Received October 2, 2018, accepted October 14, 2018, date of publication October 29, 2018, date of current version December 18, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2878425

Lexicon-Enhanced LSTM With Attention for General Sentiment Analysis

XIANGHUA FU[1], JINGYING YANG[2], JIANQIANG LI[2], MIN FANG[3], AND HUIHUI WANG[4]

[1]Faculty of Arts and Sciences, Shenzhen Technology University, Shenzhen 518118, China
[2]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
[3]Experimental and Creative Practice Education Center, Harbin Institute of Technology, Shenzhen 518055, China
[4]Department of Engineering, Jacksonville University, Jacksonbville, FL 32211, USA

Corresponding author: Jianqiang Li (lijq@szu.edu.cn)

This work was supported by the National Nature Science Foundation of China under Grant 61472258.

ABSTRACT Long short-term memory networks (LSTMs) have gained good performance in sentiment analysis tasks. The general method is to use LSTMs to combine word embeddings for text representation. However, word embeddings carry more semantic information rather than sentiment information. Only using word embeddings to represent words is inaccurate in sentiment analysis tasks. To solve the problem, we propose a lexicon-enhanced LSTM model. The model first uses sentiment lexicon as an extra information pre-training a word sentiment classifier and then get the sentiment embeddings of words including the words not in the lexicon. Combining the sentiment embedding and its word embedding can make word representation more accurate. Furthermore, we define a new method to find the attention vector in general sentiment analysis without a target that can improve the LSTM ability in capturing global sentiment information. The results of experiments on English and Chinese datasets show that our models have comparative or better results than the existing models.

INDEX TERMS Sentiment lexicon, sentiment embedding, word embedding, attention vector, sentiment analysis.

I. INTRODUCTION

Sentiment analysis aims to accurately extract people's opinions from a large number of unstructured review texts and classifying them into sentiment classes, positive or negative, or more fine-grained, very negative, negative, neutral, positive and very positive. General sentiment analysis without a target [18], calls for identifying the overall sentiment polarity of the whole text; the other sentiment analysis task, called target-dependent or aspect-level sentiment analysis [21], [22], calls for identifying the sentiment of an entity or an aspect. In our paper, we focus on the general sentiment analysis without the target. There are two general methods of sentiment analysis. One is a rule-based approach [28], [29]. The other is machine learning [30]–[33], including methods for deep learning [2], [7]. Rule-based methods pay more attention to adjectives and nouns in texts. These methods mainly rely on artificially sorted sentiment lexicons. Each word in the sentiment dictionary corresponds to an emotional tendency value. For a given text, the sentiment orientation values of multiple words in the text are combined into one according to the set rules, as the emotional tendency value of the whole text. The traditional machine learning methods mainly use naive Bayes, support vector machine (Support Vector Machines, referred to as SVMs), and maximum entropy classifiers, etc., to build sentiment analysis, but these methods require manual selection of features or design of methods to select features. Therefore, the quality of sentiment analysis results depends on the characteristics of manual selection.

In recent years, the wide application of deep learning has led to the further development of image processing and speech recognition. Due to its characteristic of automatic learning features, deep learning method has been widely used in natural language processing [3], [9], including sentiment analysis. Previous machine learning methods generally use a one-hot vector representation, the dimension of the vector increases with the increase of the text data to be processed, which is easy to cause dimensionality disaster and cannot capture the relationship between words and words. Word2vec [11], [23], [24] and Glove[13] can convert words into high-quality word embeddings representations from a large number of unlabeled texts through co-occurrence between words and words, and can capture the grammatical and semantic information of words. Nowadays, word embeddings are first used to represent a single word, and then are used to represent neural networks, such as Recurrent Neural Networks (RNN), Convolutional Neural Network (CNN), etc., models based on word vectors and deep neural networks have been extensively studied. Finally, it is a mainstream research method to automatically extract emotion-related features for analysis in the process of training models. Among them, RNN is widely used in natural language processing due to its ability to naturally model sequences. Long Short-Term Memory Network (LSTM) [6] is the most widely used. The way that use LSTMs or CNNs to combine word embeddings for text representation has become a new baseline for sentiment analysis. However, sentiment analysis of texts is still a challenge. On one hand,we argue that word embeddings are not accurate enough if directly applied to sentiment analysis tasks. The most serious problem of word embedding learning algorithms is that they only model the contexts of words but ignore the sentiment information [20]. As a result, words with opposite polarities, such as "good" and "bad," are mapped into close vectors in the embedding space. Sentiment lexicon is used as an important feature for sentiment analysis in traditional methods [4]. If we only use the word embeddings as inputs, can we capture the whole sentiment lexicon's information? Making the sentiment lexicon as the extra information is reasonable. On the other hand, LSTMs have the bias of preferring recent inputs. It's hard to capture the important information of the beginning part of text within a long sequence.

