A Novel Computer-aided Multi-label Emotion Recognition of Text Method Based on Word embedding and BiLSTM

Zheng Jia

The Harker School, Ssn Jose, CA

20jackj@students.harker.org

Keywords: Emotion recognition of text, Deep Learning (DL), Bi-directional Long Short-Term Memory (BiLSTM), Word Embedding, 7 kinds of emotions

Abstract: Emotional recognition has a great influence on people's daily behavioral interactions. With the emergence of massive data, especially various digital social media, newspapers, magazines, books, daily conversations and so on. Identifying emotion of text automatically by computer, which is of great significance to the user's sentiment analysis, network public opinion monitoring and other fields. In recent years, with the rise of deep learning, great success has been achieved in many fields, especially the emergence of gated recurrent neural networks, which are good at processing sequence data such as text and solving the problem of Long-Term dependencies. Based on this, this study proposes a multi-label emotion recognition of text based on Bi-directional Long Short-Term Memory and Word embedding technology. The experimental results show that the performance of our neural network model outperformed other neural network models. The training set, the validation set and the test set have an accuracy of 0.753, 0.6228, and 0.6136, respectively, which means that the state-of-the-art 7 kinds of emotion recognition for the text are realized, namely, happiness, anger, sadness, disgust, surprise, fear, normal.

1. Introduction

The theory of emotions has been widely concerned and studied by Darwin and James [1]. At the beginning, the research progressed very slowly. In recent decades, the research of emotions has developed rapidly.

Despite a lot of emotional theory, Ekman and Friesen found that humans have six basic emotions, namely happiness, sadness, fear, surprise, anger, and jealousy [2]. Other emotions are combined with these six basic emotions, which are compound emotions. Therefore, this article selects these six emotions in the research process, plus a neutral emotion, and judges the seven emotions of the text story.

At present, research on emotions focuses on the mechanism of emotion generation, the imaging factors of emotions, the classification of emotions, and the influence of emotions on people. There are relatively few studies on the judgment of people's emotions. The existing researches on emotional judgment are all studies on expressions, micro-expressions, and physical behaviors. Judging their emotions by observing the expressions and behaviors of others, and exploring which expressions and behaviors can accurately judge emotions. Emotion and personality, temper, individual behavior and other factors interact, not only affect people's behavior, expression, but also people's behavior and expression can reflect a person's emotions.

In recent years, with the popularization of social networks and social software, there is now a large amount of non-face-to-face communication. In these exchanges, the expressions and behaviors of the communication objects cannot be seen, and it is very difficult to judge their emotions. However, the emotions of the other party play an important role in the response and decision-making of the communicator. It is necessary to react differently according to the different emotions of the other party, so that they can communicate more harmoniously.

At present, the commonly used text sentiment analysis methods are mainly based on traditional machine learning algorithms, such as support vector machines, naive Bayes, etc., and these methods have many drawbacks: it requires a large number of manual emotional dictionary and prior knowledge. With the rise of deep learning, although researchers use convolutional neural networks and recurrent neural networks for sentiment analysis of texts, there is a problem with modeling sentences using only Long short-term memory: unable to encode information from back to front [3] And Bi-directional Long Short-Term Memory can better capture the two-way semantic dependence. Some researchers also use Long short-term memory networks and neural network language models, but only the two categories of positive and negative emotions of the text, lack of a more fine-grained classification of the text's emotional level [4] Based on this, this paper proposes a vectorization representation of text words based on word embedding as the input layer of the model, in order to enhance the effect of the model, combined with Bi-directional Long Short-Term Memory networks, and integrates from a variety of different corpus databases [5] including TEC , Tales, EmoInt for multi-label emotion recognition of text.

2. Method

2.1 Word embedding

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers. Conceptually it involves a mathematical embedding from a space with one dimension per word to a continuous vector space with a much lower dimension.

