Predicting Polarities of Tweets by Composing Word Embeddings with Long Short-Term Memory

Xin Wang[1], **Yuanchao Liu**[1], **Chengjie Sun**[1], **Baoxun Wang**[2] **and Xiaolong Wang**[1]
[1]School of Computer Science and Technology,
Harbin Institute of Technology, Harbin, China
[2]Application and Service Group, Microsoft, Beijing, China
[1]{xwang,lyc,cjsun,wangxl}@insun.hit.edu.cn
[2]baoxwang@microsoft.com

Abstract

In this paper, we introduce Long Short-Term Memory (LSTM) recurrent network for twitter sentiment prediction. With the help of gates and constant error carousels in the memory block structure, the model could handle interactions between words through a flexible compositional function. Experiments on a public noisy labelled data show that our model outperforms several feature-engineering approaches, with the result comparable to the current best data-driven technique. According to the evaluation on a generated negation phrase test set, the proposed architecture doubles the performance of non-neural model based on bag-of-word features. Furthermore, words with special functions (such as negation and transition) are distinguished and the dissimilarities of words with opposite sentiment are magnified. An interesting case study on negation expression processing shows a promising potential of the architecture dealing with complex sentiment phrases.

1 Introduction

Twitter and other similar microblogs are rich resources for opinions on various kinds of products and events. Detecting sentiment in microblogs is a challenging task that has attracted increasing research interest in recent years (Hu et al., 2013b; Volkova et al., 2013).

Go et al. (2009) carried out the pioneer work of predicting sentiment in tweets using machine learning technology. They conducted comprehensive experiments on multiple classifiers based on bag-of-words feature. Such feature is widely used because it's simple and surprisingly efficient in many tasks. However, there are also disadvantages of bag-of-words features represented by one-hot vectors. Firstly, it bears a data sparsity issue (Saif et al., 2012a). In tweets, irregularities and 140-character limitation exacerbate the sparseness. Secondly, losing sequence information makes it difficult to figure out the polarity properly (Pang et al., 2002). A typical case is that the sentiment word in a negation phrase tends to express opposite sentiment to that of the context.

Distributed representations of words can ease the sparseness, but there are limitations to the unsupervised-learned ones. Words with special functions in specific tasks are not distinguished. Such as negation words, which play a special role in polarity classification, are represented similarly with other adverbs. Such similarities will limit the compositional models' abilities of describing a sentiment-specific interaction between words. Moreover, word vectors trained by co-occurrence statistics in a small window of context effectively represent the syntactic and semantic similarity. Thus, words like *good* and *bad* have very similar representations (Socher et al., 2011). It's problematic for sentiment classifiers.

Sentiment is expressed by phrases rather than by words (Socher et al., 2013). Seizing such sequence information would help to analyze complex sentiment expressions. One possible method to leverage context is connecting embeddings of words in a window and compose them to a fix-length vector (Collobert et al., 2011). However, window-based methods may have difficulty reaching long-distance words and simply connected vectors do not always represent the interactions of context properly.

Theoretically, a recurrent neural network could process the whole sentence of arbitrary length by encoding the context cyclically. However, the length of reachable context is often limited when using stochastic gradient descent (Bengio et al., 1994; Pascanu et al., 2013). Besides that, a

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1343–1353,
Beijing, China, July 26-31, 2015.

traditional recurrent architecture is not powerful enough to deal with the complex sentiment expressions. Fixed input limits the network's ability of learning task-specific representations and simple additive combination of hidden activations and input activations has difficulty capturing more complex linguistic phenomena.

In this paper, we introduce the Long Short-Term Memory (LSTM) recurrent neural network for twitter sentiment classification by means of simulating the interactions of words during the compositional process. Multiplicative operations between word embeddings through gate structures provide more flexibility and lead to better compositional results compare to the additive ones in simple recurrent neural network. Experimentally, the proposed architecture outperforms various classifiers and feature engineering approaches, matching the performance of the current best data-driven approach. Vectors of task-distinctive words (such as *not*) are distinguished after tuning and representations of opposite-polarity words are separated. Moreover, predicting result on negation test set shows our model is effective in dealing with negation phrases (a typical case of sentiment expressed by sequence). We study the process of the network handling the negation expressions and show the promising potential of our model simulating complex linguistic phenomena with gates and constant error carousels in the LSTM blocks.

