Two-level LSTM for Sentiment Analysis with Lexicon Embedding and Polar Flipping

Ou Wu, Tao Yang, Mengyang Li, and Ming Li

Abstract—Sentiment analysis is a key component in various text mining applications. Numerous sentiment classification techniques, including conventional and deep learning-based methods, have been proposed in the literature. In most existing methods, a high-quality training set is assumed to be given. Nevertheless, constructing a high-quality training set that consists of highly accurate labels is challenging in real applications. This difficulty stems from the fact that text samples usually contain complex sentiment representations, and their annotation is subjective. We address this challenge in this study by leveraging a new labeling strategy and utilizing a two-level long short-term memory network to construct a sentiment classifier. Lexical cues are useful for sentiment analysis, and they have been utilized in conventional studies. For example, polar and negation words play important roles in sentiment analysis. A new encoding strategy, that is, ρ-hot encoding, is proposed to alleviate the drawbacks of one-hot encoding and thus effectively incorporate useful lexical cues. Moreover, the sentimental polarity of a word may change in different sentences due to label noise or context. A flipping model is proposed to model the polar flipping of words in a sentence. We compile three Chinese data sets on the basis of our label strategy and proposed methodology. Experiments demonstrate that the proposed method outperforms state-of-the-art algorithms on both benchmark English data and our compiled Chinese data.

Index Terms—Text classification, sentiment analysis, LSTM, lexicon embedding, flipping.

1 Introduction

Text is important in many artificial intelligence applications. Among various text mining techniques, sentiment analysis is a key component in applications such as public opinion monitoring and comparative analysis. Sentiment analysis can be divided into three problems according to input texts, namely, sentence, paragraph, and document levels. This study focuses on sentence and paragraph levels.

Text sentiment analysis is usually considered a text classification problem. Almost all existing text classification techniques are applied to text sentiment analysis [1]. Typical techniques include bag-of-words (BOW)-based [2], deep learning-based [3], and lexicon-based (or rule-based) methods [4].

- *Ou Wu is with Center for Applied Mathematics, Tianjin University, E-mail: wuou@tju.edu.cn*
- *Tao Yang, Mengyang Li, and Ming Li are with Civil Aviation University of China, E-mail: {yangtao087, liming2960, limengyang99}@gmail.com.*

The experimental data sets and partial codes are publicly available at Github: https://github.com/Tju-AI/two-stage-labeling-for-the-sentiment-orientations

Although many achievements have been made and sentiment analysis has been successfully used in various commercial applications, its accuracy can be further improved. The construction of a high-accuracy sentiment classification model usually entails the challenging compilation of training sets with numerous samples and sufficiently accurate labels. The reason behind this difficulty is two-fold. First, sentiment is somewhat subjective, and a sample may receive different labels from different users. Second, some texts contain complex sentiment representations, and a single label is difficult to provide. We conduct a statistical analysis of public Chinese sentiment text sets in GitHub. The results show that the average label error is larger than 10%. This error value reflects the degree of difficulty of sentiment labeling.

Negation and interrogative sentences are difficult to classify when deep learning-based methods are applied. Although lexicon-based methods can deal with particular types of negation sentences, their generalization capability is poor.

We address the above issues with a new methodology. First, we introduce a two-stage labeling strategy for sentiment texts. In the first stage, annotators are invited to label a large number of short texts with relatively pure sentiment orientations. Each sample is labeled by only one annotator. In the second stage, a relatively small number of text samples with mixed sentiment orientations are annotated, and each sample is labeled by multiple annotators. Second, we propose a two-level long short-term memory (LSTM) [5] network to achieve two-level feature representation and classify the sentiment orientations of a text sample to utilize two labeled data sets. Thirdly, in the proposed two-level LSTM network, lexicon embedding is leveraged to incorporate linguistic features used in lexicon-based methods. Lastly, the labels in a word-polar dictionary usually contain noise and the polarity of a work can also change in different contexts. A flipping model is proposed to model the sentiment polarity flipping of a word in a sentence.

Three Chinese sentiment data sets are compiled to investigate the performance of the proposed methodology. The experimental results demonstrate the effectiveness of the proposed methods. Our work is new in the following aspects.

- A highly effective labeling strategy is adopted. Labeling a high-quality training set is difficult in senti-

ment analysis. In our labeling strategy, samples are divided into ones with relatively pure sentiment orientations and ones with relatively complex sentiment orientations. This procedure is easily performed in practice.

- A two-level LSTM network is proposed. Our labeling procedure yields two training sets with different sentiment levels; therefore, we propose a two-level LSTM network that can effectively utilize the two data sets.
- Lexicon embeddings are introduced based on a new encoding strategy. To incorporate useful cues which are usually used in lexicon-based methods, an effective encoding strategy, namely, ρ-hot encoding, is proposed in this work to address the limitations of the classical one-hot encoding.
- A flipping model is proposed to model polar flipping of words. Polar words are particularly important in sentiment analysis. However, the polar labels in a polar-word dictionary are not definitely right because the dictionary is very likely to contain label noise and the polarity of a word changes according to its context. For example, the polarity of 'heavy' is positive in "the fish is heavy", whereas it is negative in "the cell phone is heavy". To this end, a flipping model[1] is established to describe the flipping of the polarity of words in texts.

The rest of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 describes our methodology. Section 4 reports the experimental results, and Section 5 concludes the study.

2 RELATED WORK

2.1 Text Sentiment Analysis

Sentiment analysis aims to predict the sentiment polarity of an input text sample. Sentiment polarity can be divided into negative, neutral, and positive in many applications.

Existing sentiment classification methods can be roughly divided into two categories, namely, lexicon-based and machine learning-based methods [8]. Lexicon-based methods [9] construct polar and negation word dictionaries. A set of rules for polar and negation words is compiled to judge the sentiment orientation of a text document. This method cannot effectively predict implicit orientations. Machine learning-based methods [10] utilize a standard binary or multi-category classification approach. Different feature extraction algorithms, including BOW [11] and part of speech (POS) [10], are used. Word embedding and deep neural networks have recently been applied to sentiment analysis, and promising results have been obtained [12] [13].

2.2 Lexion-based Sentiment Classification

Lexicon-based methods are actually in implemented in an unsupervised manner. They infer the sentiment categories of input texts on the basis of polar and negation words.

1. In theory, our model should be useful in any arbitrary method which leverages the polar labels of words as supervised information such as [6] [7].

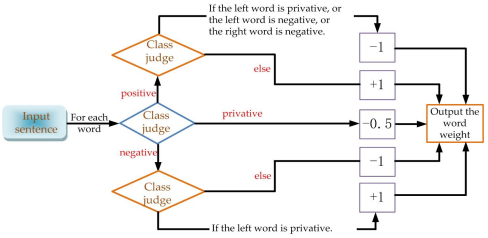


Figure 1. A lexicon-based approach for sentiment classification.

The primary advantage of these methods is that they do not require labeled training data. The key of lexicon-based methods is the lexical resource construction, which maps words into a category (positive, negative, neutral, or negation). Senti-WordNet [14] is a lexical resource for English text sentiment classification. For Chinese texts, Senti-HowNet is usually used.

Fig. 1 characterizes a typical lexicon-based sentiment classification approach. The approach iteratively checks each word in an input sentence from left to right. The weight score of each word is calculated according to the procedure shown in Fig. 1. The final sentiment score is the average score of the words with weight scores. The scores of positive, neutral, and negative sentiments are denoted as '+1', '0', and '-1', respectively. According to the lexicon-based algorithm shown in Fig. 1, the sentiment score of 'it is not bad' is 0.25, and the sentiment score of 'it is good' is 1. However, the score of 'it is not so bad' is -0.75, and this score is definitely wrong. Therefore, machine learning (including feature learning) methodologies have become mainstream in sentiment analysis.

2.3 Deep Learning-based Sentiment Classification

Deep learning (including word embedding [15]) has been applied to almost all text-related applications, such as translation [16], quality assurance [17], recommendation [18], and categorization [19]. Popular deep neural networks are divided into convolutional neural networks (CNNs) [20] and recurrent neural network (RNNs) [21] [22]. Both are utilized in sentiment classification [23]. Kim investigated the use of CNN in sentence sentiment classification and achieved promising results [3]. LSTM [24], a classical type of RNN, is the most popular network used for sentiment classification. A binary-directional LSTM [25] with an attention mechanism is demonstrated to be effective in sentiment analysis.