In order to solve the first problem, we propose a lexicon-enhanced LSTM model (LE-LSTM) to integrate sentiment lexicon into LSTM to capture more sentiment information of words. First, we use sentiment lexicon as the extra information to pre-train a word sentiment classifier. And then each word can get its sentiment embedding including the words not in sentiment lexicon. During the main training process, we concatenate the word embedding and its sentiment embedding as the input of LSTM and fine-tune the word sentiment classifier network.

In order to solve the second problem, we define a new method to calculate the attention vector in general sentiment analysis without a target and take two special circumstances as examples. In these two examples, the attention vector can be roughly considered as the global information of the whole text. So it can guide LSTM to capture the distinguishing parts of the text just like that people first get the rough idea of texts and then look for the important parts when they do the reading comprehension task.

To summarize, our contributions lie in the following three points:

1) We propose a lexicon-enhanced LSTM model for sentiment analysis, which use the prior sentiment information of words as the supplemental information to improve the quality of word representation.
2) In order to improve the semantic composition ability of LSTMs, we define a new method to calculate the attention vector in general sentiment analysis without a target and take two special circumstances as examples.
3) To verify our methods, we conduct experiments on several English datasets and Chinese datasets. The results of experiments show that our methods are effective. Furthermore, we find that LSTMs have strong ability in handling short text sequences. The attention mechanism that we use only works on the long text sequences. The analysis experiments also prove the result.

II. RELATED WORKS

Works which use deep learning method to solve sentiment analysis tasks can be divided into two directions. One is the research of word embedding. The other is the research of semantic compositions. In our work, we introduce the sentiment lexicon to improve the quality of word embedding and use the attention mechanism to improve semantic compositions.

A. WORD EMBEDDINGS

High-quality word embedding is the basis of the natural language processing task. In terms of improving the quality of word vectors, Mikolov *et al.* [9] and Pennington [13] first discovered that the word vectors learned through a RNN have interesting linear substructures conducted a comprehensive training of the global word-word co-occurrence of statistical data from the corpus and the resulting global vector (GloVe) shows interesting linear substructure in word vector space like [11]. Maas *et al.* [10] learned the word embeddings with sentiment based on the classical neuro-probabilistic language model [10]. Tang *et al.* [25] proposed three models, considering the emotional tendency of text, and learning the word embeddings with sentiment. In our experiments, we also use the GloVe to get initial word embeddings. We first use sentiment lexicon as the extra information to pre-train a word sentiment classifier and then get the sentiment embedding of each word including the words not in sentiment lexicon. Concatenating the word embedding and its sentiment embedding can improve the quality of word embeddings.

B. SEMANTIC COMPOSITIONS

However, the word embeddings can only represent a single word. For a phrase or an article, it is necessary to consider how to combine individual word embeddings into a representation of a phrase or an article. This requires consideration of the problem of semantic composition. Socher *et al.* [16], [26] use Recursive Auto-encoders (RAE) to obtain a sequence representation of the entire text through a continuous recursive combination [16], [26]. But in order to get a sequence representation, firstly, we need to parse the sentence structure, or use a greedy strategy, combining and refactoring the words with the least error each recursive. Kim [7] propose a CNN architecture for text categorization, which has multiple filters of different sizes and two different channels. Cho *et al.* [2] use RNN to process text into a sequence of words, taking into account the order of the words in the text. The RNN [6] with LSTM unit is explicitly designed to avoid gradient disappearance and has better results than RNN, such as bidirectional LSTM (referred to as Bi-LSTM), Tree-Structured Long Short-Term Memory Networks (Referred to as Tree-LSTM) [17] and Nested LSTMs (referred to as NLSTM) [27].

Because the word embeddings can only represent a single word, sentence and document representations need to consider the semantic combination problem. Semantic combination models include RAE [16], CNN [7] and RNN [2]. Among all these models, RNN with LSTM unit [6] is widely used in text sequence processing, because of the ability in modeling long sequential inputs or outputs. It is explicitly designed for avoiding the gradient vanishing problem.