2 Related Work

2.1 Microblogs Sentiment Analysis

There have been a large amount of works on sentiment analysis over tweets. Some research makes use of social network information (Tan et al., 2011; Calais Guerra et al., 2011). These works reveal that social network relations of opinion holders could bring an influential bias to the textual models. While some other works utilize the microblogging features uncommon in the formal literature, such as hashtags, emoticons (Hu et al., 2013a; Liu et al., 2012). Speriosu et al. (2011) propose a unified graph propagation model to leverage textual features (such as emoticons) as well as social information.

Semantic concept or entity based approaches lead another research direction. Saif et al. (2012a; 2012b) make use of sentiment-topic features and entities extracted by a third-party service to ease data sparsity. Aspect-based models are also exploited to improve the tweet-level classifier (Lek and Poo, 2013).

2.2 Representation Learning and Deep Models

Bengio et al. (2003) use distributed representations for words to fight the curse of dimensionality when training a neural probabilistic language model. Such word vectors ease the syntactic and semantic sparsity of bag-of-words representations. Much recent research has explored such representations (Turian et al., 2010; Huang et al., 2012).

Recent works reveal that modifying word vectors during training could capture polarity information for the sentiment words effectively (Socher et al., 2011; Tang et al., 2014). It would be also insightful to analyze the embeddings that changed the most during training. We conduct a comparison between initial and tuned vectors and show how the tuned vectors of task-distinctive function words cooperate with the proposed architecture to capture sequence information.

Distributed word vectors help in various NLP tasks when using in neural models (Collobert et al., 2011; Kalchbrenner et al., 2014). Composing these representations to fix-length vectors that contain phrase or sentence level information also improves performance of sentiment analysis (Yessenalina and Cardie, 2011). Recursive neural networks model contextual interaction in binary trees (Socher et al., 2011; Socher et al., 2013). Words in the complex utterances are considered as leaf nodes and composed in a bottom-up fashion. However, it's difficult to get a binary tree structure from the irregular short comments like tweets. Not requiring structure information or parser, long short-term memory models encode the context in a chain and accommodate complex linguistic phenomena with structure of gates and constant error carousels.

3 Recurrent Neural Networks for Sentiment Analysis

Recurrent Neural Networks (RNN) have gained attention in NLP field since Mikolov et al. (2010) developed a statistical language model based on a simple form known as Elman network (Elman, 1990). Recent works used RNNs to predict words or characters in a sequence (Chrupała, 2014; Zhang and Lapata, 2014). Treating opinion expression extraction as a sequence labelling

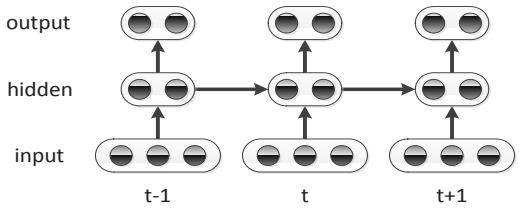

Figure 1: Illustration of simple recurrent neural network. The input of the hidden layer comes from both input layer and the hidden layer activations of previous time step.

problem, Irsoy and Cardie (2014) leverage deep RNN models and achieve new state-of-the-art results for fine-grained extraction task. The lastest work propose a tree-structured LSTM and conduct a comprehensive study on using LSTM in predicting the semantic relatedness of two sentences and sentiment classification (Tai et al., 2015).

Fig.1 shows the illustration of a recurrent network. By using self-connected layers, RNNs allow information cyclically encoded inside the networks. Such structures make it possible to get a fix-length representation of a whole tweet by temporally composing word vectors.

The recurrent architecture we used in this work is shown in Fig.2. Each word is mapped to a vector through a Lookup-Table (LT) layer. The input of the hidden layer comes from both the current lookup-table layer activations and the hidden layer's activations one step back in time. In this way, hidden layer encodes the past and current information. The hidden activations of the last time step could be considered as the representation of the whole sentence and used as input to classification layer. By storing the word vectors in LT layer, the model has reading and tuning access to word representations.