Deep learning-based methods rarely utilize the useful resources adopted in lexicon-based methods. Qiao et al. [7] incorporated lexicon-based cues into the training of an LSTM-based model. Their proposed method relies on a new loss function that considers the relationships between polar or certain types of words (e.g., negation) and those words next to them in input texts. Our study also combines lexical cues into LSTM. Nevertheless, unlike Qiao et al.'s study that implicitly used lexical cues, the present work explicitly uses lexical cues in the LSTM network. Shin et al. [26] combined the lexicon embeddings of polar words with word embeddings for sentiment classification. The difference between our approach an the method proposed by Shin et al. the is discussed in Section 3.3.5.

Numerous studies on aspect-level sentiment analysis exist [27] [28] [29]. This problem is different from the sentiment classification investigated in this study.

3 METHODOLOGY

This section first introduces our two-stage labeling procedure. A two-level LSTM is then proposed. Lexicon embedding is finally leveraged to incorporate lexical cues.

3.1 Two-stage Labeling

As stated earlier, sentiment is subjective, and texts usually contain mixed sentiment orientations. Therefore, texts sentiment orientations are difficult to label. In our study, three sentiment labels, namely, positive, neutral, and negative, are used. The following sentences are taken as examples.

S1: The service is poor. The taste is good, but the rest is not so bad.

S2: The quality of the phone is good, but the appearance is just so-so.

In user annotation, the labels of these two sentences depend on users. If a user is concerned about service, then the label of S1 may be 1negative'. By contrast, for another user who does not care about service, the label may be 'positive'. Similarly, a user may label S2 as 'positive' if he cares about quality. Another user may label it as 'negative' if the conjunction 'but' attracts the user's attention more. Another user may label it as 'neutral' if they are concerned about quality and appearance.

The underlying reason is that sentiment is more subjective than semantics. In related research on subjective categorization, such as visual aesthetics, each sample is usually repeatedly annotated by multiple annotators, and the average label is taken as the final label of the sample. This labeling strategy can also be applied to text sentiment annotation. However, we argue that this strategy is unsuitable for a (relatively) large number of samples. The reason lies in the following two aspects.

- Multiple annotators for a large number of data sets require a large budget.
- In our practice, annotators claim that their judgment criteria on sentiment become fused on texts with mixed sentiment orientations (e.g., S1 and S2) over time during labeling, and they become bored accordingly.

A two-stage labeling strategy is adopted in this study. In the first stage, each sentence/paragraph is divided into several clauses according to punctuation. The sentiment of each partitioned clause is relatively easy to annotate; therefore, each clause is labeled by only one user. In the second stage, a relatively small-sized sentence/paragraph set is labeled, and each sentence is labeled by all involved annotators. We still take the two sentences, S1 and S2, as examples. S1 and S2 are split into clauses, as shown below.

- S1:
  - S1.1: The service is poor
  - S1.2: The taste is good
  - S1.3: but the rest is not so bad.
- S2:
  - S2.1: The quality of the phone is good
  - S2.2: but the appearance is just so-so.

Each of the above clauses is labeled by only one annotator. The annotation in the first stage is easy to perform; thus, the number of clauses can be larger than the number of sentences used in the second labeling stage.

3.2 Two-level LSTM

Given two training data sets (denoted by T1 and T2), a new learning model should be utilized. LSTM[2] is a widely used deep neural network in deep learning-based text classification.

LSTM is a typical RNN model for short-term memory, which can last for a long period of time. An LSTM is applicable to classify, process, and predict time series information with given time lags of unknown size. A common LSTM block is composed of a cell, an input gate, an output gate, and a forget gate.

When LSTM is used to classify an input sentence, the hidden vectors (h_t) of each input vector are summed to form a dense vector that can be considered the feature representation of the input sentence, i.e.,

$$v_t = \sum_t h_t \tag{1}$$

In many applications, a bi-directional LSTM (Bi-LSTM) structure is usually used. In Bi-LSTM, forward and backward information are considered for information at time t; hence, the context is modeled. Bi-LSTM is thus significantly reasonable for text processing tasks. In our two-level LSTM, Bi-LSTM is used in each level.

The output hidden state at time t of a Bi-LSTM block can be described as follows:

$$\begin{aligned} \overrightarrow{h}_t &= \overrightarrow{o}_t \otimes \tanh(\overrightarrow{c}_t) \\ \overleftarrow{h}_t &= \overleftarrow{o}_t \otimes \tanh(\overleftarrow{c}_t) \\ h_t &= [\overrightarrow{h}_t, \overleftarrow{h}_t] \end{aligned} \tag{2}$$

where $\overrightarrow{h}_t$, $\overrightarrow{o}_t$, and $\overrightarrow{c}_t$ are the corresponding vectors at time t in the forward LSTM block; and $\overleftarrow{h}_t$, $\overleftarrow{o}_t$, and $\overleftarrow{c}_t$ are the corresponding vectors at time t in the backward LSTM block. $H = \{h_1, \cdots, h_T\}$. When attention is used, the dense feature vector γ of an input sentence is calculated as follows:

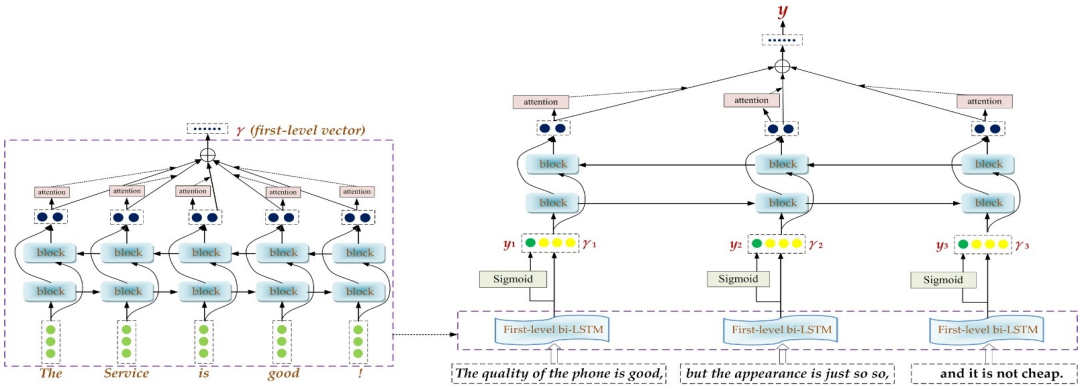
$$\begin{aligned} \alpha &= \text{softmax}(\mathbf{w}^T H) \\ \gamma &= H\alpha^T \end{aligned} \tag{3}$$

where α is the vector that consists of attention weights.

Our proposed network consists of two levels of LSTM network. In the first level, a Bi-LSTM is used and learned on the basis of the first training set T1. This level is a conventional sentiment classification process. The input of this level is a clause, and the input x_t is the embedding of the basic unit of the input texts[3]. The network is shown in Fig. 2(a).

2. CNN is another widely used text classification model. Our idea can also be applied to CNN.

3. In English texts, the basic unit is usually a word; in Chinese texts, the basic unit is a Chinese word or character.



(a) The first-level bi-LSTM (b) The whole two-level structure

Figure 2. The proposed two-level LSTM network (this network does **not** including the lexicon embedding and the polar flipping model which are introduced in Figs. 5 and 7).

In the second level, a Bi-LSTM is also used and learned on the basis of the second training set T2. The input of this level is a sentence or a paragraph. The input x_t consists of two parts[4]. The first part is the feature vector of the t-th clause. The feature vector is generated by the first-level network. In other words, the dense feature shown in Fig. 2(a) (γ) is used. The second part is the sentiment score (not predicted label) output by the first-level network. The sentence S1 (The service is poor. The taste is good, but the rest is not so bad.) used in Subsection 3.1 is taken as an illustrative example. S1 contains three clauses. Let σ be the sigmoid function. The input vector of S1 can be represented by

$$S1 : \{\eta_1, \eta_2, \eta_3\}$$

where

$$\begin{aligned} &\eta_i = \{y_i^{(1)}, \gamma_i^{(1)}\} \\ &y_i^{(1)} = \sigma(W\gamma_i^{(1)} + b) \\ &i = 1, 2, 3 \end{aligned} \tag{4}$$

where $y_i^{(1)}$ is the output score of the ith clause by the first-level LSTM and $\gamma_i^{(1)}$ is the feature representation of the ith clause by the first LSTM. The network of the whole two-level network is shown in Fig. 2(b). The loss function of the whole network is defined as follows:

$$l = \sum_n [loss(y_n, y'_n) + \frac{\lambda}{n_I} \sum_i loss(y_{ni}, y'_{ni})] \tag{5}$$

where y_n and y'_n are the true and predicted labels of the n-th sample, respectively; y_{ni} and y'_{ni} are the true and predicted label of the i-th clause of the n-th sample, respectively; λ is the parameter; n_I is the number of clauses in the i-th sample.