Methods to generate this mapping include neural networks [6] dimensionality reduction on the word co-occurrence matrix [7] probabilistic models [8] explainable knowledge base method,[6] and explicit representation in terms of the context in which words appear [9].

Word and phrase embeddings, when used as the underlying input representation, have been shown to boost the performance in NLP tasks such as syntactic parsing [8] and sentiment analysis [10].

2.2 Recurrent Neural Networks (RNNs)

The standard Recurrent Neural Network structure is shown in Fig. 1. The hidden layers of the RNN are connected. The input of the hidden layer not only has the input of the input layer, but also the output of the hidden layer at the previous moment, and the model generates an output each time combined with the current input and hidden state.

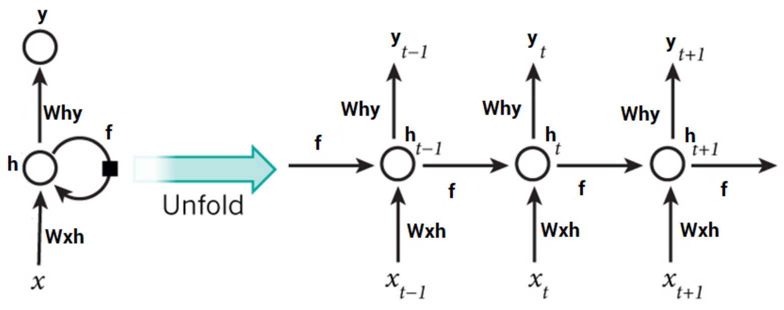


Fig. 1. RNN neural network structure

The formula for RNN is defined as follows: For a given input sequence of length T: X = (x_1, x_2, ..., x_T), it iterates the formula in order from t = 1 to t = T in chronological order, in order Obtain hidden layer neuron states H=(h_1,h_2,...,h_T) and output sequence Y=(y_1,y_2,...,y_T).

$$\mathbf{h_t} = \partial(\mathbf{W_{xh}}\,\mathbf{x_t} + \mathbf{f}\,\mathbf{h_{t-1}} + \mathbf{b_h})$$

$$\mathbf{y} = \mathbf{W_{hy}}\,\mathbf{h_t} + \mathbf{b_y}$$

The model parameter W is a weight matrix (W_{xh} is the connection weight matrix of the input layer x to the hidden layer h, W_{hy} is the connection weight matrix of the hidden layer h to the output layer y, and f is the connection of the hidden layer h to the hidden layer h Weight matrix), b is the offset vector (b_h is the hidden layer offset value, b_y is the output layer offset value), ∂ is the activation function of the hidden layer, usually set to the Sigmoid function, and the Sigmoid function formula is as follows:

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

2.3 Long Short Term Memory networks (LSTMs)

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies [11]. They were introduced by Hochreiter & Schmidhuber (1997).

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, The repeating module in an LSTM contains four interacting layers as shown in Fig. 2.

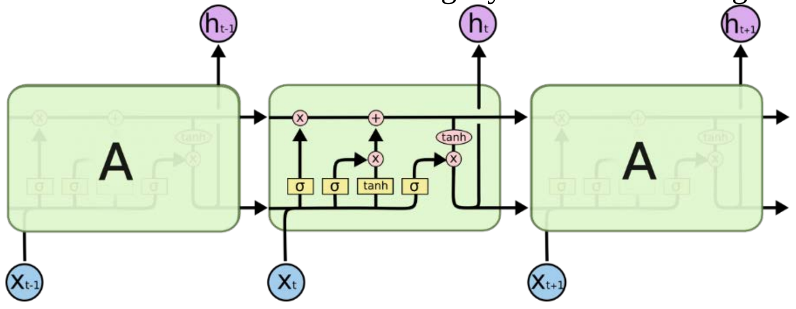


Fig. 2. The repeating module in an LSTM contains four interacting layers.