Based on such recurrent architecture, we can capture sequence information in the context and identify polarities of the tweets.

3.1 Elman Network With Fixed Lookup-Table

RNN-FLT: A simple implementation of the recurrent sentiment classifier is an Elman network (also known as simple RNN) with Fixed Lookup-Table (FLT). In such model, unsupervised pre-trained word vectors in LT layer are constant during the whole training process. The hidden layer activa-

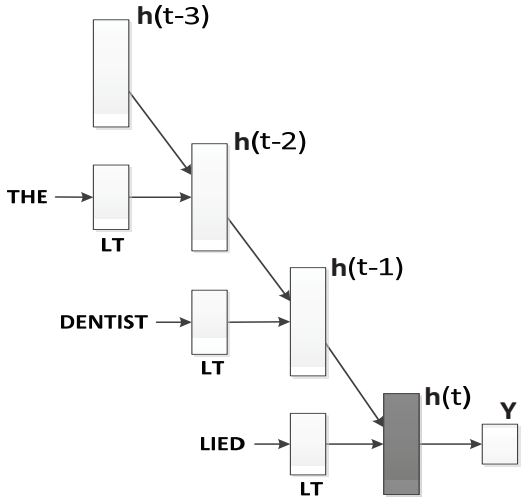

Figure 2: Illustration of the general recurrent architecture unfolded as a deep feedforward network.

tion of position h at time t is:

$$b_h^t = f\left(a_h^t\right) \tag{1}$$

$$a_h^t = \sum_i^E w_{ih} e_i^t + \sum_{h'}^H w_{h'h} b_{h'}^{t-1} \tag{2}$$

where $\mathbf{e}^t$ represents the E-length embedding of the tth word of the sentence, which stored in LT layer. w_{ih} is the weight of connection between input and hidden layer, while $w_{h'h}$ is the weights of recurrent connection (self-connection of hidden layer). f represents the sigmoid function. The binary classification loss function O is computed via cross entropy (CE) criterion and the network is trained by stochastic gradient descent using *back-propagation through time* (BPTT) (Werbos, 1990). Here, we introduce the notation:

$$\delta_i^t = \frac{\partial O}{\partial a_i^t} \tag{3}$$

Firstly, the error propagate from output layer to hidden layer of last time step T. The derivatives with respect to the hidden activation of position i at the last time step T are computed as follow:

$$\delta_i^T = f'\left(a_i^T\right) \frac{\partial O}{\partial y} v_i \tag{4}$$

where v_i represents the weights of hidden-output connection and the activation of the output layer y is used to estimate probability of the tweet bearing

a particular polarity.

$$y = f\left(\sum_{i}^{H} b_i^T v_i\right) \tag{5}$$

Then the gradients of hidden layer of previous time steps can be recursively computed as:

$$\delta_h^t = f'\left(a_h^t\right) \sum_{h'}^{H} \delta_{h'}^{t+1} w_{hh'} \tag{6}$$

3.2 Elman Network with Trainable Lookup-Table

Unsupervised trained word embeddings represent the syntactic and semantic similarity. However, in specific tasks, the importance and functions of different words vary. Negation words have similar unsupervised trained representations with other adverbs, but they make distinctive contributions in sentiment expressions. Besides the function words, tuning word vectors of sentiment words into polarity-representable ones turns out to be an effective way to improve the performance of sentiment classifiers. (Maas et al., 2011; Labutov and Lipson, 2013). Such tuned vectors work together with the deep models, gaining the ability to describe complex linguistic phenomena.

RNN-TLT: To this end, we modify the word vectors in the Trainable Lookup-Table (TLT) via back propagation to get a better embedding of words. The gradient of lookup-table layer is:

$$\delta_i^t = g'\left(a_i^t\right) \sum_{h=1}^{H} \delta_h^t w_{ih} = \sum_{h=1}^{H} \delta_h^t w_{ih} \tag{7}$$

where identity function $g(x) = x$ is considered as the activation function of lookup-table layer.

