TEXT EMOTION DETECTION BASED ON LSTM

Zhuang Zuo, Rui Xia, Ruiqi Zhong

Electrical and Computer Engineering Department,
University of Illinois at Urbana-Champaign

ABSTRACT

In this Project, the model of Long-short Term Memory model was applied in Emotion Detection from English Text.

The results from experiments

***Key words*—** Emotion, LSTM, Word Embedding, Memory

1. BACKGROUND SECTION

Emotions and feelings have been a bottle-neck in artificial intelligence research for long. The first step for AI to generate feelings and emotions is to let them detect and feel emotions. Though there are several sentiment analysis researches, none of them can be considered as real sentiment since they can only tell positive from negative, but can not recognize real emotions such as joy, sad, anger, etc. Therefore, our project is aimed at implementing a model to recognize six basic Eckman emotions proposed in Steven Handel's "Classification of Emotions"[1]: joy, sadness, surprise, anger, disgust, fear, conveyed by texts collected from Twitter based on LSTM[2].

LSTM is a recurrent network architecture introduced in Sepp Hochreiter and Jurgen Schmidhuber's paper "Long Short-Term Memory"[2] and solves complex and long time lag tasks in pattern recognition field efficiently. LSTM can learn to bridge minimal time lags in long discrete time steps by enforcing constant error flow through "constant error carrousels" within special units, provided that truncated back prop cuts off error flow trying to leak out of memory cells[1]. Two gate units learn to open and close access to error flow within the constant error carrousel of each memory cell. The multiplicative input gate affords protection of the constant error carrousel from perturbation by irrelevant inputs. Similarly, the multiplicative output gate protects other units from perturbation by currently irrelevant memory contents. The detailed implementation of our LSTM for text emotion detection will be discussed in the following sections.

2. METHOD SECTION

Our fundamental methods can be summarized into four sections: data processing, word embedding, LSTM and feature reduction. First, the raw data was processed and labeled. Second, do word embedding on the processed data to code each word in sentence as feature with 100 dimensions. Then, model of feature reduction and LSTM will be experimented on the processed data.

The flow chart is shown in Figure1.

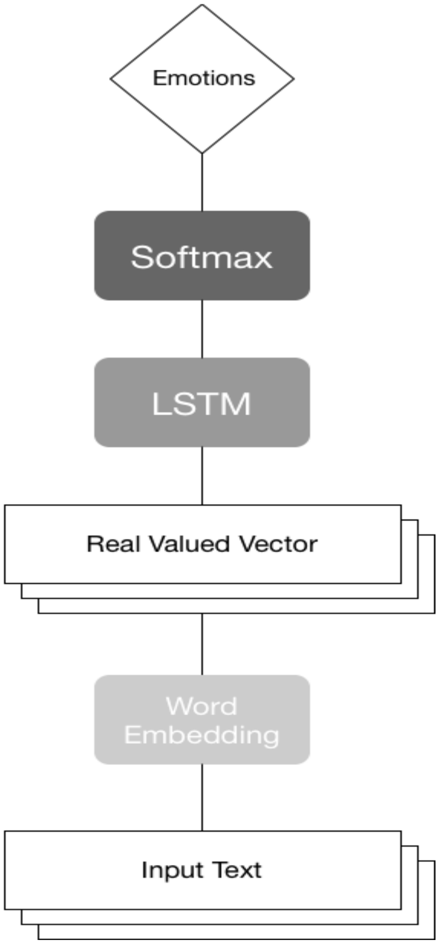

Figure 2.1 Flow chart of Methods

2.1 Data Processing

In this part, we use the split method named TweetTokenizer from Natural Language Toolkit to split lines from original data, which are collected from labeled Affective Text, which is a data set consisting of 1200 dataset, each of them annotated with the six Eckman emotions, and unlabeled

Twitter Dataset--a dataset for sentimental analysis, which is only labeled as positive or negative. We manually labeled Twitter set with the six basic Eckman emotions: joy, sadness, surprise, anger, disgust, fear. Then, all the data are sorted to make them sentences consisting of texts and labels. Then, the sentences are processed through word embedding model to shape into tensors to fit into the x and y variables in the LSTM neural network.