3.3 Lexical Embedding

Lexicon embedding aims to integrate a wide range of lexical cues into the two-level LSTM network. Based on our empirical analysis and previous studies, key lexical words, POS, and conjunction cues are considered. The proposed lexicon embedding is based on ρ-hot encoding. Therefore, ρ-hot encoding is first described.

4. The third part is lexicon embedding which will be introduced in Section 3.3.4.

3.3.1 ρ-hot encoding

For categorical data, one-hot encoding is the most widely used encoding strategy when different categories are independent[5]. For example, if one-hot encoding is used to represent three categories, namely, positive, neutral, and negative, the encoding vectors for the three categories are $[1, 0, 0]^T$, $[0, 1, 0]^T$, and $[0, 0, 1]^T$, respectively.

In this work, many lexical cues are categorical data, and different categories are independent. These lexical cues can directly be represented by one-hot encoding. The encoded vectors for lexical cues are then concatenated with other vectors, such as character/word embedding. Based on our empirical evaluations, a more effective encoding is proposed based on the conventional one-hot encoding. The encoding strategy is defined as follows.

$$\rho - \textbf{hot} \quad \textbf{encoding} : \nu_{\rho,n} = \rho \cdot \nu_1 \otimes \mathbf{1}_{n\times 1} \tag{6}$$

where $\nu_{\rho,n}$ is the ρ-hot encoded vector; ρ is the proportion parameter; ν_1 is the one-hot encoded vector; $\mathbf{1}_{n\times 1}$ is an n-dimensional vector; $\otimes$ is the tensor product. If both ρ and n are equal to 1, then ρ-hot encoding is reduced to one-hot encoding. The parameter n is applied to increasing the length of the final encoded vector.

When ρ equals 0.6 and n equals 4. For a three-dimensional one-hot encoded vector ν_1, we have

$$\begin{aligned} \text{if} \quad & \nu_1 = [1, 0, 0]^T, \\ & \nu_\rho = 0.6 \cdot \nu_1 \otimes \mathbf{1}_{4\times 1} \\ & \quad = [0.6, 0.6, 0.6, 0.6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T \\ \text{if} \quad & \nu_1 = [0, 1, 0]^T, \\ & \nu_\rho = 0.6 \cdot \nu_1 \otimes \mathbf{1}_{4\times 1} \\ & \quad = [0, 0, 0, 0, 0.6, 0.6, 0.6, 0.6, 0, 0, 0, 0, 0, 0, 0, 0]^T \\ \text{if} \quad & \nu_1 = [0, 0, 1]^T, \\ & \nu_\rho = 0.6 \cdot \nu_1 \otimes \mathbf{1}_{4\times 1} \\ & \quad = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.6, 0.6, 0.6, 0.6]^T \end{aligned} \tag{7}$$

The numerical examples in Eq. (7) indicate that the obtained ρ-hot encoded vector (ν_ρ) can have varied values for non-zero elements and longer lengthes compared with the corresponding one-hot encoded vector (ν_1). Similar yet

5. When different categories are correlated, sophisticated encoding strategies can be utilized. For example, one-hot is a traditional encoding for words. Many word-embedding methods are proposed with consideration of the relation among words.

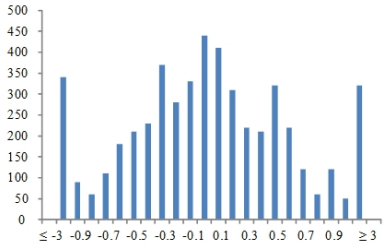


Figure 3. The histogram of the values in word embedding vectors. Most values are smaller than 1.

simple tricks for the increasing of the vector length have been successfully used in previous studies [30] [31].

So far, there is no accurate theoretical explanation for our proposed ρ-hot encoding. Nevertheless, one-hot encoding presents two main limitations when the encoded vector is concatenated with other vectors.

- The value difference between the elements of one-hot encoded vectors and those of other encoded vectors (e.g., word embedding vectors) may be large. Fig. 3 shows the histogram of the values of the elements of the word embedding vectors. The magnitude of most elements are smaller than 1.
- The lengths of one-hot encoded vectors are usually shorter than those of other encoded vectors. Consequently, the proportion of one-hot encoded part is small in the concatenated vectors.

The above two limitations may affect the final sentiment analysis performance, whereas our proposed ρ-hot encoding alleviated these two limitations.

3.3.2 Embedding for key lexical words

Most lexicon-based sentiment methods rely on four types of words, namely, positive, negative, neutral, and negation. These words are useful cues for predicting the sentiment labels of input texts. The incorporation of these words should also be useful. A previous study has shown that a typical document comprises approximately 8% of such words [32]. Sentiments expressed in a conditional sentence can be difficult to determine due to the semantic condition. The sentiment polarities of interrogative sentences are also difficult to classify according to our empirical study. Because the automatically judging whether a sentence is conditional or interrogative is challenging, we directly consider suppositive and interrogative words.

Five types of words, namely, positive (Pos), negative (Neg), negation (Nt), suppositive (Sup), and interrogative (Int), are represented by the proposed encoding method. The rest words, which do not belong to any of the above five types, are named "others (Oth)" instead of "neutral" because some words, such as "the", are unrelated to "sentiment". The value of n in Eq. (6) is set as 10. The encoded vectors are as follows.

$$\begin{aligned}
&\text{Pos} : [\boldsymbol{\rho}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}]^T \\
&\text{Neg} : [\mathbf{0}_{1\times 10}, \boldsymbol{\rho}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}]^T \\
&\text{Nt} : [\mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \boldsymbol{\rho}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}]^T \\
&\text{Sup} : [\mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \boldsymbol{\rho}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}]^T \\
&\text{Int} : [\mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \boldsymbol{\rho}_{1\times 10}, \mathbf{0}_{1\times 10}]^T \\
&\text{Oth} : [\mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \boldsymbol{\rho}_{1\times 10}]^T
\end{aligned}$$

In the proposed ρ-hot embedding, the parameter ρ can be learned during training.

Certain types (e.g., positive, negative, and negation) of words should play more important roles than other words do in texts; therefore, their embeddings are also used in the attention layer. A new LSTM based on our lexicon embedding is proposed, as shown in Fig. 4. The attention layer and final dense vector of the network in Fig. 2(a) are calculated as follows.

$$\begin{aligned}
\alpha_t &= \text{softmax}(W_\alpha \begin{bmatrix} h_t \\ l_t \end{bmatrix} + b_i) \\
\gamma &= \sum_t \alpha_t \begin{bmatrix} h_t \\ l_t \end{bmatrix}
\end{aligned} \tag{8}$$

where α_t is the attention weight for the t-th input, l_t is the lexicon embedding for key lexical words for the t-th input, and γ is the final dense vector. Eq. (7) is used in the first-level LSTM.

3.3.3 Embedding for POS

POS is usually used as a key cue in sentiment analysis [33]. Intuitively, adjective and adverb words are very likely to be more important than some other types of words such as adjective and article. To this end, we use additional lexicon embedding based on POS information. This new lexicon embedding is also applied to the attention layer. The motivation lies in that certain types of POS should play important roles in sentiment.

