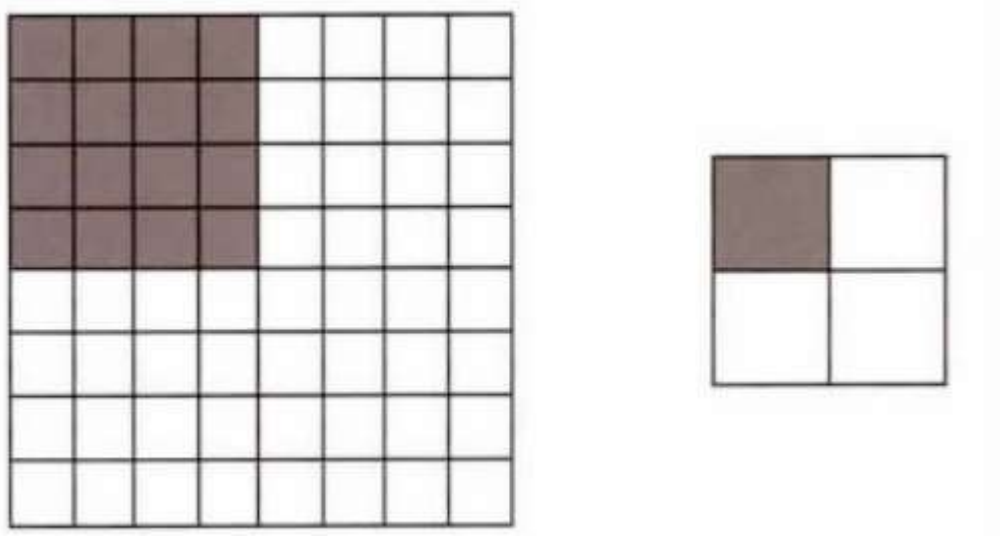
Each convolutional layer can have multiple filters. After each filter is convolved with the original image, a Feature Map can be obtained. Therefore, the depth (number) of the Feature Map after convolution and the number of filters in the convolutional layer are the same.

Since the size of data from convolutional layer is too large, and there exists a fact that in many places several values are used to express one value that is helpful for experiment, the pooling layer is used to downsize the dimension of features and compress numbers of data and parameters while maintain the more important features. After performing two convolutions, we perform Maxpooling on the Feature Map. The maximum pooling model takes the maximum value in the pooling domain as the sub-sampling feature value. We suppose that the input feature map is matrix F, the sub-sampling pooling domain is c × c matrix P, the offset is b2, the sub-sampling feature map obtained is S, and the moving step of the pooling process is set to c.

The formula is :

$$S_{ij} = \max_{i=1,j=1}^{c} \left(F_{ij}\right) + b_2 \tag{3}$$

In the formula above, $\max_{i=1,j=1}^{c}$ represents the largest element taken from the pooling domain of the input feature map F whose size is c × c. The picture below shows the working principle of pooling layer[6]:



(a) Original Feature Map (b) Feature Map After Pooling

Figure 2. Pooling Sketch

ISSN: 2414-1895 DOI: 10.6919/ICJE.202112_7(12).0042

The data we read is 64*64. After two convolutions and one pooling, we get a data size of 32*32. After repeating the above process, data of 16*16 size is finally obtained, and the data is passed to the fully connected layer.

The working principle of the fully connected layer is to connect each node inside with each node on the upper layer, as shown below[7].

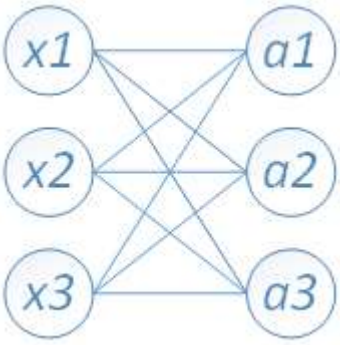


Figure 3. FC Calculate Principle

In this picture, x1、x2、x3 are represented as the inputs of fully connected layer, while a1, a2, a3 are the output. Fully connected layer is always placed at the tail end of process. It gathers all the features given before and calculates them according to weight given by y = W * x, where y is the output, x is the input and W means weight.