Six one-hot vectors stand for six types of emotions.
According to the convention of split of file, we put the first 60 percent of lines into the training data set for training purpose, the next 20 percent of lines into the evaluating data set for, the last 20 percent of lines into the testing data set.

To obtain the feature space, read every word in each sentence to a list and find the largest length of sentences (60). Adding zeros to each sentence, to make the length of feature of each data consistent.

2.2 Word Embedding

The Word2vec is a highly efficient tool for natural language processing which is released by Google in 2013. It maps the word to a real vector domain using deep-learning model. The Word2vec is a three-layer neural network which contains the input layer, hidden layer and the output layer. The theory of the Word2vec function mainly involves statistical language model including N-gram model, neural network language model, continuous bag-of-words model and continuous skip-gram model. The neural network language model(NNLM) is constructed to build the language model. In the NNLM, the model uses Eigen vectors to represent the features of every word. After the word is transferred to a vector, it corresponds to a dot in the feature space. At the same time, the feature dimensions of every word is less than the total number of words in the vocabulary. The basis of NNLM is a joint probability expressed as follows:

$$f(w_t, \dots, w_{t-n+1}) = P(w_t | w_1^{t-1})$$

And for arbitrary w_1^{t-1},

$$\sum_{i=1}^{|V|} f(i, w_{t-1}, \dots, w_{t-n+1}) = 1$$

$$f > 0$$

The model can be further resolved. In the dictionary V, C is a function mapping from word i to its vector C(i), C is a matrix whose dimension is |V|*m. The probability function among words can be represented as C.

$$f(i, w_{t-1}, \dots, w_{t-n+1}) = g\big(i, C(w_{t-1}), \dots, C(w_{t-n+1})\big)$$

$$P(w_t = i | w_1^{t-1})$$

$$i\ th\ output = P(w_t = i | context).$$

The parameter in the neural network is $\theta = (C, w)$. The second parameter is $g(w)$. The purpose of training is to maximize the likelihood function:

$$L = \frac{1}{T}\sum_t \log f(w_t, w_{t-1}, \dots, w_{t-n+1}; \theta) + R(\theta).$$

For the output layer, the softmax function is applied.

$$P(w_t | w_{t-1}, \dots, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

$$y = b + Wx + U\tanh(d + Hx)$$

$$\theta = (C, b, d, W, U, H)$$

The parameters that need to be trained is θ. So the gradient descent method is used to derive θ.

In our model, the word embedding is used as a model to handle all texts. After the data is processed, the Word2Vec function in gensim package is applied to convert the words in texts of the sentences into vectors which are of 100 dimensions. That is, all words are mapped into a real vector domain, which will bring the convenience that the similarity of words will be represented as the distance in the vector domain. So, every word can be regarded as an input of 100 dimensions. In addition, zeros added to regularize the feature space will be converted to 100 zeros for each as well.

So, after doing processing and word embedding, the data are prepared for training and testing:
Features Space: $\{0,1\}^{100\times 60}$
Label Space: $\{0,1\}^6$

2.3 LSTM (Long Short-Term Memory)

The LSTM network has its advantage over other neural networks. It is especially efficient when the magnitude of weights in the transposition matrix is very large, it can also efficiently avoid vanishing gradients and performs very well in learning long-term dependencies in data. The key structure of LSTM neural network is called a memory cell, which is composed of four main elements: an input gate, a neuron with a self-recurrent connection(a connection to itself), a forget gate and an output gate. The self-recurrent connection has a weight of 1.0 and ensures that, barring any ourside interference, the state of a memory cell can remain constant from one step to another. The gate is set to modulate the interactions between the memory cell itself and its environment. The input gate allows the state of the memory cell to have an effect on other neurons or prevent it. At last, the forget gate can modulate the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state as needed. We believe that the LSTM network will perfrom well because it can remember the relationship between words and phrases in a text.