3.3 Long Short-Term Memory

The simple RNN has the ability to capture context information. However, the length of reachable context is often limited. The gradient tends to vanish or blow up during the back propagation (Bengio et al., 1994; Pascanu et al., 2013). Moreover, Elman network simply combines previous hidden activations with the current inputs through addictive function. Such combination is not powerful enough to describe a complex interactions of words.

An effective solution for these problems is the Long Short-Term Memory (LSTM) architecture (Hochreiter and Schmidhuber, 1997; Gers, 2001). Such architecture consists of a set of recurrently connected subnets, known as memory blocks. Each block contains one or more self-connected memory cells and the input, output and forget gates. Fig.3 gives an illustration of an LSTM block. Once an error signal arrives Constant Error Carousel (CEC), it remains constant, neither growing nor decaying unless the forget gate squashes it. In this way, it solves the vanishing gradient problem and learns more appropriate parameters during training.

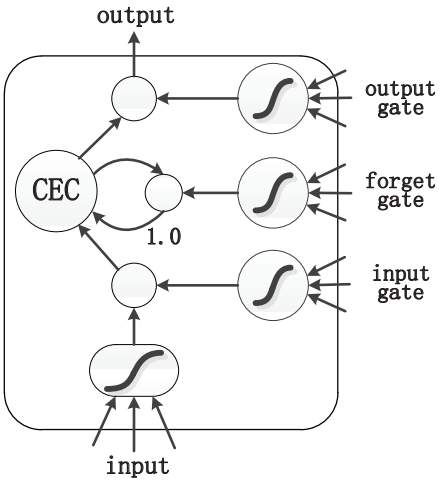

Figure 3: Illustration of LSTM memory block with one cell. Constant Error Carousel (CEC) maintains the internal activation (called *state*) with a recurrent connection of fixed weight 1.0, which may be reset by the forget gate. The input and output gates scale the input and output respectively. All the gates are controlled by the maintained state, network input and hidden activation of previous time step.

Moreover, based on this structure, the input, output and stored information can be partial adjusted by the gates, which enhances the flexibility of the model. The activations of hidden layer rely on the current/previous state, previous hidden activation and current input. These activations interact to make up the final hidden outputs through not only additive but also element-wise multiplicative functions. Such structures are more capable to learn a complex composition of word vectors than simple RNNs.

These gates are controlled by current input, previous hidden activation and cell state in CEC unit:

$$G_I^t = f\left(U_I x^t + V_I h^{t-1} + W_I s^{t-1}\right) \tag{8}$$

$$G_F^t = f\left(U_F x^t + V_F h^{t-1} + W_F s^{t-1}\right) \quad (9)$$

$$G_O^t = f\left(U_O x^t + V_O h^{t-1} + W_O s^t\right) \quad (10)$$

where G^t indicates the gate activation at time t, x^t, h^t and s^t is input, hidden activation and state in CEC unit at time t respectively, while U, V and W represent the corresponding weight matrices connect them to the gates. Subscript I, F and O indicate input, forget and output respectively. The CEC state and block output are computed by the functions with element-wise multiplicative operation:

$$s^t = G_F^t s^{t-1} + G_I^t f\left(U_S x^t + V_S h^{t-1}\right) \quad (11)$$

$$a^t = G_O^t s^t \quad (12)$$

where U_S indicates connection weight between input and state, while V_S represents the weight matrix connecting hidden layer to state.

LSTM-TLT: By replacing the conventional neural units in RNN-TLT with LSTM blocks, we can get the LSTM network with Trainable Lookup-Table. Such model achieves a flexible compositional structure where the activations interact in a multiplicative function. It provides the capacity of describing diverse linguistic phenomenon by learning complex compositions of word embeddings.

4 Experiments

4.1 Data Set

We conduct experiments on the Stanford Twitter Sentiment corpus (STS)[1]. The noisy-labelled dataset is collected using emoticons as queries in Twitter API (Go et al., 2009). 800,000 tweets containing positive emoticons are extracted and labelled as positive, while 800,000 negative tweets are extracted based on negative emoticons. The manually labelled test set consists of 177 negative and 182 positive tweets.