Our framework includes two fully connected layers to better process data to achieve the desired goals. We define the calculation formula for the access node of the first fully connected layer as follows:

$$(image\_size\ //\ 4)\ *\ (image\_size\ //\ 4)\ *\ 32 \tag{4}$$

The input image size is 64, divided by 4 is because the image size is reduced after two pooling, and multiplied by 32 is because our convolutional layer depth is 32. The output nodes of the first fully connected layer are 128, so the access points of the second fully connected layer are 128. Since the data set we use only has two emotions of praise and criticism, the output of our last fully connected layer is 2.

2.2 Framework optimization

We regularize the framework using the L2 LOSS method, which is equivalent to adding a new variable to the original loss of the framework, namely loss = loss + loss_new. The formula is:

$$S = \sum_{i=0}^{n}(y_i - h(x_i))^2 \tag{5}$$

Its function is to do the subtraction between the target value y_i and the model output (estimated value) $h(x_i)$ and then square the error. We define a function named apply_regularization to achieve this function. The model here uses the tf.nn.l2_loss() function, which is used to calculate the error value of the tensor using the L2 norm, but there is no square root and only half of the value of the L2 norm, which is:

$$Output = \frac{(\sum_{i=0}^{n} {t_i}^2)}{2}\ (n = 1, \ldots, n) \tag{6}$$

In the training phase, we use the drop out method to randomly drop some nodes in the fully connected layer with a given probability. Therefore, when the value is propagated forward, the activation value of a certain neuron will stop working with a certain probability p, which can make the model more generalized, because it will not rely too much on some local features.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202112_7(12).0042

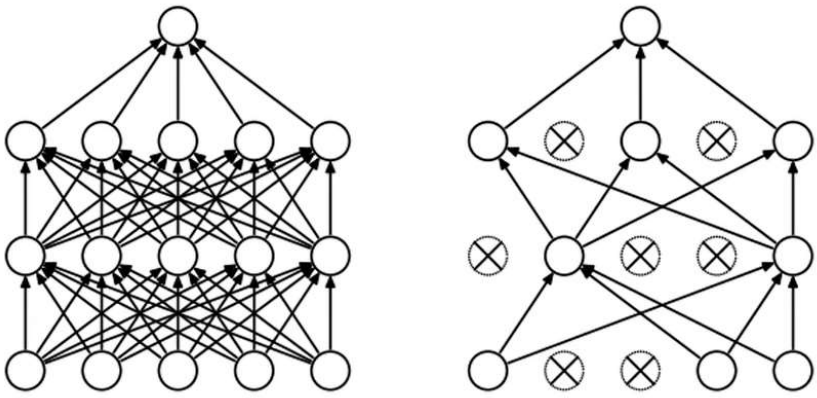


Figure 4. Dropout sketch

In order to achieve this function, we only need to use tf.nn.dropout, whose function is to output the input tensor proportionally according to the given probability parameter.

In the learning rate optimization part, we use Adam optimization, which designs independent adaptive learning rates for different parameters by calculating the first-order moment estimation and the second-order moment estimation of the gradient.

Adam (adaptive moment estimation) is a stepwise optimization algorithm of random objective function based on low-order adaptive moment estimation [8].Compared with the basic stochastic gradient descent algorithm, Adam is not easy to fall into the local optimum and the update speed is fast. The Adam algorithm update parameter method is as follows:

While θ_t not converge do

$$t = t + 1 \tag{7}$$

$$g_t = \nabla_\theta f_t(\theta_{t-1}) \tag{8}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t \tag{9}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\, {g_t}^2 \tag{10}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{m_t}{(1-\beta_1^t)(\sqrt{\frac{v_t}{1-\beta_2^t}}+\varepsilon)} \tag{11}$$

End while

The θ_{t-1} is the parameter to be updated; α is the learning rate; g_t is the gradient of the random objective function; m_t is the partial first-order moment estimation, $m_0 = 0$; v_t is the partial second-order moment estimation, $v_0 = 0$; β_1 and β_2 are the exponential decay rate of moment estimation; ε is a small positive number. The default values of these parameters in machine learning problems are:

$$\alpha = 0.01, \beta 1 = 0.9, \beta 2 = 0.999, \varepsilon = 10 - 8\ [8]$$

After establishing the algorithm, we need to optimize the learning rate, and decay with a given step size for a given learning rate. This can avoid the calculation cycle being too long due to the step size being too large, or the calculation result is not obtained after an infinite number of iterations due to the step size being too small.

