Attention-based Hierarchical LSTM Model for Document Sentiment Classification

Bo Wang1, * and Binwen Fan2
1, 2Harbin Institute of Technology, Shenzhen, Shenzhen, China
*wb_hitsz@foxmail.com

Abstract. Document sentiment classification is a fundamental task in data mining, contains extensive underlying commercial value. With the development of deep learning, we can extract features in an automatic way, instead of design it by oneself. Which can help us use semantic information to classify the document in a better way. Base that, in this paper, we present a hierarchical network structure according to the structure in real document. Based on LSTM to encode semantic information; then combine with attention mechanism to improve the accuracy of classification. And last, conduct experiment on two dataset, analyse the accuracy result of different model, and study some tricks in parameter selection.

1. Introduction
Document sentiment classification [1] is used to analyse public opinion towards commodity, current event and so on, which is an important task in field of Nature Languages Processing. For this task, dominant approaches exploit machine learning algorithm to build document classifier[2], which need researchers to design hand-crafted features[3, 4]. With the development of deep-learning approaches, many methods appears such as recurrent neural network [5] and convolutional neural network [6] to do this work. Recently years, attention mechanism [7] appears, and achieves state-of-art in many different fields.

In this paper, we will introduce an attention-based hierarchical bidirectional long short-term memory (LSTM) to achieve the task of document sentiment classification. Make use of information of document structure [8, 9] in review document, combine attention mechanism. And we will demonstrate the effectiveness of hierarchical network structure and attention mechanism in task of document sentiment classification.

2. Model

2.1. Long-short term memory network
Long short-term memory (LSTM) [10] has been widely replace the recurrent neural network (RNN). The LSTM uses gate mechanism to update information of memory cell. There are three types of gates, the forget gate \( f_t \), the input gate \( i_t \), and output gate \( o_t \). They control how to update the cell state together. The architecture of LSTM is shown in Figure 1.

At time \( t \), we need to compute how much past information would be forget with previous hidden state \( h_{t-1} \) and current input information \( x_t \). Forget gate \( f_t \) is computed by:
Figure 1. The architecture of LSTM
\[ f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \] (1)

Where \( \sigma \) denote the sigmoid activation function. Then use input gate and candidate cell state to get the new cell state: \( x_{t-1}, x_{t+1} \)

\[ i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \] (2)
\[ \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \] (3)
\[ C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \] (4)

At last, output the cell state and hidden state of current unit. Cell state has been computed before, and now use output gate which control how much information will be output to hidden state to decide the hidden state, the output gate and hidden state are updated as:

\[ o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \] (5)
\[ h_t = o_t \cdot \tanh(c_t) \] (6)

2.2. Attention based hierarchical LSTM
In this part, we will focus on document structure. We divide a document into word-level and sentence-level two part, use LSTM compute the sentence representation and document representation separately. The structure viewed in Figure 2 below.

Figure 2. The architecture of proposed model
2.3. Word-level
Assume that there are M sentences in a document, and N words for each sentence. We use \( s_m \), \( m \in [1, M] \) to denote sentences of document. And \( w_{mn} \), \( n \in [1, N] \) to denote words in sentence \( m \).

Now we will present how to build the model to represent document vector from the raw corpus.

Firstly, we need to represent the words which is the least unit in a document in a vector representation by the embedding matrix \( W_e \), so we get input vector of word-level encoder. We use the bidirectional LSTM to encoder words information in both forward and backward direction, which could cover the contextual information more effectively. And we use \( h_{mn} \) (the brief representation of hidden state) to represent both forward and backward direction encoder information \([\bar{h}_{mn}, \tilde{h}_{mn}]\), which represent the whole sentence information of \( s_m \) centres on \( w_{mn} \).

By now, we have get the sentence vector representation, every \( h_{mn} \) can be used to represent the information of \( s_m \). Generally speaking, the hidden state \( h_{mn} \) of the last word or mean vector of all words will be used to represent the sentence information. In our model, we will introduce the attention mechanism to extract more important information in all hidden state.

\[
\begin{align*}
    u_{mn} &= \tanh(W_e h_{mn} + b_e) \\
    \alpha_{mn} &= \frac{\exp(u_{mn}^T u_m)}{\sum_n \exp(u_{mn}^T u_m)} \\
    s_m &= \sum_n \alpha_{mn} h_{mn}
\end{align*}
\]

In word attention, we introduce \( u_{mn} \) and \( \alpha_{mn} \) to represent the hidden representation of \( h_{mn} \) and importance of each hidden representation. Firstly, we feed \( h_{mn} \) into a one-layer MLP to get hidden representation \( u_{mn} \), then use a softmax layer to compute the importance of \( u_{mn} \). And last we combine \( u_{mn} \) and \( \alpha_{mn} \) to get the sentence representation \( s_m \).