4.2 Experimental Settings

Recurrent Neural Network: We implement the recurrent architecture with trainable lookup-table layer by modifying RNNLIB (Graves, 2010) toolkit.

Early Stopping: From the noisy labelled data, we randomly selected 20,000 negative and 20,000 positive tweets as validation set for early stopping. The rest 1,560,000 tweets are used as training set.

Parameter Setting: Tuned on the validation set, the size of the hidden layer is set to 60.

Word Embeddings: We run *word2vec* on the training set of 1.56M tweets (without labels) to get domain-specific representations and use them as initial input of the model. Limited to the input format of the toolkit, we learned 25-dimensional (relatively small) vectors. *Skip-gram architecture* and *hierarchical softmax algorithm* are chosen during training.

4.3 Comparison with Data Driven Approaches

Classifier	Accuracy(%)
SVM	81.6
MNB	82.7
MAXENT	83.0
MAX-TDNN	78.8
NBoW	80.9
DCNN	**87.4**
RAE	77.6
RNN-FLT	80.2
RNN-TLT	86.4
LSTM-TLT	**87.2**

Table 1: Accuracies of different classifiers.

Naive Bayes, Maximum Entropy and SVM are widely used classifiers. Go et al. (2009) presented the results of three non-neural models using unigram and bigram features.

Dynamic Convolutional Neural Network (DCNN) (Kalchbrenner et al., 2014) is a generalization of MAX-TDNN (Collobert et al., 2011). It has a clear hierarchy and is able to capture long-range semantic relations. While the Neural Bag-of-Words (NBoW) takes the summation of word vectors as the input of a classification layer. Kalchbrenner et al. (2014) reported performances of the above three neural classifiers.

Recursive Autoencoder (RAE) has proven to be an effective model to compose words vectors in sentiment classification tasks (Socher et al., 2011). We run RAE with randomly initialized word embeddings. We do not compare with RNTN (Socher et al., 2013) for lack of phrase-level sentiment labels and accurate parsing results.

Table 1 shows the accuracies of different classifiers. Notably, RNN-TLT and LSTM-TLT out-

[1] http://twittersentiment.appspot.com/

perform the three non-neural classifiers. Trained on the considerable data, these classifiers provide strong baselines. However, bag-of-words representations are not powerful enough. Sparsity and losing sequence information hurt the performance of classifiers. Neural models overcome these problems by using distributed representations and temporally encoding the contextual interaction.

We notice a considerable increase in the performance of the RNN-TLT with respect to the NBoW, whose embeddings are also tuned during supervised training. It suggests that recurrent models could generate better tweet-level representations for the task by composing the word embeddings in a temporal manner and capturing the sequential information of the context.

Convolutional neural networks have outstanding abilities of feature extraction, while LSTM-TLT achieves a comparable performance. It suggests that LSTM model is effective in learning sentence-level representations with a flexible compositional structure.

RAE provides more general representations of phrases by learning to reconstruct the word vectors. Recurrent models outperform RAE indicates that task-specific composing and representation learning with less syntactic information lead to a better result.

Comparing RNN-FLT with RNN-TLT, we can easily figure out that the model with trainable lookup-table achieves better performance. This is due to the fact that tuned embeddings capture the sentiment information of text by distinguishing words with opposite sentiment polarities and providing more flexibility for composing. LSTM-TLT does not outperform RNN-TLT significantly. And the situations are almost the same on short-sentence (less than 25 words) and long-sentence (not less than 25 words) test set. Such results indicate that the ability of LSTM getting access to longer-distance context is not the determinant of improvement, while the capacity of LSTM handling complex expressions plays a more important role. Such capacity will be further discussed in subsection 4.7.

Since the training set is large enough, we have not observed strong overfitting during the training process. Therefore, no regularization technology is employed in the experiments.

4.4 Comparison with Feature Engineering Approaches

Method	Craft feature	Accuracy(%)
Speriosu et al. (2011)	emoticon hashtag	84.7
Saif et al. (2012a)	sentiment-topic semantic	86.3 84.1
Lek and Poo (2013)	aspect-based	**88.3**
This work		**87.2**

Table 2: Comparison with different feature engineering methods.

