Sentiment Embeddings with Applications to Sentiment Analysis

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, Ming Zhou

Abstract—We propose learning sentiment-specific word embeddings dubbed sentiment embeddings in this paper. Existing word embedding learning algorithms typically only use the contexts of words but ignore the sentiment of texts. It is problematic for sentiment analysis because the words with similar contexts but opposite sentiment polarity, such as *good* and *bad*, are mapped to neighboring word vectors. We address this issue by encoding sentiment information of texts (e.g. sentences and words) together with contexts of words in sentiment embeddings. By combining context and sentiment level evidences, the nearest neighbors in sentiment embedding space are semantically similar and it favors words with the same sentiment polarity. In order to learn sentiment embeddings effectively, we develop a number of neural networks with tailoring loss functions, and collect massive texts automatically with sentiment signals like emoticons as the training data. Sentiment embeddings can be naturally used as word features for a variety of sentiment analysis tasks without feature engineering. We apply sentiment embeddings to word-level sentiment analysis, sentence level sentiment classification and building sentiment lexicons. Experimental results show that sentiment embeddings consistently outperform context-based embeddings on several benchmark datasets of these tasks. This work provides insights on the design of neural networks for learning task-specific word embeddings in other natural language processing tasks.

Index Terms—Natural Language Processing, Word Embeddings, Sentiment Analysis, Neural Networks

1 Introduction

Word representation attempts to represent aspects of word meanings. For example, the representation of "*cellphone*" may capture the facts that cellphones are electronic products, that they include battery and screen, that they can be used to chat with others, and so on. Word representation is a critical component of many natural language processing systems [4], [5] as word is usually the basic computational unit of texts.

A straight forward way is to represent each word as a one-hot vector, whose length is vocabulary size and only one dimension is 1, with all others being 0. However, one-hot word representation only encodes the indices of words in a vocabulary, but fails to capture rich relational structure of the lexicon. To solve this problem, many studies represent each word as a continuous, low-dimensional and real-valued vector, also known as **word embeddings** [6], [7], [8]. Existing embedding learning approaches are mostly on the basis of distributional hypothesis [9], which states that the representations of words are reflected by their contexts. As a result, words with similar grammatical usages and semantic meanings, such as "*hotel*" and "*motel*", are mapped into neighboring vectors in the embedding space. Since word embeddings capture semantic similarities between words, they have been leveraged as inputs or extra word features for a variety of natural language processing tasks, including machine translation [10], syntactic parsing [11], question answering [12], discourse parsing [13], etc al.

Despite the success of the context-based word embeddings in many NLP tasks [14], we argue that they are not effective enough if directly applied to sentiment analysis [15], [16], [17], which is the research area targeting at extracting, analyzing and organizing the sentiment/opinion (e.g. thumbs up or thumbs down) of texts. The most serious problem of context-based embedding learning algorithms is that they only model the contexts of words but ignore the sentiment information of text. As a result, words with opposite polarity, such as *good* and *bad*, are mapped into close vectors in the embedding space. This is meaningful for some tasks such as pos-tagging [18] because the two words have similar usages and grammatical roles. However, it becomes a disaster for sentiment analysis as they have opposite sentiment polarity labels.

In this paper, we propose learning sentiment-specific word embeddings dubbed **sentiment embeddings** for sentiment analysis. We retain the effectiveness of word contexts and exploit sentiment of texts for learning more powerful continuous word representations. By capturing both context and sentiment level evidences, the nearest neighbors in the embedding space are not only semantically similar but also favor to have the same sentiment polarity, so that it is able to separate *good* and *bad* to opposite ends of the spectrum. In order to learn sentiment embeddings effectively, we develop a number of neural networks to capture sentiment of texts (e.g. sentences and words) as well as contexts of words with dedicated loss functions. We learn sentiment embeddings from tweets[1], leveraging positive and negative emoticons as pseudo sentiment labels of sentences without manual

- *Duyu Tang, Bing Qin and Ting Liu are with Harbin Institute of Technology, Harbin, China. E-mail: {dytang, qinb, tliu}@ir.hit.edu.cn. Corresponding author is Bing Qin.*
- *Furu Wei, Nan Yang and Ming Zhou are with Microsoft Research, Beijing, China. E-mail: {fuwei, mingzhou}@microsoft.com, nyang.ustc@gmail.com*
- *This work was partly done during Duyu Tang's internship at Microsoft Research, Beijing, China. It is a substantial extension of our previous works in [1], [2], [3].*

[1] https://twitter.com

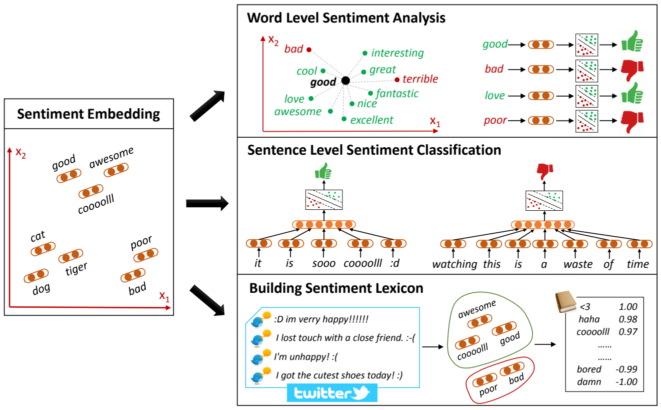


Fig. 1. An illustration of sentiment embeddings with applications to sentiment analysis tasks, including word level sentiment analysis, sentence level sentiment classification and building sentiment lexicon.

annotations. We obtain lexical level sentiment supervision from Urban Dictionary[2] based on a small list of sentiment seeds with minor manual annotation.

We evaluate the effectiveness of sentiment embeddings empirically by applying them to three sentiment analysis tasks. Word level sentiment analysis on benchmark sentiment lexicons [19], [20] can help us see whether sentiment embeddings are useful to discover similarities between sentiment words. Sentence level sentiment classification on tweets [21], [22] and reviews [23] help us understand whether sentiment embeddings are helpful in capturing discriminative features for predict the sentiment of text. Building sentiment lexicon [24] is useful for measuring the extent to which sentiment embeddings improve lexical level tasks that need to find similarities between words. Experimental results show that sentiment embeddings consistently outperform context-based word embeddings, and yields state-of-the-art performances on several benchmark datasets of these tasks.

The major contributions of the work presented in this paper are as follows.

- We propose learning sentiment embeddings that encode sentiment of texts in continuous word representation.
- We develop a number of neural networks with tailoring loss functions to learn sentiment embeddings. We learn sentiment embeddings from tweets with positive and negative emoticons as distant-supervised corpora without any manual annotations.
- We verify the effectiveness of sentiment embeddings by applying them to three sentiment analysis tasks. Empirical experimental results show that sentiment embeddings outperform context-based embeddings on several benchmark datasets of these tasks.

This article is organized as follows. We introduce the background of word embeddings in Section 2. We then present the methodology for learning sentiment embeddings in Section 3. The use of sentiment embeddings in three applications are given in Section 4 (word level sentiment analysis), Section 5 (sentence level sentiment classification) and Section 6 (building sentiment lexicon). We conclude this paper in Section 7.

2 BACKGROUND

In this section, we describe the background on learning continuous word representation.

Word representation aims to represent aspects of word meaning. A straight-forward way is to encode a word w_i as a *one-hot* vector, whose length is vocabulary size with 1 in the w_i^{th} position and zeros everywhere else. However, such *one-hot* word representation only encodes the indices of words in a vocabulary, without capturing rich relational structure of the lexicon. One common approach to discover the similarities between words is to learn a clustering of words [25], [26]. Each word is associated with a discrete class, and words in the same class are similar in some respects. This leads to a *one-hot* representation over a smaller vocabulary size. Instead of characterizing the similarity with a discrete variable based on clustering results which corresponds to a soft or hard partition of the set of words, many researchers target at learning a continuous and real-valued vector for each word, also known as word embeddings. Existing embedding learning algorithms are mostly based on the distributional hypothesis [9], which states that words in similar contexts have similar meanings. Many matrix factorization methods can be viewed as modeling word representations. For example, Latent Semantic Indexing (LSI) [27] can be regarded as learning a linear embedding with a reconstruction objective, which uses a matrix of "term-document" co-occurrence statistics, e.g. each row stands for a word or term and each column corresponds to an individual document in the corpus. Hyperspace Analogue to Language [28] utilizes a matrix of "term-term" co-occurrence statistics, where both rows and columns correspond to words and the entries stand for the number of times a given word occurs in the context of another word. Hellinger PCA [29] is also investigated to learn word embeddings over "term-term" co-occurrence statistics.