In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent point wise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. Denote * as element wise multiplication and ignore bias term, LSTM calculates a hidden state ht as:

$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Here, i, f, o are called the input, forget and output gates, respectively. Note that they have the exact same equations, just with different parameter matrices (W is the recurrent connection at the previous hidden layer and current hidden layer, U is the weight matrix connecting the inputs to the current hidden layer). They care called gates because the Sigmoid function squashes the values of these vectors between 0 and 1, and by multiplying them element-wise with another vector you define how much of that other vector you want to "let through". The input gate defines how much of the newly computed state for the current input you want to let through. The forget gate defines how much of the previous state you want to let through. Finally, The output gate defines how much of the internal state you want to expose to the external network (higher layers and the next time step). All the gates have the same dimensions $\boldsymbol{d_h}$, the size of your hidden state.

$\tilde{C}$ is a "candidate" hidden state that is computed based on the current input and the previous hidden state. C is the internal memory of the unit. It is a combination of the previous memory, multiplied by the forget gate, and the newly computed hidden state, multiplied by the input gate. Thus, intuitively it is a combination of how we want to combine previous memory and the new input. We could choose to ignore the old memory completely (forget gate all 0's) or ignore the newly computed state completely (input gate all 0's), but most likely we want something in between these two extremes. $\boldsymbol{h_t}$ is output hidden state, computed by multiplying the memory with the output gate. Not all of the internal memory may be relevant to the hidden state used by other units in the network.

Intuitively, plain RNNs could be considered a special case of LSTMs. If fix the input gate all 1's, the forget gate to all 0's (say, always forget the previous memory) and the output gate to all 1's (say, expose the whole memory), it will almost get a standard RNN.

2.4 Bi-directional Long Short-Term Memory (BiLSTM)

The forward LSTM is combined with the backward LSTM to form BiLSTM. For example, we code the phrase "I love China", the model is shown in Fig. 3.

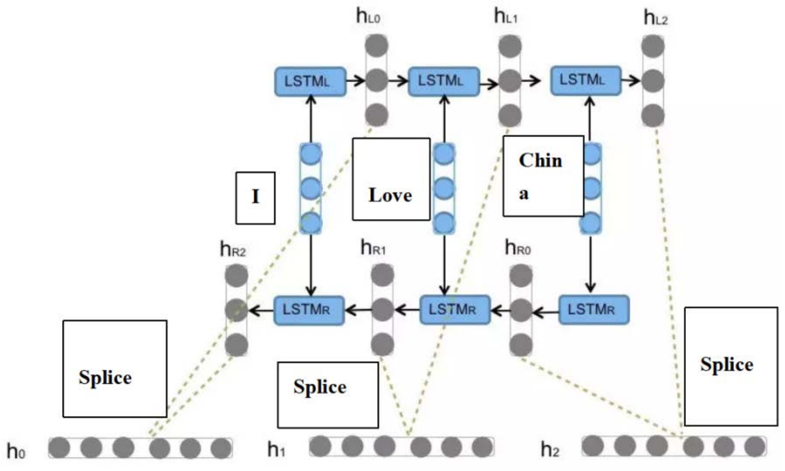


Fig. 3. Bi-directional LSTM encoding sentence

Enter "I", "Love", and "China" in the forward direction LSTM_L to get three vectors {h_{L0}, h_{L1} ,h_{L2}}. Enter "China", "Love", and "I" in the backward direction LSTM_R to get three vectors {h_{R0}, h_{R1},h_{R2}}. Finally, the forward and backward hidden vectors are spliced to get {[h_{L0}, h_{R2}], [h_{L1}, h_{R1}], [h_{L2}, h_{R0}]}, ie {h_0,h_1,h_2}.

For sentiment classification tasks, the representation of the sentences we use is often [h_{L2},h_{R2}]. Because it contains all the information of forward and backward, as shown in Fig. 4.