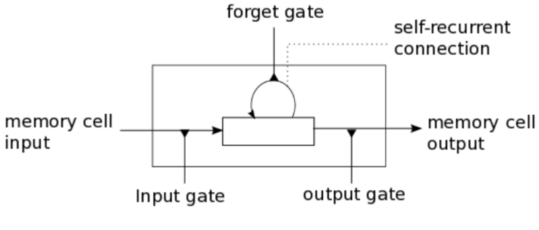

Figure 2.2 LSTM

The equations below describe how a layer of memory cells is updated at every time step t. In these equations:

1. x_t is the input to the memory cell layer at time t.
2. $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$ and V_o are weight matrices
3. b_i,b_f, b_c and b_o are bias vectors

First, compute the values for f_t, the activation of the memory cell's forget gates at time t:

$$f_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

Given the value of the input gate activation i_t, the forget gate activation f_t and the candidate state value $\widetilde{C_t}$, we can compute C_t the memory cell's new state at time t:

$$C_t = i_t * \widetilde{C_t} + f_t * C_{t-1}$$

With the new state of the memory cells, we can compute the value of their output gates and, subsequently, their outputs:

In the LSTM model, we regard each text in a sentence as x variable, which is of 60 time steps and 100 input features. Actually, 60 is the maximal length of a text and 100 is the dimension of a word vector. The label for a text is the y variable, which is converted to a one-hot vector for the convenience of calculation. Every time the LSTM network will remember a word and at the next time step, it moves to another word of the text. To reduce the complexity of OUR our That is, in our model, we take advantage of LSTM's 'remember' property to analysis the relationship between the combinations of different words and sentences (in the form of real valued vectors) and emotions to express.

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

In the LSTM model, we regard each text in a sentence as x variable, which is of 60 time steps and 100 input features. Actually, 60 is the maximal length of a text and 100 is the dimension of a word vector. The label for a text is the y variable, which is converted to a one-hot vector for the convenience of calculation. Everytime the LSTM network will remember a word and at the next timestep, it moves to another word of the text. To reduce the complexity of That is, in our model, we take advantage of LSTM's 'remember' property to analysis the relationship between the combinations of different words and sentences (in the form of real valued vectors) and emotions to express.

2.4 Feature Reduction

To increase the accuracy of our model, a model for feature reduction is built to find the least representative words for all types of emotions. The most representative words for each type of emotion are also found for reference. Our target is to remove the least representative words in texts to reduce the working load and increase the testing accuracy of our model. When choosing the words that are to be removed, we created a stop word list which is consisted of words that are sorted to be least representative. The most representative words are removed from the words in the stop word list. The models of feature reduction will be discussed in the following paragraph. Dictionaries are created for texts of each type of emotion. To find the most representative words for each type, we calculate the frequency number of every word showing in that emotion type texts over the total occurrence of the word in the common dictionary, which is a dictionary that consists of all words in the six type emotion texts. The above-mentioned dictionaries for each type of text is sorted according to the occurring probability of the words in that dictionary. To make the sorting for the most representative words more meaningful, we only choose the words whose occurrence time is among the occurrence time of the first 1/12 words with the highest occurrence times.

To find the least representative words, the following two aspects are taken into our consideration: 1. the occurrence of every word in the common dictionary over the sum of occurring times of all words in the common dictionary, which is called occurrence probability. 2. for every word in the common dictionary, the variance among its six frequencies, which stand for the occurring times of that word in a certain type of emotion texts over the total number of occurrence of that word in all the texts. The words in the common dictionary are sorted according to the product of the negative logarithm value of the occurrence probability and the variance. The negative logarithm function is applied to amplify the influence of the occurrence probability. That is, the least representative words should also occur quite often to be selected.

Then, when the list for the words to be removed is being created, the words in the least representative list should be selected except for that if it occurs in the most representative word list, too.

In a word, through our feature reduction model, the number of words that are to be converted to word vectors is reduced. The relatively representative words remain.

3. EXPERIMENT SECTION

3.1 Description of Experiments

3.1.1 Tuning Parameters

At the beginning, we have tried experiments on different learning rate, iteration number and batch size. The results showed that the parameters have light effects on the results. We chose the best set of parameters, which are shown in Table 3.1.

