Sparse Self-Attention LSTM for Sentiment Lexicon Construction

Dong Deng, Liping Jing, Jian Yu, and Shaolong Sun

***Abstract*—Sentiment lexicon is a very important resource for opinion mining. Recently, many state-of-the-art works employ deep learning techniques to construct sentiment lexicons. In general, they firstly learn sentiment-aware word embeddings, and then use it as word features to construct sentiment lexicons. However, these methods do not consider the importance of each word to the distinguish of documents' sentiment polarities. As we know, most words among a document do not contribute to understand documents' semantic or sentiment. For example, in the tweet *It's a good day, but i can't feel it. I'm really unhappy.* The words 'unhappy', 'feel' and 'can't' are much more important than the words 'good', 'day' in predicting the sentiment polarity of this twitter. Meanwhile, many words, such as 'the', 'in', 'it' and 'I'm' are uninformative. In this paper, we propose a novel sparse self-attention LSTM (*SSALSTM*) to efficiently capture the above intuitive facts, and then construct a large scale sentiment lexicons in twitter. In SSALSTM, we use a novel self-attention mechanism to capture the importance of each words to the distinguish of documents' sentiment polarities. In addition, a L_1 regularize is applied in the attentions which can ensure the sparsity characters that most words in a document are semantic and sentiment indistinguishable. Once we learn an efficient sentiment-aware word embedding, we train a classifier which uses sentiment-aware word embedding as features to predict the sentiment polarities of words. Extensive experiments on four publicly available datasets, SemEval 2013–2016, indicate that the sentiment lexicon generated by our proposed model achieves state-of-the-art performance on both supervised and unsupervised sentiment classification tasks.**

***Index Terms*—Sentiment lexicon, sentiment analysis, sparsity, deep learning, text mining.**

I. Introduction

With the rapid growth of social media, delivering, searching or sharing opinions on-line have become a common activate in our daily life. Such huge opinionated data contains much valuable information for content providers, governments, sellers, etc. For instance, we can review lots of on-line comments on a specific dress before we determine to buy it. Companies can also directly collect an abundant of publicly information from networks instead of conducting opinion polls towards to their services or products. Thus, it's important to extract opinions from such data which contributes to know more about the preferences or intentions of users. Sentiment analysis, as a technique to extract opinions, contains many related tasks, such as document-level sentiment classification, aspect-level sentiment classification, sentiment lexicon construction and so on [1]–[3]. Among all these tasks, sentiment lexicon is primary and valuable, because it can provide a prior knowledge of many sentiment words to other tasks, such as *good* is a positive word and *bad* is a negative word.

Sentiment lexicon is an importance resource which contains many words or phrases with their sentiment polarities. It plays a crucial rule in many sentiment analysis tasks. So far, sentiment lexicons can be roughly classified into two categories, domain independent and domain specific lexicons. The first one includes many existing manually annotated sentiment lexicons, like MPQA [4], GI [5], HowNet [6], the Bing Liu's lexicon [7], or lots of lexicons using diverse co-occurrence information between words and sentiment labels [8]–[14]. Manually constructed lexicons usually have a high precision, however, they are also labour intensive, time consuming and not big enough in many applications. In contrast, Lexicons, constructed through co-occurrence information, are usually much bigger. Although they don't have a high precision, they can get a good recall and capture many implicit sentiment words. Thus, recently, more and more researchers construct domain independent lexicons through co-occurrence information among documents. The second category includes some lexicons which usually utilize domain specific features [15]–[17]. These methods can discover much more domain specific sentiment words, such as *warm*, *family*, *cloudy* and so on. However, they also need much more domain specific datasets and can't apply well in other domains. In this paper, we aim at constructing a large scale domain independent sentiment lexicons.

In order to construct domain independent sentiment lexicons, many approaches have been proposed. Recently, with the rapid growth of deep learning in natural language processing, many works have been proposed [14], [18]–[21]. Although such methods can extract much more contextual semantic relations between words. They ignore an intuitive fact that only a few words contribute to the distinguish of documents' sentient polarities.

Manuscript received November 24, 2018; revised March 30, 2019 and June 11, 2019; accepted July 19, 2019. Date of publication August 5, 2019; date of current version August 15, 2019. This work was supported in part by the National Natural Science Foundation of China under Grants 61822601, 61773050, and 61632004; in part by the Beijing Natural Science Foundation under Grant Z180006; in part by the Beijing Municipal Science & Technology Commission under Grant Z181100008918012; and in part by National Key Research and Development Program under Grant 2017YFC1703506. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Min Zhang. *(Corresponding author: Liping Jing.)*

D. Deng, L. Jing, and J. Yu are with the School of Computer and Information Technology, Beijing Key Lab of Traffic Data Analysis and Mining, Beijing Jiaotong University, Beijing 100044, China (e-mail: 15112074@bjtu.edu.cn; lpjing@bjtu.edu.cn; jianyu@bjtu.edu.cn).

S. Sun is with the School of Management, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: sunshl@amss.ac.cn).

Digital Object Identifier 10.1109/TASLP.2019.2933326

many words are uninformative, or even convey the opposite sentiments. Thus, it's very important to design a model which can capture the importance of each word to the documents. Furthermore, we also need to keep the sparsity that only a few words works for classifying the documents' sentiment labels.

In this paper, in order to solve or alleviate the above problems, we propose a novel sparse self-attention LSTM to construct a large scale domain independent sentiment lexicon. Firstly, we design a self-attention layer to capture the weight of each word in documents. Then, a L_1 regularization is applied in these weights to ensure that only a minority of words are used to distinguish the sentient polarities of documents. After we obtain a sentiment-aware word embedding, we follow the strategy in [18] that builds a classifier to convert learnt word embeddings to a sentiment lexicon.

In sum, the main contribution of this work are as follows:

- In this paper, we use a novel self-attention mechanism to construct sentiment lexicon. Different from the most existing sentiment lexicon construction approaches, SSALSTM does not equally treat each word in a document. It can efficiently focus on those sentiment and semantic important words to construct lexicons.
- Different from the standard attention mechanism, our proposed self-attention mechanism applies the weights into the word embeddings of the input words, which can guarantee that we can learn an efficiently sentiment-aware word embedding. When we learn such word embeddings, we can easily construct a large scale sentiment lexicon.
- As we know, SSALSTM is the first work which considers the sparse character of sentiment words in a document. In SSALSTM, we intuitively observe that only a few sentiment words contribute to distinguishing the sentiment polarities of documents, By introducing a $L1$ regularizer into the attention mechanism, we can capture the sparse character of sentiment words in a document.
- Experiments over SemEval 2013–2016 datasets indicate that SSALSTM achieves the state-of-the-art performance on both supervised and unsupervised binary sentiment classification task.

The rest of this paper is organized as follows. Section II introduces some related work. Section III describes the proposed SASLSTM model and its training process. Section V shows how to obtain the training data and how to construct the sentiment lexicon based on our proposed method. Meanwhile, a series of experiments on SemEval 2013–2016 datasets are also conducted in this section. The results have shown that the proposed SASLSTM lexicon gives better accuracies on these applications comparing with the state-of-the-art lexicons. A brief conclusion and future work are given in Section VI.

II. Related Work

Automatic domain independent sentiment lexicon construction is recently popular in sentiment analysis, many approaches have been proposed. They can be roughly classified into two categories: *dictionary-based* and *corpus-based* approach.

Dictionary-based approaches are usually based on an existing dictionary (e.g., WordNet [22]). Such dictionaries contain words' synonyms and antonyms. In detail, a small set of sentiment seeds with known positive or negative orientations are firstly collected manually. Secondly, they expand this set by searching in the dictionary for their synonyms and antonyms. Then, the newly discovered words are added to the seed list and the next iteration begins. Finally, the iterative process ends with no more words added to the seed list. Based on this intuitive techniques, approaches in [7], [23]–[25] use different strategies to clean up or expand the resulting seed list, and then get a sentiment lexicon. Except for these methods, some approaches are proposed to use more sophisticated mechanisms. For instance, [11] uses a WordNet distance based technique to determine the sentiment orientation of a given adjective. The method in [26] exploit a directed, weighted semantic graph where neighbouring nodes are synonyms or antonyms of words in WordNet. In [27], it uses WordNet synonyms and homonyms to build a word relatedness graph and apply random walk to construct a sentiment lexicon. *Dictionary-based* methods heavily rely on the quality of existing dictionary. Furthermore, with the ubiquitous domain diversity, it cannot capture the domain-specific implicit sentiment words.

Corpus-based approaches usually originate from the assumption [28] that words appearing in similar contexts have similar meanings. Based on this intuitive assumption, in [29], the authors firstly define a set of seeds, it calculates point-wise mutual information (PMI) between the candidate words and the seeds. Then, these words' sentiment orientation (SO) are computed by the difference of PMI score between positive and negative seed words. Furthermore, with the rapid growth of social media, short texts, like tweets, microbiology, are shown to contain a few emoticons which may convey different sentiment polarities. Recent work [13] utilizes such emoticons as distant supervision and calculates PMI between words and the distant supervised sentiment labels. By using this technique, it constructs a sentiment lexicon and achieves a good performance in tweet sentiment classification task. Recently, due to the capability of deep neural network for representation learning, many works employ deep learning techniques to construct sentiment lexicon. In [14], the authors utilizes the popular Skip-Gram neural language model to capture the semantic relations between words and integrate the sentiment information of tweets into its hybrid loss function to learn sentiment-aware embeddings. A similar work in [18] use both document-level and word-level supervision to learn word embeddings, and then construct the sentiment lexicon using the same way. On the other hand, the authors in [19], [20] utilize neural network to directly learn the sentiment polarity of words.