With the revival of interest in deep learning and neural network [30], [31], [32], a surge of studies learn word embeddings with neural network. A pioneered work in this field is given by Bengio et al. [6]. They introduce a neural probabilistic language model that learns simultaneously a continuous representation for words and the probability function for word sequences based on these word representations. Given a word w_i and its preceding context words, the algorithm first maps each context word to its continuous vector with a shared lookup table. Afterwards, context word vectors are fed to a feed-forward neural network with $softmax$ as output layer to predict the conditional probability of next word w_i. The parameters of neural network and lookup table are jointly learned with back propagation. Following Bengio et al. [6]'s work, a lot of approaches are proposed to speed-up the training processing or capturing richer semantic information. Bengio et al. [33] introduce a neural architecture by concatenating the vectors of context words and current word, and use importance sampling to effectively optimize the model with observed "positive sample" and sampled "negative samples". Morin and Bengio [34] develops *hierarchical softmax* to decompose

[2] http://www.urbandictionary.com/

the conditional probability with a hierarchical binary tree. Mnih and Hinton [35] introduce a log-bilinear language model. Collobert and Weston [36] train word embeddings with a ranking-type hinge loss function [37] by replacing the middle word within a window with a randomly selected one. Mikolov et al. [7], [38] introduce Continuous Bag-of-Words (CBOW) and Continuous Skip-gram, and release the popular *word2vec*[3] toolkit. CBOW model predicts the current word based on the embeddings of its context words, and Skip-gram model predicts surrounding words given the embeddings of current word. Mnih et al. [39] accelerate the embedding learning procedure with Noise Contrastive Estimation [40]. There are also many algorithms developed for capturing richer semantic information, including global document information [41], Chinese character radical [42], dependency based contexts [43], multi sense information [44] and semantic lexical information [45].

Some studies in recent years attempt to learn sentiment-tailored word embeddings by encoding the sentiment polarity of texts. Maas et al. [46] introduce a probabilistic topic model by inferring the polarity of a sentence based on the embeddings of each word it contains. Labutov and Lipson [47] re-embed existing word embeddings with logistic regression by regarding sentiment supervision of sentences as a regularization item.

3 Methodology

We present the methods for learning sentiment embeddings in this section. We first describe standard context-based neural network methods for learning word embeddings. Afterwards, we introduce our extension for capturing sentiment polarity of sentences before presenting hybrid models which encode both sentiment and context level information. We then describe the integration of word level information for embedding learning.

3.1 Notation

We notate the meaning of variables used in this paper. In particular, w_i means a word whose index is i in a sentence, h_i is context words of w_i in one sentence, e_i is the embedding vector of w_i. In this work, we implement the neural network approaches with some basic neural layers, including $lookup$, $hTanh$, $linear$ and $softmax$. For each neural layer, O_{layer} means the output vector. The implementations of these layers can be found at: http://ir.hit.edu.cn/~dytang.

3.2 Modeling Contexts of Words

Dominating word embedding learning algorithms are on the basis of distributional hypothesis [9], which states that the representations of words can be reflected by their contexts. In this subsection, we describe a prediction model (Section 3.2.1) and a ranking model (Section 3.2.2) to encode contexts of words for learning word embeddings. These context-based models will be naturally incorporated with sentiment-specific models (Section 3.3) for learning sentiment embeddings.

3.2.1 *Prediction Model*

An effective way to encode contexts of words into word representation is "context prediction" [7], [39], [48]. Given a target word w_i and its context words h_i, "context prediction" aims to predict w_i based on h_i, which can be viewed as language modeling. The contexts of a target word could be preceding, following or surrounding words occurred in a piece of text. Since we do not focus on an exact language model, we investigate surrounding words $h_i = \{w_{i-c}, w_{i-c+1}, ...w_{i-1}, w_{i+1}, ...w_{i+c-1}, w_{i+c}\}$ as context in this work. The method can be naturally extend to preceding or following context words.

We build a prediction model analogous to the representative "context prediction" neural language model given by Bengio et al. [6]. They model the conditional probability $P(w_i|h_i)$ of predicting a target word w_i based on its contexts h_i by taking word embeddings as a parameter. The scoring function is a feed-forward neural network consisting of $lookup \rightarrow linear \rightarrow hTanh \rightarrow linear \rightarrow softmax$.

Lookup layer (also referred to as projection layer) contains a lookup table $LT \in \mathbb{R}^{d\times|V|}$ which maps each word to its continuous vector, where d is the dimension of each word vector and $|V|$ is the vocabulary size. The lookup operation can be viewed as a projection function that uses a binary vector idx_i which is zero in all positions except at the i-th index.

$$e_i = LT \cdot idx_i \in \mathbb{R}^{1\times d} \tag{1}$$

We concatenate the embeddings of context words $\{w_{i-c}, w_{i-c+1}, ..., \ w_{i-1}, w_{i+1} \ , ...w_{i+c-1}, w_{i+c}\}$ as the output of lookup layer, which is formalized as below.

$$O_{lookup} = [e_{i-c}; ... \ e_{i-1}, e_{i+1} \ ...; e_{i+c}] \in \mathbb{R}^{1\times d\cdot 2c} \tag{2}$$

Afterwards, O_{lookup} is fed to a linear layer for dimension transformation, where $W_{l1} \in \mathbb{R}^{len\times 2c}$ is the position-dependent weights and $b_{l1} \in \mathbb{R}^{1\times len}$ is the bias of linear layer, len is the length of output vector O_{l1} of linear layer.

$$O_{l1} = W_{l1} \cdot O_{lookup} + b_{l1} \tag{3}$$

The linear layer is followed by a non-linear function layer for adding element wise non-linearity. S-tandard non-linear layers include $hyperbolic\ tangent$, $hard\ hyperbolic\ tangent$ $(hTanh)$, $sigmoid$ and $rectifier$. We use $hTanh$ in this work for its computational efficiency and effectiveness in literature [14]. We denote the output vector of $hTanh$ as $O_{htanh} \in \mathbb{R}^{1\times len}$.

$$hTanh(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases} \tag{4}$$

The output layer of standard neural language model is a $softmax$ layer whose output length is vocabulary size. $Softmax$ [49] is suitable for this case as its outputs can be interpreted as conditional probabilities, which is calculated as below.

$$softmax_i = \frac{exp(z_i)}{\sum_{i'} exp(z_{i'})} \tag{5}$$

Conditional probability $P(w_i|h_i)$ is the idx_{w_i}-th value in the output vector of $softmax$ layer, where idx_{w_i} is the index of

[3] https://code.google.com/p/word2vec/

w_i in vocabulary. The optimizing objective of a given word-context pair (w_i, h_i) is to maximize $P(w_i|h_i)$.

However, directly predicting the probability of $P(w_i|h_i)$ is time-costly as the length of output $softmax$ layer is vocabulary size, which is typically hundreds of thousands. To speed-up the training process, we use noise contrastive estimation [39], and transfer the problem of "context prediction" to distinguish a word-context pair as real case or artificial noise by means of logistic regression. The probability that the given sample came from the data is $P(D|w,\theta)$

$$P(D|w,\theta) = \frac{exp(f_\theta(w_i,h_i))}{exp(f_\theta(w_i,h_i)) + k \cdot exp(f_\theta(w^n,h_i))} \tag{6}$$

where w^n is an artificial noise such as a randomly selected word from vocabulary. The scaling factor k accounts for the fact that noise samples are k times more frequent than data samples [39]. The score function $f_\theta(w,h)$ quantifies the compatibility between context h_i and target word w_i, which can be naturally defined as a feed forward neural network consisting of $lookup \rightarrow linear \rightarrow hTanh \rightarrow linear$. The input of lookup layer is the concatenation of the current word w and context words h. The output is a linear layer with output length as 1, which stands for the compatibility between context h and word w. We implement $P(D|w,\theta)$ with a $softmax$ layer and maximize the log probability of the $softmax$ for parameter estimation.

$$loss_{cPred} = -\sum_{w \in T} logP(D|w,\theta) \tag{7}$$

3.2.2 Ranking Model

Collobert and Weston use a pairwise ranking approach [36] to capture the contexts of words for learning word embeddings. It holds the similar idea with noise contrastive estimation but the optimizing objective is to assign a real word-context pair (w_i, h_i) a higher score than an artificial noise (w^n, h_i) by a margin. They minimize the following hinge loss function, where T is the training corpora.

$$loss_{cRank} = \sum_{(w_i,h_i) \in T} max(0, 1 - f_\theta(w_i,h_i) + f_\theta(w^n,h_i)) \tag{8}$$

The scoring function $f_\theta(w,h)$ is achieved with a feed forward neural network. Its input is the concatenation of the current word w_i and context words h_i, and the output is a linear layer with only one node which stands for the compatibility between w and h. During training, an artificial noise w^n is randomly selected over the vocabulary under a uniform distribution.

3.3 Modeling Sentiment Polarity of Sentences

We present the approaches to encode sentiment polarity of sentences in sentiment embeddings in this part. We describe two neural networks including a prediction model and a ranking model to take considerations of sentiment of sentences.

3.3.1 Prediction Model

The basic idea of the prediction model is regarding sentiment prediction as a multi-class classification task. It predicts positive/negative categorical probabilities of a word sequence by regarding word embeddings as parameters. In particular, given a variable sized sentence, we slide fixed window of words across a sentence and predict the sentiment polarity of each window based on local word embeddings. This assumption is suitable for tweets if window size is large, since tweets are typically short and sentiment condensed. However, it might be not appropriate for document level review texts where sentiment shifting indicators (e.g. negation and contrast) are frequently used.

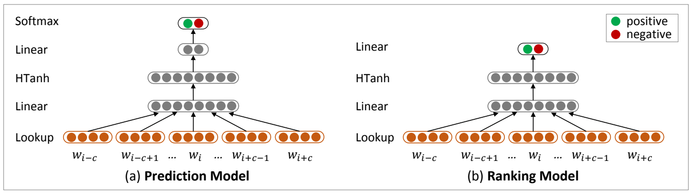


Fig. 2. An illustration of two neural networks that model the sentiment polarity of sentences for learning sentiment embeddings.