Parameters	Learning Rate	#Iteration	Batch Size
Values	0.002	2,000,000	100

Table3.1 The Choice of Parameters

3.2.2 Tuning the Model

Then, we fixed the parameters as the best ones from previous experiments and pay attention on different model structures.

In conclusion, we have mainly done 5 group of experiments to find the best model:
A. Word embedding+ Deleting stop words + 1 layer LSTM
B. Word embedding+ Containing stop words + 1 layer LSTM
C. Word embedding+ Deleting stop words + 3 layer LSTM
D. Word embedding+ Containing stop words + 3 layer LSTM
E. Word embedding+ Containing stop words + 1 layer LSTM +Softmax Layer

The results form experiments of tuning parameters

3.2 Results

3.2.1 Accuracy

By observation, the training accuracy will converge after iterating for 2,000,000 times, which will take around 5 hours.

Experiment	A	B	C	D	E
Train Accuracy	67%	91%			31%
Test Accuracy		48%			27%

Table 3.2 Accuracy after 2,000,000 iterations

3.2.2 Learning Curve

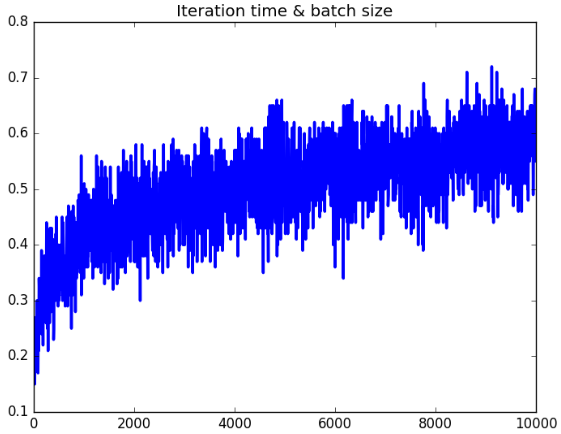

Figure 3.1 Learning Curve for A

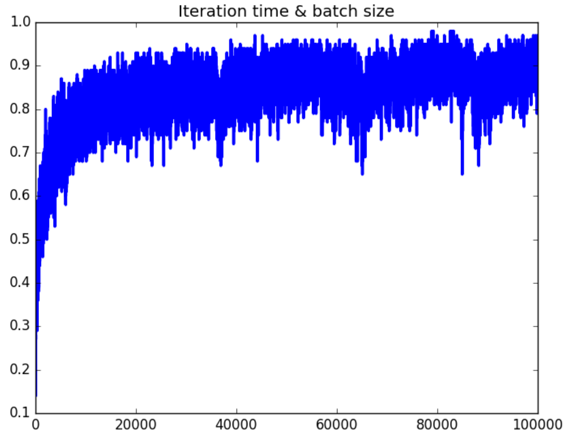

Figure 3.2 Learning Curve for B

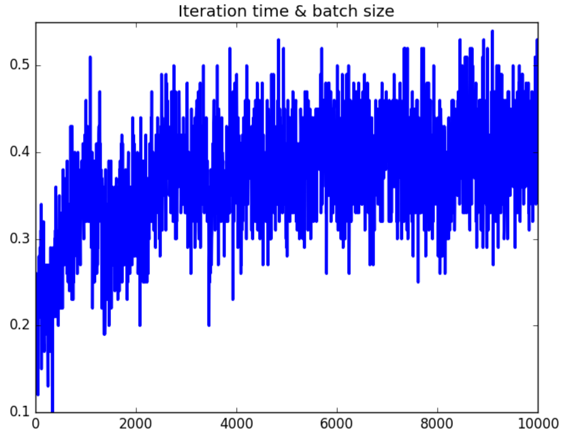

Figure 3.3 Learning Curve for C

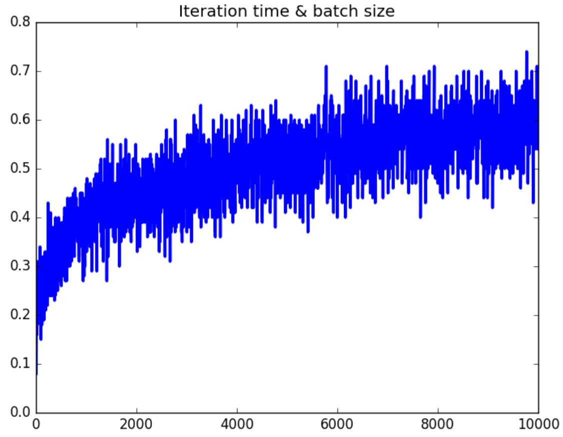

Figure 3.4 Learning Curve for D

Figure 3.5 Learning Curve for E

As shown above, the best results shown on Experiments B, in which we applied word embedded, contained the stop words and use a one layer LSTM. The best cur is shown as follows:

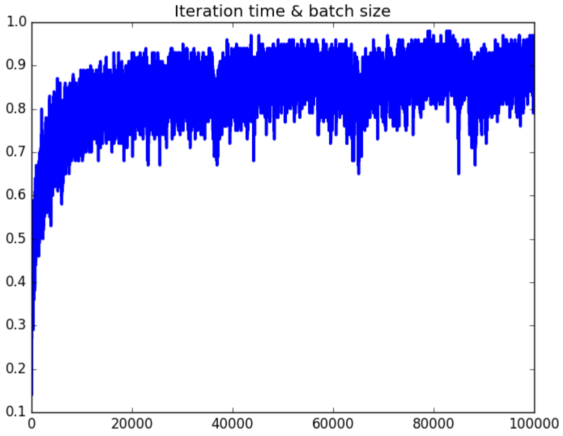

Figure 3.6 Best Learning Curve

3.2.3 Confusion Matrix (Best Model)

4. CONCLUSION AND FUTURE WORK

4.1 Conclusion and Analysis

In conclusion, the best models applied in our experiments can reach a 90 percent accuracy on training data and successfully classify half of the emotions from test datasets. So, the memory ability of LSTM Model and the application of word embedding make improvements in finding the relations between words and analyze the emotions well.

In addition, the decrease of accuracy after doing feature reduction and deleting indecisive 'stopping words' implies that the more important element in LSTM Model is the relations between words or features instead of whether the feature is decisive. In other words, for a model with dependency on long time memory, the logistic relations between features will be more decisive than individual feature. So, LSTM model is appropriate for solving problems similar with text emotion detection, whose features, like words in sentence, will have logic relations with each other.