2.4. Sentence-level
We have get sentence representation according to above process. Then we will get document representation in a similar way. Firstly we use \( h_m \) to represent forward and backward direction encoder information \([\bar{h}_m, \tilde{h}_m]\). After to get more important sentence information, we introduce attention mechanism in sentence-level continue.

\[
\begin{align*}
    u_m &= \tanh(W_m h_m + b_m) \\
    \alpha_m &= \frac{\exp(u_m^T u_m)}{\sum_m \exp(u_m^T u_m)} \\
    d &= \sum_m \alpha_m h_m
\end{align*}
\]

This time we use \( d \) to represent document representation. And introduce \( u_m \) and \( \alpha_m \) represent the hidden representation and importance.

From the above, we have got the representation of document, which can be regard as feature vector of document. And now we will add a linear layer to transform the feature vector into another vector \( v \) whose length is the number of class, followed:

\[
v = W_v d + b_d
\]
Then add a softmax layer to convert $v$ to a probability value:

$$p = \frac{\exp(v)}{\sum \exp(v)} \quad (14)$$

In training process, we use cross-entropy loss function compute training loss:

$$\text{loss} = -\sum_i \log(p_{ij}) \quad (15)$$

And $j$ is the real label of document $i$.

3. Experimental and analysis

3.1. Dataset
For test the effectiveness of our model, we choose two datasets as experiment data, Yelp and IMDB. The statistics of datasets shown in Table 1. The Yelp is an unbalance dataset, and have five levels from 1 to 5, higher is better. The IMDB is a balance dataset have positive and negative two classes.

| dataset | documents | class | length/mean | vocabulary |
|---------|-----------|-------|-------------|------------|
| Yelp    | 230907    | 5     | 151.2       | 204982     |
| IMDB    | 50000     | 2     | 349.2       | 149393     |

All datasets divided into train data and test data with 9:1 randomly.

3.2. Hyper parameter setting
For raw document, we first use NLTK tools to split sentences and replace low-frequency word appear less than five times, after use Word2vec tool to train our word embedding as the initial value of word vector.

We set hyper parameters after some tests. In experience, we set dimension of word embedding is 200, the hidden state of LSTM unit is 64 and the activation function of LSTM is tanh. We set the max length of document is 100 and max number of sentence is 10. We use Adam gradient descent algorithm as optimizer. We use mini-batch to train our model, and set mini-batch size is 64. We use early-stopping strategy to avoid over-fitting.

3.3. Results and analysis

![Figure 3. Model training curve](image)

We present the results of accuracy between different models in Table 2 and the training curve is depict in Figure 3. From the Table 2, our hierarchical attention model achieves better result than hierarchical model and LSTM attention model. Compare with use LSTM only, add both an attention model and a hierarchical model will contribute to make a little improvement for classification accuracy. And from
the training curve we can see, model with attention mechanism will convergence faster. Besides, when training process get to third epoch, the accuracy begin to drop, that is because the over-fitting of model, so we introduce early-stopping strategy, when monitor training get into over-fitting phase, stop the training process ahead of time to avoid over-fitting.

### Table 2. Accuracy for experiment in different model

| model/dataset | Yelp  | IMDB  |
|---------------|-------|-------|
| LSTM          | 59.91 | 89.30 |
| LSTM-AM       | 62.80 | 90.35 |
| Hi-LSTM       | 62.56 | 90.70 |
| Hi-LSTM-AM    | **64.44** | **91.51** |

#### 3.4. Word embedding dimension

In this experiment, we fixed number of hidden units as 64, choose the tanh as activation function to test the effect of word embedding. We present results in Table 3.