Although above sentiment lexicons have shown good performance in some tasks, they ignore an intuitive fact that different words in a sentence or document usually has different importance to the distinguish of documents' sentiment polarities. Furthermore, most of the time, majority of words in a sentence is uninformative and do not have an impact on the sentiment. Thus, in this paper, we propose a novel sparse self-attention LSTM to construct sentiment lexicons which can fully take advantage of the importance of each word to the sentiment polarities of documents. It will firstly learn a sentiment-aware word embeddings, and then the sentiment strength of each word is predicted by a classifier which is modelled through the learnt word embeddings and some prior known sentiment words.

TABLE I
NOTATION USED THROUGHOUT THE PAPER

Symbol	Description
N	number of documents in corpus.
V	number of unique words in the vocabulary.
y^k	sentiment label of the k-th document.
d^k	hidden representation of k-th document in corpus.
$d^{k'}$	weighted hidden representation of k-th document in corpus.
h_i^k	hidden representation of i-th word in k-th document in LSTM
w_i^k	the i-th word in k-th document.
α	importances or weights of words in a document.
a_{pos}	bias for positive words.
a_{neg}	bias for negative words.
$S(wi)$	the semantic similar words in Urban Dictionary.
$Pos(S(wi))$	the number of positive words in $S(w_i)$.
$Neg(S(wi))$	the number of negative words in $S(w_i)$.
$\hat{y}_i$	the predicted sentiment polarities of word w_i.
y_i	the true sentiment polarities of word w_i.

III. SPARSE SELF-ATTENTION LSTM FOR AUTOMATIC SENTIMENT LEXICON CONSTRUCTION

Different words among a document have different importance to document's semantic and sentiment. Thus, it's important to consider or distinguish the effect of each word in predicting the sentiment polarities of documents. In this paper, we design a self-attention [30], [31] mechanism to capture the weight of each word. Those words which are unimportant are marked with small weights. Furthermore, we apply a L_1 regularize to these weights which can ensure that most of uninformative words are ignored. In this section, we'll describe how we design a novel sparse self-attention LSTM to obtain a sentiment-aware word embeddings. Then, a large scale domain-specific sentiment lexicon can be constructed by exploiting learnt word embeddings and some seed words.

A. Premiliary

Let $\mathcal{D} = \{d_1, \ldots, d_k, \ldots, d_N\}$ be a collection of training data, where N is the size of documents in corpus, d_k is the k-th document. $d_k = w_1^k, \ldots, w_i^k, \ldots, w_{n_k}^k$, where n_k represents the length of k-th document, w_i^k indicates the i-th word in k-th document. The size of unique word in dictionary is V. The notation is summarized in Table I.

B. Architecture of Sparse Self-Attention LSTM

The architecture of sparse self-attention LSTM is shown in Fig. 1. The whole architecture contains four parts which correspond to the input layer, LSTM layer, sparse self-attention layer and prediction layer. The input layer maps each word to a low dimensional real-value vector. The LSTM layer are aimed to capture the contextual semantic relations between words. Sparse self-attention layer are used to capture the weight of each word in documents and ensure the sparsity character of these weights. Then, a weight summation of all words in . Once we obtain a representation of the whole document, the sentiment polarity of this document is predicted through a fully connected layer. The detail description of each part is given as follows.

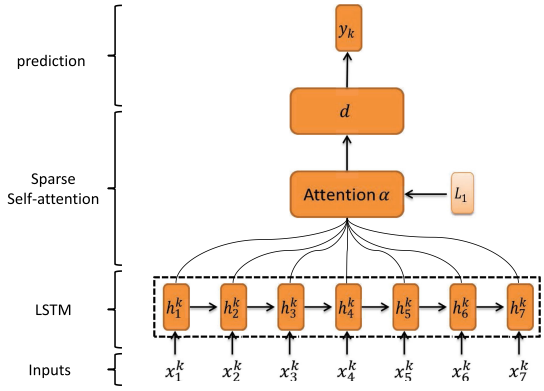


Fig. 1. The architecture of the proposed model, SSALSTM. It contains four parts.

C. Input Layer

For k-th document in the corpus, $d_k = w_1^k, \ldots, w_i^k, \ldots, w_{n_k}^k$, we use d_k as our input. In this paper, each word is represented by word embedding which is a recently popular representation technique. It maps each word to a dense low dimensional real-value vector. Compared with traditional *bag-of-words* representation, word embedding can capture more semantic and syntactic information among words. Let $x_i^k \in \mathcal{R}^C$ be a C-dimensional word embedding corresponding to the i-th word w_i^k in the k-th document. As shown in Fig. 1, the k-th document contains seven words $w_1^k, w_2^k, \ldots, w_7^k$. They are represented as word embedding $x_1^k, x_2^k, \ldots, x_7^k$ which are input to the LSTM layer.

D. LSTM Layer

Long Short Term Memory(LSTM) [32] has been proven to be an effective technique to model sequential data [33]. Different from the standard recurrent neural network [34], the architecture of LSTM contains a set of recurrently connected blocks which have complicated structures. Specifically, each block consists of one or more self-connected memory cells and three multiplicative units (the input, forget and output gates). The memory cells are some internal neurons which store the accumulated information in sequences. The input gate is used to extract information from the input, the forget gate is designed to filter information which is useless, the output gate is explicated to decide which part of information in memory cells to output. Such complicated structures in LSTM can efficiently capture the contextual semantic and syntactic relations between words which are even far from each other. In detail, the calculation process of LSTM neural network can be shown as follows:

$$\text{forget}_i = \text{sigmoid}(W_f \cdot [h_{i-1}, x_i] + b_f) \quad (1)$$

$$\text{input}_i = \text{sigmoid}(W_i \cdot [h_{i-1}, x_i] + b_i) \quad (2)$$

$$\text{output}_i = \text{sigmoid}(W_o \cdot [h_{i-1}, x_i] + b_o) \quad (3)$$

$$\widetilde{C_i} = \tanh(W_C \cdot [h_{i-1}, x_i] + b_C) \quad (4)$$

$$C_i = \text{forget}_i * C_{t-1} + \text{input}_i * \widetilde{C_i} \quad (5)$$

$$h_i = \text{output}_i * \tanh(C_i) \quad (6)$$

where (W_f, b_f), (W_i, b_i) and (W_o, b_o) are the parameters of forget gate, input gate and output gate. W_C is the parameter of memory cell. From (1), (2), (3) and (4), we can observe that the memory cell is updated through the state of the last memory cell and the current input information. The sigmoid function is defined by

$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}} \tag{7}$$

In LSTM layer, sigmoid is the activation function of input gate, output gate and forget gate and tanh is the activation function of memory. The tanh function is defined by:

$$\tanh(z)_j = \frac{e^{z_j}}{\sum_{j=1}^{L}}, z = (z_1, \ldots, z_j, \ldots, z_L) \tag{8}$$

In *SSALSTM*, as shown in Fig. 1, we use h_i^k to represent the hidden representation of i-th word in k-th document in LSTM. Thus, the hidden representation of the whole sequence can be represented as follows:

$$d^k = [h_1^k, h_2^k, \ldots, h_{n_k}^k] \tag{9}$$

In SSALSTM, the representation of the whole sequence is used as input to the sparse self-attention layer.

E. Sparse Self-Attention Layer

Recently, attention mechanism have been popular introduced in many LSTM based neural networks [35]–[37]. It's used to capture the weights of words to the representation of the whole documents. In this paper, we use attention mechanism to show the intuitive facts that different words in a document have different importance to the distinguish of document's semantic or sentiment polarities. Specifically, given the document representation d^k calculated in (9), importance or weights of words are calculated through an attention mechanism as follows:

$$\alpha^k = \text{softmax}(W^1 \cdot d^k + b^1) \tag{10}$$

where W^1 indicates the weights of neural network and b^1 is the bias. $\alpha^k = [\alpha_1^k, \ldots, \alpha_i^k, \ldots, \alpha_{n_k}^k]$ represents the probability of each word contributing to the sentiment polarities of documents. The bigger the value of α_i^k, the more important the i-th word.

Furthermore, in order to ensure that only a few words which contribute to the semantic and sentiment of documents, we apply a L_1 regularization on α^k. In a word, the sparse self-attention mechanism can be formulated as follows:

$$|\alpha^k|_{L_1} = |\text{softmax}(W^1 \cdot d + b^1)|_1 \tag{11}$$

Through the sparse gate mechanism, our method can efficiently filter out those words which make no contribution to the distinguish of the documents.

F. Prediction Layer

Once we estimate the importance or weight of each word to the sentiment polarities of documents, a weighted summation of word embeddings in this document is formed to predict the document's sentiment label.

$$d^{k'} = \alpha_1^k x_1^k + \cdots + \alpha_i^k x_i^k + \cdots + \alpha_{n_k}^k x_{n_k}^k \tag{12}$$

where x_i^k denotes the embedding of the i-th word in k-th document, n_k indicates the length of the k-th document. Finally, we can get the output of the sparse self-attention layer.

$$\hat{y}^k = \text{softmax}(W d^{k'} + b) \tag{13}$$

where $\hat{y}^k$ is a 2-dimensional vector which indicates the predicted sentiment label of k-th document ((1,0) indicates the positive sentiment label and (0,1) denotes the negative sentiment label). We use cross entropy as our loss function. Finally, the objective function of SSALSTM is given by

$$\mathcal{J} = \sum_{k=1}^{N} y^k \log \hat{y}^k \tag{14}$$

G. Obtaining Sentiment-Aware Word Embedding

Our proposed approach, *SSALSTM*, is trained by minimizing Eq. 14. Once all of the parameter are estimated after training, each word's word embedding, $x_i^k \in \mathcal{R}^C$, contains much sentiment information from documents. Thus, $x_i^k \in \mathcal{R}^C$ is the sentiment-aware word embedding that we learn from *SSALSTM*.