Above semantic composition methods consider that each input has the same importance. But when you look at a picture, you always focus on a certain point, and only assign less attention to other areas. Attention mechanism [1] is designed to solve the problem. That the attention mechanism [1] combined with LSTM or CNN can be considered as another effective compositional function. The function is selected to decide ''where to look'' by assigning weights or importance to each components. Bahdanau *et al.* [1] applied the attention mechanism to machine translation for the first time. Subsequently, attention mechanisms have also been widely applied to specific target (aspect) sentiment analysis tasks. Wang *et al.* [21] propose the LSTM model with attention mechanism for the sentiment analysis of specific aspects. Yang *et al.* [22] proposed an LSTM model based on the attention mechanism for the sentiment analysis of feature targets. The attention mechanism assigns weights to each input word based on a particular aspect (goal). However, in the sentiment analysis of non-specific aspects (targets), attention mechanisms are rarely used. Even for general sentiment classification tasks, the importance of each word is different. But it rarely used in general sentiment analysis without target because of the difficulty in determining the attention vector. In our paper, we systematically define a method to find the attention vector in general sentiment analysis without target and take two special circumstances as examples. Reference [14] also models the linguistic role of sentiment word in sentiment analysis tasks by adding regularizer to the loss function. But it can't handle the sentiment word not in sentiment lexicon and don't consider the different importance of words.

III. THE MODELS

In this study, we propose a lexicon-enhanced LSTM model (LE-LSTM) to introduce sentiment lexicon into LSTM (subsection A). Also, we introduce the attention mechanism into general sentiment analysis without a target. It can capture the distinguishing parts of the text according to the defined attention vector (subsection B).

A. LEXICON-ENHANCED LSTM MODEL

Given a word embedding lookup L ($L \in R^{d\times|V|}$, initialized by word vector file) and one-hot vector e (of length $|V|$), where d is the dimension of word embedding and $|V|$ is the vocabulary size, we can get a word embedding: $x_t = Le_t$. Given a sentiment lexicon, we can pre-train a word sentiment classifier. According to the quality of sentiment lexicon, we can design how many layers the sentiment classifier has and how many categories each word can be classified into different categories. In FIGURE 1 and the following equations, we just use one layer as for a example. The final layer output of word sentiment classifier can be considered as the sentiment embedding of each word. The word sentiment classifier and word embedding lookup can be updated during the main training process.

We concatenate the word sentiment embedding, denoted as s_t, and its corresponding word embedding, denoted as x_t. The final input of LSTM is $w_t = s_t \oplus x_t$. Other steps are the same as the basic LSTM unit. A detailed structure of the basic LSTM unit can be seen from [6]. We describe the LE-LSTM model in FIGURE 1 and the following equations. y_t^s represents the corresponding sentiment label of the word at t time step. According to the sentiment score of words in sentiment lexicon, we divide the words into different categories.

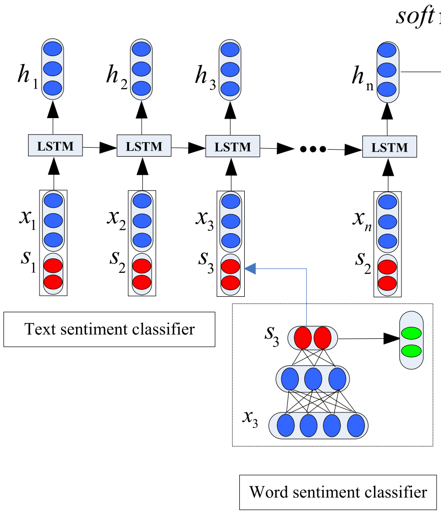

FIGURE 1. We propose a LE-LSTM model. We concatenate the word sentiment embedding, and its corresponding word embedding. The final layer output of word sentiment classifier can be considered as the sentiment embedding of each word. The word sentiment classifier and word embedding lookup can be updated during the main training process.

Word sentiment classifier:

$$s_t = g(W^{(s)}x_t + b^{(s)}) \tag{1}$$

$$p_t^s = soft\max(s_t) \tag{2}$$

$$L_s = -\sum_t y_t^s \log p_t^s \tag{3}$$

LE-LSTM model:

$$i_t = \sigma(W^{(i)}x_t + W_s^{(i)}s_t + U^{(i)}h_{t-1} + b^{(i)}) \tag{4}$$

$$f_t = \sigma(W^{(f)}x_t + W_s^{(f)}s_t + U^{(f)}h_{t-1} + b^{(f)}) \tag{5}$$

$$o_t = \sigma(W^{(o)}x_t + W_s^{(o)}s_t + U^{(o)}h_{t-1} + b^{(o)}) \tag{6}$$

$$u_t = \tanh(W^{(u)}x_t + W_s^{(u)}s_t + U^{(u)}h_{t-1} + b^{(u)}) \tag{7}$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1} \tag{8}$$

$$h_t = o_t \odot \tanh(c_t) \tag{9}$$

B. LEXICON-ENHANCED LSTM MODEL WITH ATTENTION

By using the lexicon-enhanced LSTM model we proposed above, we can get the outputs of each time-step $[h_1, h_2, ..., h_n]$ $(h_i \in R^m)$, where m is the dimension of the hidden outputs and n is the length of text. Unlike the standard LSTM model that uses the final output as the text representation, attention mechanism means combining the hidden outputs of each time step according to the calculated attention weights.