Table 2 shows the comparison with different feature engineering methods. In Speriosu et al. (2011)'s work, sentiment labels propagated in a graph constructed on the basis of contextual relations (e.g. word presence in a tweet) as well as social relations. Saif et al. (2012a) eased the data sparsity by adding sentiment-topic features that extracted using traditional lexicon. While Lek and Poo (2013) extracted tuple of *[aspect, word, sentiment]* with hand-crafted templates. With the help of opinion lexicon and POS tagger especially designed for twitter data, their approach achieved a state-of-the-art result.

Even though these methods rely on lexicons and extracted entities, our data-driven model outperforms most of them, except the aspect-based one that introduced twitter-specific resources. This is due to the fact that traditional lexicons, even emoticons added, are not able to cover the diversification of twitter sentiment expressions, while LSTM learns appropriate representations of sentiment information through compositional manner.

4.5 Experiments on Manually Labelled Data

Different from STS dataset deciding the polarity based on emoticons, the benchmark dataset in SemEval 2013 (Nakov et al., 2013) is labelled by human annotators. In this work we focus on the binary polarity classification and abandon the neutral tweets. There are 4099/735/1742 available tweets in the training/dev/test set respectively. Since the training set is relatively small, we don't apply fine tuning on word vectors. Namely we use fixed lookup-table for both RNN and LSTM. 300-dimensional vectors are learned on the 1.56M tweets of STS dataset using word2vec. Other settings stay the same as previous experiments.

Method	Accuracy(%)
SVM	74.5
RAE	75.4
RNN-FLT	83.0
LSTM-FLT	**84.0**

Table 3: Accuracies of different methods on SemEval 2013

Table 3 shows our work compared to SVM and Recursive Autoencoder. From the result, we can see that the recurrent models outperforms the baselines by exploiting more context information of word interactions.

4.6 Representation Learning

Recent works reveal that modifying word vectors during training could capture polarity information for the sentiment words effectively (Socher et al., 2011; Tang et al., 2014). However, it would be also helpful to analyse the embeddings that changed the most.

Function words: We choose 1000 most frequent words. For each word, we compute the distance between unsupervised vector and tuned vector. 20 words that change most are shown in Fig.4.

It's noteworthy that there are five negation words (not, no, n't, never and Not) in the notably-change group. The representations of negation words are quite similar with other adverbs in unsupervised learned embeddings, while the proposed model distinguishes them. This indicate that our polarity-supervised models identify negation words as distinctive symbols in sentiment classification task, while unsupervised learned vectors do not contain such information.

Besides the negation words and sentiment words, there are also other prepositions, pronouns and conjunctions change dramatically (e.g. *and* and *but*). Such function words also play a special role in sentiment expressions (Socher et al., 2013) and the model in this paper distinguishes them. However, the contributions of these words to the task are not that explainable as negation words (at least without sentiment strength information).

To further explain how the tuned vectors work together with the network and describe interactions between words, we study the process of the model classifying negation phrases in the following subsection.

Sentiment words: In order to study the embedding change of sentiment words, we choose the most frequent sentiment words in our training data, 20 positive and 20 negative, and observe the dissimilarity of the vectors in a two-dimensional space. An alternative least-square scaling is implemented based on Euclidean distance between word vectors. Figure 5 shows sentiment-specific tuning reduces the overlap of opposite polarities. Polarities of words are identified based on a widely-used sentiment lexicon (Hu and Liu, 2004).

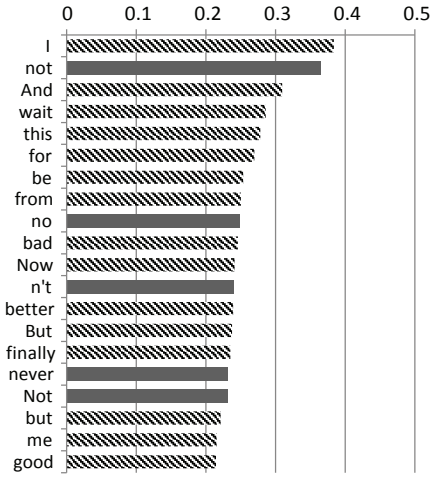

Figure 4: Word change scale to [0,1]. Distances are computed by reversing cosine similarity.