An illustration of prediction model with binary sentiment categories (positive and negative) is given in Figure 2 (a). It contains five layers, namely $lookup \rightarrow linear \rightarrow hTanh \rightarrow linear \rightarrow softmax$. The input is a fixed-length word sequence $\{w_{i-c}, w_{i-c+1}, ..., \ w_i \ , ...w_{i+c-1}, w_{i+c}\}$, where w_i is the current word and c is window size. Lookup, linear and $htanh$ layers are described in Section 3.2.1. The output of $htanh$ layer is used as features to predict the positive and negative probabilities of input, as the continuous representation of higher layers in a neural network can be interpreted as abstractive and discriminative features describing the input. To predict the probabilities of positive and negative categories, we feed $htanh$ to a linear layer to convert the vector length to category number C, which is 2 in this binary classification case. The parameters of the second linear layer are $W_{l2} \in \mathbb{R}^{C \times len}$ and $b_{l2} \in \mathbb{R}^{1 \times C}$. We then add a $softmax$ layer as the output layer to generate conditional probabilities over positive and negative categories.

During training, we are given the gold sentiment polarity of an input sentence. Let $f^g(t) \in \mathbb{R}^{1 \times C}$ be the gold distribution of an input t, where C is the number of sentiment polarity labels, which is 2 for positive and negative classification. $f^g(t)$ has a 1-of-C coding scheme, which has the same dimension as the number of classes, and only the dimension corresponding to the ground truth is 1, with all others being 0. For example, $f^g(t) = [1,0]$ stands for a sentence with positive polarity and $f^g(t) = [0,1]$ represents a sentence with negative polarity. We use cross entropy error between gold sentiment distribution and predicted distribution as the loss function of $softmax$ layer. For a corpus T, the loss function is given below, and the parameters can be learned with standard back propagation [50].

$$loss_{sPred} = -\sum_{t}^{T} \sum_{k=\{0,1\}} f_k^g(t) \cdot log(f_k^{pred}(t)) \tag{9}$$

3.3.2 Ranking Model

We describe an alternative of prediction model, which is a ranking model that outputs two real-valued sentiment s-cores for a word sequence with fixed window size. The basic idea of ranking model is that if the gold sentiment polarity of a word sequence is positive, the predicted positive score should be higher than the negative score. Similarly, if the gold sentiment polarity of a word sequence is negative, its positive score should be smaller than the negative score. For example, if a word sequence is associated with two scores $[f^{rank}_{pos}, f^{rank}_{neg}]$, then the values of [0.7,0.1] can be interpreted as a positive case because the positive score 0.7 is greater than the negative score 0.1. By that analogy, the result with [-0.2,0.6] indicates a negative polarity.

Based on this consideration, we develop a neural network based ranking model, as illustrated in Figure 2 (b), which shares some similarities with [36]. As is shown, the ranking model is a feed-forward neural network consisting of four layers ($lookup \rightarrow linear \rightarrow hTanh \rightarrow linear$). Compared with the prediction model as shown in Figure 2 (a), the $softmax$ layer is removed because thisz' objective does not require probabilistic interpretation. Let us denote the output vector of ranking model as $f^{rank} \in \mathbb{R}^{1\times C}$, where $C = 2$ for binary positive and negative classification. We design a margin ranking loss function for model training, which is described as below.

$$loss_{sRank} = \sum_{t}^{T} max(0, 1 - \delta_s(t) f^{rank}_0(t) + \delta_s(t) f^{rank}_1(t)) \tag{10}$$

where T is the training corpus, f^{rank}_0 is the predicted positive score, f^{rank}_1 is the predicted negative score, $\delta_s(t)$ is an indicator function which reflects the gold sentiment polarity (positive or negative) of a sentence.

$$\delta_s(t) = \begin{cases} 1 & \text{if } f^g(t) = [1, 0] \\ -1 & \text{if } f^g(t) = [0, 1] \end{cases} \tag{11}$$

3.4 Modeling Sentiment of Sentences and Contexts of Words

Up till now, we describe the neural network methods for learning word embeddings by modeling contexts of words (Section 3.2) and sentiment of sentences (Section 3.3) separately. In this section, we introduce two hybrid models, which naturally capture both sentiment of sentences and contexts of words for learning a more powerful sentiment embeddings based on aforementioned models.

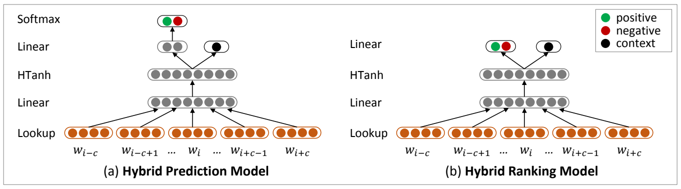


Fig. 3. An illustration of hybrid models that capture contexts of words as well as sentiment of sentences for learning sentiment embeddings.

3.4.1 Hybrid Prediction Model

We combine the context-based prediction model (Section 3.2.1) and the sentiment-based prediction model (Section 3.3.1) to get a hybrid prediction model in this part. We design a hybrid loss function, which is weighted linear combination of the sentiment loss $loss_{sPred}$ (Equation 9) and the context loss $loss_{cPred}$ (Equation 7), where $0 \leq \alpha_{pred} \leq 1$ weights the two parts.

$$loss_{hyPred} = \alpha_{pred} \cdot loss_{sPred} + (1 - \alpha_{pred}) \cdot loss_{cPred} \tag{12}$$

Furthermore, in order to encode sentence and context level information in word representation rather than in the parameters of sentiment-specific of context-specific linear layers, we use shared parameter for the neural layers below $hTanh$, which is illustrated in Figure 3 (a). It is worth noting that, the artificially generated "noises" are only used for calculating context-based loss $loss_{cPred}$, while has no influences on the sentiment loss $loss_{sPred}$.

3.4.2 Hybrid Ranking Model

Similar with the hybrid prediction model, we merge context ranking model (Section 3.2.2) and sentiment ranking model (Section 3.3.2) to get a hybrid ranking model in this part. We design a hybrid loss function, which is weighted linear combination of the sentiment loss $loss_{sRank}$ (Equation 10) and the context loss $loss_{cRank}$ (Equation 8), where $0 \leq \alpha_{rank} \leq 1$ weights two parts. The model is shown in 3 (b). The artificially generated "noises" are only used for calculating context-based loss $loss_{cRank}$, while does not contribute to the sentiment loss $loss_{sRank}$.

$$loss_{hyRank} = \alpha_{rank} \cdot loss_{sRank} + (1 - \alpha_{rank}) \cdot loss_{cRank} \tag{13}$$

Some recent studies [43], [51], [52], [53] also use neural networks to encode additional evidences to word embeddings. They typically design tailored objective function to incorporate task-specific information. Our work is in line with these work as we develop neural models to learn sentiment embeddings by using sentence level sentiment information as task-specific evidences.

3.5 Modeling Lexical Level Information

We investigate lexical level information for enhancing sentiment embeddings in this part. We use two kinds of lexical-level information, namely word-word associations and word-sentiment associations. We develop two regularizers to naturally incorporate them into aforementioned sentiment, context and hybrid neural models.

3.5.1 Integrating Word-Word Association

We model word-word association in this part, holding the consideration that the words from same cluster should be as close with each other in the embedding space. In this work, the word clusters used in this part are obtained automatically from Urban Dictionary, which will be detailed in Section 3.6.

Given two words within the same cluster, our objective is minimizing the distance between them in the embedding space. We formalize it as:

$$loss_{ww} = \lambda_{ww} \sum_{(w,v)\in E} ||e_w - e_v||^2_2 \tag{14}$$

where E is the set of word clusters, e_w and e_v are the embeddings of word w and v, respectively. The above loss function can be naturally integrated with the embedding learning models introduced in Section 3.2, 3.3 and 3.4 as a regularizer.

3.5.2 Integrating Word-Sentiment Association

We integrate word-sentiment association by directly predicting the sentiment polarity of each word regarding the embedding values of each word as features. Accordingly, the optimizing objective is to minimize the negative sum conditional probabilities:

$$loss_{ws} = -\lambda_{ws}(\sum_{w \in Ps} P_{pos}w + \sum_{w \in Ns} P_{neg}w) \qquad (15)$$

where $P_{pos}w$ and $P_{neg}w$ represent the conditional probabilities of classifying a word as positive and negative; Ps and Ns are the set of words with prior positive and negative polarity. The training data will be detailed in Section 3.6.

We give some real examples to better explain the lexical information used for training sentiment embeddings. The top 10 positive words are: *:) :d love :-) like good lol happy thanks haha*, and top 10 negative sentiment words are: *:(:-(sorry sad bad hate ill shit sick fuck*. During training, lexical level sentiment information is directly encoded in the embeddings of each word. Since the learning objective of ultimate model is modeling context and sentiment evidences simultaneously, the words not covered in the lexicon (e.g. "*coooolll*") but sharing similar contexts with the words in the lexicon (e.g. "*cool*" and "*haha*") could be also mapped to close vectors in the embedding space.

In practice, we develop a feed forward neural network consisting of $lookup \rightarrow linear \rightarrow softmax$. The output of lookup layer is the embedding of a word, and the length of linear layer and $softmax$ layer are both C, which is the class number and is 2 for binary sentiment/negative case. The outputs of $softmax$ represent the conditional probabilities of classifying a word as positive and negative categories. The training objective is to minimize the cross entropy error of $softmax$ layer. In this way, word-sentiment association can be conveniently integrated with previously introduced neural networks as a regularizer.