In target-dependent or aspect-level sentiment analysis tasks, a target or an aspect in text is generally used as an attention vector. It is reasonable, just like that people first look for the sentiment target entity, and then focus on the related content. However, it is difficult to choose the attention vector in general sentiment analysis without target. In our paper, we propose a new method to calculate the attention vector T according to the following equation.

$$T = \sum_{i=1}^{n} P_i(x)Q_i(x) \tag{10}$$

In the above equation, $P_i(x)$ $(\sum_{i=1}^{n} P_i(x) = 1)$ represents the sentiment weight of each word in text, which can be calculated according to the relevance or similarity between the word sentiment and the global sentiment of text. For test dataset, we can use the traditional method to get a preliminary global sentiment polarity of text. $Q_i(x)$ represents the semantic information of each word in a text, such as the word embedding or each time-step hidden output of LSTM. The calculated attention vector T carries not only the semantic information but also the sentiment information.

When we have to get the attention vector, we need to consider how to calculate the attention weights according to the attention vector. In recent researches, there are many methods which are proposed to calculate attention weights [1], [22]. We try some of them, and finally, use the method [21] proposed to calculate attention weights.

If we have designed the specific method to calculate the sentiment weight $P_i(x)$ between each word sentiment and the global sentiment, we can calculate the attention vector T according to equation 7. In our paper, we take two special circumstances as examples. These two attention vectors can be roughly considered as the global information of the whole text. So it can guide LSTM to capture the distinguishing parts of the text just like that people first get the rough idea of texts and then look for the important parts when they do the reading comprehension task.

In one setting, we set $P_n = 1$, $P_i = 0(i \neq n)$ and $Q_i(x) = h_i$. We simply denote the model as ALE-LSTM, which uses the final hidden output of LSTM as the attention vector. Reference [15] also uses the method to calculate the attention vector for key term extraction and dialogue act detection. The attention mechanism produces an attention weight vector ∂ according to T and a text representation z by weighting every hidden output. ∂ represents how much contributions that each hidden output makes to the global text sentiment polarity. In the back-propagation process, the loss of model is not only backpropagated to the beginning part from h_n step by step but also backpropagated directly from z.

In the other settings, we set $P_i(x) = \frac{1}{n}$ and $Q_i(x) = x_i$. It uses the average vector of the word embeddings as an attention vector. To the best knowledge of ours, it's the first time to use the average vector of word embeddings as the attention vector. The average vector carries more the original semantic information rather than the information processed by a neural network, so it can more fairly to choose the distinguishing parts of h_i. We simply denote the model as WALE-LSTM. Other steps are the same as the first setting. Detailed information can be found in FIGURE 2, and the following equations.

$$M_i = \tanh(W_h h_i) \oplus tanh(W_v T) \tag{11}$$

$$M = [M_1, M_2, ..., M_n] \tag{12}$$

$$H = [h_1, h_2, ..., h_n] \tag{13}$$

$$\partial = soft\max(W^T M) \tag{14}$$

$$z = H\partial^T \tag{15}$$

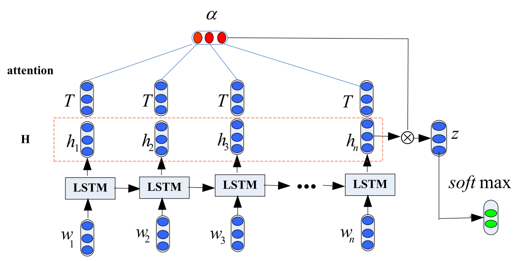

FIGURE 2. The attention mechanism produces a text representation z by weighting every hidden output. In the back-propagation process, the loss of model is not only backpropagated to the beginning part from h_n step by step but also backpropagated directly from z. The calculated attention vector T carries not only the semantic information but also the sentiment information.

In the above equations, $W_h \in R^{m\times m}$, $W_v \in R^{r\times r}$, $M_i \in R^{m+r}$, $M \in R^{(m+r)\times n}$, $H \in R^{m\times n}$, $W \in R^{(m+r)}$, $\partial^T \in R^n$ and $z \in R^m$. r is size of $Q_i(x)$.

Finally, we can get the sentiment distribution of each training text by adding a softmax layer in each final text representation z.

$$p_\theta(y|x) = soft\max(w^l z + b^l) \tag{16}$$

w^l and b^l are the weight and bias parameters, which need to optimize during training. Objective function of the model is cross-entropy error:

$$J_\theta = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p_\theta(y_i|x_i) + \beta\frac{1}{N}\sum_{i=1}^{N} L^s + \lambda||\theta||^2 \quad (17)$$

In the above equations, N is the number of train examples. y is the true sentiment labels of texts. L^s denotes the loss of word sentiment classifier defined in equation (3). β is used for fine-tuning the word sentiment classifier network. The final item is used for avoiding the over-fitting problem. Other parameters are updated by using the 'adam' [8].