The proposed ρ-hot embedding is still applied to POS types in this study. According to our initial case studies, eight POS types are considered. They are noun, adjective, verb, pronoun, adverb, preposition, accessory, and others. The eight POS types are represented by the proposed ρ-hot encoding. We let n in Eq. (6) be 10. The first three POS types are as follows.

$$\begin{aligned}
&\text{Noun} : [\boldsymbol{\rho}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \cdots, \mathbf{0}_{1\times 10}]^T \\
&\text{Adj} : [\mathbf{0}_{1\times 10}, \boldsymbol{\rho}_{1\times 10}, \mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \cdots, \mathbf{0}_{1\times 10}]^T \\
&\text{Verb} : [\mathbf{0}_{1\times 10}, \mathbf{0}_{1\times 10}, \boldsymbol{\rho}_{1\times 10}, \mathbf{0}_{1\times 10}, \cdots, \mathbf{0}_{1\times 10}]^T
\end{aligned}$$

When POS embedding is used, the attention layer and final outputs of the network in Eq. (3) become

$$\begin{aligned}
\alpha_t &= \text{softmax}(W_\alpha \begin{bmatrix} h_t \\ l_t \\ \eta_t \end{bmatrix} + b_i) \\
\gamma &= \sum_t \alpha_t \begin{bmatrix} h_t \\ l_t \end{bmatrix}
\end{aligned} \tag{9}$$

where η_t is the lexicon embedding for POS of the t-th input.

3.3.4 Embedding for conjunction

Conjunction words play important roles in sentiment analysis [34]. For example, conjunctions such as 'but' and 'moreover' usually indicate the focus of texts and attract readers attention. These words may be useful cues for attention inference in the second level of our network. Therefore, conjunctions are considered in the input of the second-level LSTM.

Once a set of conjunction words is compiled, ρ-hot embedding is used. In our experiments, the number of conjunction words (including thesaurus) is 13 for Chinese

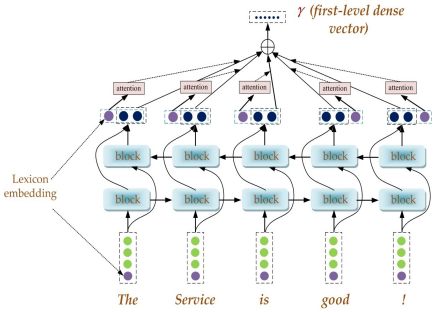


Figure 4. The first-level LSTM with lexicon embedding in both the input and attention layers.

texts and 26 for English texts. Therefore, the parameter n for Chinese conjunction words in Eq. (6) is set as 1.

When conjunction embedding is used for the second-level layer, the attention layer and final outputs of the network in Fig. 2(b) are calculated as follows.

$$\beta_t = \text{softmax}(W_\beta \begin{bmatrix} y_t^{(1)} \\ h_t^{(2)} \\ \omega_t^s \\ \omega_t^e \end{bmatrix} + b'_i) \qquad (10)$$
$$\gamma^{(2)} = \sum_t \beta_t \begin{bmatrix} h_t^{(2)} \\ y_t^{(1)} \end{bmatrix}$$

where β_t is the attention weight for the t-th input clause; $h_t^{(2)}$ is the hidden vector of the second-level LSTM; ω_t^s and ω_t^e are the conjunction embeddings for the first and last words in the t-th input clause, respectively; and $\gamma^{(2)}$ is the final dense vector used for the final classification.

The two-level network with lexicon embedding is shown in Fig. 5. The lexicon embedding is used in both input and attention layers.

3.3.5 Differences between our and existing lexicon embedding

Shin et al. [26] also embedded lexical information into sentiment analysis. Three major differences exist between our method and the method proposed by Shin et al. [26].

- The lexicon embedding proposed by Shin et al. uses one-hot encoding, whereas the proposed method uses a new encoding strategy that can be considered a soft one-hot encoding.
- The lexicon embedding proposed by Shin et al. extends the lengths of raw encoded vectors. However, the extension aims to keep the lengths of lexical and word embeddings equal. Their extension method also only relies on zero padding and is thus different from the proposed method.
- Only sentimental words are considered in the lexicon embedding proposed by Shin et al. On the contrary, sentimental words, POS, and conjunctions are considered in our work.

3.4 The Polar Flipping Model

The polarity of words is crucial in sentiment analysis. Most rule-based methods mainly rely on polar and negation words. The word polarity is also usually applied as (partial) supervised information in learning-based methods. Qian et

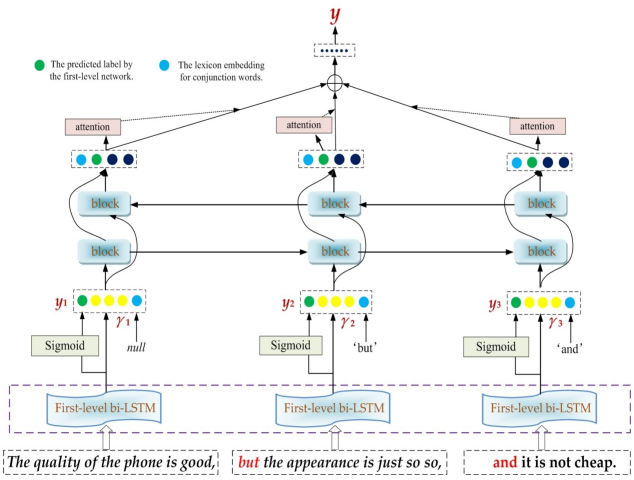


Figure 5. The whole two-level LSTM network with lexicon embedding in both the input and attention layers.

al. [7] leveraged the polar labels to regularize the hidden state of each word in training texts. They utilized the KL divergence between the polar label and the predicted label of a word as the regularization term during training. Although word level polar information is proven to be useful in sentiment analysis, the application of word-level polar information suffers from the following defects:

- Because the number of words usually exceeds ten thousand, the labeling of all words' polarity is not an easy task. Labeling error is thus unavoidable.
- The polarities of some words highly depend on its contexts. Therefore, the polarity of a word may vary in different texts.

In this work, the word-level polar labels are considered noisy labels. In noisy-label learning [35], a flipping model is usually used to model the relationships between (possibly) noisy and true labels. Inspired by noisy-label learning, a polar flipping model is proposed. Let P_1, P_2, and P_3 represent the three polar labels 'positive', 'negative', and 'neural', respectively. Without of loss generalization, the label of a given word is assumed to be P_1. Let x be the representation of the given word and C be the representation of the context information. Then, the polar flipping model is described as follows:

$$\begin{aligned} P=&\sigma_1(x,C)P_1 \\ &+[1-\sigma_1(x,C)]\{\sigma_2(x,C)P_2+[1-\sigma_2(x,C)]P_3\} \end{aligned} \qquad (11)$$

where both σ_1 and σ_2 are sigmoid functions. Eq. (10) describes the flipping of the polarity of a word given its context. If $\sigma_1(x,C) == 1$, then $P = P_1$; else if $\sigma_2(x,C) == 1$, then $P = P_2$; else $P = P_3$.

In this work, the hidden state vector h of the word in LSTM is used to represent (x, C) in Eq. (10). Therefore, Eq. (10) becomes:

$$P=\sigma_1(h)P_1+[1-\sigma_1(h)]\{\sigma_2(h)P_2+[1-\sigma_2(h)]P_3\} \qquad (12)$$

In noisy-label learning, the noisy rate is usually assumed to be small. In our model, to control the flipping rate, we

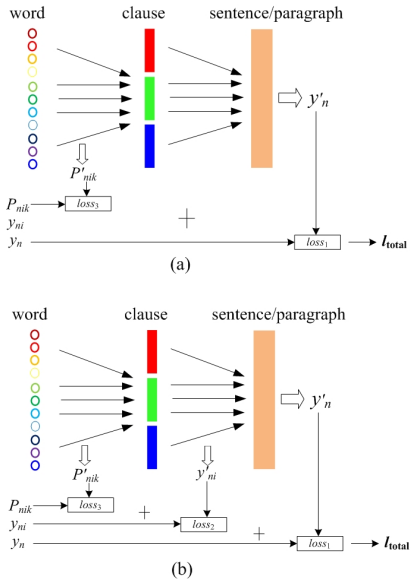


Figure 6. The illustrative network structure with supervised information in middle layers.