3.6 Training Data and Parameter Learning

We describe the training datasets (Section 3.6.1) and the parameter estimation strategy (Section 3.6.2) for learning sentiment embeddings in this part.

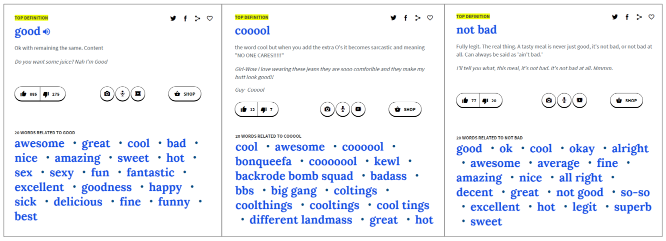
3.6.1 Datasets

As mentioned in Section 3.3 and 3.4, the sentiment-specific models require sentence level sentiment information as supervision to learn sentiment embedding. To model word level information as described in Section 3.5, we need resources containing word-word and word-sentiment associations.

We collect sentence level sentiment information automatically from Twitter. This is based on the consideration that larger training data usually leads to more powerful word representation [1], and it is not practical to manually label sentiment polarity for huge number of sentences. Specifically, we leverage massive tweets containing emoticons as weakly supervised corpora without any manual annotations. We crawl tweets from April 1st, 2013 to April 30th, 2013 with TwitterAPI. We use the carefully selected positive and negative emoticons [54][4], and save the tweets only containing positive emoticons or negative emoticons[5]. These automatically collected tweets contain noises so that they are not good enough if directly used as gold training data to build sentiment classifiers. However, they are effective enough to provide weakly supervised signals for training sentiment embeddings.

Several heuristic rules are employed to filter the noises from the automatically collected tweets. For example, we tokenize each tweet with TwitterNLP [57], remove the @user and URLs of each tweet, and filter the tweets that are too short ($<$ 7 words). Finally, we collect 10M tweets, selected by 5M tweets with positive emoticons and 5M tweets with negative emoticons. The statistics are given in Table 3.

In order to collect resources containing massive word-word associations, we leverage Urban Dictionary without using any manual annotation. Urban Dictionary is a crowd-sourcing resource, examples of which are illustrated in Figure 4. For an word included in Urban Dictionary, such as "good", there exists a web page[6] expressing its definition, sample sentences and related words. We use the related words which we name as word clusters in this work. Specifically, we collect words whose prefix ranges from "a" to "z". There are 799,430 items containing similar words, each of which has about 10.27 similar words on average.



good

Ok with remaining the same. Content

Do you want some juice? Nah I'm Good

awesome • great • cool • bad • nice • amazing • sweet • hot • sex • sexy • fun • fantastic • excellent • goodness • happy • sick • delicious • fine • funny • best

cooool

the word cool but when you add the extra O's it becomes sarcastic and meaning "NO ONE CARES!!!!!"

Girl-Wow i love wearing these jeans they are sooo comforible and they make my butt look good!!

Guy- Cooool

cool • awesome • coooool • bonqueefa • cooooool • kewl • backrode bomb squad • badass • bbs • big gang • coltings • coolthings • cooltings • cool tings • different landmass • great • hot

not bad

Fully legit. The real thing. A tasty meal is never just good, it's not bad, or not bad at all. Can always be said as 'ain't bad.'

I'll tell you what, this meal, it's not bad. It's not bad at all. Mmmm.

good • ok • cool • okay • alright • awesome • average • fine • amazing • nice • all right • decent • great • not good • so-so • excellent • hot • legit • superb • sweet

Fig. 4. Illustrated examples in Urban Dictionary.

In order to collect sentiment information of words, we use the aforementioned word clusters from Urban Dictionary to expand a small size of manually labeled sentiment seeds. Specifically, we manually label the top frequent 500 words from the vocabulary of sentiment embedding as positive, negative or neutral. After removing the ambiguous ones, we obtain 125 positive, 109 negative and 140 neutral words, which are regarded as the sentiment seeds. Afterwards, we use the similar words from Urban Dictionary to expand the sentiment seeds. We formulate this procedure as a k-nearest neighbors (KNN) classifier by regarding sentiment seeds as gold standard. We apply the KNN classifier to the items of Urban Dictionary word cluster, and predict a three-dimensional discrete vector $[knn_{pos}, knn_{neg}, knn_{neu}]$ for each item. Each value reflects the hits numbers of sentiment seeds with different sentiment polarity in its similar words. For example, the vector value of "*coooolll*" is $[10, 0, 0]$, which means that there are 10 positive seeds, 0 negative seeds and 0 neutral seeds occur in its similar words. To ensure the quality of the expanded words, we set threshold for each category to collect the items with

[4] The positive emoticons are :) :) :-) :D =), and the negative emoticons are :(: (:-(.

[5] How to automatically collecting large-scale and robust tweets with neutral sentiment remains as an open problem, even though there are some preliminary attempts [55], [56].

[6] http://www.urbandictionary.com/define.php?term=good

high quality as expanded words. Take positive category as an example, we keep an item as positive expanded word if it satisfies $knn_{pos} > knn_{neg} + threshold_{pos}$ and $knn_{pos} > knn_{neu} + threshold_{pos}$ simultaneously. We empirically set the thresholds of positive, negative and neutral as 6,3,2 respectively by balancing the size of expanded words in three categories. After seed expansion, we collect 1,512 positive, 1,345 negative and 962 neutral words. We also tried the propagation methods to expand the sentiment seeds, namely iteratively adding similar words of sentiment seeds from Urban Dictionary into the expanded word collection. However, the quantity of expanded words is less than the KNN-based results and the quality is worse.

3.6.2 Parameter Learning

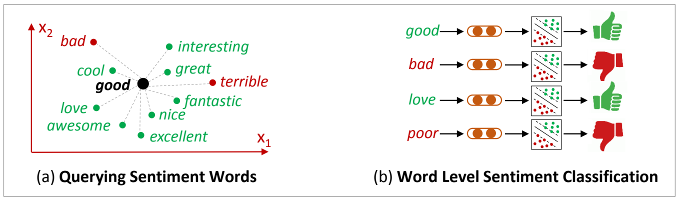
We learn sentiment embeddings by taking the derivative of the loss through back-propagation with respect to the whole set of parameters [14]. We initialize the values of word vectors from a uniform distribution $U(-0.01, 0.01)$, and initialize the values of linear layers by fan-in the input length [14], namely from a uniform distribution $U(\frac{-0.01}{InputLength}, \frac{0.01}{InputLength})$. We empirically set the window size as 7 (preceding 3 words and following 3 words), the embedding length as 50, the length of hidden layer as 20. For parameter update, we use AdaGrad [58], which is widely applied in deep learning literature [59], [60], [61]. The learning rate in AdaGrad is adaptively changed for different parameters at different steps.

$$\theta_t = \theta_{t-1} - \frac{\epsilon}{\sqrt{\sum_{\tau=1}^{t} g_\tau^2}} g_t \tag{16}$$

where θ_t is the value of parameter θ at time step t, g_t is the gradient of θ at time step t, ϵ is the learning rate and is set to 0.1 in our approach.

4 Word Level Sentiment Analysis

We investigate whether sentiment embeddings are useful for discovering similarities between sentiment words in this section. We conduct experiments on word level sentiment analysis in two settings, namely querying neighboring sentiment words in embedding space (Section 4.1) and word level sentiment classification (Section 4.2).



(a) **Querying Sentiment Words**

(b) **Word Level Sentiment Classification**

Fig. 5. An illustration of word level sentiment analysis tasks, including querying sentiment words and word level sentiment classification.

4.1 Querying Sentiment Words

A better sentiment embedding should have the ability to map positive words into close vectors, to map negative words into close vectors, and to separate positive words and negative words apart. Accordingly, in the vector space of sentiment embedding, the neighboring words of a positive word like "good" should be dominated by positive words like "cool", "awesome", "great", etc., and a negative word like "bad" should be surrounded by negative words like "terrible" and "nasty". Based this consideration, we query neighboring sentiment words in existing sentiment lexicon to investigate whether sentiment embeddings are helpful in discovering similarities between sentiment words. Specifically, given a sentiment word as input, we first find out the top N_w closest words in the sentiment lexicon. The closeness of two words is measured by the similarity (e.g. cosine) between their word embeddings. Afterwards, we calculate how much percentage of those neighboring words have the same sentiment polarity with the target sentiment word. This percentage of polarity consistency corresponds to an $Accuracy$ evaluation metric, which is calculated as follows.

$$Accuracy = \frac{\sum_{i=1}^{\#Lex} \sum_{j=1}^{N_w} \delta_w(w_i, c_{ij})}{\#Lex \times N_w} \tag{17}$$

where $\#Lex$ is the number of words in the sentiment lexicon, w_i is the *i-th* word in the lexicon, c_{ij} is the *j-th* closest word to w_i in the lexicon, $\delta_w(w_i, c_{ij})$ is an indicator function whose value is 1 if w_i and c_{ij} have the same sentiment polarity and 0 for the opposite case. Higher accuracy refers to a better sentiment embedding in capturing similarities between sentiment words.