H. Training Setting

In *SSALSTM*, the word embeddings and the parameters in different layers can be calculated by applying differentiation chain rule. Given a training set $\mathcal{D}$ consisting of N samples, we optimize the model through *Adam* over mini-batches. It's worth noting that although our proposed model, SSALSTM, is more complicated than the existing methods, like [14], [18], [20], we can solve 1.4 million tweets within 40 minutes on a CPU platform. Furthermore, when we run the same model and datasets on a GPU platform, we just use nearly 10 minutes.

IV. From Sentiment-Aware Word Embeddings to Sentiment Lexicon

Once we obtain sentiment-aware word embeddings, we use these word-level features to construct a sentiment lexicon. The construction process can be divided into two steps, the first step is to get enough training data which contains some already known positive and negative words. The second step is to build a classifier which predicts the sentiment strength of words by using the sentiment-aware word embeddings as features. Through this strategy, we can construct a large scale sentiment lexicon.

A. Training Data Construction

In is section, we propose three ways to construct training data for word-level sentiment annotation which are referred as *Seed*, *Lexicon* and *Urban*. Their corresponding statistics are shown in Table II. The following is the detail description about how to construct them.

1) Seed: Constructing From Seed Words: Seed is manually constructed and contains 125 positive words, 109 negative words. These words are commonly used in our daily life. This training data is the most easy way for us to obtain.

2) Lexicon: Constructing from Publicly Available Sentiment Lexicons: Lexicon comes from the publicly available sentiment

TABLE II
STATISTICS OF THE TRAINING DATA FOR WORD-LEVEL SENTIMENT LEXICON CONSTRUCTION

Trining data	#pos	#neg	Total
Seed	125	109	234
Lexicon	2003	2003	4006
Urban	6966	7549	14515

TABLE III
DATA SUMMARY OF SEMEVAL 2013–2016 DATASETS

Dataset	#pos	#neg	Total
SemEval2013-test	1474	559	2033
SemEval2014-test	982	202	1184
SemEval2015-test	1038	365	1403
SemEval2016-test	7059	3231	10290

lexicons which are also manually constructed. In this paper, *Lexicon* is constructed to use *Liu's* lexicon [7]. It's worth noting that, except for the *Liu's* lexicon, any publicly available sentiment lexicons can be used here. Meanwhile, the *Liu's* lexicon actually contains 2003 positive words and 4783 negative words. In order to keep balance between positive and negative words, we only randomly utilize 2003 negative words in *Lexicon*.

3) Urban. Constructing From Urban Dictionary: In order to get enough training data, *Urban* is constructed with the same strategy as that in [14] which expands seed words with Urban Dictionary. [1] Urban Dictionary is an online dictionary which contains more than ten million definitions. Different from the other web-based dictionary, such as WordNet [22], it was primary founded to collect slang or cultural words which are not typically found in standard dictionaries. For each token in Urban Dictionary, it also shows a list of semantic similar words contributed by volunteers. For instance, the similar words of "cool" are "coool", "awesome", "col", *et al.* These similar words are usually semantically close and tend to have the same sentiment polarities. Thus, it's an important resource for us to collect our training data. In this paper, we use the data released by [14] which contains nearly 800 thousands items from 'a' to 'z'. Furthermore, each of items has about 10 similar words on average.

In detail, we firstly use 125 positive and 109 negative words as seed which are manually labelled [14]. Secondly, a variant-KNN classifier is used to extend the positive and negative seed words based on the synset of Urban Dictionary. Specifically, we use w_i indicates the i-th unique word in the dictionary. $S(w_i) = w_1^i, w_2^i, \ldots, w_j^i$ represents the semantic similar words of w^i. Thus, Any word will be considered as positive words when the following condition is satisfied:

$$CPOS(w_i) = a_{pos} + Pos(S(w_i)) - Neg(S(w_i)) \quad (15)$$

where a_{pos} is a bias parameter for positive sentiment, it is usually a negative value. $Pos(S(w_i))$ and $Neg(S(w_i))$ indicates the number of positive and negative words in $S(w_i)$ respectively. It's worth noting that, if we do not explicitly know the sentiment polarity of a word in $S(w_i)$, it will not be considered. From (15), if the value of $CPOS(w_i)$ is positive, then the word w_i is a positive word. Meanwhile, Any word will be considered as negative words when the following condition is satisfied:

$$CNEG(w_i) = a_{neg} + Neg(S(w_i)) - Pos(S(w_i)) \quad (16)$$

where a_{nge} is a bias parameter for negative sentiment, it is usually a positive value. From (16), if the value of $CNEG(w_i)$ is positive, then the word w_i is a negative word.

[1]https://www.urbandictionary.com/

B. Classifier for Sentiment-Aware Word Embeddings

Once we learn the sentiment-aware word embeddings, we can build a classifier for sentiment lexicon construction. In this paper, we use the logistic regression. Specifically, for i-th word w_i in the vocabulary, it's represented by a C dimensional real-value vector, $x_i \in R^C$. Meanwhile, the true sentiment polarities of this word is represented as y_i. Thus, we define a classifier as follows:

$$\hat{y_i} = f(x_i) \quad (17)$$

In (17), $\hat{y_i}$ is the predicted value of sentiment polarities of w_i. f is a function which in this paper corresponds to a logistic regression classifier.

V. EXPERIMENTS

A. Datasets and Settings

We use the public available tweets[2] as our datasets to construct sentiment lexicon. It contains 0.8 million positive and 0.8 million negative tweets. We evaluate the F_1 and accuracy ($Accuracy$) score of these lexicons on the SemEval 2013–2016 datasets in positive and negative class respectively. The summary information of these dataset is sown in Table III. It's worth noting that, in order to satisfy space limitation, we also use **SemEval13-16**, to represent SemEval 2013–2016. Meanwhile, we do not include the training data of SemEval 2013–2016 into sentiment140.

B. Lexicon Based Supervised and Unsupervised Sentiment Classification

The resulting lexicons can be used to solve the sentiment classification tasks. In this paper, we evaluate the performance of different lexicons in both supervised and unsupervised binary sentiment classification.

1) Lexicon Based Unsupervised Sentiment Classification: The unsupervised sentiment classification is calculated by summing up the sentiment score of all words in a document. Then, the sentiment polarity of the document is obtained by checking the total sentiment strength. If the sum is greater than 0, the document will be considered as positive, otherwise, negative. We use F_1 and $Accuracy$ as our metrics.

2) Lexicon Based Supervised Sentiment Classification: To evaluate supervised sentiment classification, we follow the strategy in [18] which manually extracts some lexicon features to evaluate the performance of different sentiment lexicons.

- Non-zero score of the last negative word in the tweet;
- Non-zero score of the last positive word in the tweet;

[2]http://help.sentiment140.com/for-students

- Total count of words in the tweet score of which is less than 0;
- Total count of words in the tweet score of which is greater than 0;
- The sum of scores for all word less than 0;
- The sum of scores for all word great than 0;
- The min score less than 0;
- The max score greater than 0;

After we have such features of each document, a logistic regression classifier is exploited to test the performance. Specifically, given a set of document-label pair $(d_k, y_k), k = 1, \ldots, N, y_k \in \{-1, +1\}$, where d_k indicates the lexicon features extracted from the document and y_k is the sentiment label of k-th document. N is the number of documents in the corpus. $y_k = +1$ represents that the sentiment polarity of the k-th document is positve and $y_k = -1$ represents that the sentiment polarity of the k-th document is negative. Then, we utilize logistic regression to build classifier for supervised sentiment classification. Logistic regression can be defined as follows:

$$\min_w L = \sum_{k=1}^{N} \xi(w; d_k, y_k) + C\frac{1}{2}w^T w \tag{18}$$

where $\xi(w; d_k, y_k)$ is the loss function, $C > 0$ is a penalty parameter and w is the weight vector. Meanwhile, $\xi(w; d_k, y_k)$ is defined as follows:

$$\xi(w; d_k, y_k) = \log\left(1 + e^{y_k w^T d_k}\right) \tag{19}$$

It's worth noting that the weight variable w includes a bias term b, and we handle this term by augmenting the vector w and each instance d_k with an additional dimension, $w \leftarrow [w; b]$ and $d_k \leftarrow [d_k; 1]$.

C. Baselines

We compare the proposed SSALSTM with seven popular sentiment lexicons.

- MPQA [4]: MPQA is a manually annotated sentiment lexicons. It usually has high precision, but low recall.
- NRC [13]: The NRC lexicon is constructed by using hastag(#) within a tweet which usually indicates the sentiment, such as 'sadness', 'angry' are ordinary indicators to show the negative emotion of users. 'happy' and 'surprised' are expressing the positive emotion [38].
- Sentiment140 [13]: The sentiment140 lexicon is generated through the same way as NRC. The only difference is that they use the different tweet datasets. NRC uses the tweets collected by the Twitter API every four hours from April to December 2012. However, Sentiment140 is constructed from the setniment140 corpus [38] which contains 1.6 million tweets.
- HIT [14]: HIT is constructed by deep learning techniques which combine a Skip-Gram based neural language model and document-level sentiment classification. Once sentiment-aware word embeddings are generated, a logistic regression classifier is used to predict the sentiment polarities of unknown words.