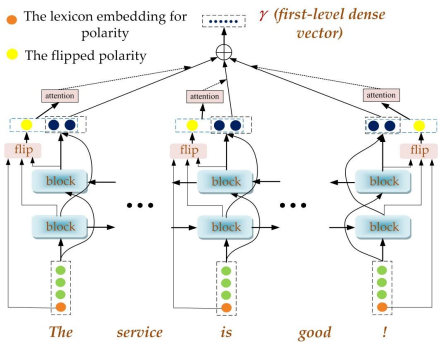


Figure 7. The first-level network with the flipping model (To simplify the illustration, only polar information is shown in the lexicon embedding).

take the flipping loss as an additional regularization term. Consequently, the loss function defined in Eq. (5) becomes:

$$l_{total} = \sum_{n} [loss_1(y_n, y'_n) + \frac{\lambda_1}{n_I} \sum_{i} loss_2(y_{ni}, y'_{ni}) + \frac{\lambda_2}{n_{IK}} \sum_{k} loss_3(P_{nik}, P'_{nik})] \quad (13)$$

where P_{nik} and P'_{nik} are the true and the flipped polar labels of the k-th word in the i-th clause in the n-th sample; λ_1 and λ_2 are balance parameters; n_{IK} is the number of words in the i-th clause of the n-th sample. The functions $loss_1$ and $loss_2$ in Eq. (12) are the cross-entropy loss; the function $loss_3$ is calculated as follows:

$$loss_3(P_{nik}, P'_{nik}) = 1 - \sigma_1(h_{nik}) \quad (14)$$

where h_{nik} is the hidden vector of the k-th word in the i-th clause in the n-th sample. According to the loss function defined in Eq. (12), our method can be viewed as being added supervised information in the middle layers (both word and clause levels) of the whole network as shown in Fig. 6. The first-level LSTM in the proposed two-level network described in Fig. 2(b) is shown in Fig. 7.

Based on Fig. 7, there are three main differences between our and existing hierarchal networks:

- Our network depends on a relatively new labeling strategy, namely, two-stage labeling.
- Each level of our network is associated with supervised information, whereas only the top level is with supervised information in almost all existing hierarchal networks.
- Each level of our network is associated with a loss term in our network, whereas there is usually only one loss in existing hierarchal networks.
- Plentiful lexical cues are embedded into our network, whereas there is usually only one type of lexical cues considered in existing networks.

3.5 The Learning Details

The algorithmic steps of the entire learning procedure for the proposed ρ-hot lexicon embedding-based two-level LSTM (called ρTl-LSTM) are shown in **Algorithm** 1. In **Algorithm** 1, T1 refers to the training data that consist of clauses and the labels obtained in the first-stage labeling procedure. T2 refers to the training data that consist of sentences and the labels obtained in the second-stage labeling procedure.

ALGORITHM 1: ρTl-LSTM

Input: Training sets T1 and T2; dictionary of key lexical words; POS for each word; dictionary of conjunction words; character/word embeddings for each character/word; parameters λ_1 and λ_2.

Output: A trained two-level LSTM for sentiment classification.
Steps:

1. Construct the ρ-hot-based embedding vector for each word (including punctuation) in the clauses in T1. The embeddings include the character/word and lexicon embeddings of each character/word;
2. Construct the embedding vector for each conjunction word in each clauses as the input for the second-level LSTM;
3. Train the two-level LSTM on the basis of the input embedding vectors and labels of polar words, the T1 text clauses, and the T2 text samples; the loss function is defined in Eq. (12).

The proposed two-level LSTM can be applied to texts with arbitrary languages. Word information is required in lexical construction regardless of whether character or word embedding is used. The reason is that the three types of lexicon embeddings are performed at the word level. Therefore, when character embedding is used, the lexicon embedding of each character is the lexicon embedding of the word containing it.

4 EXPERIMENTS

This section shows the evaluation of the proposed methodology in terms of the two-level LSTM network and each part of the lexicon embedding. Both English benchmark text corpora and Chinese text corpora are used.

4.1 Results on Chinese Corpora

In the experiments, three competing algorithms, namely, BOW, CNN, and (conventional) LSTM, are used. The lexicon embedding-based method proposed by Shin et al. [26] discussed in Section 3.3.5 is also compared.

For BOW, term frequency-inverse document frequency is utilized to construct features. Ridge regression [36] is used as a classifier. For CNN, a three-channel CNN is used. For LSTM, one-layer and two-layer Bi-LSTM with attention are adopted, and the results of the network with superior performance are presented. CNN and LSTM are performed on TensorFlow, and default parameter settings are followed.

The key parameters are searched as follows. The embedding dimensions of characters and words are searched in [100, 150, 200, 250, 300]. The parameter n in ρ-hot encoding is searched in $[1, 3, \cdots, 15]$. The parameters λ_1 and λ_2 are searched in $[0.001, 0.01, 0.1, 1, 10]$. Baidu Chinese word segmentation API is used.

4.1.1 Experimental Data and Labeling

We compile three Chinese text corpora from online data for three domains, namely, 'hotel', 'mobile phone (mobile)', and 'travel'. All texts are about user reviews. Each text sample collected is first partitioned into clauses according to Chinese tokens[6]. Three clause sets are subsequently obtained from the three text corpora.

The labels '+1', '0.5', and '0' correspond to the three sentiment classes 'positive', 'neutral', and 'negative', respectively. The text data are labeled according to our two-stage labeling strategy.

- In the first stage, only one user is invited to label each clause sample as the sentiment orientations for clauses (or sub-sentences) are easy to label.
- In the second stage, five users[7] are invited to label each text sample in the three raw data sets. The average score of the five users on each sample is calculated. Samples with average scores located in [0.6, 1] are labeled as 'positive'. Samples with average scores located in [0, 0.4] are labeled as 'negative'. Others are labeled as 'neutral'. The details of the labeling results are shown in Table 1.

TABLE 1
Details of the three data corpora. Each corpus consists of raw samples (sentences or paragraphs) and partitioned clauses (sub-sentences).

Data corpus		raw	clauses
Travel	Pos.	1567	3490
	Neu.	576	5168
	Neg.	1957	2633
	Total	4100	11291
Hotel	Pos.	1586	3987
	Neu.	401	2123
	Neg.	1838	5154
	Total	3825	11264
Mobile	Pos.	1400	2788
	Neu.	589	2375
	Neg.	1494	2955
	Total	3483	8118

6. Token-based token is inaccurate for English text partition. Nevertheless, the segment results for Chinese texts are acceptable. A more reasonable way will be investigated in our future work.

7. Five graduate students, including three males and two females, were invited to label the data.

All the training and test data and the labels are available online[8].

In our experiments, the five types of key lexical words introduced in Subsection 3.3.2 are manually constructed. The details of the five types of words are listed in Table 2[9]. The conjunction words are also manually constructed. The number of conjunction words used in the experiments is 169.

TABLE 2
Numbers of five types of key lexical words.

Data corpus	Travel	Hotel	Mobile
Positive	366	254	358
Negative	327	194	382
Negation	61	61	61
Interrogative	48	48	48
Suppositive	18	18	18

In each experimental run, the training set is compiled on the basis of the training data listed in Table 1. The compiling rule is specified before each experimental run. The test data are fixed to facilitate experimental duplication and comparison by other researchers. The data with the fixed split are available at our Github page.

4.1.2 Results of existing methods

In this subsubsection, each of the three raw data sets (associated with their labels) shown in Table 1 is used. The clause data are not used. In other words, the training data used in this subsubsection are the same as those used in previous studies. For each data corpus, 1000 raw data samples are used as the test data, and the rest are used as the training data. The involved algorithms are detailed as follows.

- **CNN-C** denotes the CNN with (Chinese) character embedding.
- **CNN-W** denotes the CNN with (Chinese) word embedding.
- **CNN-Lex-C** denotes the algorithm which also integrates polar words in CNN which is proposed by Shin et al. [26]. The (Chinese) character embedding is used.
- **CNN-Lex-W** denotes the algorithm which also integrates polar words in CNN which is proposed by Shin et al. [26]. The (Chinese) word embedding is used.
- **Bi-LSTM-C** denotes the BI-LSTM with (Chinese) character embedding.
- **Bi-LSTM-W** denotes the Bi-LSTM with (Chinese) word embedding.
- **Lex-rule** denotes the rule-based approach shows in Fig. 1. This approach is unsupervised.
- **BOW** denotes the conventional algorithm which is based of bag-of-words features. Four shallow classifiers are used, namely, logistic regression (LR), support vector machine (SVM), naive Bayes (NB), and random forest (RF). For SVM, the parameters C and g

8. https://github.com/Tju-AI/two-stage-labeling-for-the-sentiment-orientations/tree/master/data

9. The five types of key lexical words are also available and introduced in our Github project page.

are searched via five-fold cross validation from $\{0.1, 1, 5, 10, 100\}$ and $\{0.01, 0.1, 1, 5, 10\}$, respectively. For RF, the number of trees is searched via five-fold cross validation from $\{10, 20, 50, 100, 200, 500\}$.