We use a series of codes to complete this step of operation. In this code, we define the initial learning rate, each batch is equivalent to one step of operation, and how many steps are the learning rate decay and decay rate. For example, assuming that the learning rate is 0.8 at the beginning and the decay rate

ISSN: 2414-1895 DOI: 10.6919/ICJE.202112_7(12).0042

is 0.9, then after a period of our definition, the learning rate becomes 0.72 (0.8*0.9), so the step length and learning rate can be excellently controlled and improved the degree of model fit.

3. Experiment Phase

3.1 Data set

The data set used by the project is a film review file (.tsv file) provided by the project instructor. There are three columns, namely id, sentiment, and review, with a total of 25,000 rows of data. cf. Table 1:

Table 1. Part of Sentiment Dataset

id	sentiment	review
5814_8	1	With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd documentary here and there, watched The Wiz and watched Moonwalker again....
2381_9	1	\The Classic War of the Worlds\" by Timothy Hines is a very entertaining film that obviously goes to great effort and lengths to faithfully recreate H. G. Wells' classic book. Mr. Hines succeeds in doing so. I, and those who watched his film with me,...
7759_3	0	The film starts with a manager (Nicholas Bell) giving welcome investors (Robert Carradine) to Primal Park. A secret project mutating a primal animal using fossilized DNA, like ¨Jurassik Park¨, and some scientists resurrect one of nature's most fearsome predators,...
3630_4	0	It must be assumed that those who praised this film (\the greatest filmed opera ever,\" didn't I read somewhere?) either don't care for opera, don't care for Wagner, or don't care about anything except their desire to appear Cultured. Either as a representation of Wagner's swan-song,...

In the data set, 0 and 1 correspond to criticism and praise. Film reviews are English texts, with many stop words, punctuation marks, modal particles, etc. In order for the machine to achieve a better training effect, the text needs to be preprocessed. We treated each rating as a category, and there are two categories of praise and criticism. In order to keep the number of examples in different categories in balance, 70% of the data is used for training and 2% for validation testing to maintain statistical properties when evaluating the model.

3.2 Data Process(load.py)

Regarding the preprocessing of the text, we used the nltk library. First of all, we defined two functions review_to_word and Clear_reviews. In review_to_words function, we mainly removed the HTML and non-letters of the read text, and converted it to lowercase, and then used the split () function to split the sentence into independent words. The set () function was used to build a list of no repeated words, and finally removed stop words, connected the processed words and output the simplified text. The main purpose of the Clear_reviews function is to store reviews in a list, which is convenient for subsequent operations. First obtained the number of reviews according to the column size of the data, and then looped through each review. For each review, we used the review_to_words defined above to process the read text and used the append () function to add the corresponding processed review to the end of the list, and finally returned an updated clean_review list.

After defining these two functions, we used pandas to read the tsv file, call the Clear_reviews function operation on the file ‘review’ column, and get clean_train_reviews.

Because machine learning does not read pictures or text directly like humans, we need to convert the data into a language that can be recognized by the machine, that is, an array. Here in our model, we

ISSN: 2414-1895 DOI: 10.6919/ICJE.202112_7(12).0042

used the word vector model and one-hot encoding. We used the CountVectorizer function in the sklearn library to build a word vector model. Simply put, the word vector model is to number all words, such as 1: movie. We used the word vector model, and we only need to compare the finished model with the comments. First, we initialized the bag of words vector. Here we selected the first 4096 words with the highest occurrence frequency as they have higher weights. Then we used the fit_transform() function, whose role is to fit the model and learn vocabulary; and transform our training data into feature vectors. In this way, we get train_data_features, and convert the obtained train_data_features into an array. So far, each comment was converted into an array of (4096, 1). After these operations, we defined _train_samples, _test_samples, train_labels, and test_labels. _train_samples and train_labels use processed text as samples, while _test_samples and test_labels use unprocessed text as samples. After getting the sample, we defined a reformat function. The function reads the sample and label, reshapes them into the form of 64*64*1 and makes them into one-hot encoding.

After writing all the functions, we defined the data size format as 64*64, with two types of labels, commendatory and derogatory, and single-channel processing. We have written the data processing part into a load.py file so that it can be called when training the model.

Because there are too many data samples, one-time reading causes the running speed to be too slow, we choosed to process the data in batches. We divided the data into blocks of size 500, and read one data block at a time for operation.