TABLE IV
STATISTICS AND OVERLAP RATE OF THE EXISTING LEXICONS

Lexicon	#pos	#neg	Total	#overlap rate
MPQA	2289	4559	6848	0.07
NRC	32048	22081	54129	0.58
Sentiment140	38312	24156	62468	0.84
ETSL	706	571	1277	0.02
HIT	178781	171508	347626	0.45
NN	13071	171508	184579	0.99
HSSWE	27206	23685	50891	0.97
SSASLSTM	30526	21390	51916	1.00

- ETSL [39], [40]: ETSL refers to SemEval-2015 English Twitter Sentiment Lexicon, [3] which is constructed through Best-Worst Scaling.
- NN [20]: NN is also constructed from a deep learning based methods which uses a simple feed-forward neural network, and directly learns the sentiment polarities of words. Different from HIT, NN uses a two-dimensional vectors to represent each word. The first and second value in this vector are corresponding to the sentiment strength of positive and negative.
- HSSWE [18]: HSSWE is constructed by a neural network model which considers both the document-level and the word-level sentiment supervision.

It's worth noting that HIT, NN and HSSWE are three deep learning based methods. Sentiment140, NRC, and ETSL are three sentiment lexicons which are based on statistics information. MPQA is a manually labeled sentiment lexicon. Furthermore, For MPQA, if a word is labeled positive, the sentiment strength is set to 1, otherwise the sentiment strength of this word is set to -1. Meanwhile, in order to explicitly exhibit that *HSSWE*and *HSSWE* are also constructed with the help of Urban Dictioary, we use *HSSWE-urban* and *HIT-urban* to represent these two sentiment lexicons.

D. Existing Sentiment Lexicons Comparison

Existing sentiment lexicons, including **SSALSTM**, are constructed through different training data. But all of them are equally evaluated in four publicly available dataset. In this paper, the lexicons used in this paper have the different lexicon size and the overlap rate. As shown in Table IV, the left three columns show the statistics of the existing lexicons. the last column indicates their overlap rate with SSALSTM. Comparing with NN, HSSWE and Sentiment140, SSALSTM gets 99%, 97% and 84% overlap respectively. The reason is that the training data of NN, HSSWE, SSALSTM and Sentiment140 contain the same publicly available tweets,[4] Noting that NN also uses other tweets as training data, thus, NN contains much more sentiment words than HSSWE and SSALSTM. Meanwhile, MPQA and ETSL get a very low overlap with SSALSTM, because these two lexicons only contain thousands of words. Other lexicons, including NRC, NRC and HIT, also get a nearly 50% overlap with SSALSTM, because these lexicons are all constructed

[3]http://www.saifmohammad.com/WebDocs/lexiconstoreleaseonsclpage/SemEval2015-English-Twitter-Lexicon.zip

[4]http://help.sentiment140.com/for-students

TABLE V
SUPERVISED AND UNSUPERVISED SENTIMENT CLASSIFICATION ON SEMEVAL2013 (Pre, $Recall$, F_1 AND Acc SCORE ON BOTH POSITIVE AND NEGATIVE SENTIMENT)

Methods		Positive			Negative			All	
		Precision	Recall	F1	Precision	Recall	F1	Accuracy	Improvements
Supervised	**MPQA**	78.59	87.86	82.97	53.51	36.85	43.64	73.84	12.43%
	NRC	79.18	**93.36**	85.69	66.78	35.24	46.14	77.38	8.231%
	Sentiment140	83.90	90.10	86.89	67.56	54.38	60.26	80.29	4.779%
	ETSL	83.58	88.34	85.89	63.79	54.20	58.61	78.96	6.357%
	NN	84.79	90.31	87.46	69.11	57.25	62.62	81.22	3.676%
	SSALSTM-seed	87.76	88.95	88.35	69.76	67.26	68.49	82.99	1.577%
	SSALSTM-lexicon	86.65	88.00	87.32	66.98	64.22	65.57	81.47	3.380%
	HIT-urban	85.13	90.03	87.51	68.99	58.50	63.31	81.37	3.499%
	HSSWE-urban	87.86	89.29	88.57	**70.47**	67.44	68.92	83.28	1.233%
	SSALSTM-urban	**90.88**	87.12	**88.96**	69.35	**76.92**	**72.94**	**84.32**	-
Unsupervised	**MPQA**	80.12	78.44	79.27	46.10	48.66	47.35	70.26	14.22%
	NRC	85.16	80.14	82.57	54.64	63.15	58.59	75.47	7.862%
	Sentiment140	88.39	78.98	83.42	56.70	72.63	63.69	77.24	5.701%
	ETSL	83.29	88.54	85.84	63.73	53.13	57.95	78.81	3.785%
	NN	**90.34**	76.68	82.95	56.01	**78.35**	**65.32**	77.13	5.836%
	SSALSTM-seed	84.05	**91.80**	87.75	**71.39**	54.03	61.51	81.42	0.598%
	SSALSTM-lexicon	87.21	85.49	86.34	63.61	66.91	65.21	80.38	1.868%
	HIT-urban	86.19	86.71	86.45	64.36	63.33	63.84	80.29	1.978%
	HSSWE-urban	86.43	85.97	86.20	63.49	64.40	63.94	80.03	2.295%
	SSALSTM-urban	85.64	90.17	**87.85**	69.85	60.11	64.62	**81.91**	-

through tweets which are collected with different data size and date.

Meanwhile, from Table IV, we can easily observe that deep learning and most statistical based lexicons are usually much larger than that of manually constructed lexicons. Meanwhile, NN is much more imbalanced than other lexicons

E. Parameter Selection in Experiments

In this paper, The embedding size K, the parameters of deep neural network and the hyper-parameter of sentiment bias b_{pos} and b_{neg} are set through different strategies. For embedding size K, we set this value from 20 to 95 with a interval 15, then we choose the value which can generates the best performance in document sentiment classification task. The detail description of experiments about embedding size will be described in subsection V-K. As for sentiment bias a_{pos} and a_{neg}, we choose the values from two sets respectively, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and [0, −1, −2, −3, −4, −5, −6, −7]. Then we tune these two parameters in classification tasks which are the same as that of embedding size. When we obtain the best performance, we choose the corresponding values for a_{pos} and a_{neg}. For parameters of deep neural network, we use a single LSTM and set the hidden layer to 64. The coefficient of L_1 regularize is 0.05. Meanwhile, we set the maximum epochs to 10 and set the early stopping mechanism. In detail, when the performance of SSALSTM is not improved in the successive three times in validation datasets. we stop the programme.

F. Performance of Document Sentiment Classification

We report the supervised and unsupervised sentiment classification results on the SemEval 2013–2016 datasets in Table V, VI, VII and VIII respectively. The best results are highlighted in bold. In this section, we show the *Precision*, *Recall*, *F1* and *Accuracy* of all methods in both positive and negative sentiment. The last column, *Improvements*, shows the improvements of SSALSTM compared with methods.

Meanwhile, from Table V we can easily observe that the statistical based lexicons, NRC, Sentiment140, ETSL and deep learning based lexicons, HIT, NN, HSSWE and SSALSTM are better than manually constructed lexicons, MPQA, in both supervised and unsupervised classification tasks. It's mainly because that the manually annotated lexicons usually do not contain many implicit sentiment words which are important in distinguishing the sentiment polarities of documents. On the other hand, most of the deep learning based methods, HIT, HSSWE and SSALSTM perform better than statistical based methods, NRC, Sentiment140 and ETSL, except for NN. For instance, the average accuracy of deep learning based lexicons (including *HIT-urban*, *HSSWE-urban*, *NN* and *SSALSTM-urban*) is 82.54 and 79.41 which is better than that of statistical based lexicons (including Sentiment140, ETSL and NRC), 78.88 and 77.19 in supervised and unsupervised classification task respectively. The main reason is that statistical based methods, such as NRC and Sentiment140 only take advantage of the co-occurrence information between words and sentiment labels, however, deep learning based methods additionally learn word embedding which can efficiently capture the contextual sentiment and semantic relations among words. It's worth noting that NN do not show an obviously improvements over statistical based methods. It's because that NN automatically learn sentiment lexicons without relying on any manual resources or word embeddings. Almost all of the lexicons get a better performance on positive document

TABLE VI
SUPERVISED AND UNSUPERVISED SENTIMENT CLASSIFICATION ON SEMEVAL2014 (Pre, $Recall$, F_1 AND Acc SCORE ON BOTH POSITIVE AND NEGATIVE SENTIMENT)

Methods		Positive			Negative			All	
		Precision	Recall	F1	Precision	Recall	F1	Accuracy	Improvements
Supervised	**MPQA**	86.42	86.25	86.34	33.82	34.16	33.99	77.36	13.02%
	NRC	86.00	**96.95**	91.14	61.04	23.27	33.69	84.38	5.127%
	Sentiment140	92.35	93.38	92.86	65.97	62.38	64.12	88.09	0.958%
	ETSL	90.37	88.9	89.63	50.00	53.96	51.90	82.94	6.746%
	NN	92.15	93.18	92.66	64.92	61.39	63.10	87.75	1.338%
	SSALSTM-seed	93.46	91.65	92.55	62.90	68.81	65.72	87.75	1.338%
	SSALSTM-lexicon	92.84	89.82	91.30	57.27	66.34	61.47	85.81	3.519%
	HIT-urban	90.52	90.43	90.47	53.69	53.96	53.83	84.21	5.318%
	HSSWE-urban	92.36	92.36	92.36	62.87	62.87	62.87	87.33	1.810%
	SSALSTM-urban	**93.64**	92.97	**93.30**	**66.99**	**69.31**	**68.13**	**88.94**	-
Unsupervised	**MPQA**	89.49	79.74	84.33	35.60	54.46	43.05	75.42	13.89%
	NRC	89.65	89.10	89.38	48.56	50.00	49.27	82.43	5.880%
	Sentiment140	93.92	83.30	88.29	47.60	73.76	57.86	81.67	6.748%
	ETSL	90.22	89.21	89.71	50.23	52.97	51.57	83.02	5.207%
	NN	**95.45**	83.30	88.96	49.85	**80.69**	61.63	82.85	5.401%
	SSALSTM-seed	91.47	**95.01**	**93.21**	**70.12**	56.93	**62.84**	**88.51**	-1.062%
	SSALSTM-lexicon	92.86	88.70	90.73	54.88	66.83	60.27	84.97	2.980%
	HIT-urban	91.71	87.88	89.76	51.03	61.39	55.73	83.36	4.818%
	HSSWE-urban	92.45	91.04	91.74	59.45	63.86	61.58	86.40	1.347%
	SSALSTM-urban	90.73	94.70	92.68	67.30	52.97	59.28	87.58	-