The accuracies of the above algorithms are listed in Table 3. Overall, Bi-LSTM significantly outperforms CNN and BOW ($p < 0.01$) based on t-test. This conclusion is in accordance with the conclusion that RNN performs efficiently against CNN in a broad range of natural language processing (NLP) tasks on the basis of extensive comparative studies [37]. In addition, CNN-Lex outperforms CNN under both character and word embeddings ($p < 0.05$), which suggests that lexicon cues are useful in sentiment analysis. Lex-rule achieves the lowest accuracies on all the three data sets. Considering that the performances of (traditional) CNN, Lex-rule, and BOW are relatively poor, they are not applied in the remaining parts.

TABLE 3
The classification accuracies of existing algorithms on raw samples.

Data corpus	Travel	Hotel	Mobile
CNN-C	0.723	0.698	0.727
CNN-W	0.731	0.729	0.748
CNN-Lex-C	0.744	0.734	0.731
CNN-Lex-W	**0.758**	0.764	0.755
Bi-LSTM-C	0.754	0.753	0.805
Bi-LSTM-W	0.746	**0.785**	**0.809**
Lex-rule	0.556	0.539	0.684
BOW+LR	0.713	0.678	0.702
BOW+SVM	0.746	0.735	0.718
BOW+NB	0.698	0.712	0.687
BOW+RF	0.733	0.756	0.743

4.1.3 Results of two-level LSTM without lexicon embedding

In this experimental comparison, the proposed two-level LSTM is evaluated, whereas lexicon embedding is not used in the entire network. The primary goal is to test whether the introduced two-stage labeling and the two-level network structure are useful for sentiment analysis.

The raw and clause data listed in Table 1 are used to perform the two-level LSTM. Tl-LSTM denotes the two-level LSTM. 'R+C' refer to the mixed data of raw and clause data. The test data are still the 1000 samples used in Section 4.1.2 for each corpus. Table 4 shows the classification accuracies. To ensure that the results differ from those in Table 3, we explicitly add 'R+C' after each algorithm in Table 4. In the last line of Table 4, the base results for each corpus in Table 3 are also listed.

On all the three data corpora, the proposed two-level network (without lexicon embedding) with character embedding, Tl-LSTM-C, do not significantly outperform other involved competing methods by conducting t-test. However, it achieves the highest accuracies on all the three data corpora. The results in Table 4 indicate that the performances of Tl-LSTM on the mixed training and test data (R+C) are better than those of Bi-LSTM. This comparison indicates that the proposed two-level LSTM is useful.

TABLE 4
The classification accuracies of competing algorithms.

Data corpus	Travel	Hotel	Mobile
Tl-LSTM-C(R+C)	**0.801**	**0.813**	**0.820**
Tl-LSTM-W(R+C)	0.770	0.772	0.820
CNN-Lex-C (R+C)	0.755	0.770	0.749
CNN-Lex-W (R+C)	0.773	0.776	0.786
Bi-LSTM-C (R+C)	0.781	0.784	0.817
Bi-LSTM-W (R+C)	0.762	0.789	0.813
Baseline (best in Table 3)	0.758	0.785	0.809

In addition, for the involved algorithms, most accuracies achieved on 'R+C' are higher than the best results only achieved on 'R' listed in Table 3. This comparison suggests that the introduced two-stage labeling is useful.

4.1.4 Results of the whole two-level LSTM

In this experimental run, both lexicon embedding and the flipping model are used in the proposed two-level LSTM or ρTl-LSTM. Table 5 shows the results. In order to assess the usefulness of the flipping model, the results of ρTl-LSTM without considering flipping are also present.

The performances of TI-LSTM with both lexicon embedding and the flipping model (i.e., ρTl-LSTM) are consistently better than those of TI-LSTM without lexicon embedding (i.e., Tl-LSTM) listed in Table 4 and ρTl-LSTM without flipping ($p < 0.05$). ρTl-LSTM with flipping also slightly outperforms ρTl-LSTM without flipping indicating that the flipping modular is useful in the network. The improved accuracies of ρTI-LSTM over Tl-LSTM on the three data corpora are listed in Table 6.

TABLE 5
The classification accuracies of two-level LSTM with lexicon embedding.

Data corpus	Travel	Hotel	Mobile
ρTl-LSTM-C without flipping	0.816	0.837	0.826
ρTl-LSTM-W without flipping	0.800	0.810	0.841
ρTl-LSTM-C	0.817	0.837	0.837
ρTl-LSTM-W	0.804	0.821	0.847

TABLE 6
The accuracy improvement of two-level LSTM when lexicon embedding was used over those of two-level LSTM without lexicon embedding.

Travel	Hotel	Mobile
+1.6%	+2.4%	+1.7%
+3.4%	+3.9%	+2.7%

4.2 Results on English Corpora

Two benchmark datasets are used for evaluating the proposed models: Movie Review (MR) [38] and Stanford Sentiment Treebank (SST) [39]. The former consists 10,662 sentences with binary classes (positive and negative); while the

latter consists of 11,885 sentences with five classes {very negative, negative, neutral, positive, very positive}.

The involved competing algorithms are CNN, LSTM/Bi-LSTM, Tree-LSTM [40], NCSL [6], and LR-Bi-LSTM [7]. The last two methods are two state-of-the-art methods which also utilize the word-level polar information. The results of these competing algorithms are obtained directly from the results presented in Qian et al. work [7]. The polar and negation words are compiled by following the method used in Qian et al. work [7]. The number of conjunction words in the dictionary is 207 which are attributed to 13 classes of words.

TABLE 7
The accuracies of the competing methods on English corpora in ρ-hot encoding.

Method	MR	SST
CNN	0.815	0.469
Bi-LSTM	0.793	0.465
Tree-LSTM	0.807	0.481
NCSL	0.829	0.471
LR-Bi-LSTM	0.821	0.486
ρTl-LSTM	**0.851**	**0.498**

The proposed method achieves the highest accuracies among all the competing methods including the state-of-the-art NCSL and LR-Bi-LSTM. The increased accuracies on both sets are smaller than those on datasets reported in Section 4.1. The main reason lies in that the average numbers of clauses on both MR and SST are less than 1.5.

4.3 Disussion

The experimental evaluation discussed in Subsection 4.2 verifies the effectiveness of the proposed method, ρTl-LSTM, on fixed training/testing data split. In this subsection, we conducted more experiments via 10-cross validation on each data corpus to further evaluate the proposed method. More over, unlike the conventional RNN, ρTl-LSTM contains lexicon embedding that consists of new technique and components, including ρ-hot encoding, embedding for polar words, embedding for POS, and embedding for conjunctions. Therefore, this subsection evaluates the performances of the involved technique and embeddings separately.

4.3.1 The 10-fold cross validation results on the Chinese corpora

The 10-fold cross validation results for the main competing methods in Section 4.1.2 are listed in Table 8. Based on the t-test, ρTl-LSTM significantly outperforms the involved competing methods ($p < 0.01$) on all the three data sets. Overall, the performances of the methods with character embedding are comparable to those of the methods with word embedding.

4.3.2 The effect of different parameters on ρ-hot encoding

Our ρ-hot encoding differs from one-hot encoding in two aspects. The first aspect is that the nonzero values in one-hot encoding are only equal to 1, whereas the nonzero values in ρ-hot encoding are ρ. The second aspect is that only one element in one-hot encoding is nonzero, whereas n elements in ρ-hot encoding are nonzero.

TABLE 8
The classification accuracies of competing methods via 10-fold cross validation.

Data corpus	Travel	Hotel	Mobile
CNN-Lex-C (R+C)	0.751	0.755	0.751
CNN-Lex-W (R+C)	0.766	0.769	0.763
Bi-LSTM-C (R+C)	0.768	0.765	0.801
Bi-LSTM-W (R+C)	0.751	0.778	0.803
ρTl-LSTM-C	0.812	0.829	0.831
ρTl-LSTM-W	0.801	0.815	0.841

In this experiment, we test whether ρ-hot encoding is useful in two experimental runs. In the first run, the value of ρ is manually set to 0.5 and 1 in the experimental run without optimization. The parameter n in Eq. (6) is set as 15. The classification accuracies vary according to different ρ values on all the three data corpora. When ρ equals 1, the accuracies are the lowest in most cases shown in Fig. 8.