However, the results of testing didn't reach as high as our expected compared to the accuracy shown on training data. The possible reasons for it and some possible solving methods are analyzed as follows:
First, the lack of previous work on this area lead to the lack of available data. In addition, so much work needed to be done on finding and processing the raw data.
Second, most of the data were labeled by ourselves, which means the different understanding of sentences will lead to some errors before the experiments. For example, the similar sentences could be labeled as different classes. So, in such limited time, our database is not complete enough to obtain a higher accuracy which could be reached in future works.
Third, our database is mainly from Twitter and Facebook, where people always use some simple words and symbols without attention on the structure of sentence to express their emotion. As a result, the effect of linkage of a whole sentence was weakened, which lead to the reduction of advantages from LSTM model.
Moreover, as mentioned in the method description, the model of word embedding also needed a large number of data to train for a better accuracy to detect similar word. As a result, the lack of data will also lead to an inaccurate feature space.

4.2 Future Work

1. Enlarge the database:

The lack of data in our experiments is a crucial problem, so if more datasets could be used in experiments, the results will be more general and convincing.

2. Weight more decisive words

The model in feature reduction also found the most decisive words from each classes of training data. In the future work, the decisive words could be weighted in model to obtain a better performance.

3. Intensify the logic connections

Figure out some methods to intensify the logic connection of a whole sentence to better use of the advantage of LSTM's long term memory. For example, manually add some link word.

5. CODE DESCRIPTION

12. REFERENCES

[1] Steven., Handel. "Classification of Emotions." The Emotion Machine, , 2011. Accessed 7 Dec. 2016. www.theemotionmachine.com/classification-of-emotions/

[2] Sepp Hochreiter and Jurgen Schmidhuber, Long Short-Term Memory. Neural Computation 9(8):1735-1780, 1997.