IV. EXPERIMENTS

A. DATASET

To verify the proposed models, we conduct experiments on five datasets. Detailed information can be seen from TABLE 1. The first three datasets are English datasets and the last two are Chinese datasets.

TABLE 1. Dataset partition.

Corpus	Classes	Instances	Avg length	Words in lexicon/Total Words	Train/(valid/)test
IMDB	2	50000	257	14924/106249	25000/2500
Yelp2013	5	78966	179	11250/73691	62522/7773/8671
MR	2	10662	20	6752/18758	9662/1000
NB4000	2	4000	42	2257/7594	3600/400
Book4000	2	4000	94	6467/17141	3600/400

- **IMDB** created by [10], is widely used in document sentiment analysis tasks, which includes larger amounts of user reviews and recommendations.
- **Yelp2013** created by [19], are often classified into five classes, very negative, negative, neutral, positive and very positive.
- **MR** created by [12] is an authoritative and commonly used sentiment dataset.
- **NB4000** includes lots of reviews of the notebook and **Book4000** includes reviews of books after removing the duplicate. Both of them are created by Songbo Tan.

B. EXPERIMENT SETTING

We compare the results of experiments with RAE model [16], the standard LSTM model [2], Bidirectional LSTM model [5], CNN [7] model and Tree-LSTM model [17]. All of them are the main methods used in handling sentiment classification. In English datasets, we only split each text into words by using punctuation and spaces and delete some special tokens without any other preprocessing. Chinese datasets are split into single words first by using the Chinese word segment system ICTCLAS2013[1] and then processed like the English datasets.

For all the datasets, we use the mini-batch size of 64, the learning rate of 0.001, and the dropout rate of 0.5. The number of hidden layer units is the same as word embedding size. All the hyperparameters are chosen by using cross-validation on Yelp2013. In other datasets, we use the same setting except for early stopping. We set the word embedding size of 100. In English dataset, we use a word vector file downloaded from the website.[2] In Chinese dataset, we use Glove to train word vector file on large Chinese Wikipedia dataset. In English dataset, SentiWordNet is used as sentiment lexicon. Due to the lack of the standard Chinese sentiment lexicon, we collected the following sentiment lexicons as the final used Chinese sentiment lexicon.

- Chinese sentiment polarity dictionary NTUSD[3] created by National Taiwan University.
- Chinese praise and derogatory dictionary[4] created by Tsinghua University.
- HowNet sentiment word set,[5] including Chinese emotion words and Chinese evaluation words.

C. EXPERIMENT RESULTS

All results of experiments can be seen from TABLE 2. Bi-LSTM and Tree-LSTM have better accuracy than LSTM model on all datasets, but does not make significant improvements. Furthermore, Tree-LSTM needs an external tool to parse the text to get a tree structure. Our methods get the best performance on the four datasets including English and Chinese datasets. Although the CNN model has the best accuracy on the MR dataset, our method (LE-LSTM) gets a comparative result, 80.8%, which improves the accuracy about 2% compared with the standard LSTM model. According to these results, we can conclude that our proposed methods are effective.

TABLE 2. Experiments results on different datasets.

Methods	IMDB	Yelp2013	MR	NB4000	Book4000
RAE	85.9	57.5	77.7	91.5	93.5
LSTM	86.2	58.5	78.5	91.75	94
Bi-LSTM	86.6	59	78.8	92	94.75
CNN	87	58	**81**	92	95
Tree-LSTM	88	59	80.7	92.5	95
LE-LSTM	87.4	59.4	80.8	**93.25**	95
ALE-LSTM	89.3	60.5	80	93	95.5
WALE-LSTM	**89.5**	**60.6**	79.9	93	**96**

Making comparison among our proposed methods,we find that ALE-LSTM and WALE-LSTM don't work at all times compared with LE-LSTM model. It only improves the accuracy of the datasets which have long text sequence. such as the IMDB, the Yelp2013 and the Book4000. These three datasets have longer text sequence than the other two datasets, which can be seen from TABLE 1. The results show that the standard LSTM has strong ability in handling short text sequence

[1]http://ictclas.nlpir.org/

[2]https://nlp.stanford.edu/projects/glove/

[3]http://www.datatang.com/data/44317

[4]http://nlp.csai.tsinghua.edu.cn/ lj/sentiment.dict.v1.0.zip

[5]http://www.datatang.com/datares/go.aspx?dataid=603399

(on the MR and the NB4000 datasets). But it also has long-term dependent problem when it meets long text sequence. Using the attention mechanism can improve this problem. Besides, WALE-LSTM has slightly higher accuracy than ALE-LSTM on the three datasets.