The results shown in Fig. 8 indicate that the value of ρ does affect the performance of the entire network. Consequently, the classical one-hot encoding, which fixes the value of nonzero elements as 1, is ineffective. In our experiments, the learned value of ρ is approximate 0.4.

In the second run, the performances under different n (i.e., 1, 5, 10, 15) are tested. Table 9 shows the comparison results. The value of n does affect the performance of the entire network, thereby indicating that the introduction of the n-duplicated strategy in encoding is effective. In the experiments, when n is increasing, the accuracies first increase and then decrease. The main reason may lie in the fact that when n becomes large, the proportion of lexicon embedding becomes large accordingly. An over-length input feature vector may incur 'curse of dimensionality' and thus weaken the performance of the proposed two-level network.

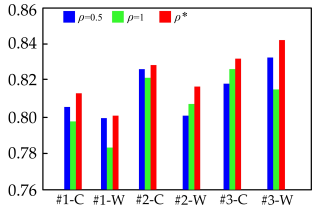


Figure 8. Classification accuracies under different ρ values. #1-C and #1-W represent ρTl-LSTM-C and ρTl-LSTM-W on the first (travel) corpus, respectively; #2-C and #2-W represent ρTl-LSTM-C and ρTl-LSTM-W on the second (hotel) corpus, respectively; #3-C and #3-W represent ρTl-LSTM-C and ρTl-LSTM-W on the third (hotel) corpus, respectively. ρ^* is the searched optimal value.

4.3.3 The effect of polar words

In this experimental run, we test whether the labeled polar (negative and positive) words do affect the performance of the entire method when they are used in lexicon embedding. To this end, we order the polar words according to their frequencies in the training data. Top 0%, 50%, 100% polar words are used. The corresponding classification accuracies are depicted in Fig. 9.

TABLE 9
The accuracies of ρTl-LSTM with different n values in ρ-hot encoding.

	n	Travel	Hotel	Mobile
ρTl-LSTM-C	1	0.790	0.814	0.809
	5	0.803	0.821	0.815
	10	0.798	0.815	0.820
	searched	0.812	0.829	0.831
ρTl-LSTM-W	1	0.796	0.798	0.820
	5	0.802	0.809	0.829
	10	0.800	0.801	0.820
	searched	0.801	0.815	0.841

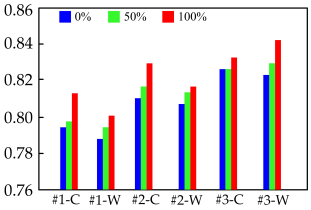


Figure 9. Classification accuracies under different proportions of polar words. #1-C and #1-W represent ρTl-LSTM-C and ρTl-LSTM-W on the first (travel) corpus, respectively; #2-C and #2-W represent ρTl-LSTM-C and ρTl-LSTM-W on the second (hotel) corpus, respectively; #3-C and #3-W represent ρTl-LSTM-C and ρTl-LSTM-W on the third (hotel) corpus, respectively.

In most cases, the accuracies are the lowest when no polar words are used in the lexicon embedding. When all polar words are used, the proposed network achieves the highest accuracies.

In the experiment, only one user is invited to manually compile the dictionary for a data corpus. One and a half hour is needed for each data corpus. In our viewpoint, it is worth manually compiling the polar words for sentiment analysis by considering the performance improvement and time-consumption.

4.3.4 The effect of POS cues

In this experimental run, we test whether POS cues do play positive roles in the entire model. To this end, we remove POS in the lexicon embedding of the proposed method. The results are shown in Fig. 10.

In most cases, the accuracies with POS embedding are greater than those without POS embedding, thereby indicating that the application of POS to lexicon embedding is useful.

4.3.5 The effect of conjunction cues

In this experimental run, we test whether conjunction cues do play positive roles in the entire model. To this end, the lexicon embedding for conjunction words is also removed from the proposed method. The results are shown in Fig. 11.

The algorithm with conjunction embedding outperforms that without conjunction embedding consistently, thereby indicating that the application of conjunction to lexicon embedding is useful.

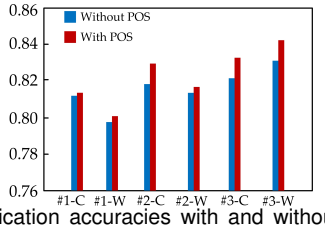


Figure 10. Classification accuracies with and without POS in lexicon embedding. #1-C and #1-W represent ρTl-LSTM-C and ρTl-LSTM-W on the first (travel) corpus, respectively; #2-C and #2-W represent ρTl-LSTM-C and ρTl-LSTM-W on the second (hotel) corpus, respectively; #3-C and #3-W represent ρTl-LSTM-C and ρTl-LSTM-W on the third (hotel) corpus, respectively.

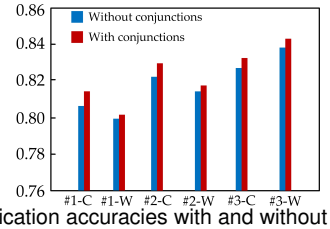


Figure 11. Classification accuracies with and without conjunction in lexicon embedding. #1-C and #1-W represent ρTl-LSTM-C and ρTl-LSTM-W on the first (travel) corpus, respectively; #2-C and #2-W represent ρTl-LSTM-C and ρTl-LSTM-W on the second (hotel) corpus, respectively; #3-C and #3-W represent ρTl-LSTM-C and ρTl-LSTM-W on the third (hotel) corpus, respectively.

4.3.6 The effect of different training tricks

In this experimental run, we test the main competing methods under different training tricks including dropout, batch normalization (BN), and pooling. Table 10 presents the accuracies of ρTl-LSTM based on 10-cross validation. These two training tricks do not improve the performances. Table 11 presents the accuracy comparison between average-pooling and max-pooling when CNN-lex is used. The average-pooling significantly outperforms max-pooling ($p < 0.01$).

TABLE 10
The classification accuracies with different training tricks.

Data corpus	Travel	Hotel	Mobile
ρTl-LSTM-C+BN	0.800	0.817	0.830
ρTl-LSTM-W+BN	0.791	0.804	0.825
ρTl-LSTM-C+droput	0.796	0.820	0.819
ρTl-LSTM-W+dropout	0.795	0.801	0.832
ρTl-LSTM-C+BN+droput	0.793	0.808	0.823
ρTl-LSTM-W+BN+dropout	0.794	0.809	0.832

TABLE 11
The classification accuracies with two pooling strategies.

Data corpus	Travel	Hotel	Mobile
CNN-Lex-C (R+C) with average	0.768	0.767	0.767
CNN-Lex-C (R+C) with max	0.751	0.755	0.751
CNN-Lex-W (R+C) with average	0.779	0.773	0.778
CNN-Lex-W (R+C) with max	0.766	0.769	0.763

4.3.7 Summarization of the experimental comparisons

According to above experimental comparisons, our proposed methodology has the follows advantages:

- The two-stage labeling strategy can provide more supervised information which is useful for model training.
- The proposed ρChot encoding is more flexible than the one-hot encoding and is thus more useful for cues embedding.
- The embedding of more lexical cues integrates more useful information related to sentiment orientations. The flipping module is also helpful in performance improvement.

5 Conclusion

High-quality labels are crucial for learning systems. Nevertheless, texts with mixed sentiments are difficult for humans to label in text sentiment classification. In this study, a new labeling strategy is introduced to partition texts into those with pure and mixed sentiment orientations. These two categories of texts are labeled using different processes. A two-level network is accordingly proposed to utilize the two labeled data in our two-stage labeling strategy. Lexical cues (e.g., polar words, POS, conjunction words) are particularly useful in sentiment analysis. These lexical cues are used in our two-level network, and a new encoding strategy, that is, ρ-hot encoding, is introduced. ρ-hot encoding is motivated by one-hot encoding. However, the former alleviates the drawbacks of the latter. Due to labeling noise or context, the polarity of a word varies in different texts. A flipping model is proposed to model the polarity flipping process. Three Chinese sentiment text data corpora are compiled to verify the effectiveness of the proposed methodology. Our proposed method achieves the highest accuracies on these three data corpora. On English data corpora, the proposed method outperforms state-of-the-art algorithms.