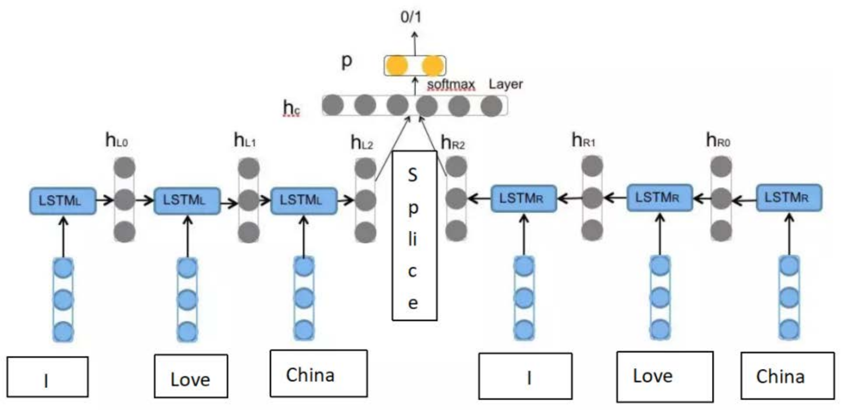


Fig. 4. Splice vector for sentiment classification

3. Dataset Processing

Our dataset has a total of 39,597 text data, of which the training set is 33855, the validation set is 3762, the test set is 1980, the test set is 5% of the total data set, and the training set and the validation set are 9:1. The division is in accordance with the neural network engineering standards. At the same time, our datasets are rich in sources, mainly from the following seven areas: tec, emotion-cause, grounded_emotions, electoral_tweets, dailydialogues, tales-emotion, emoint, the number of texts in each field is shown in the following table:

Table 1. Nums of Text of seven Domains

Domain Name	**Nums of Text**
tec	10713
Emotion-cause	1789
Grounded_emotions	1055
electoraltweets	2298
dailydialogues	10584
Tales-emotion	9611
emoint	3547

Classified from the source of the data, our text sentiment database comes from the following four databases: tweets, conversations, artificial_sentences, fairy_tale_sentences, and the number of texts in each database is shown in the following table:

Table 2. Four databases of our text sentiment database

Databases Name	**Nums of databases**
tweets	17613
conversations	10584
artificial_sentences	1789
fairytale_sentences	9611

Finally, in order to balance the seven expressions: joy, anger, sadness, disgust, fear, surprise, noemo, the number of texts, the data is balanced, that is, the number of text data of each expression is relatively balanced, avoiding the model learning In the process of data characteristics, there is tendency and data deviation, and the objectivity of the data is kept as much as possible. The balanced data set for the seven expressions is shown in the following table:

Table 3. 7 emotions of balanced dataset

Emotion Name	**Nums of Text**
joy	6062
anger	6062
sadness	6062
disgust	3225
fear	6062
surprise	6062
noemo	6062

4. Experiment Design And Data Analysis

3.1 Hyper parameters setting

Next, we start the model training process. Through experiments, the best hyper parameter settings are shown in the following table:

Table 4. Best Training parameters of our Neural Network

Parameter Name	**Value(or Proper noun)**
Batch_size	32
epochs	4
optimizer	‘Adam’
Units of LSTM	100
dropout	0.5

Our neural network structure is as follows:

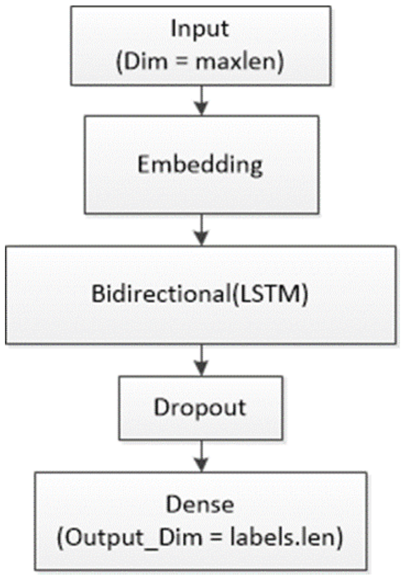
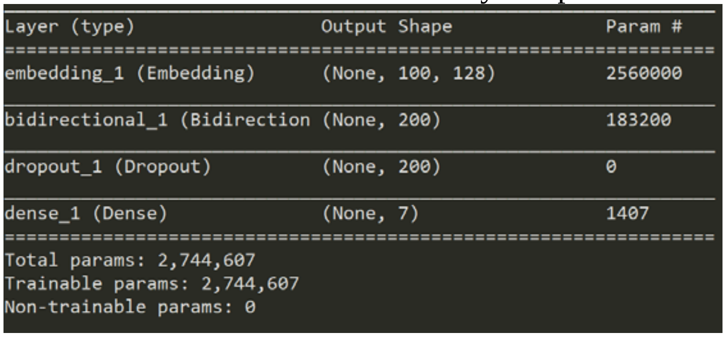


Fig. 5. Our devised model structure

In the model input and output parameters, the maximum length of our sequence is set to 100, the maximum dictionary length is 20000, the dimension embedded in each word is set to 128, followed by the bidirectional Long Short-Term memory network, and its output tensor is Two-dimensional array, the length of the latter dimension is 200. At the same time, we use the dropout layer. Finally, we set the full connection layer of the label length. The activation function uses 'softmax' to output the probability distribution of the seven labels. The hierarchy and parameter output are as follows:

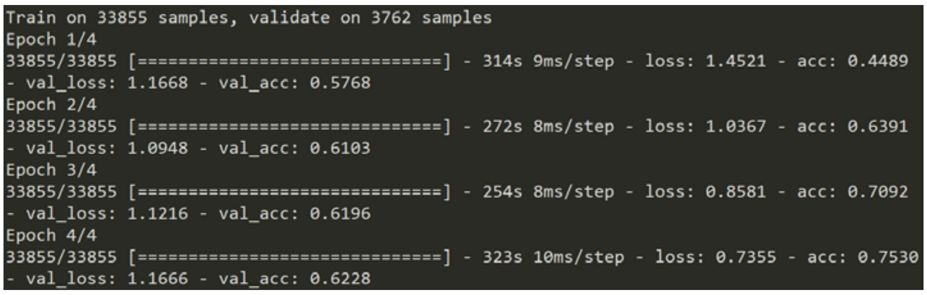


```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 128)          2560000
_________________________________________________________________
bidirectional_1 (Bidirection (None, 200)               183200
_________________________________________________________________
dropout_1 (Dropout)          (None, 200)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 1407
=================================================================
Total params: 2,744,607
Trainable params: 2,744,607
Non-trainable params: 0
```

Fig. 6. The hierachy and parameter of model

3.2 Training process

After determining the network structure and network nodes, we began to train our model on the computer, the hardware is cpu. The training process is as follows:



```
Train on 33855 samples, validate on 3762 samples
Epoch 1/4
33855/33855 [==============================] - 314s 9ms/step - loss: 1.4521 - acc: 0.4489
- val_loss: 1.1668 - val_acc: 0.5768
Epoch 2/4
33855/33855 [==============================] - 272s 8ms/step - loss: 1.0367 - acc: 0.6391
- val_loss: 1.0948 - val_acc: 0.6103
Epoch 3/4
33855/33855 [==============================] - 254s 8ms/step - loss: 0.8581 - acc: 0.7092
- val_loss: 1.1216 - val_acc: 0.6196
Epoch 4/4
33855/33855 [==============================] - 323s 10ms/step - loss: 0.7355 - acc: 0.7530
- val_loss: 1.1666 - val_acc: 0.6228
```