Let us take "good" and $N_w = 10$ as an example to further explain the experimental protocol, which is illustrated in Figure 5 (a). We first find the top 10 closest words in the embedding space, including "cool", "love", "awesome", "bad", etc. We can find that among these 10 neighbors, eight words (except for "bad" and "terrible") sharing the same sentiment polarity with "good". As a result, the accuracy for the word "good" is 80%. We conduct experiments on three benchmark sentiment lexicons, BL-Lexicon [19][7], MPQA [20][8] and NRC-Lexicon [62][9]. The statistics of the lexicons are given in Table 6. We set N_w as 10 and 30 separately to evaluate the performance of sentiment embeddings. We compare to the following embedding learning algorithms by apply these algorithms on our tweet dataset to learn word embeddings.

- C&W: C&W model is a representative algorithm [36] for learning word embeddings, which is the context-ranking model as described in Section 3.2.2.
- Word2vec: Mikolov et al. [38] develop CBOW and SkipGram to learn continuous word vectors, and embed these two algorithms in a widely used toolkit "word2vec". We utilize CBOW in the experiments, which is analogous to our context-prediction model.
- ReEmbed: Labutov and Lipson [47] learn task-specific embeddings based on an existing background embedding and sentiment-specific corpus. *Re(C&W)* and *Re(Word2vec)* stand for the use of different word embeddings as background embedding.

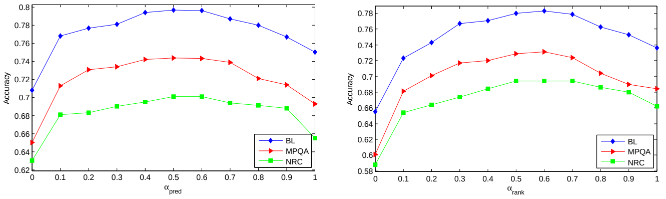
[7] http://www.cs.uic.edu/liub/FBS/sentiment-analysis.html#lexicon
[8] http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/
[9] http://saifmohammad.com/WebPages/lexicons.html

TABLE 1
Accuracy of querying sentiment words in different sentiment lexicons using different word embeddings. The best method is in **bold**.

Embedding	$N_w = 10$			$N_w = 30$		
	BL	MPQA	NRC	BL	MPQA	NRC
C&W	65.5	60.1	58.8	64.4	59.4	58.1
Word2vec	70.8	65.0	63.0	68.8	63.2	61.3
Re(C&W)	67.1	61.8	59.6	65.9	60.9	58.6
Re(word2vec)	71.2	65.7	63.4	69.6	64.1	61.8
SE-SPred	75.0	69.3	65.5	74.6	69.0	65.1
SE-SRank	73.6	68.4	66.2	73.8	68.3	65.9
SE-HyPred	**79.7**	**74.4**	**70.1**	77.4	72.1	67.9
SE-HyRank	78.3	73.1	69.4	**77.9**	**72.6**	**68.9**

From Table 1, we can find that experimental results with different neighbor size N_w are almost consistent. Among all these methods, context-based embedding learning algorithms (C&W, word2vec) perform lower than other sentiment-specific models. The reason lies in that they encode contexts of words into word embeddings while ignoring sentiment information of texts, so that they are not capable of distinguishing the words with similar context but opposite sentiment polarity like "good" and "bad". After re-embedding background word embeddings with sentiment of texts, the accuracy of querying sentiment words improves. This shows the importance of sentiment of texts for boosting word embeddings in capturing similarities between sentiment words. In all these experiments, we can find that sentiment models ("SE-SPred" and "SE-SRank") and hybrid models ("SE-HyPred" and "SE-HyRank") outperform baseline methods. The proposed methods differ from context-based models in that we capture sentiment information of texts, which provides crucial evidences for capturing similarities between sentiment words. The hybrid models can be viewed as the "joint" version of ReEmbed by simultaneously encoding contexts of words and sentiment of sentences into word representation from scratch. Two hybrid models yields best performances as they capture not only contexts of words but also sentiment information of sentences. Since we evaluate sentiment embeddings on word level sentiment analysis lexicons, we do not compare with the embeddings learned with word level information for fair comparison in this part.

We study the influences of context parts and sentiment parts in hybrid models by varying α_{pred} and α_{rank} from 0 to 1, increased by 0.1. The SE-HyRank model with α_{rank} = 0 is essentially the C&W model, which does not capture any sentiment information of text. On the contrary, SE-HyRank model with α_{rank} = 1 is the sentiment ranking model as described in Section 3.3.2. SE-HyPred model shares a similar characteristic. We conduct experiments on three sentiment lexicons (BL, MPQA and NRC), and set N_w = 10 in this setting. Experimental results are given in Figure 6. We can find that the trends of Figure 6 (a) and Figure 6 (b) are similar with each other. There is an obvious performance boost at α = 0.1, where sentiment of text is first integrated into context-based model. This results show that sentiment information is very important, adding of which can help to discover similarities between sentiment words. Both hybrid models yield best performances when α is in range [0.5,0.6], which balances context and sentiment information.



(a) SE-HyPred with different α_{pred}.

(b) SE-HyRank with different α_{rank}.

Fig. 6. Experimental results on querying sentiment words on three sentiment lexicons. We set N_w as 10.

4.2 Word Level Sentiment Classification

We conduct word level sentiment classification to further investigate the effectiveness of sentiment embeddings in capturing similarities between sentiment words. Specifically, we conduct binary polarity classification on three sentiment lexicons (BL, MPQA and NRC) to infer whether a sentiment word expresses a positive or a negative meaning. The continuous word embeddings of a word are considered as its features for classification. We conduct N-Fold cross validation for each dataset, regarding (N-1) parts as training data and one part as test data. We train supervised classifiers N times and average the classification accuracies as the final evaluation metric. Experimental results with N = 5 and N = 10 are given in Table 2.

TABLE 2
Accuracy of word level sentiment classification (positive vs negative) using different word embeddings. The best method is in **bold**.

Embedding	N-Fold = 5			N-Fold = 10		
	BL	MPQA	NRC	BL	MPQA	NRC
C&W	76.2	70.5	68.1	76.6	69.8	67.5
Word2vec	80.1	75.7	71.7	81.8	73.4	71.2
Re(C&W)	75.0	71.3	68.6	76.6	71.4	68.2
Re(word2vec)	81.5	76.9	71.6	83.1	75.3	73.1
SE-SPred	80.8	78.2	73.3	80.4	76.0	72.4
SE-SRank	78.6	76.2	71.3	77.8	71.1	71.2
SE-HyPred	**86.0**	**85.3**	**77.5**	**87.0**	**83.4**	76.9
SE-HyRank	83.9	80.1	75.7	85.6	79.2	**77.0**

We can find that the results with different fold size are consistent. Purely context-based methods including C&W and Word2vec performs relatively poor as they do not use any sentiment information of text. Re-Embedding them with sentiment information slightly improve the classification accuracy. Among the bottom four sentiment embedding learning algorithms, we can see that hybrid models (SE-HyPred, SE-HyRank) obviously show superior performances than sentiment models (SE-SPred, SE-SRank). This shows that both context and sentiment information are helpful for discovering similarities between sentiment words, and incorporating sentiment information significantly boost the classification accuracy. SE-HyPred performs best in this task.

Comparing between SE-HyPred and SE-HyRank, we can find in most cases SE-HyPred preforms slightly better. These two methods use the same information, including contexts of words and sentiment of sentences. The difference is that SE-HyPred uses sentiment information in a classification way while SE-HyRank uses sentiment information in a ranking fashion. In this binary case, we believe that SE-HyPred is more suitable as the sentiment labels (positive vs. negative) are two individual categories. There does not exist an apparent ranking relation between these sentiment labels. For the rating stars in review cites (e.g. Amazon and Yelp), SEHyRank might be better.

5 Sentence Level Sentiment Classification

In this part, we apply sentiment embedding as features to sentiment level sentiment classification. This helps us to investigate whether sentiment embedding is capable of capturing discriminative features for classifying the polarity labels (e.g. thumbs up or thumbs down) of text. We first present our strategy of using sentiment embedding as features for sentiment classification. We then describe experimental settings and empirical results.

5.1 Sentence Level Sentiment Classification

We apply sentiment embeddings in a supervised learning framework for sentiment classification of sentences [63], [64]. Instead of using hand-crafting features, we use sentiment embeddings to compose the feature of a sentence. The sentiment classifier is built from sentences with manually annotated sentiment polarity.

Specifically, we use a semantic composition based framework [1] to get sentence representation. The basic idea is to compose sentence level features from sentiment embeddings of words. This is based on the principal of compositionality [65], which states that the meaning of a longer expression (e.g. a sentence) is determined by the meaning of words it contains.

We use max, $average$ and min pooling layers [1], [3] to obtain the sentence representation, which have been used as simple and effective methods for compositionality learning in vector-based semantics [66]. Each pooling layer $pooling_p$ employs the embedding of words and conducts matrix-vector operation of p on the sequence represented by columns in each lookup table. $z(s)$ is the concatenation of results obtained from different pooling functions. In this way, there is no additional parameters in the composition component, so that we can naturally build a SVM classifier with the composed sentence representation and compare to other state-of-the-art features [64], [67] for fair comparison. Furthermore, it is also convenient to investigate whether there is a further performance boost by integrating sentence embedding feature with existing feature sets. Sentiment embeddings can also be naturally fed to other semantic composition models like Recursive Neural Network and Convolution Neural Network.