TABLE VII
SUPERVISED AND UNSUPERVISED SENTIMENT CLASSIFICATION ON SEMEVAL2015 (Pre, $Recall$, F_1 AND Acc SCORE ON BOTH POSITIVE AND NEGATIVE SENTIMENT)

Methods		Positive			Negative			All	
		Precision	Recall	F1	Precision	Recall	F1	Accuracy	Improvements
Supervised	**MPQA**	79.60	84.20	81.84	46.23	38.63	42.09	72.35	9.608%
	NRC	81.74	88.82	85.13	57.82	43.56	49.69	77.05	3.736%
	Sentiment140	83.76	**90.46**	**86.98**	**64.89**	50.14	56.59	79.97	0.087%
	ETSL	81.89	86.71	84.23	54.61	45.78	49.63	75.98	5.072%
	NN	85.71	82.66	84.16	55.22	60.82	57.89	76.98	3.823%
	SSALSTM-seed	86.14	85.65	85.89	59.84	60.82	60.33	79.19	1.062%
	SSALSTM-lexicon	85.77	86.51	86.14	60.67	59.18	59.92	79.40	0.800%
	HIT-urban	83.81	87.76	85.74	59.81	51.78	55.51	78.40	2.049%
	HSSWE-urban	85.83	84.59	85.20	57.89	60.27	59.06	78.26	2.224%
	SSALSTM-urban	**87.82**	84.78	86.27	60.60	**66.58**	**63.45**	**80.04**	-
Unsupervised	**MPQA**	83.53	74.28	78.63	44.38	58.36	50.41	70.14	12.67%
	NRC	86.10	71.58	78.17	45.37	67.12	54.14	70.42	12.33%
	Sentiment140	87.77	78.13	82.67	52.61	69.04	**59.72**	75.77	5.665%
	ETSL	81.78	86.90	84.26	54.67	44.93	49.32	75.98	5.403%
	NN	**90.55**	65.51	76.02	45.09	**80.55**	57.82	69.42	13.57%
	SSALSTM-seed	82.98	**90.17**	86.43	62.91	47.40	54.06	79.04	1.594%
	SSALSTM-lexicon	85.84	84.10	84.96	57.25	60.55	58.85	77.98	2.913%
	HIT-urban	85.38	83.82	84.59	56.25	59.18	57.67	77.41	3.623%
	HSSWE-urban	85.59	85.84	85.71	59.39	58.90	59.15	78.83	1.855%
	SSALSTM-urban	84.89	**89.31**	**87.04**	**64.31**	54.79	59.17	**80.32**	-

classification than that on negative document classification. The main reason is that these lexicons do not consider the effect of imbalance data (It's common that capturing positive documents are easier than negative documents). In a word, our proposed method, *SSALSTM-urban* achieves nearly 12.43% and 14.22% improvements over manually constructed lexicons, MPQA, nearly 6.456% and 5.783% improvements over statistical based lexicons, NRC, Sentiment140 and ETSL in supervised and unsupervised tasks. Meanwhile, *SSALSTM = urban* also gains 1.233% and 2.295% improvements over deep learning based lexicon, *HSSWE-urban*. Meanwhile, if *SSALSTM* does not use the pre-knowledge information from Urban Dictionary, *SSALSTM-seed* and *SSALSTM-lexicon* still perform much better than other sentiment lexicons. For example, in supervised

TABLE VIII
SUPERVISED AND UNSUPERVISED SENTIMENT CLASSIFICATION ON SEMEVAL2016 (Pre, $Recall$, F_1 AND Acc SCORE ON BOTH POSITIVE AND NEGATIVE SENTIMENT)

Methods		Positive			Negative			All	
		Precision	Recall	F1	Precision	Recall	F1	Accuracy	Improvements
Supervised	**MPQA**	73.94	86.07	79.54	52.56	33.70	41.07	69.63	12.83%
	NRC	79.55	**92.32**	**85.46**	**74.17**	48.16	58.40	78.45	1.790%
	Sentiment140	78.04	90.98	84.01	69.09	44.07	53.82	76.25	4.544%
	ETSL	79.75	85.51	82.53	62.40	52.55	57.06	75.16	5.909%
	NN	80.23	87.09	83.52	65.32	53.11	58.59	76.42	4.331%
	SSALSTM-seed	82.66	85.71	84.16	66.04	60.72	63.27	77.86	2.529%
	SSALSTM-lexicon	82.39	86.37	84.34	66.71	59.67	63.00	77.99	2.366%
	HIT-urban	79.60	88.51	83.82	66.78	50.45	57.48	76.56	4.156%
	HSSWE-urban	81.53	87.56	84.44	67.59	56.67	61.65	77.86	2.529%
	SSALSTM-urban	**85.55**	85.04	85.29	67.74	**68.62**	**68.17**	**79.88**	-
Unsupervised	**MPQA**	78.49	76.26	77.36	51.17	54.35	52.71	69.38	11.10%
	NRC	85.71	78.99	82.21	60.81	71.22	**65.6**	76.55	1.909%
	Sentiment140	82.13	79.35	80.71	57.98	62.27	60.05	73.98	5.202%
	ETSL	79.42	85.28	82.25	61.66	51.72	56.25	74.74	4.229%
	NN	**86.11**	70.51	77.53	53.84	**75.15**	62.73	71.96	7.791%
	SSALSTM-seed	78.23	**89.89**	83.65	67.23	45.34	54.16	75.90	2.742%
	SSALSTM-lexicon	81.61	82.96	82.28	61.37	59.15	60.24	75.48	3.280%
	HIT-urban	79.75	84.36	81.99	60.89	53.2	56.79	74.58	4.434%
	HSSWE-urban	79.69	86.56	82.98	63.82	51.81	57.19	75.65	3.063%
	SSALSTM-urban	80.89	89.04	**84.77**	**69.29**	54.04	60.72	**78.04**	-

sentiment classification tasks on *SemEval13* dataset, *SSALSTM-seed* and *SSALSTM-lexicon* achieve 82.99% and 81.47% in accuracy metric respectively. Except for *HSSWE-urban* and *SSALSTM-urban*, they got the best performance and obtain significant improvements over ETSL and MPQA. In unsupervised sentiment classification tasks on *SemEval13* dataset, *SSALSTM-seed* and *SSALSTM-lexicon* achieve 81.42% and 80.38% in accuracy metric respectively. They even outperform the *HIT-urban* and *HSSWE-urban*.

For SemEval 2014–2016 dataset, the experimental results are shown in Table VI, VII and VIII respectively. From these three tables, we can see nearly the similar observation shown in V, Our proposed lexicons, *SSALSTM-urban* still achieves the best performance when comparing both with the statistical based lexicons, manually constructed lexicons and the deep learning based lexicons, NN, *HIT-urban* and *HSSWE-urban*. Meanwhile, comparing with other sentiment lexicons which do not use Urban Dictionary, like ETSL, NRCMPQA and Sentiment140, *SSALSTM-seed* and *SSALSTM-lexicon* get better performance in most of time. It's also interesting to observe that NN can get a better Recall or F1 score in unsupervised classification tasks in Table V, VI, VII and VIII. It's because that NN contains much more negative sentiment words than positive sentiment words (nearly 10:1), However, our proposed lexicon, SASLSTM, still performs better than NN overall.

G. Comparing SSALSTM With Deep Learning Based Sentiment Lexicons

Compared to *HSSWE* and other methods, *SSALSTM* achieves state-of-the-art performance in all of the four datasets on overall accuracy. It's because that, firstly, *SSALSTM* can capture much more semantic relations among words than other models. In *SSALSTM*, we utilize Long Short-Term Memory (LSTM) neural network to model input document which can capture relations between words even they are far from each other. Meanwhile, LSTM also considers the sequence of words in documents. In contrast, *HSSWE* uses a simple feed forword neural network to model semantic relations among words. It cannot capture the semantic relations between words when they are far from each other and do not consider the sequence of words. Secondly, *SSALSTM* consider the importance of each word to distinguish the document's sentiment label, while *HSSWE* and other approaches treat each word equally. As we know, most words in a document do not contribute to understand the documents' sentiment or semantic. Thus, it's important to decrease the weights of those meaningless words and increase the weights of those important sentiment words. Thirdly, *SSALSTM* additionally apply a L_1 regularize to the weights of words in each document which can further filter out some meaningless words and decrease their interruption.