3.3 Result Analyze

After training the model with a given data set, we tested the model and got an accuracy of about 80%. In order to optimize the accuracy, we analyzed the two necessary parameters of the model, calculated the standard deviation of the results to better describe the impact of parameters on accuracy. The larger the value of the standard deviation, the farther the return from the past average value, the more unstable the return and the higher the risk. On the contrary, the smaller the value of the standard deviation, the more stable the return and the smaller the risk.

Table 2. Biases variable influence

Biases variable	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Average Accuracy	80.80%	81.40%	80.06%	81.10%	80.20%	80.30%	80.40%	81.05	79.90%
Standard Deviation	1.49	1.54	1.455	1.785	1.318	1.462	2.487	1.857	1.733

Table 3. Iteration times influence

Iteration times	500	1000	1500	2000	2500	3000
Average Accuracy	81.228	81.975	81.343	81.271	80.343	80.429
Standard Deviation	1.237	1.627	1.505	1.192	1.857	1.168

Two parameters were taken out to show how they affect the results. The record sheet is shown above, cf. Table 2 and Table 3. For the deviation variable, the change of the result is irregular and increases linearly. After many attempts, the highest average accuracy of the product is considered to be 0.2. The standard deviation shows that these changes are irregular. When the iteration is reduced to about 500 times, the average accuracy will increase, and the irregular standard deviation changes will still exist. After these attempts, set the two parameters to deviation variable = 0.2 and iteration step size = 1000 to obtain the highest accuracy.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202112_7(12).0042

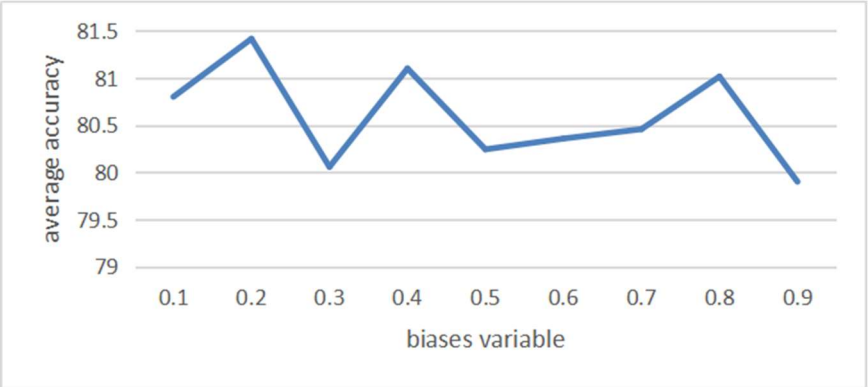


Figure 5. Relationship between Biases and Average accuracy

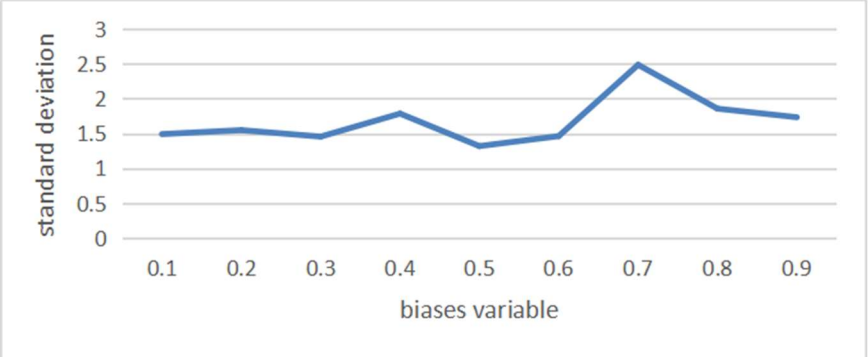


Figure 6. Relationship between Biases and Standard deviation

The above two figures show the change in the results while maintaining the number of iterations, each time increasing the deviation variables 0.1. The results are considered irregular. Based on comprehensive judgment, it is believed that biases can achieve a better prediction effect at 0.2, so this parameter is selected as a fixed value for iterative testing.