To explicitly evaluate it, we selected embeddings of 2000 most frequent sentiment words (1000 each polarity) and compute the centers of both classes. If an embedding is closer to the opposite polarity center, we consider it as an *overlap*. Experimentally, the proportion of *overlap* of unsupervised learned vectors is 19.55%, while the one of tuned vectors is 11.4%. Namely the overlap ratio is reduced by 41.7%. Experimentally, such polarity separating relies on tuning through lookup-table layer rather than LSTM structure. With the decrease of overlap of polarities, sentiment of word turns more distinguishable, which is helpful for polarity prediction.

4.7 Case Study: Negation

Negation phrases are typical cases where sentiment is expressed by sequence rather than words. To evaluate the ability of the model dealing with such cases, we select most frequent 1000 negative and 1000 positive words in the training data and generate the corresponding negation phrases (such

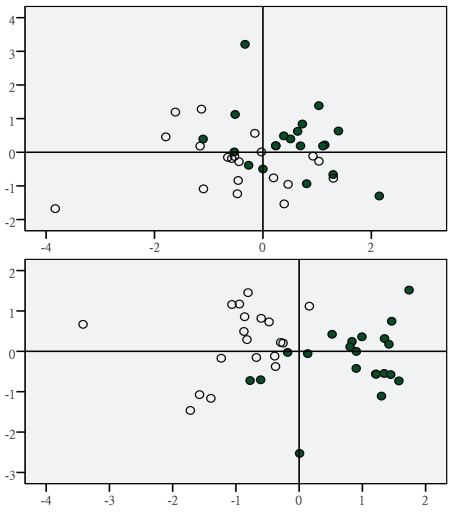

Figure 5: Distance of word vectors shown in two-dimensional space. The above figure shows the distribution of unsupervised learning vectors and the below figure indicates the tuned one. The solid and hollow points represent the positive and negative words respectively.

as *not good*).

Classifier	Accuracy(%)
MNB+unigram+bigram	32.98
RNN-TLT	52.00
LSTM-TLT	**64.85**

Table 4: Accuracy on generated negation phrases test set.

Statistical result shows that only 37.6% of the negation phrases appeared in the training text. It sets a theoretical upper bound to the classifiers based on the unigram and bigram features. Experimental result shown in Table 4 indicates that LSTM model effectively handles the sequential expressions of negation. By composing word vectors, recurrent models ease the sparsity of bag-of-word features and achieve a significant improve than MNB using unigram and bigram features. LSTM outperform RNN by 12.85%, such result suggests the element-wise multiplicative compositional function of LSTM provides more flexibility to simulate interactions between word vectors. A clear process of LSTM handling negation phrases is observed, which is described in the rest of the subsection, while the one of RNN is not that obvious.

As mentioned in 4.6, the task-distinctive function words are distinguished. It would be insightful to show how it works together with the LSTM structure.

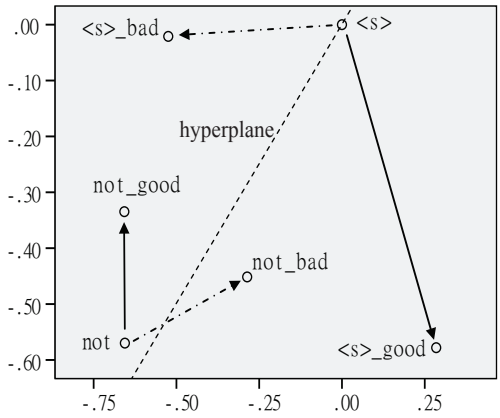

Figure 6: Hidden activations of negation phrases. <s> represent the beginning of sentences. *not bad* and *good* lead to positive outputs, while *not good* and *bad* result in negative values. The dotted line indicates the classification hyperplane. The solid arrows represent the hidden vector changes when the network take the word *good* as input, while the dotted arrows indicate the changes when the word *bad* is input. The sentiment words are input in two situations (as initial input or after negation word), while the changes of hidden vectors of same word are opposite in the two situations.