H. Attended Words by Self-Attention Mechanism

Our proposed method, SSALSTM, adopts self-attention mechanism to capture the importance or weights of words to the distinguish of documents' sentiment. The weights of words in k-th document is indicated by $\alpha^k = [\alpha_1^k, \ldots, \alpha_i^k, \ldots, \alpha_{n_k}^k]$. In order to demonstrate the effective of self-attention mechanism, 10 tweets (5 positive tweets and 5 negative tweets.) are selected in the training data. We rank the weights of words in each tweet and show the rank of top-5 words in Table IX. From Table IX, we can observe that SSALSTM can extract some meaningful or sentiment words in tweets and assign their a

TABLE IX
WE RANK THE WEIGHTS OF WORDS IN DOCUMENTS AND SHOW THE TOP 5 WORDS. THE FIGURE IN BRACKET SHOWS THEIR RANK IMPORTANCE TO THE DOCUMENTS' SENTIMENT

Label	Tweet
Positive	⟨**user**⟩**(2)** the concert was soooo **goood(3)** last night !!! had an **awesome(1) time(5) !!!!!(4)**
Positive	**look(5)** what came in the mail today **!(4) i(2) love(1) river(3)** ⟨url⟩
Positive	**finally(5)** got to **talk(4)** to today and so **excited(2)** about **next(1)** week **!!!!(3)**
Positive	**thanks(2) for(4)** my **new(5) followers(3) cheers(1)**
Positive	**proof(2) that(4) i'm(5)** a total **nerd(1)** : ⟨url⟩
Negative	**i(3)** think **i'm(4) getting(5) sick(2)** .... **fml(1)**
Negative	less than two hours in twitter and **i(4) already(2) have(3)** 2 **spam(1) messages(5)**
Negative	⟨**user**⟩**(5) lol(3) cool(4)** . **sry(2)** i **missed(1)** your party
Negative	also ; **i(3) have(4) noooo(1)** idea **what's(2)** going on right **now(5)**
Negative	⟨**user**⟩**(2) im(3) gonna(5) miss(1)** you too **!(4)**

TABLE X
TOP 20 SENTIMENT WORDS OF THREE DEEP LEARNING BASED LEXICONS SSALSTM, HSSWE AND NN

SSALSTM		HSSWE		NN	
Positive	Negative	Positive	Negative	Positive	Negative
smile	#inaperfectworld	proud	**sad**	#whereismike	#sad
congratulations	**sadd**	worries	**sadly**	#conceptfollowspree	#janoskiansnewsong
proud	#dontyouhate	**smile**	bummed	#16factsaboutme	heartbreaking
welcome	disapointed	congratulations	unfortunately	:d	himu00a0sad
pleasure	fawcett	**smiling**	gutted	d	maajjoorrlyyy
relieve	#pakcricket	blessed	poor	dn	ud83dudc95n
smiling	**saddens**	=(	rip	dnn	#thatswhat···
loving	boohoo	thank	ruined	#askdemilovato	nnud···
#musicmonday	**saddd**	pleasure	bummer	#rcl600kgiveaway	#madeinaus
emailunlimited	**saddened**	excellent	disappointed	#rcl500kgiveaway	teasay
thank	upsetting	vip	depressing	dni	ud83cudf···
stoked	**sad**	congrats	fathers	#8thstarmagicball	saddest
amusing	condolences	#followfriday	upset	7026350948	#1dalbumfour···
excited	disappointed	fantastic	depressed	iloveyouu	#heffrondrivepgh
glad	**saddest**	welcome	died	#happy	#preorder···
smiles	**sadface**	hehehe	cancelled	#itstylersbithday	#preorderhurri···
blessed	owie	hehe	missing	#excited	#followmecarter
congrats	hurtful	hilarious	disappointing	#happy21st···	ud83dudc95ncan
inspired	ftl	woohoo	miss	replacment	#justinnoticekati
thankyou	#ac	wait	sucks	#followmejenn	u2665ncan

high weights. In particular, for tweet'⟨user⟩ lol cool. sry missed your party', It contains both positive sentiment words, 'cool' and negative sentiment words, 'sry'. SSALSTM can precisely discover the most meaning words, 'missed' and 'sry' ahead of positive sentient words 'cool'. Thus, different from other deep learning based methods [14], [18], [20], SSALSTM can extract meaningful semantic words and explicit and implicit sentiment words by introducing self-attention mechanism.

I. Sentiment Lexicon Analysis

In order to obtain more insight of SSALSTM and observe the effectiveness of our constructed sentiment lexicon. We extract top 20 positive and negative sentiment words in three deep learning based sentiment lexicons, NN, HSSWE and SSALSTM. These sentiment words are shown in Table X. From Table X, we can observe that SSALSTM and HSSWE can discover much more commonly used positive and negative sentiment words than NN. Furthermore, It's easily to see that most of the sentiment words in NN are some hashtag words. It's mainly because that NN uses the average sentiment polarities of words to directly predict the sentiment polarities of documents. Meanwhile, documents' sentiment is marked by distant supervised sentiment labels which is determined by some pre-defined hashtag in the tweets [38].

Compared with HSSWE in Table X, we can easily observe that both SSALSTM and HSSWE can discover the most positive and negative words in tweets. However, different from HSSWE, SSALSTM can also discover much more synonyms which have the same stem. For instance, in SSALSTM, simile, similing and smiles can be discovered within top 20 positive sentiment words; sadd, saddens, saddd, sad and sadface are discovered within top 20 negative sentiment words. As we know, the rank of sentiment words in Table X means to what extent the corresponding word is positive or negative. In this case, the words with similar meaning (e.g., sharing the same stem) should obviously be close to each other.

J. Comparison Within the Model

In this subsection, we carried out the internal comparison within our model. In detail, except for LSTM neural network,

TABLE XI
SUPERVISED AND UNSUPERVISED EVALUATION FOR INTERNAL COMPARISON (F_1 AND ACCURACY SCORE)

Metric		Lexicon	SemEval13	SemEval14	SemEval15	Semeval16	Average	Improvements
Supervised	F_1	SSARNN-urban	79.86	79.49	73.85	75.85	77.26	1.416%
		SSABILSTM-urban	**81.43**	79.06	74.22	75.66	77.59	0.995%
		SSALSTM-urban	81.06	**80.72**	**74.94**	**76.74**	**78.37**	-
	$Accuracy$	SSARNN-urban	83.58	88.34	78.83	79.40	82.54	1.209%
		SSABILSTM-urban	**84.76**	88.01	79.19	79.29	82.81	0.886%
		SSALSTM-urban	84.32	**89.94**	**80.04**	**79.88**	**83.55**	-
Unsupervised	F_1	SSARNN-urban	69.60	71.93	67.10	65.96	68.65	8.125%
		SSABILSTM-urban	72.27	73.75	70.40	67.63	71.01	5.105%
		SSALSTM-urban	**76.42**	**76.34**	**73.30**	**73.27**	**74.83**	-
	$Accuracy$	SSARNN-urban	78.02	86.40	77.69	72.93	78.76	3.904%
		SSABILSTM-urban	79.84	87.16	**79.62**	74.50	80.28	2.050%
		SSALSTM-urban	**81.91**	**87.58**	**80.32**	**78.04**	**81.96**	-

we use bi-directional LSTM and basic RNN to replace LSTM to evaluate the performance. They are referred as *SSABILSTM-urban* and *SSARNN-urban*. Noting that *SSARNN-urban* is constructed to replace the LSTM with a basic RNN architecture. A basic RNN is a very simple recurrent neural network, and very easy to implement and train. In our paper, we use Elman network [41] as our basic RNN neural network. An Elman network is a three-layer network, It has an input layer, a hidden layer and output layer. Specifically, given the t-th word w_t in a document, x_t is the word embedding of w_t. Then, our basic RNN is calculated as follows:

$$h_t = \text{sigmoid}(W_h x_t + U_h h_{t-1} + b_h) \tag{20}$$

$$y_t = \text{softmax}(W_y h_t + b_y) \tag{21}$$

where h_t is the hidden representation and y_t is the output in the t-th time. W_h, U_h and b_h are the parameters of hidden layer. W_y and b_y are the parameter of output layer. From the above equation. h_{t-1} is the $(t-1)$-th time hidden representation.

- **SSARNN-urban**: we use a simple three-layer recurrent neural network (RNN) instead of LSTM to model the semantic relations between words. SARNN-urban denotes the sentiment lexicon constructed by self-attention recurrent neural network.
- **SSABILSTM-urban**: bi-directional LSTM is utilized to replace the LSTM to model the associations between words. SABILSTM-urban indicates the sentiment lexicon constructed by self-attention bi-directional LSTM model.

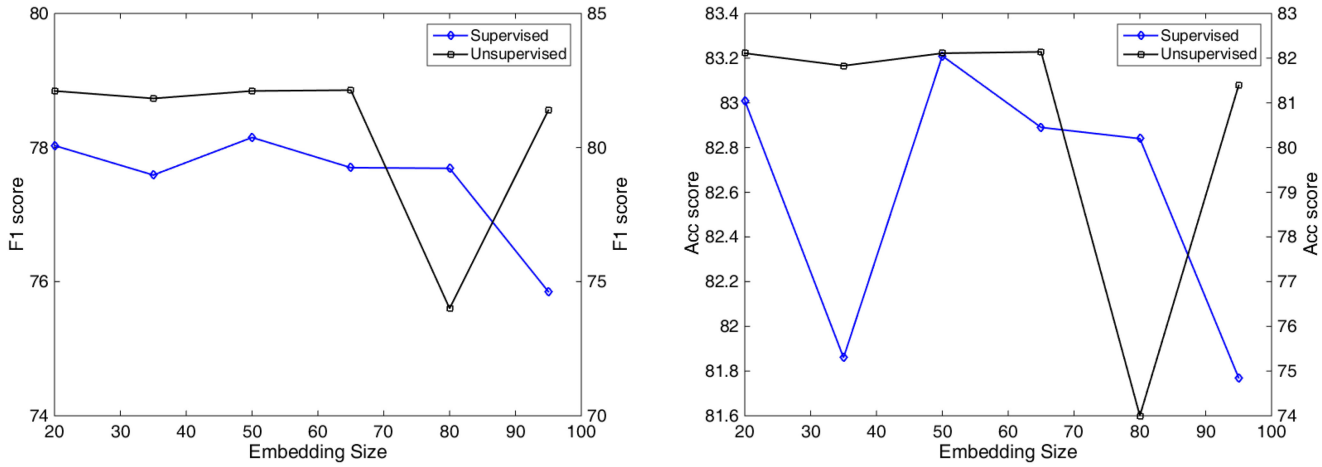
The experimental results is shown in Table XI. Noting that we only exhibit the experimental results which correspond to the sentiment lexicons constructed through *Urban*. It's because that we almost see the similar observation in sentiment lexicons constructed through *Seed* and *Lexicon*. Thus, we use *SSALSTM-urban*, *SSABILSTM-urban* and *SSARNN-urban* to represent them. For supervised sentiment classification, we can observe that SSALSTM-urban achieves the best performance except for in SemEval 2013 dataset. SSALSTM-urban outperforms SSARNN-urban 1.416 in F_1 score and 1.209 in $Accuracy$ score on the average of four datasets. It also outperforms SSABILSTM-urban 0.995 in F_1 score and 0.886 in $Accuracy$ score on the average of all datasets. For unsupervised sentiment classification, we obtain the experimental results that SSALSTM-urban gets the best performance on most of the datasets. It obtains 8.125 improvement in F_1 score and 3.904 improvement in $Accuracy$ score than SSARNN-urban on the average of all datasets. Meanwhile, SSALSTM-urban gains 5.105 improvement in F_1 score and 2.050 improvement in $Accuracy$ score on the average of four datasets. The main reason is that LSTM based methods can remember longer semantic relations between words than standard recurrent neural network. Meanwhile, SSABILSTM-urban performs a little worse than SSALSTM-urban, the probably reason is that tweet is a kind of short data. Applying bi-directional LSTM is easy to be over-fitting and can't obtain a better performance on test data.