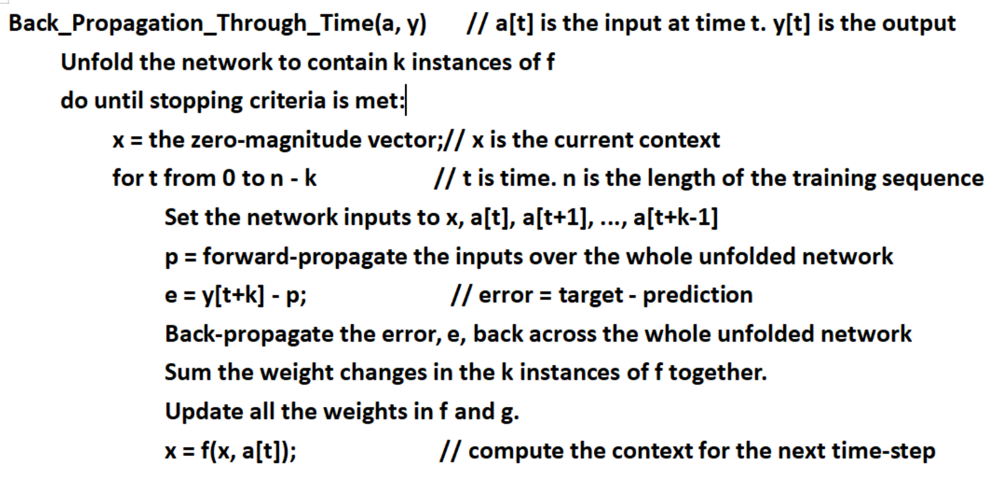
Fig. 7. The training process

From the above figure we can see that each epoch is about 300s, and our cpu can complete the training of the model normally. In the training process, we use the minimum batch training method, that is, the model randomly extracts the sample data of the batch_size size from the training set, first performs forward propagation, obtains the actual output, and then calculates the loss according to the difference between the actual output and the real label, our The loss function C uses a categorical cross_entropy function, and the calculation formula is as follows:

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)]$$

Above the formula, y is the desired output and a is the actual output of the neuron.

At the same time, we calculate the gradient according to the loss C, and use the gradient to update the weight through the time(BPTT),Pseudo-code for a truncated version of BPTT as follows, where the training data contains n input-output pairs, but the network is unfolded for k time steps:



```
Back_Propagation_Through_Time(a, y)    // a[t] is the input at time t. y[t] is the output
    Unfold the network to contain k instances of f
    do until stopping criteria is met:
        x = the zero-magnitude vector;// x is the current context
        for t from 0 to n - k               // t is time. n is the length of the training sequence
            Set the network inputs to x, a[t], a[t+1], ..., a[t+k-1]
            p = forward-propagate the inputs over the whole unfolded network
            e = y[t+k] - p;                  // error = target - prediction
            Back-propagate the error, e, back across the whole unfolded network
            Sum the weight changes in the k instances of f together.
            Update all the weights in f and g.
            x = f(x, a[t]);                  // compute the context for the next time-step
```

Fig. 8. Pseudo-code for back propagation through time of LSTM

In this way we complete an update of the network parameters based on the sample data of a batch. As the number of training iterations increases, the actual output of the model is closer to the real output by back propagation through time, and the parameters of the model are close to the optimal parameters. That is, the complex mapping from the input of text data to the text label is implemented as follows:

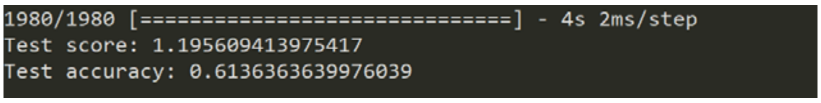
$$g = f(i)$$

Above the formula, g represents the target label of emotion, f is a Nonlinear complex function, i refers to the input text.

The purpose of our training model is to better find such a function f to achieve the best fit from input to output. However, this kind of fitting is not artificially setting the parameters of the function f. Instead, it automatically discovers the patterns and dataset features from the data through the neural network model, thereby realizing the higher-level feature representation of the text. This is also the charm of neural networks and deep learning.