We conduct experiment on two types of datasets, one Twitter dataset from SemEval [21], [68] and another review dataset from Rotten Tomatoes [23]. The statistics of the datasets are given in Table 3. On SemEval dataset, we follow the experimental protocols officially provided by SemEval 2014 [68]. We train sentiment classifier from "SemEval-2013Train", tune parameters on "SemEval-2013Dev" and test the performance of the classifier on "SemEval-2013Test" and "SemEval-2014Test". The "SemEval-2013Train" and "SemEval-2013Dev" datasets were completely in full to task participants of SemEval 2013. However, we were unable to download all the training and development sets because some tweets were deleted or not available due to modified authorization status.

We conduct binary classification (positive vs negative) on SemEval dataset as the sentiment embeddings are trained with only positive and negative sentiment supervisions. As a reference, we also conduct ternary classification (positive vs negative vs neutral) on SemEval dataset. It is worth noting that, Twitter sentiment classification evaluation in SemEval asks participants to do ternary classificaton over positive, negative and neutral categories. However, the official evaluation metric is macro-F1 score over positive and negative categories. In this work, we use macro-F1 score over all categories for both binary and ternary classification. For ternary classification, Macro-F1 is the average of F1-Score of positive, negative and neutral categories.

$$MacroF1 = (F1_{positive} + F1_{negative} + F1_{neutral})/3 \quad (18)$$

where each $F1$ score is calculated as an combination of precision (P) and recall (R).

$$F1 = 2 * (P + R)/P * R \quad (19)$$

TABLE 3
Statistics of datasets. #Pos, #Neg and #Neu are the number of positive, negative and neutral instances. l_{avg} is average length of sentences, $|V|$ is the vocabulary size.

Dataset	#Pos	#Neg	#Neu	l_{avg}	$\|V\|$
Tweets-Emoticon	5M	5M	0	13.6	1.84M
SemEval-2013Train	2,642	994	3,436	21.5	17,153
SemEval-2013Dev	408	219	493	21.6	4,731
SemEval-2013Test	1,572	601	1,640	21.5	11,345
SemEval-2014Test	982	202	669	21.3	6,042
Review-Embed-Train	0.22M	0.22M	0	23.2	0.14M
Review-All	5,331	5,331	0	21.0	20,263

5.2 Experimental Results

We show empirical experiments on tweet level sentiment classification in this part. We compare with several standard and strong baseline methods as follows.

- Dist+Ngrams. We use the tweets containing positive and negative emoticons as distant supervised training data to build binary sentiment classifier [69]. We use unigram, bigram and trigram features and train SVM classifier with LibLinear [70].
- SVM+Ngrams. In sentiment analysis community, SVM classifier with bag of ngrams [63] is a standard baseline for sentiment classification. We try unigram, bigram and trigram featurs.
- SVM+TextFeatures. We build a state-of-the-art baseline [64], [67], which uses many complexed features such as word ngrams, character ngrams, lexicon features, cluster features, etc.

The sentiment embeddings learned with Hybrid models as well as lexical level information are named as "SE-HyPred-Lex" and "SE-HyRank-Lex". We use sentiment embeddings as features and build sentiment classifier with LibLinear [70]. Experimental results are given in Table 4.

From Table 4, we can find that distant supervision ("Dist") does not perform well. The reason is that the automatically collected tweets contain noises, which are regarded as the

TABLE 4
Macro-F1 of Twitter sentiment classification on SemEval datasets with different classification algorithms.

Method	Positive/Negative		Pos/Neg/Neu	
	2013Test	2014Test	2013Test	2014Test
Dist+unigrams	73.7	73.3	–	–
Dist+{1, 2}-grams	74.0	75.2	–	–
Dist+{1, 2, 3}-grams	74.0	75.7	–	–
SVM+unigrams	72.1	68.4	55.8	54.8
SVM+{1, 2}-grams	72.7	70.0	57.4	57.6
SVM+{1, 2, 3}-grams	72.7	69.6	56.9	57.1
SVM+TextFeatures	83.8	83.9	66.6	66.8
SE-HyPred-Lex	81.2	80.5	63.3	62.9
SE-HyPred-Lex+TF	84.7	84.4	67.3	67.1
SE-HyRank-Lex	84.0	84.3	64.7	62.8
SE-HyRank-Lex+TF	**86.1**	**86.6**	**67.9**	**67.5**

gold standard to build the Twitter sentiment classifier. We can see that bag of ngram features are not powerful enough for Twitter sentiment classification as it can not well capture the sophisticated semantics of tweet. Complexed text level feature is an extremely strong performer on Twitter sentiment classification, which shows the effectiveness of feature engineering. For binary classification, we can find that only using sentiment embedding feature can obtain comparable performance with text features on both datasets. This shows that sentiment embeddings are helpful in capturing discriminative features for predict the positive/negative sentiment of text. For ternary classification, sentiment embeddings are still worse than complexed text features. The reason is that sentiment embeddings are learned from tweets with only positive or negative supervisions, without adequate neutral supervisions of tweets. As a result, it does not capture sufficient discriminative information for ternary sentiment classification. Concatenating sentiment embeddings with text features ("SE-Hy**+TF") yields further improvements for both binary and ternary classification.

We compare sentiment embeddings with several baseline embedding learning algorithms for Twitter sentiment classification. The word embeddings are learned from the same tweet dataset and further applied as the unique sentence feature with the composition strategy as described in Section 5.1. We build Twitter sentiment classifier with LibLinear [70].

TABLE 5
Macro-F1 of Twitter sentiment classification on SemEval datasets with different word embeddings features.

Method	Positive/Negative		Pos/Neg/Neu	
	2013Test	2014Test	2013Test	2014Test
C&W	70.2	69.1	53.2	53.1
Word2vec	70.7	70.6	51.8	50.8
Re(C&W)	74.1	76.2	58.3	58.7
Re(Word2ve)	72.5	74.8	56.8	57.2
SE-Pred	80.8	79.1	61.4	61.8
SE-Rank	79.8	81.5	60.9	59.0
SE-HyPred	80.5	79.2	62.0	61.6
SE-HyRank	82.3	81.8	63.4	61.5
SE-HyPred-Lex	81.2	80.5	63.3	**62.9**
SE-HyRank-Lex	**84.0**	**84.3**	**64.7**	62.8

From Table 5, we can find that the results on binary and ternary classifications are almost consistent. Context-based embeddings are relatively weak in this setting as they do not use the crucial sentiment factors of texts. Reembedding context-based embeddings with sentiment of texts can slightly boosts the performances. Compared with baseline methods, our Hybrid models (SE-HyPred and SE-HyRank) get better classification performances by modeling contexts of words and sentiment of sentences simultaneously from scratch. After interacting with lexical level information, sentiment embeddings get further improvements. These empirical results further demonstrate that sentiment embeddings are helpful in capturing discriminative features for predict the sentiment of text.

5.3 Sentiment Classification of Reviews

We apply sentiment embeddings for sentiment classification of reviews to further investigate its ability in discovering discriminative features from different domains. We run supervised learning pipeline regarding word embeddings as features. The sentiment embeddings are learned from movie reviews ("Review-Embed-Train"), whose statistical information are given in Table 3. We conduct 10-fold cross-validation on "Review-All" from Rotten Tomatoes [23] and use accuracy as the evaluation metric.

Classification accuracies using different word embeddings are shown in Figure 7. We can find that re-embedding Re(**) always improves classification performances, e.g. Re(C&W) improves C&W and Re(Word2Vec) improves Word2vec. These results indicate that encoding sentiment information of texts could boost the accuracy of sentiment classification. Compared with baseline word embeddings, hybrid models ("SE-HyPred" and "SE-HyRank") which use context and sentiment level information yield best performances on sentence level sentiment classification of reviews. The difference between hybrid models and reembeddings is that hybrid models learn sentiment embeddings directly while reembeddings modify a pretrained context based word embeddings. From this perspective, the hybrid models can be viewed as joint models that simultaneously capture context and sentiment information.

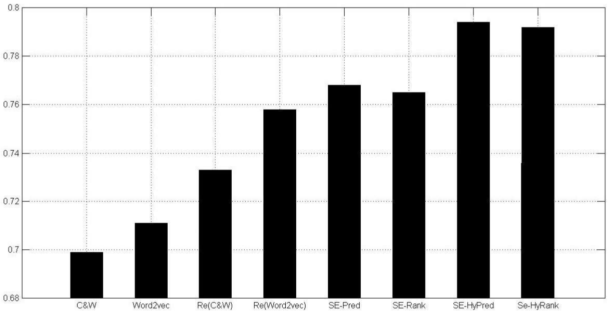


Fig. 7. Accuracy of binary sentiment classification (positive/negative) of review sentences. Word embeddings are used as features.

It is worth noting that sentiment embeddings could be naturally extended to multi-dimensional sentiment classification (e.g. emotions in Experience Project dataset or 1-5 stars in review site). Sentiment embeddings are regarded as feature representations in a machine learning based pipeline. One could design task-oriented objective function (e.g. cross entropy error over multi categories) in order to fit for the multi-dimensional sentiment labels. We conduct

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2489653, IEEE Transactions on Knowledge and Data Engineering

multi-dimensional sentiment classification on restaurant reviews from Yelp [71], and find that sentiment embeddings performs slightly better than SkipGram in terms of classification accuracy (see [71] for more details).

6 Building Sentiment Lexicon

We apply sentiment embeddings to building sentiment lexicon, which is useful for measuring the extent to which sentiment embeddings improve lexical level tasks that need to find similarities between words. We introduce a classification approach to build sentiment lexicon by regarding sentiment embeddings as word features, and then describe experimental settings and the results.