D. MODEL ANALYSIS

Attention mechanism: From TABLE 2, we find that the proposed attention mechanism cannot work on some short datasets. In order to exclude the possibility that the characteristics of these datasets themselves cause the problem, we do some experiments on the IMDB dataset. We respectively select 713 samples from test data of IMDB to test if ALSTM (ALE-LSTM removes the lexicon-enhanced mechanism) and WALSTM (WALE-LSTM removes the lexicon-enhanced mechanism) only work better than LSTM on long text sequences. Their lengths are in the range (0,100), (200,400) and (600,800). The number here is just an example, you can change it based on your experiments. All other settings are the same as before. The accuracy of per 100 training epochs of the different test datasets can be seen from FIGURE 3. From the subfigure (a), we find ALSTM and WALSTM have higher accuracy than LSTM at the beginning training epochs. But they almost haven't promotion compared with LSTM when the models converge. However, in the subfigure (b) and (c), the accuracy of ALSTM and WALSTM are higher than LSTM when the models converge. Furthermore, on the (600,800) dataset, the accuracy of ASLTM and WALSTM improves about 6%-7% compared with LSTM, while the accuracy improves about 2%-3% on the (200,400) dataset. Comparing (a), (b) and (c), we can conclude that the attention mechanism has a better performance on the longer dataset.

TABLE 3. **Words with high or low attention weights.**

	Words in Yelp2013
High attention weights	tasty, nicely, amazing, popular, best, happy, worse, unfortunate, disappointing, painfully
Low attention weights	a, the, about, which, and, of, an, your, to, for, was, if, there

In TABLE 3, we list some words with high attention weights and low attention weights in the Yelp2013 examples. We can see that these two attention mechanisms can find the important words, such as "best" and "unfortunate," and ignore some stop words, such as "the,""a" and "your."

Sentiment lexicon: In TABLE 4, we take MR dataset as an example and list some texts which are correctly predicted by LE-LSTM, but misclassified by LSTM. We find that some examples have obvious sentiment words, which can largely affect the sentiment polarity of the whole text. But LSTM cannot predict the sentiment polarity of texts correctly, such as "good" in the first text and "lazy" in the third text.

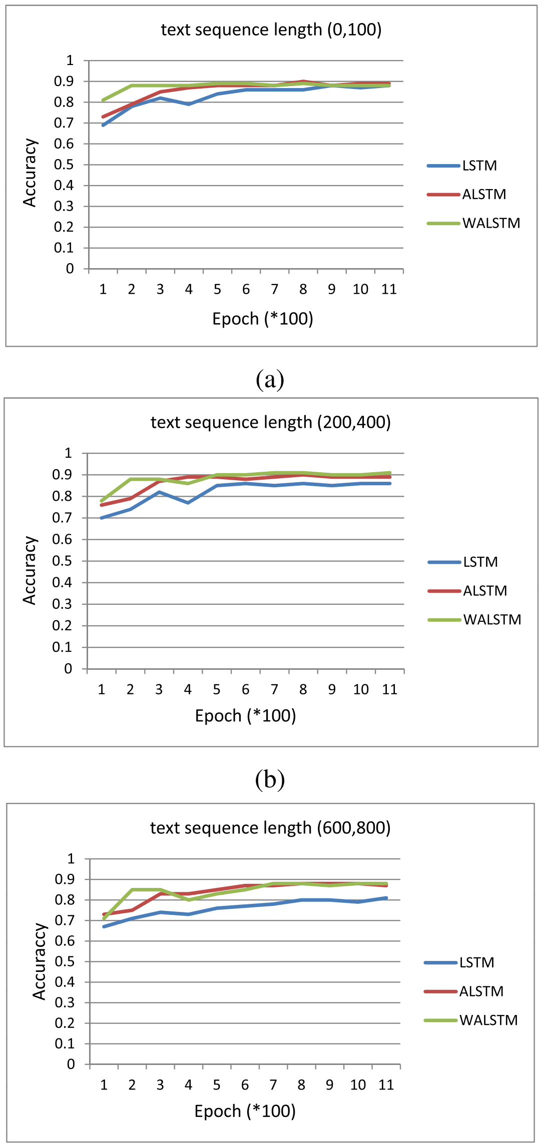

FIGURE 3. **From the subfigure (a), we find ALSTM and WALSTM have higher accuracy than LSTM at the beginning training epochs. But they almost haven't promotion compared with LSTM when the models converge. However, in the subfigure (b) and (c), the accuracy of ALSTM and WALSTM are higher than LSTM when the models converge.**
(a) Test accuracy on length (0,100). (b) Test accuracy on length (200,400). (c) Test accuracy on length (600,800).