The proposed two-level network and lexicon embedding can also be applied to other types of deep neural networks. In our future work, we will extend our main idea into several networks and text mining applications.

References

[1] B. Liu, *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 2015.

[2] A. Pak and P. Paroubek, "Twitter as a corpus for sen-timent analysis and opinion mining," in *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, 2010, pp. 17–23.

[3] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of International Conference on Empirical Methods in Natural Language (EMNLP)*, 2014, pp. 1746–1751.

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

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

[6] Z. Teng, D.-T. Vo, and Y. Zhang, "Context-sensitive lexicon features for neural sentiment analysis," *Proc. EMNLP*, vol. 1629–1638, 2016.

[7] Q. Qian, M. Huang, J. Lei, and X. Zhu, "Linguistically regularized lstms for sentiment classification," *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1679–1689, 2017.

[8] W. Zhao, Z. Guan, L. Chen, X. He, D. Cai, B. Wang, and Q. Wang, "Weakly-supervised deep embedding for product review sentiment analysis." *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 1, pp. 185–197, 2018.

[9] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of ACM SIGKDD International Conference Knowledge Discovery Data Mining (KDD)*, 2004, pp. 168–177.

[10] T. Mullen and N. Collier, "Sentiment analysis using support vector machines with diverse information sources," in *Proceedings of International Conference on Empirical Methods in Natural Language (EMNLP)*, 2004, pp. 412–418.

[11] G. Paltoglou and M. Thelwall, "A study of information retrieval weighting schemes for sentiment analysis," in *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, 2010, pp. 1386–1395.

[12] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of International Conference on Machine Learning (ICML)*, 2011, pp. 513–520.

[13] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective lstms for target dependent sentiment classification," in *Proceedings of International Conference on Computational Linguistics (COLING)*, 2016, pp. 3298–3307.

[14] A. E. Baccianella, Stefano, and F. Sebastiani, "Senti-wordnet 3.0: An enhanced lexical resource for sentiment analy-sis and opinion mining," in *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, 2010, pp. 2200–2204.

[15] S. Lai, K. Liu, L. Xu, and J. Zhao, "How to generate a good word embedding," *IEEE Intelligent Systems*, vol. 31, no. 6, pp. 5–14, 2015.

[16] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of International Conference on Empirical Methods in Natural Language (EMNLP)*, 2014, pp. 1724–1734.

[17] M.-C. Yang, N. Duan, M. Zhou, and H.-C. Rim, "Joint relational embeddings for knowledge-based question answering," in *Proceedings of International Conference on Empirical Methods in Natural Language (EMNLP)*, 2014, pp. 645–650.

[18] A. M. Dai, C. Olah, and Q. V. Le, "Document embedding with paragraph vectors," *CoRR*, vol. abs/1507.07998, 2015.

[19] D. Tang, B. Qin, and T. Liu, "Document modeling with gat-ed recurrent neural network for sentiment classification," in *Proceedings of International Conference on Empirical Methods in Natural Language (EMNLP)*, 2015, pp. 1422–1432.

[20] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014, pp. 655–665.

[21] R. Socher, C. C.-Y. Lin, A. Y. Ng, and C. D. Manning, "Parsing natural scenes and natural lan-guage with recursive neural networks," in *Proceedings of International Conference on Machine Learning (ICML)*, 2011, pp. 129–136.

[22] M. Zhang, Y. Zhang, and D. Vo, "Gated neural networks for targeted sentiment analysis," in *Proceedings of AAAI Conference (AAAI)*, 2016, pp. 3087–3093.

[23] C. Santos, N. Dos, and M. Gattit, "Deep convolu-tional neural networks for sentiment analysis of short texts," in *Proceedings of International Conference on Computational Linguistics (COLING)*, 2014, pp. 69–78.

[24] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with lstm," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[25] X. Zhou, X. Wan, and J. Xiao, "Attention-based lstm network for cross-lingual sentiment classification," in *Proceedings of International Conference on Empirical Methods in Natural Language (EMNLP)*, 2016, pp. 247–256.

[26] B. Shin, T. Lee, and J. D. Choi, "Lexicon integrated cnn models with attention for sentiment analysis," in *Proceedings of WASSA@EMNLP*, 2017, pp. 149–158.

[27] K. Schouten and F. Frasincar, "Survey on aspect-level sentiment analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 813–830, 2016.

[28] K. Schouten, O. van der Weijde, F. Frasincar, and R. Dekker, "Supervised and unsupervised aspect category detection for sentiment analysis with co-occurrence data," *IEEE Transactions on Cybernetics*, vol. 48, no. 4, pp. 1263–1275, 2018.

[29] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based lstm for aspect-level sentiment classification," *Proc. EMNLP*, vol. 606–615, 2016.

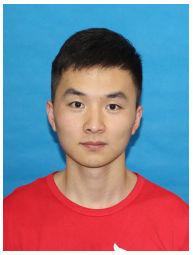
[30] L. He, K. Lee, M. Lewis, and L. Zettlemoyer, "Deep semantic role labeling: What works and what's next," in *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017, pp. 473–483.

[31] B. Wang and W. Lu, "Learning latent opinions for aspect-level sentiment classification," in *Proceedings of AAAI Conference (AAAI)*, 2018, pp. 5537–5544.
[32] R. Narayanan, B. Liu, and A. Choudhary, "Sentiment analysis of conditional sentences," in *Proceedings of International Conference on Empirical Methods in Natural Language (EMNLP)*, 2009, pp. 180–189.
[33] E. Cambria, S. Poria, A. F. Gelbukh, and M. Thelwall, "Sentiment analysis is a big suitcase," *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 74–80, 2017.
[34] X. Ding and B. Liu, "The utility of linguistic rules in opinion mining," in *Proceedings of the 30th Annual international ACM SIGIR Conference (SIGIR)*, 2007, pp. 811–812.
[35] Y. Duan and O. Wu, "Learning with auxiliary less-noisy labels," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 28, no. 7, pp. 1716–1721, 2017.
[36] E. Dobriban and S. Wager, "High-dimensional asymptotics of prediction: Ridge regression and classification," *The Annals of Statistics*, vol. 46, no. 1, pp. 247–279, 2018.
[37] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of CNN and RNN for natural language processing," *CoRR*, vol. abs/1702.01923, 2017.
[38] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 115–124, 2005.
[39] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, , and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," *Proc. EMNLP*, vol. 1631–1642, 2013.
[40] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1556–1566, 2015.

Ou Wu received the BSc degree in Electrical Engineering from Xi'an Jiaotong University, China, in 2003, and the MSc degree and the PhD degree in Computer Science from the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, China, in 2006 and 2012. From April 2007, he joined NLPR as an Assistant Professor. From February 2017, he joined Center for Applied Mathematics (CAM), Tianjin University as a full Professor. His research interests include data mining and machine learning.

Tao Yang received the B.Eng. degree in electronic information engineering from Civil Aviation University of China, Tianjin (CAUC), China, in 2016. Since Sept. 2016, he has been in M.Eng. degree program of 3 years in CAUC and majored in control engineering. Now he is a research intern in Center for Applied Mathematics (CAM), Tianjin University, China. His research interests include text mining and deep learning.

Mengyang Li received the B.Eng. degree in automation from Zhengzhou University of Aeronautics, Zhengzhou(ZUA), China, in 2015. Since Sept. 2015, he has been in M.Eng. degree program of 3 years in electronic information and automation college of Civil Aviation University of China, Tianjin(CAUC),and majored in pattern recognition and intelligent control. Now he is a research intern in Center for Applied Mathematics (CAM), Tianjin University, China. His research interests include text mining and deep learning.



Ming Li received The B.Eng degree in automation from Zhengzhou University of Aeronautics, Zhengzhou(ZUA), China, in 2015. Since Sept. 2016, he has been in M.Eng degree program of 3 years in Civil Aviation University of China Tianjin (CAUC),and majored in control science and engineering. Now he is a research intern in Center for Applied Mathematics (CAM), Tianjin University, China. His research interests include text mining and deep learning.