6.1 A Classification Approach for Building Sentiment Lexicon

We describe a classification approach for building large-scale sentiment lexicon, which is illustrated in Figure 8. We cast sentiment lexicon learning as a word-level classification task, which consists of two part: (1) an embedding learning algorithm to effectively learn the continuous representation of words, which are used as features for word-level sentiment classification, (2) a seed expansion algorithm that expands a small list of sentiment seeds to collect training data for building the word-level classifier. The learned sentiment embeddings are naturally regarded as continuous word features. Seed expansion procedure has been introduced in Section 3.6.1.

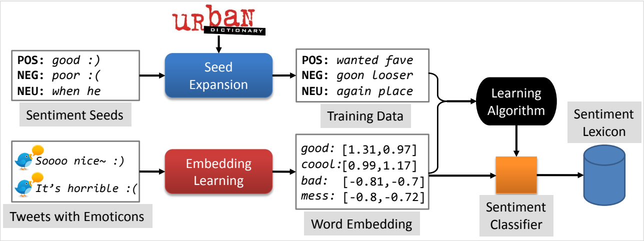


Fig. 8. A classification approach for building sentiment lexicon.

After obtaining the training data and feature representation of words, we build a word-level sentiment classifier with $softmax$, whose length is two for the positive *vs* negative case:

$$\mathbf{y}(w) = softmax(\theta \cdot e_i + b) \tag{20}$$

where θ and b are the parameters of classifier, e_i is the embedding of a word w_i, $\mathbf{y}(w)$ is the predicted sentiment distribution of w_i. We employ the classifier to predict the sentiment distribution of each word in the vocabulary of sentiment embeddings, and save the words as well as their sentiment probability in the positive (negative) lexicon if the positive (negative) probability is larger than 0.5.

6.2 Experimental Settings and Results

We evaluate the effectiveness of the automatically generated sentiment lexicons. As it is labor intensive to manually check the lexicon accuracy, we apply it as features to existing supervised learning pipeline for Twitter sentiment classification. We compare it with traditional sentiment lexicons including BL-Lexicon, MPQA and NRC-Lexicon as well as automatically generated sentiment lexicons including HashtagLex and Sentimenet140Lex [64]. The statistics of baseline lexicons and our lexicons are given in Table 6. Traditional sentiment lexicons with a relative small lexicon size. HashtagLex and Sentiment140Lex are Twitter-specific sentiment lexicons. Our lexicons (SE-HyPred Lexicon and SE-HyRank Lexicon) are larger than traditional sentiment lexicons, and have comparable size with automatically generated lexicons.

TABLE 6
Statistics of sentiment lexicons (unigram only).

Sentiment Lexicon	Size	#Positive	#Negative
BL-Lexicon	6,786	2,006	4,780
MPQA	6,451	2,301	4,150
NRC-Lexicon	5,555	2,231	3,324
HashtagLex	54,129	32,048	22,081
Sentiment140Lex	62,468	38,312	24,156
SE-HyPred Lexicon	65,854	33,829	32,025
SE-HyRank Lexicon	64,603	31,591	33,012

We evaluate the effectiveness of sentiment lexicons by applying them as features for Twitter sentiment classification in a state-of-the-art supervised learning pipeline [64], [67]. We use lexicon feature [64] as the unique feature to build sentiment classifier. Specifically, the lexicon features for each sentiment polarity (positive or negative) are:

- total count of tokens in the tweet with score greater than 0;
- the sum of the scores for all tokens in the tweet;
- the maximal score;
- the non-zero score of the last token in the tweet;

We conduct experiments on the SemEval datasets (see Table 3), training model on "SemEval-2013Train", tuning parameters on "SemEval-2013Dev" and testing on "SemEval-2013Test" as well as "SemEval-2014Test". We conduct both binary (positive/negative) and ternary (positive/negative/neutral) classification. Experiment results with different sentiment lexicons are given in Table 7.

TABLE 7
Macro-F1 of Twitter sentiment classification on SemEval datasets with different sentiment lexicon features (Unique).

Method	Positive/Negative		Pos/Neg/Neu	
	2013Test	2014Test	2013Test	2014Test
BL-Lexicon	66.9	56.8	36.5	37.8
MPQA	65.0	58.4	39.1	39.4
NRC-Lexicon	62.3	55.4	34.8	37.7
HashtagLex	63.0	60.1	36.1	34.7
Sentiment140Lex	70.8	**76.7**	46.1	**47.1**
SE-HyPred	**76.0**	72.4	**51.5**	46.4
SE-HyRank	73.7	74.0	50.2	47.0

From Table 7, we can find that the performances of traditional lexicons are relatively low. This is because that in this setting sentiment lexicon feature is the only feature, however, standard sentiment lexicons are typically with a small size and low coverage. As a result, many ill-formed expressions on Twitter can not be covered. Sentiment140Lex is the strongest performer among all baseline sentiment lexicons. This indicates the effectiveness of the automatically generated lexicon. In addition, this show that emoticons are better sentiment signals than sentimental hashtags when used for building sentiment lexicons. On "2013Test" dataset, our lexicons show superior performances over all baseline methods. On "2014Test" dataset our lexicons perform comparable with the best baseline lexicon Sentiment140Lex.

We further compare sentiment embeddings with context-based word embeddings in building sentiment lexicon. From Table 8, we can find that on binary classification, sentiment embeddings obviously outperform baseline word embeddings, which verifies its ability in finding similarities between words. However, the improvement is not so significant on ternary classification because the training data of sentiment embeddings do not contain neutral supervisions.

TABLE 8
Macro-F1 of Twitter sentiment classification on SemEval datasets with different sentiment lexicon features (Unique).

Method	Positive/Negative		Pos/Neg/Neu	
	2013Test	2014Test	2013Test	2014Test
C&W	68.0	63.5	42.5	41.8
Word2vec	69.8	66.8	45.2	42.9
Re(C&W)	72.5	65.4	46.5	45.0
Re(Word2vec)	71.1	69.6	49.0	46.2
SE-Pred	74.8	71.4	50.1	45.7
SE-Rank	72.8	72.2	49.6	45.9
SE-HyPred	**76.0**	72.4	**51.5**	46.4
SE-HyRank	73.7	**74.0**	50.2	**47.0**

We also conduct experiments in an "Append" setting by appending lexicon features to existing basic features. We use the feature sets of [64] excluding lexicon feature as the basic feature, including bag of words, pos-tagging, emoticons, hashtags, elongated words, etc. Experimental results in "Append" setting are given in Table 9. We can find that the classification performances with different lexicons are close. This is because the influence of lexicon features is weakened when combined with hundreds of thousands of discrete features like bag of words. Our lexicons perform comparably with the best performed baseline lexicon in this setting.

TABLE 9
Macro-F1 of Twitter sentiment classification on SemEval datasets with different sentiment lexicon features (Append).

Method	Positive/Negative		Pos/Neg/Neu	
	2013Test	2014Test	2013Test	2014Test
BL-Lexicon	81.2	80.9	64.2	**66.4**
MPQA	79.2	79.8	63.2	65.7
NRC-Lexicon	79.8	79.4	62.6	64.0
HashtagLex	80.2	81.5	62.8	64.0
Sentiment140Lex	80.1	81.7	63.3	65.5
SE-HyPred Lexicon	**81.7**	81.4	**64.9**	64.5
SE-HyRank Lexicon	81.0	**81.9**	63.8	65.2

We discuss about some limitations and future directions of the classification based lexicon learning approach introduced in this paper. It is well accepted that feature representation is the key to build a powerful classifier. One constrain of the approach is that each word in the training/test process requires to have an embedding vector, which is regarded as the feature representation. However, the vocabulary of word embeddings cannot cover every word due to limit of the training corpus. The approach proposed in this work cannot handle the words not covered in the embedding vocabulary. How to deal with the newly generated words in social media is an interesting future work.

7 Conclusion

We learn sentiment-specific word embeddings (named as sentiment embeddings) in this paper. Different from majority of exiting studies that only encode word contexts in word embeddings, we factor in sentiment of texts to facilitate the ability of word embeddings in capturing word similarities in terms of sentiment semantics. As a result, the words with similar contexts but opposite sentiment polarity labels like "good" and "bad" can be separated in the sentiment embedding space. We introduce several neural networks to effectively encode context and sentiment level informations simultaneously into word embeddings in a unified way. The effectiveness of sentiment embeddings are verified empirically on three sentiment analysis tasks. On word level sentiment analysis, we show that sentiment embeddings are useful for discovering similarities between sentiment words. On sentence level sentiment classification, sentiment embeddings are helpful in capturing discriminative features for predicting the sentiment of sentences. On lexical level task like building sentiment lexicon, sentiment embeddings are shown to be useful for measuring the similarities between words. Hybrid models that capture both context and sentiment information are the best performers on all three tasks.

Acknowledgments

We gratefully acknowledge the helpful discussions with Yaming Sun. We thank the editor and anonymous reviewers for their helpful comments and feedbacks. This work was supported by the National High Technology Development 863 Program of China (No. 2015AA015407), National Natural Science Foundation of China (No. 61133012 and No. 61273321). Duyu Tang is supported by Baidu Fellowship and IBM Ph.D. Fellowship.

References

[1] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *Procedding of the 52th Annual Meeting of Association for Computational Linguistics.*, 2014, pp. 1555–1565.