Furthermore, in order to exclude the possibility that a larger word embedding size(caused by concatenating with the sentiment embedding) leads to the improvement of performance. we set different word embedding size to do experiments on MR dataset. The results can be seen from TABLE 5. We find that the accuracy of LE-LSTM with 100 embedding size is higher than the accuracy of LSTM with 300 embedding size. But the sentiment embedding size is only set 3-21 according to the quality of sentiment lexicon. So we can conclude that the prior sentiment information of words can really help to improve the performance.

TABLE 4. Examples predicted by LE-LSTM but LSTM.

Example	LE-LSTM	LSTM	Label
the farcical elements seemed too pat and familiar to hold my interest , yet its diverting grim message is a good one .	positive	negative	positive
off the hook is overlong and not well-acted , but credit writer-producer-director adam watstein with finishing it at all .	negative	positive	negative
lazy filmmaking , with the director taking a hands-off approach when he should have shaped the story to show us why it's compelling .	negative	positive	negative

TABLE 5. Results on different word embedding size.

Methods	100	200	300
LSTM	78.5	78.8	79.2
LE-LSTM	80.8	81	81.2

V. CONCLUSION

In this paper, we propose a lexicon-enhanced LSTM model to solve the problem that word embeddings carry more semantic information rather than sentiment information. By using the sentiment lexicon to train a word sentiment classifier, we can get the sentiment embedding of each word. Concatenating the word embedding and its sentiment embedding as the input of LSTM can make a higher performance in sentiment analysis tasks. We also propose a new method to calculate the attention vector in general sentiment analysis without target, and take two special circumstances as examples. But in both two circumstances, the attention mechanism only works better on long sequences, which also shows that LSTM has strong ability in modeling short sequences.

Compared with the existing methods, the results of experiments on Chinese and English datasets show that our models are effective. Besides, the quality of English and Chinese sentiment lexicons is very important. More abundant and accurate sentiment lexicon resources will make our models better.

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. NIPS*, Montreal, QC, Canada, 2014, pp. 1–15.

[2] K. Cho, B. V. Merrienboer, C. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. EMNLP*, Doha, Qatar, 2014, pp. 1724–1734.

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12 pp. 2493–2537, Aug. 2011.

[4] J. Fang and B. Chen, "Incorporating lexicon knowledge into SVM learning to improve sentiment classification," in *Proc. SAAIP*, 2011, pp. 94–100.

[5] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Germany: Springer-Verlag, 2012.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] Y. Kim. (2014). "Convolutional neural networks for sentence classification." [Online]. Available: https://arxiv.org/abs/1408.5882

[8] D. P. Kinga and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, San Diego, CA, USA, 2015, p. 115.

[9] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Trans. Assoc. Comput. Linguistics*, vol. 3, pp. 211–225, May 2015.

[10] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. ACL*, Stroudsburg, PA, USA, 2011, pp. 142–150.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (2013). "Efficient estimation of word representations in vector space." [Online]. Available: https://arxiv.org/abs/1301.3781

[12] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proc. ACL*, Ann Arbor, MI, USA, 2005, pp. 115–124.

[13] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. EMNLP*, vol. 14, 2014, pp. 1532–1543.

[14] Q. Qian, M. Huang, and X. Zhu. (2016). "Linguistically regularized LSTMS for sentiment classification." [Online]. Available: https://arxiv.org/abs/1611.03949

[15] S. S. Shen and H. Y. Lee. (2016). "Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection." [Online]. Available: https://arxiv.org/abs/1604.00077

[16] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proc. EMNLP*, Stroudsburg, PA, USA, 2011, pp. 151–161.

[17] K. S. Tai, R. Socher, and C. D. Manning. (2015). "Improved semantic representations from tree-structured long short-term memory networks." [Online]. Available: https://arxiv.org/abs/1503.00075

[18] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proc. EMNLP*, Lisbon, Portugal, 2015, pp. 1422–1432.

[19] D. Tang, B. Qin, and T. Liu, "Learning semantic representations of users and products for document level sentiment classification," in *Proc. ACL*, Beijing, China, 2015, pp. 1014–1023.

[20] D. Tang, F. Wei, B. N. Yang, T. Liu, and M. Zhou, "Sentiment embeddings with applications to sentiment analysis," *IEEE Trans. Knowl Data Eng.*, vol. 28, no. 2, pp. 496–509, Feb. 2016.

[21] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in *Proc. EMNLP*, Austin, TX, USA, 2016, pp. 606–615.