K. Effect of the Embedding Size

In this section, we evaluate the impact of the dimension size of word embeddings. In the experiment, we set the embedding size from 20 to 95 with a interval 15. The average performance of our proposed model, SSALSTM, is shown in Fig. 2. As we can see, SSALSTM achieves the best results when the embedding size is 50 in both supervised and unsupervised sentiment classification task. When the embedding size is bigger than 50, the performance of SSALSTM decreases apparently.

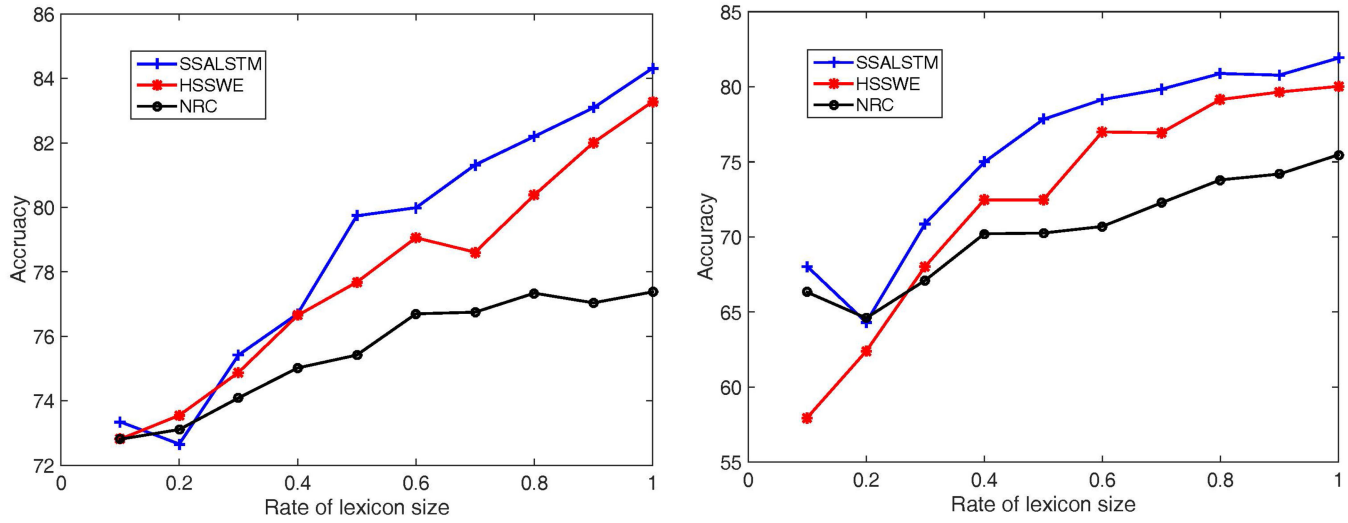
L. Effect of the Lexicon Size

In this section, we evaluate the effect of the lexicon size. Firstly, we use the SemEval2013 dataset and choose three sentiment lexicons as an example, including SSALSTM-urban, HSSWE and NRC. SSALSTM-urban and HSSWE are two deep learning based sentiment lexicons and NRC is a statistical based lexicon. Secondly, we randomly extract a subset of words from these three lexicons respectively with the rate from $\{10\%, 20\%, \ldots, 80\%, 90\%, 100\%\}$. Then, we conduct the supervised and unsupervised sentiment classification tasks to evaluate their performance. The results are shown in Fig. 3. From Fig. 3, we can observe that the performance of these three sentiment lexicons decreases with the reduction of the sentiment lexicon size. The main reason is that many commonly used sentiment words and implicit sentiment words are filtered out when the lexicon size descreas. For example, when we randomly extract 10% sentiment words from these three lexicons (nearly 5000 words). The accuracy metric of SSALSTM-urban decreases from 84.32% to 73.21% on supervised sentiment classification. Meanwhile, due to the same reason, although



(a) F1 (b) Accuracy

Fig. 2. Average performance of SASLSTM ((a) **F1** score (b) **Acc** score) on supervised and unsupervised classification tasks with different value of embedding size.



(a) supervised (b) unsupervised

Fig. 3. Supervised (a) and unsupervised (b) classification tasks with different rate of the lexicon size.

MPQA and ETSL contain only a few sentiment words (nearly 1000-3000), MPQA can still obtain 73.84% accuracy on supervised sentiment classification. Because MPQA are manually constructed to contain nearly all of the commonly used sentiment words.

M. Impact of L_1 Regularize

In order to evaluate the effect of L_1 regularize in our proposed models, we compare the performance of two models which respectively use self-attention lstm with or without L_1 regularize in supervised and unsupervised classification tasks. We use *SSALSTM-urban* and *SALSTM-urban* to represent them respectively. The experimental results are shown in Table XII. From the table, we can easily observe that the overall performance of our proposed model with L_1 regularize is better than that without L_1 regularize. For instance, in supervised classification task, SSALSTM-urban obtains 2.302% and 2,957% improvements over SALSTM-urban in F1 score. Meanwhile, SSALSTM-urban gets 1.180% and 1.134% improvements over SALSTM-urban in Accuracy score. It's also verified that most of words among a document do not contribute the distinguish of document's sentiment polarity.

N. Analysis of SSALSTM With or Without Self-Attention Mechanism

In order to analyze the impact of self-attention to SSALSTM, we compare the performance of SSALSTM with or without

TABLE XII
SUPERVISED AND UNSUPERVISED EVALUATION FOR SSALSTM WITH OR WITHOUT L1 REGULARIZE (F_1 AND ACCURACY SCORE)

Metric		Lexicon	SemEval13	SemEval14	SemEval15	Semeval16	Average	Improvements
Supervised	F_1	**SALSTM-urban**	78.54	80.30	73.09	74.66	76.65	2.302%
		SSALSTM-urban	**81.13**	**80.88**	**75.20**	**76.60**	**78.45**	-
	$Accuracy$	**SALSTM-urban**	82.74	88.85	79.11	78.79	82.37	1.180%
		SSALSTM-urban	**84.37**	**88.94**	**80.33**	**79.80**	**83.36**	-
Unsupervised	F_1	**SALSTM-urban**	71.69	**77.52**	72.00	69.27	72.62	2.957%
		SSALSTM-urban	**76.42**	76.34	**73.30**	**73.27**	**74.83**	-
	$Accuracy$	**SALSTM-urban**	79.40	**88.68**	**80.47**	75.60	81.04	1.134%
		SSALSTM-urban	**81.91**	87.58	80.33	**78.05**	**81.97**	-

TABLE XIII
SUPERVISED AND UNSUPERVISED EVALUATION FOR SSALSTM WITH OR WITHOUT SELF-ATTENTION MECHANISM (F_1 AND ACCURACY SCORE)

Metric		Lexicon	SemEval13	SemEval14	SemEval15	Semeval16	Average	Improvements
Supervised	F_1	**SLSTM-urban**	77.38	79.37	72.58	72.88	75.55	3.70%
		SSALSTM-urban	**81.13**	**80.88**	**75.20**	**76.60**	**78.45**	-
	$Accuracy$	**SLSTM-urban**	81.96	87.75	78.62	77.33	81.42	2.33%
		SSALSTM-urban	**84.37**	**88.94**	**80.33**	**79.80**	**83.36**	-
Unsupervised	F_1	**SLSTM-urban**	68.98	72.09	67.40	63.07	67.89	9.27%
		SSALSTM-urban	**76.42**	**76.34**	**73.30**	**73.27**	**74.83**	-
	$Accuracy$	**SLSTM-urban**	77.78	86.57	77.90	71.64	78.47	4.27%
		SSALSTM-urban	**81.91**	**87.58**	**80.33**	**78.05**	**81.97**	-

self-attention mechanism. In this subsection, we use SLSTM-urban to represent SSALSTM without self-attention mechanism and SSALSTM-urban to indicate the original SSALSTM model. The results are shown in Table XIII. From Table XIII, we can easily observe that SSALSTM-urban performs better than SLSTM-urban in all datasets. For instance, on supervised sentiment classification, SSALSTM-urban gets 3.70% and 2.33% improvements than SLSTM-urban in F1 and Accuracy metric respectively. On unsupervised sentiment classification, SSALSTM-urban also obtains a significant improvement (9.27%) than SLSTM-urban in F1 metric. Thus, self-attention is very important for sentiment lexicon construction. It also verifies that different words in a sentence contribute different to the sentiment polarities of documents.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel sparse self-attention LSTM (SSALSTM) model to automatic construct sentiment lexicons. SSALSTM can solve two issues which block the performance of current deep learning based methods. SSALSTM uses LSTM to capture complex contextual semantic and syntactic relations between words. Then, it exploits self-attention mechanism to capture the importance of words to the distinguish of documents' sentiment polarities. Furthermore, we apply L_1 regularize to the weights of words in a document, in this way, we can ensure the intuitions that only a minority of words contribute to the distinguish of documents' sentiment label. A series of experiments on the SemEval 2013–2016 datasets indicate that the sentiment lexicon generated by our proposed method achieves state-of-the-art performance on both supervised and unsupervised sentiment classification tasks.