### Table 3. Experiment of word embedding dimension

| Database/embed dim | 50   | 100  | 200  | 300  |
|--------------------|------|------|------|------|
| Yelp               | 64.13| 64.38| **64.44** | 64.37 |
| IMDB               | 91.16| 91.35| **91.51** | 91.43 |

The dimension of word embedding represent the semantic features of the word, generally speaking, the more features, the better efficiency of experiment have. However, from the table, when embedding dimension increase from 50 to 200, the accuracy of classification have some growth. But when embedding dimension increase to 300, the accuracy go down. That is because a large number of features will contribute to distinguish different words in some degree, but it will reduce the character of features.

#### 3.5. Number of hidden units

This time we will study the effect of number of hidden units, word embedding dimension fixed as 200 and activation function choose tanh. As we know, the hidden units is the storage of long term information. So as we can see results from the Table 4, the accuracy will increase follows the increase of number of hidden units. But the speed of increase will be slow.

### Table 4. Experiment of number of hidden units

| Database/number hidden | 32  | 64  | 128 | 256 |
|------------------------|-----|-----|-----|-----|
| Yelp                   | 63.28| 64.44| 64.63| **64.66** |
| IMDB                   | 90.90| 91.51| 91.65| **91.75** |

#### 3.6. Activation function

![Activation Functions](attachment:image.png)

At last, let us study the effect of different activation function. We used to use sigmoid and tanh as activation function. Recent years, an activation function named Relu appears, which have the advantages of simplify computation and prevent vanish of gradient in positive interval. This time we fixed word embedding dimension as 200 and number of hidden is 128. The Figure 4 shows different activation function curves.
Table 5. Experiment of activation function

| database/activation function | sigmoid | tanh  | Relu  |
|------------------------------|---------|-------|-------|
| Yelp                         | 61.94   | 64.63 | 65.25 |
| IMDB                         | 90.15   | 91.65 | 91.90 |

From the Table 5, Relu function achieve best result during three function. Compare with sigmoid function, tanh function solve the zero-mean problem, prevent ‘shift’ during training process. For Relu, the function always keep a gradient in are interval, as for sigmoid and tanh, there are vanishing gradient problem happened when input value too large or too small. But the Relu still need to be improved in negative interval.

4. Summary
In this paper, we present an attention based hierarchical LSTM model. And conduct experiments on two database with classification accuracy to demonstrate the effectiveness of our model. Analysis how word embedding dimension, number of hidden units and activation function effect the classification results, demonstrate the effectiveness of hierarchical network structure and attention mechanism in task of document sentiment classification.

Acknowledgments
This work was supported by Special funds for the development of strategic emerging industries in Shenzhen (no. JCYJ20150625142543448).

References
[1] Pang, Bo, and Lee, Lillian. "Opinion Mining and Sentiment Analysis." Foundations & Trends in Information Retrieval 2.1–2(2008):1-135.
[2] Pang, Bo, L. Lee, and S. Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques," Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, July 2002, pp. 79-86.
[3] Ifrim, Georgiana, and G. Weikum. "The Bag-of-Opinions Method for Review Rating Prediction from Sparse Text Patterns." Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), Beijing, August 2010, pages 913–921.
[4] Paltoglou, Georgios, and M. Thelwall. "A Study of Information Retrieval Weighting Schemes for Sentiment Analysis." Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1386–1395, Uppsala, Sweden, 11-16 July 2010.
[5] Vo D T, Zhang Y. Target-dependent twitter sentiment classification with rich automatic features. International Conference on Artificial Intelligence. AAAI Press, 2015:1347-1353.
[6] Kalchbrenner, Nal, Grefenstette, Edward, Blunsom, Phil. A Convolutional Neural Network for Modelling Sentences. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 655–665, Baltimore, Maryland, USA, June 23-25 2014.
[7] Xu K, Ba J, Kiros R, et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Proceedings of the 32nd International Conference on Machine Learning, 2015, p. 2048-2057.
[8] Yang, Zichao, et al. "Hierarchical Attention Networks for Document Classification." Proceedings of NAACL-HLT 2016, pages 1480–1489, San Diego, California, June 12-17, 2016.
[9] Tang, Duyu, B. Qin, and T. Liu. "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification." Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1422–1432, Lisbon, Portugal, 17-21 September 2015.
[10] Hochreiter S, and Schmidhuber J. "Long short-term memory." Neural Computation. Vol.9.8(1997): 1735-1780.