3.3 Test process

After completing the training of the model, we use keras, a pure Python advanced neural network and machine learning library to save the weight of our trained network model, which is the h5 file. Next, we load our network weight file and perform emotion recognition on the test data of 1980 texts. The results are as follows:



```
1980/1980 [==============================] - 4s 2ms/step
Test score: 1.195609413975417
Test accuracy: 0.6136363639976039
```

Fig. 9. Result on the test set

From the above figure, we can see that the accuracy of our model in the test set is 0.6136. That is to say, in the 1980 text data, we can correctly classify 7 emotions among 1215 text data. The model is for text. Semantic interpretation and emotional fine-grained analysis are successful. Of course, we have further explored some textual sentiment data that cannot be correctly classified. Some text data cannot be correctly labeled with a single label. However, in the data processing process, the method adopted is that only one type of emotion tag is given for each piece of text, and the possibility that a piece of text corresponds to multiple emotion tags is excluded.

3.4 Model inference

To further explore our model's ability to adapt to new data, that is, generalization capabilities. We made a browser-based model inference test. For any user input, the model interprets the text on the webpage and outputs the emo expression corresponding to the seven labels. It is inferred that we are divided into two levels of experiments:

(1) Text recognition with keywords such as happy, sad, etc.

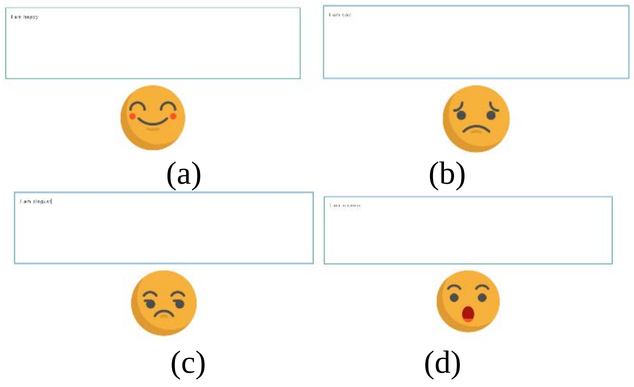
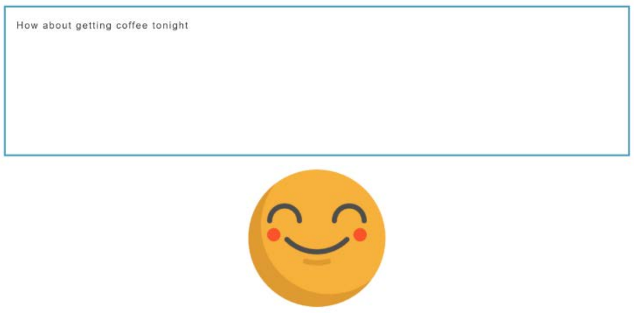


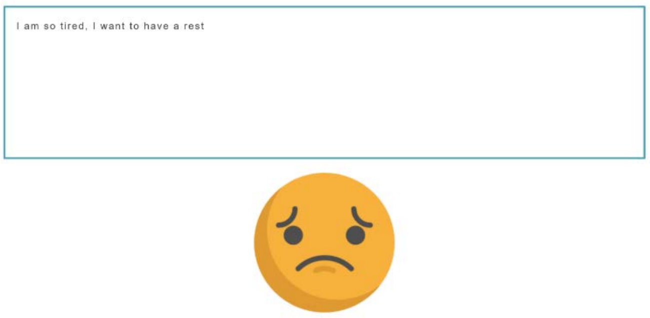
Fig. 10. Inference On the Browser by inputting some text, which will generating the corresponding emoji emotion of the text.(a)input text: ’I am happy’,emoji:happy; (b)input text:’I am sad’, emoji:sad; (c)input text:’I am disgust’, emoji:disgust; (d)input text:’I am surprise’, emoji:surprise.