[2] D. Tang, F. Wei, B. Qin, M. Zhou, and T. Liu, "Building large-scale twitter-specific sentiment lexicon : A representation learning approach," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, 2014, pp. 172–182.

[3] D. Tang, F. Wei, B. Qin, T. Liu, and M. Zhou, "Coooolll: A deep learning system for twitter sentiment classification," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 2014, pp. 208–212.

[4] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*. MIT press, 1999.

[5] D. Jurafsky and H. James, "Speech and language processing an introduction to natural language processing, computational linguistics, and speech," 2000.

[6] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *The Conference on Neural Information Processing Systems*, 2013.

[8] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[9] Z. S. Harris, "Distributional structure." *Word*, 1954.

[10] N. Yang, S. Liu, M. Li, M. Zhou, and N. Yu, "Word alignment modeling with context dependent deep neural network." in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013, pp. 166–175.

[11] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in *Annual Meeting of the Association for Computational Linguistics*, 2013.

[12] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. Daumé III, "A neural network for factoid question answering over paragraphs," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 633–644.

[13] J. Li, R. Li, and E. Hovy, "Recursive deep models for discourse parsing," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 2061–2069.

[14] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[15] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and trends in information retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.

[16] B. Liu, "Sentiment analysis and opinion mining," *Synthesis Lectures on Human Language Technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[17] R. Feldman, "Techniques and applications for sentiment analysis," *Communications of the ACM*, vol. 56, no. 4, pp. 82–89, 2013.

[18] J. Ma, Y. Zhang, and J. Zhu, "Tagging the web: Building a robust web tagger with neural network," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014, pp. 144–154.

[19] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2004, pp. 168–177.

[20] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2005, pp. 347–354.

[21] P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter, and T. Wilson, "Semeval-2013 task 2: Sentiment analysis in twitter," in *Proceedings of the International Workshop on Semantic Evaluation*, vol. 13, 2013.

[22] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," in *Proceedings of Special Interest Group on Information Retrieval (SIGIR)*. ACM, 2015, pp. 959–962.

[23] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 2005, pp. 115–124.

[24] A. Severyn and A. Moschitti, "On the automatic learning of sentiment lexicons," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, 2015.

[25] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[26] L. D. Baker and A. K. McCallum, "Distributional clustering of words for text classification," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1998, pp. 96–103.

[27] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *JASIS*, vol. 41, no. 6, pp. 391–407, 1990.

[28] K. Lund and C. Burgess, "Producing high-dimensional semantic spaces from lexical co-occurrence," *Behavior Research Methods, Instruments, & Computers*, vol. 28, no. 2, pp. 203–208, 1996.

[29] R. Lebret, J. Legrand, and R. Collobert, "Is deep learning really necessary for word embeddings?" *NIPS workshop*, 2013.

[30] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[31] N. Jones, "Computer science: The learning machines." *Nature*, vol. 505, no. 7482, p. 146, 2014.

[32] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," 2015, book in preparation for MIT Press.

[33] Y. Bengio, J.-S. Senécal *et al.*, "Quick training of probabilistic neural nets by importance sampling," in *AISTATS Conference*, 2003.

[34] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proceedings of the international workshop on artificial intelligence and statistics*, 2005, pp. 246–252.

[35] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 641–648.

[36] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[37] N. A. Smith and J. Eisner, "Contrastive estimation: Training log-linear models on unlabeled data," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005, pp. 354–362.

[38] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *International Conference on Learning Representations*, 2013.

[39] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Advances in Neural Information Processing Systems*, 2013, pp. 2265–2273.

[40] M. U. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 307–361, 2012.

[41] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *ACL*. ACL, 2012, pp. 873–882.

[42] Y. Sun, L. Lin, N. Yang, Z. Ji, and X. Wang, "Radical-enhanced chinese character embedding," *Neural Information Processing - 21st International Conference, ICONIP*, 2014.

[43] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014, pp. 302–308.

[44] J. Li and D. Jurafsky, "Do multi-sense embeddings improve natural language understanding?" in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1722–1732.

[45] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," *North American Chapter of the Association for Computational Linguistics*, 2015.

[46] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2011.

[47] I. Labutov and H. Lipson, "Re-embedding words," in *Annual Meeting of the Association for Computational Linguistics*, 2013.

[48] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014, pp. 238–247.

[49] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing*. Springer, 1990, pp. 227–236.

[50] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, 1988.

[51] K. Hashimoto, P. Stenetorp, M. Miwa, and Y. Tsuruoka, "Task-oriented learning of word embeddings for semantic relation classification," in *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, 2015, pp. 268–278.

[52] S. Gouws and A. Søgaard, "Simple task-specific bilingual word embeddings," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 1386–1390.

[53] Z. Chen, W. Lin, Q. Chen, X. Chen, S. Wei, H. Jiang, and X. Zhu, "Revisiting word embedding for contrasting meaning," in *Proceed-*

ings of the 53rd Annual Meeting of the Association for Computational Linguistics, 2015, pp. 106–115.

[54] X. Hu, J. Tang, H. Gao, and H. Liu, "Unsupervised sentiment analysis with emotional signals," in *Proceedings of the International World Wide Web Conference*, 2013, pp. 607–618.

[55] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in *Proceedings of Language Resources and Evaluation Conference*, vol. 2010, 2010.

[56] K.-L. Liu, W.-J. Li, and M. Guo, "Emoticon smoothed language models for twitter sentiment analysis," in *The Association for the Advancement of Artificial Intelligence*, 2012.

[57] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith, "Part-of-speech tagging for twitter: Annotation, features, and experiments," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2011, pp. 42–47.

[58] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, pp. 2121–2159, 2011.

[59] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1631–1642.

[60] J. Li and E. Hovy, "A model of coherence based on distributed sentence representation," in *Proceedings of Empirical Methods in Natural Language Processing*, 2014.

[61] J. Li, T. Luong, D. Jurafsky, and E. Hovy, "When are tree structures necessary for deep learning of representations?" in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 2304–2314.

[62] S. M. Mohammad and P. D. Turney, "Crowdsourcing a word–emotion association lexicon," *Computational Intelligence*, vol. 29, no. 3, pp. 436–465, 2013.

[63] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002, pp. 79–86.

[64] S. M. Mohammad, S. Kiritchenko, and X. Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," *Proceedings of the International Workshop on Semantic Evaluation*, 2013.

[65] G. Frege, "On sense and reference," *Ludlow (1997)*, pp. 563–584, 1892.

[66] J. Mitchell and M. Lapata, "Composition in distributional models of semantics," *Cognitive Science*, vol. 34, no. 8, pp. 1388–1429, 2010.

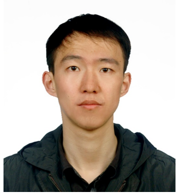
[67] S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," *Journal of Artificial Intelligence Research*, pp. 723–762, 2014.

[68] S. Rosenthal, A. Ritter, P. Nakov, and V. Stoyanov, "Semeval-2014 task 9: Sentiment analysis in twitter," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 2014, pp. 73–80.

[69] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, pp. 1–12, 2009.

[70] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

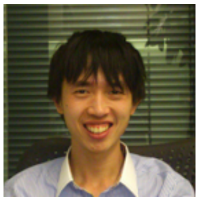
[71] D. Tang, B. Qin, T. Liu, and Y. Yang, "User modeling with neural network for review rating prediction," in *Proc. IJCAI*, 2015, pp. 1340–1346.



Duyu Tang received the Master's Degree in Jul. 2012 from the Department of Computer Science, Harbin Institute of Technology, Harbin, China. Since 2012, he is a Ph.D. candidate at the Department of Computer Science, Harbin Institute of Technology. His current research interests include natural language processing, sentiment analysis and deep learning.

Furu Wei received his Ph.D Degree in Jun. 2009 from Department of Computer Science, Wuhan University. He is a lead researcher in Natural Language Computing group at Microsoft Research Asia, Beijing, China. Before joining MSRA-NLC in Nov. 2010, he has been a staff researcher at IBM Research - China (IBM CRL) since Jul. 2009. His research interests include natural language processing, information retrieval and machine learning.

Bing Qin received her Ph.D Degree in 2005 from the Department of Computer Science, Harbin Institute of Technology, Harbin, China. She is a full professor of Department of Computer Science, and the deputy director of Research Center for Social Computing and Information Retrieval (HIT-SCIR) from Harbin Institute of Technology. Her research interests include natural language processing, information extraction, document-level discourse analysis and sentiment analysis.



Nan Yang received the Ph.D Degree in Jun. 2014 from University of Science and Technology, Hefei, China. He currently works as an associate researcher in Microsoft Research, Beijing, China. His research interests lie in the interaction between natural language processing and machine learning, especially in machine translation and deep learning.

Ting Liu received his Ph.D Degree in 1998 from the Department of Computer Science, Harbin Institute of Technology, Harbin, China. He is a full professor of Department of Computer Science, and the director of Research Center for Social Computing and Information Retrieval (HIT-SCIR) from Harbin Institute of Technology. His research interests include information retrieval, natural language processing and social media analysis.

Ming Zhou received his Ph.D. Degree in 1991 from computer science, Harbin Institute of Technology, Harbin, China. He is a principal researcher, manager of Microsoft Research Asia Natural Language Computing Group, Microsoft Research, Beijing, China. He came to Microsoft in 1999 from his post as an associate professor of computer science at Tsinghua University. He is an expert in the areas of machine translation and natural language processing.