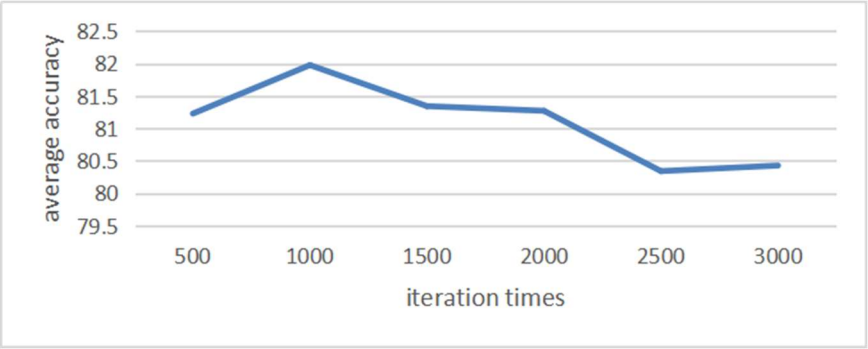


Figure 7. Relationship between Iteration and Average accuracy

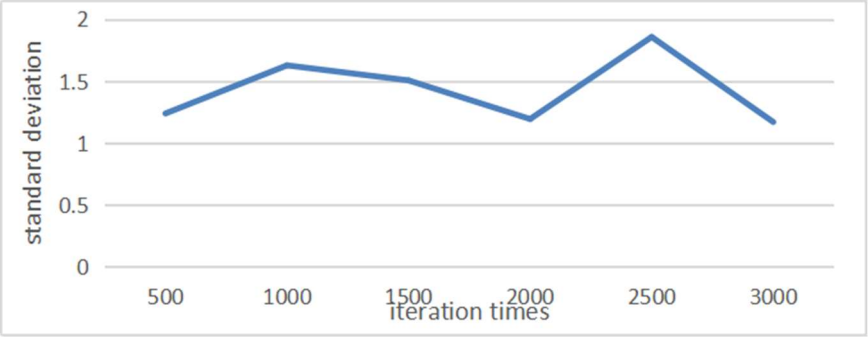


Figure 8. Relationship between Iteration and Standard deviation

ISSN: 2414-1895 DOI: 10.6919/ICJE.202112_7(12).0042

The two graphs above show the changes in the results while keeping the deviation constant, each time increasing the number of iterations by 500. They showed that 500 iterations is an available option for maintaining high accuracy.

After the iterative process, the accuracy rate increased from 45% (that is, the recognition accuracy rate is lower than the random selection) to 82%, indicating an acceptable high accuracy rate. The increase in average accuracy indicates that the Model algorithm is correct. By improving the parameters, the accuracy that can be convinced by the public can be obtained.

4. Conclusion

This paper proposes a natural language sentiment analysis model based on a self-designed deep learning framework. This model uses the traditional process of convolutional layer to pooling layer to fully connected layer, which can more accurately judge the emotion of the text. In this case, considering the predictive power of artificial intelligence, an accuracy of 80% is considered acceptable. High accuracy also proved that this model can have more space for applications, such as monitoring public opinion and so on.

This experiment did not consider the fact that the text was not completely read. This issue will be studied in subsequent experiments. At the same time, this model still has room for improvement at the optimization level.

Acknowledgments

Thanks to the project instructor Li Shuangshuang for his strong support to our team.

References

[1] Cen Xuelin, Hou Yong. Influencing factors and economic value of online film reviews in the Internet[J]. Contemporary Economy,2018(07):68-70.

[2] Cen Xuelin, Hou Yong. Influencing factors and economic value of online film reviews in the Internet[J]. Contemporary Economy,2018(07):68-70.

[3] Goodfellow, I., Bengio, Y., Courville, A.. Deep learning (Vol. 1). Cambridge：MIT press, 2016 / Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, L., Wang, G. and Cai, J., 2015. Recent advances in convolutional neural networks. arXiv preprint arXiv:1512.07108.

[4] Zhang, W., 1988. Shift-invariant pattern recognition neural network and its optical architecture. In Proceedings of annual conference of the Japan Society of Applied Physics.

[5] kaisa158, difference between padding='SAME' and 'VALID', CSDN, [Online]. Available at: https://blog.csdn.net/kaisa158/article/details/81416975.

[6] Liu Wanjun, Liang Xuejian, Qu Haicheng. Study on the Learning Performance of Convolutional Neural Networks with Different Pooling Models[J]. Journal of Image and Graphics,2016,21(09):1178-1190.

[7] Li Yandong, Hao Zongbo, Lei Hang. Summary of Convolutional Neural Network Research[J]. Computer Applications,2016,36(09):2508-2515+2565.

[8] KINGMA D P, BA J. Adam: A Method for Stochastic Optimization［J］. Computer Science, 2014.