[22] M. Yang, W. Tu, J. Wang, F. Xu, and X. Chen, "Attention based LSTM for target dependent sentiment classification," in *Proc. AAAI*, San Francisco, CA, USA, 2017, pp. 5013–5014.

[23] T. Mikolov *et al.*, "Distributed representations of words and phrases and their compositionality," in *Proc. NIPS*, Lake Tahoe, Spain, 2013, pp. 3111–3119.

[24] T. Mikolov, W. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proc. NAACL*, Atlanta, GA, USA, 2013, pp. 746–751.

[25] D. Y. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for Twitter sentiment classification," in *Proc. ACL*, Baltimore, MD, USA, 2014, pp. 1555–1565.

[26] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proc. EMNLP-CoNLL*, Jeju Island, South Korea, 2012, pp. 1201–1211.

[27] J. Moniz and D. Krueger, "Nested LSTMs," in *Proc. ACML*, Seoul, South Korea, 2017, pp. 1–15.

[28] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Comput. Linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[29] P. D. Turney, "Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews," in *Proc. ACL*, Philadelphia, PA, USA, 2002, pp. 417–424.

[30] S. Li, Z. Wang, G. Zhou, and S. Y. M. Lee, "Semi-supervised learning for imbalanced sentiment classification," in *Proc. IJCAI*, Barcelona, Spain, 2011, p. 1862.

[31] B. Pang, L. Lee, and S. Vaithyanathan, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proc. ACL*, Barcelona, Spain, 2004, p. 271.

[32] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: Sentiment classification using machine learning techniques," in *Proc. ACL*, Philadelphia, PA, USA, USA, 2002, pp. 79–86.

[33] E. Riloff, J. Wiebe, and T. Wilson, "Learning subjective nouns using extraction pattern bootstrapping," in *Proc. HLT-NAACL*, Edmonton, BC, Canada, 2003, pp. 25–32.

XIANGHUA FU received the Ph.D. degree from Xi'an Jiaotong University, Shanxi, China.

He currently presides over one item of the National Natural Science Foundation, one Shenzhen Basic Research Project, hosts a project of the Natural Science Foundation of Guangdong Province, manages two basic research projects in Shenzhen, and participates in many projects, such as the National Natural Science Foundation Project and the National Support Program. He has published more than 60 articles in lots of important journals and international conferences at home and abroad. He also participated in the completion of two provincial-level quality courses. He is mainly engaged in data mining, machine learning, natural language processing, information retrieval, and the Internet of Things. He received the first prize of provincial teaching achievements. As an Associate Editor, he has participated in the preparation of four textbooks, and published and translated one book.

He started his teaching career with the School of Computer and Software, Shenzhen University, in 2005. Since 2008, he has been a master's Instructor of the School of Computer and Software, Shenzhen Technology University, where he is currently a Professor.

JINGYING YANG received the B.S. degree from the School of Information Science and Technology, Hainan University, Haikou, Hainan, in 2016.

She is currently pursuing the master's degree with Shenzhen University, instructed by Prof. Fu. Her main research directions are machine learning and natural language processing.

JIANQIANG LI received the B.S. and Ph.D. degrees from the South China University of Technology in 2003 and 2008, respectively.

He started teaching with Shenzhen University in 2008. He had published more than 40 papers in the fields of robotics, Internet of Things, hybrid systems, mobile medicine, optimization control and artificial intelligence, and more than thirty papers have been received in the three major indexes. He has successfully applied for ten national patents and eight software copyrights. He has been engaged in research work on artificial intelligence, robotics, Internet of Things, and mobile medical, for many years. And, he has hosted and completed three projects of the National Natural Science Foundation of China, including one key project, and one for face and youth project.

He is currently the Deputy Dean of the School of Computer Software, Shenzhen University, the Executive Director of the Institute of Network and Information Security, and the Deputy Director of the Mobile Internet Application Middleware Technology Engineering Laboratory of Guangdong Province. In 2016, he was selected as a member of the Guanyuan Youqingİ Training Plan of Shenzhen University.

MIN FANG received the B.S. and M.S. degrees from the South China University of Technology in 2004 and 2007, respectively.

She is currently a Teacher with the Harbin Institute of Technology, Shenzhen. Her major research interests include data analytics, Internet of Things, and cloud computing.

HUIHUI WANG received the Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, USA, in 2013.

In 2013, she joined the Department of Engineering, Jacksonville University, Jacksonville, FL, USA, where she is currently an Assistant Professor and the Founding Chair of the Department of Engineering. In 2011, she was an Engineering Intern with Qualcomm, Inc. She has authored more than 30 articles. She holds one U.S. patent. Her research interests include cyber-physical systems, Internet of Things, and healthcare and medical engineering based on smart materials, robotics, haptics based on smart materials/structures, ionic polymer metallic composites, and MEMS.