In the future work, we still have lots of work to do. Firstly, existing lexicons, including SSALSTM and other baselines, are performing much better in positive class than that in negative class, such imbalance is a challenging problem in sentiment lexicon construction task. Thus, we're interested to solve this problem in our next step. Secondly, this paper only considers the binary sentiment classification task, however, as we know, classifying documents or sentences into positive, neutral, and negative is more practically. Finally, the proposed method, SSALSTM, can not only apply in the social media datasets, but also in the domain of e-commence, like the movie review datasets. Thus, in the future, we will consider the performance of SSALSTM in other domains.

REFERENCES

[1] L. Kaushik, A. Sangwan, and J. H. L. Hansen, "Automatic sentiment detection in naturalistic audio," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 8, pp. 1668–1679, Aug. 2017.
[2] Z. Wang, S. Y. M. Lee, S. Li, and G. Zhou, "Emotion analysis in code-switching text with joint factor graph model," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 3, pp. 469–480, Mar. 2017.
[3] A. Neviarouskaya, H. Prendinger, and M. Ishizuka, "Sentiful: A lexicon for sentiment analysis," *IEEE Trans. Affective Comput.*, vol. 2, no. 1, pp. 22–36, Jan./Jun. 2011.
[4] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proc. Conf. Human Lang. Technol. Empirical Methods Natural Lang. Process.*, 2005, pp. 347–354.
[5] R. Gilman, "The general inquirer: A computer approach to content analysis," *Amer. J. Sociol.*, vol. 73, no. 5, pp. 634–635, 1968.
[6] Z. Dong and Q. Dong, *Hownet and the Computation of Meaning*. Singapore: World Scientific, 2006.
[7] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2004, pp. 168–177.
[8] A. Esuli and F. Sebastiani, "Sentiwordnet: A publicly available lexical resource for opinion mining," in *Proc. 5th Conf. Lang. Resources Eval.*, 2006, pp. 417–422.
[9] L. Barbosa and J. Feng, "Robust sentiment detection on twitter from biased and noisy data," in *Proc. 23rd Int. Conf. Comput. Linguistics, Posters*, 2010, pp. 36–44.
[10] L. Gatti, M. Guerini, and M. Turchi, "Sentiwords: Deriving a high precision and high coverage lexicon for sentiment analysis," *IEEE Trans. Affective Comput.*, vol. 7, no. 4, pp. 409–421, Oct./Dec. 2016.
[11] J. Kamps, M. Marx, R. J. Mokken, and M. de Rijke, "Using wordnet to measure semantic orientations of adjectives," in *Proc. 4th Int. Conf. Lang. Resources. Eval.*, 2004, vol. 4, pp. 1115–1118.

[12] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in *Proc. Int. Conf. Lang. Resources Eval.*, 2010, pp. 2200–2204.
[13] S. Mohammad, S. Kiritchenko, and X. Zhu, "Nrc-Canada: Building the state-of-the-art in sentiment analysis of tweets," in *Proc. 1th Int. Workshop Semantic Evaluation Exercises (SemEval-2013)*, 2013, pp. 321–327.
[14] D. Tang, F. Wei, B. Qin, M. Zhou, and T. Liu, "Building large-scale twitter-specific sentiment lexicon: A representation learning approach," in *Proc. 25th Int. Conf. Comput. Linguistics, Tech. Papers*, 2014, pp. 172–182.
[15] H. Saif, M. Fernandez, L. Kastler, and H. Alani, "Sentiment lexicon adaptation with context and semantics for the social web," *Semantic Web*, vol. 8, no. 5, pp. 643–665, 2017.
[16] M. Yang, D. Zhu, M. Rashed, and K.-P. Chow, "Learning domain-specific sentiment lexicon with supervised sentiment-aware LDA," *Frontiers Artif. Intell. Appl.*, vol. 263, pp. 927–932, 2014.
[17] S. Deng, A. P. Sinha, and H. Zhao, "Adapting sentiment lexicons to domain-specific social media texts," *Decis. Support Syst.*, vol. 94, pp. 65–76, 2017.
[18] L. Wang and R. Xia, "Sentiment lexicon construction with representation learning based on hierarchical sentiment supervision," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 502–510.
[19] Q. Li and S. Shah, "Learning stock market sentiment lexicon and sentiment-oriented word vector from stocktwits," in *Proc. 21st Conf. Comput. Natural Lang. Learn.*, 2017, pp. 301–310.
[20] D. Vo and Y. Zhang, "Don't count, predict! an automatic approach to learning sentiment lexicons for short text," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, vol. 2, pp. 219–224.
[21] E. Cambria, S. Poria, D. Hazarika, and K. Kwok, "Senticnet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings," in *Proc. 32th Int. Conf. Assoc. Adv. Artif. Intell.*, 2018, pp. 1795–1802.
[22] G. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, pp. 39–41, 1995.
[23] A. Valitutti, C. Strapparava, and O. Stock, "Developing affective lexical resources," *Psychology J.*, vol. 2, no. 1, pp. 61–83, 2004.
[24] S. Kim and E. Hovy, "Determining the sentiment of opinions," in *Proc. 20th Int. Conf. Comput. Linguistics*, 2004, Art. no. 1367.
[25] S. Mohammad, C. Dunne, and B. Dorr, "Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2009, pp. 599–608.
[26] S. Blair-Goldensohn *et al.*, "Building a sentiment summarizer for local service reviews," in *Proc. WWW Workshop NLP Inf. Explosion Era*, 2008, vol. 14, pp. 339–348.
[27] A. Hassan and D. Radev, "Identifying text polarity using random walks," in *Proc. 48th Annu. Meeting Assoc. Comput. Linguistics*, 2010, pp. 395–403.
[28] Z. Harris, "Distributional structure," *Word*, vol. 10, no. 23, pp. 146–162, 1954.
[29] D. Peter, "Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2002, pp. 417–424.
[30] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier, "Effective attention modeling for aspect-level sentiment classification," in *Proc. 27th Int. Conf. Comput. Linguistics*, Aug. 2018, pp. 1121–1131.
[31] G. Letarte, F. Paradis, P. Giguère, and F. Laviolette, "Importance of self-attention for sentiment analysis," in *Proc. EMNLP Workshop BlackboxNLP, Analyzing Interpreting Neural Netw. NLP*, Brussels, Belgium, Nov. 2018, pp. 267–275.
[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[33] M. Sundermeyer, R. Schlter, and H. Ney, "LSTM neural networks for language modeling," in *Proc. 13th Annu. Conf. Int. Speech Commun. Assoc.*, 2012, pp. 194–197.
[34] T. Mikolov, M. Karafiát, L. Burget, J. Černock, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1045–1048.
[35] P. Zhou *et al.*, "Attention-based bidirectional long short-term memory networks for relation classification," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, vol. 2, pp. 207–212.
[36] Y. Wang *et al.*, "Attention-based LSTM for aspect-level sentiment classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 606–615.
[37] A. Vaswani *et al.*, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf
[38] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," Stanford University, Stanford, CA, USA, CS224N Project Report, 2009.
[39] S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," *J. Artif. Intell. Res.*, vol. 50, pp. 723–762, 2014.
[40] S. Rosenthal *et al.*, "Semeval-2015 task 10: sentiment analysis in twitter," in *Proc. 9th Int. Workshop Semantic Eval*, 2015, pp. 451–463.
[41] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

Dong Deng received the B.S. degree in computer science from the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China, in 2014. He is currently working toward the Ph.D. degree with the School of Computer and Information Technology, Beijing Jiaotong University, China. His research interests include sentiment analysis, data mining and machine learning.

Liping Jing received the Ph.D degree in applied mathematics from the University of Hong Kong, Hong Kong, in 2007. She was a Research Associate with the Department of Computer Science, University of Texas at Dallas, Dallas, TX, USA, from 2007 to 2008, and a Visiting Scholar with ICSI and the AMPLab, University of California at Berkeley, Berkeley, CA, USA, from 2015 to 2016. She is currently a Professor with the Beijing Key Laboratory of Traffic Data Analysis and Mining, School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. She has authored of co-authored more than 70 peer-reviewed research papers in various journals and conferences. Her research interests include high-dimensional subspace learning and machine learning. She served as a Regular Reviewer and Program Committee Member for a number of international journals and conferences.

Jian Yu received the B.S. and M.S. degrees in mathematics and the Ph.D. degree in applied mathematics from Peking University, Beijing, China, in 1991, 1994, and 2000, respectively. He is currently a Professor with Beijing Jiaotong University, Beijing, China, and the Director with the Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing, China. His current research interests include machine learning, image processing, and pattern recognition.

Shaolong Sun received the Ph.D. degree majoring in management science and engineering from the Institute of Systems Science, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing, China. He is currently a Professor with the Department of Management Science, School of Management, Xi'an Jiaotong University, Xi'an, China. His research interests include artificial intelligence, big data mining, machine learning, social networks analysis, knowledge management, and economic and financial forecasting. He has authored or coauthored more than ten papers in journals including Applied Energy and Journal of Environmental Management.