From the above figure, we can find that once the text contains keywords: sad, happy, disgust, fear, joy, etc., the text words with obvious emotional tendency, the model can be well recognized.

(2) Implicit test of text emotions


How about getting coffee tonight

(1)


I am so tired, I want to have a rest

(2)

Fig. 11. No explicitly emotion words in the sentences, (1)input text is ‘how about getting coffee tonight’, the output demonstrates the happiness of the speaker; (2)input text is ‘I am so tired. I want to have a rest’, the output demonstrates the sadness of the speaker.

As can be seen from the above figure, although the text such as 'how about getting Coffee tonight' does not appear emotional words such as joy, happy, but the model can interpret the joy component according to the semantics, indicating that our LSTM is processing sequences such as text. The data can indeed capture the context information of the context. Of course, the method based on world embedding is also beneficial to better represent the relationship between words and improve the effect of the model. Similarly, our inferred text 'I am so tired. I want to have a rest' is based on the same principle.

5. Discussion And Conclusion

This study uses a variety of corpus databases from different sources, covering social networks (tweets), daily conversations, artificial_sentences, and fairy tales. The data sources are widely representative and can be more realistically reflected. The essence of textual emotions in natural language. At the same time, we use the current popular deep learning technology, especially the distributed representation technology of words [11] and the technique proposed by Schmidhuber et.al to solve the problem of long-term dependence distance of the original circulating neural network [12]. The Neural Network Language Model (NNLM) and the Bidirectional Long-Term Memory Network (BiLSTM) are used for automatic multi-label emotion recognition of text.

Because natural language is a more abstract and advanced cognitive process of human beings, the semantic information contained is more complicated. Some textual information has multiple semantic information that is close to emotions. In the process of data set processing, we only use hard text statements. Single-label mood tagging, there may be cases where the semantic information of the text cannot be completely covered. The next step is to perform multi-label proportional labeling on the semantic information of the text, combined with an effective deep neural network model and other more advanced natural language processing techniques for achieving deeper semantic information mining of text emotions.

References

[1] Darwin, C. (2007) [1872]. The expression of the emotions in man and animals. New York: Filiquarian. ISBN 0-8014-1990-5

[2] Ekman, P; Friesen, W (1971). "Constants across cultures in the face and emotion". Journal of Personality and Social Psychology. 17 (2): 124–9.

[3] Bao Yi. Text sentiment analysis based on deep neural network [D], 2017.

[4] Chen Junqing, Zhang Wei. Text Emotion Analysis of Neural Network Based on Language Model [J]. Modern Computer (Professional Edition), 2018(7).

[5] Bostan L A M, Klinger R. An Analysis of Annotated Corpora for Emotion Classification in Text[C]//Proceedings of the 27th International Conference on Computational Linguistics. 2018: 2104-2119.

[6] Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg; Dean, Jeffrey (2013). "Distributed Representations of Words and Phrases and their Compositionality". arXiv:1310.4546

[7] Lebret, Rémi; Collobert, Ronan (2013). "Word Emdeddings through Hellinger PCA". Conference of the European Chapter of the Association for Computational Linguistics (EACL). 2014. arXiv:1312.5542

[8] Globerson, Amir (2007). "Euclidean Embedding of Co-occurrence Data". Journal of Machine Learning Research.

[9] Levy, Omer; Goldberg, Yoav (2014). Linguistic Regularities in Sparse and Explicit Word Representations . CoNLL. pp. 171–180.

[10] Socher, Richard; Perelygin, Alex; Wu, Jean; Chuang, Jason; Manning, Chris; Ng, Andrew; Potts, Chris (2013). Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. EMNLP.

[11] Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural computation, 1997, 9(8): 1735-1780.

[12] Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model [J]. Journal of machine learning research, 2003, 3(Feb): 1137-1155.