We train the network on STS dataset and test it on few words and phrases (*good*, *bad*, *not good* and *not bad*). For the convenience of analysis the activation within the network, we set the size of hidden layer to 2. Such setting reduces the performance by about 7% on the public test set, but the trained model still work effectively. Fig.6 shows the activations of LSTM hidden layers. Both sentiment words and negation phrases are classified into correct categories. Furthermore, when sentiment words like *good* (i) input as the first word of sentence and (ii) input after negation word, it cause opposite change in hidden layer. These behaviours simulate the change of sentiment in the negation expressions.

As mentioned in 3.3, gates' activations are controlled by current input, state in CEC unit and output of hidden layer of previous time step. They are many possible ways for the model to simulating the sentiment change. In the experiment, the observed situation is shown in Fig.7:

Negation word contains both polarities. The

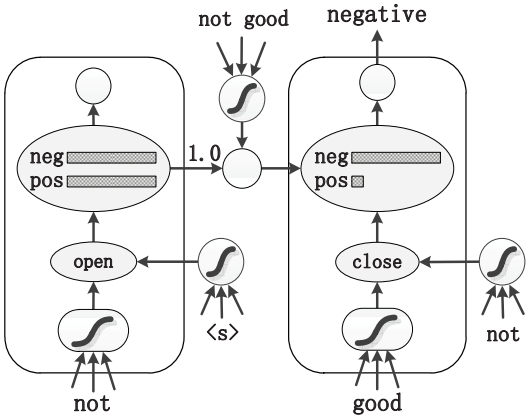

Figure 7: Observed process of LSTM block handling negation phrase *not good*. Some less important connections are omitted in this figure.

positive-axle and negative-axle are almost orthogonal. Negation word has large components on both axles.

***not* make input gate close.** Experiments show recurrent activations make the input gate close, namely previous word *not* squashes the input (both current and recurrent input) to a very small value.

Choose a polarity to forget. The combination of the recurrent input *not* and current input *good* make the CEC unit forget the positive information, namely they make forget gate reduce state's component on positive-axle while leaving a large projection on negative-axle. A significant dissimilarity of forget gate activations between positive and negative words is observed in the experiment, when they are input after *not*.

In this way, the temporally-input phrase *not good* shows a negative polarity. Correspondingly, phrase *not bad* turns positive after reducing the negative components of the negation word. Such case shows the process of the gates and CEC unit cooperating in the LSTM structure. Together with tuned vectors, the architecture has a promising potential of capture sequence information by simulating complex interactions between words.

5 Conclusion

In this paper we have explored to capture twitter sentiment expressed by interactions of words. The contributions of this paper can be summarized as follows: (i) We have described long short-term memory based model to compose word representations through a flexible compositional function. Tested on a public dataset, the proposed architecture achieves result comparable to the current best data-driven model. The experiment on negation test set shows the ability of the model capturing sequential information. (ii) Beyond tuning vectors of sentiment words, we put forward a perspective of distinguishing task-distinctive function words only relying on the label of the whole sequence. (iii) We conduct an interesting case study on the process of task-distinctive word vectors working together with deep model, which is usually considered as a black-box in other neural networks, indicating the promising potential of the architecture simulating complex linguistic phenomena.

Acknowledgments

We thank Deyuan Zhang, Lei Cui, Feng Liu and Ming Liu for their great help. We thank the anonymous reviewers for their insightful feedbacks on this work. This research is supported by National Natural Science Foundation of China (No.613400114), Specialized Research Fund for the Doctoral Program of Higher Education (No.20132302120047), the Special Financial Grant from the China Postdoctoral Science Foundation (No.2014T70340), China Postdoctoral Science Foundation (No.2013M530156)

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Pedro Henrique Calais Guerra, Adriano Veloso, Wagner Meira Jr, and Virgílio Almeida. 2011. From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158. ACM.

Grzegorz Chrupała. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 680–686, Baltimore, Maryland, June. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa.

2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Felix Gers. 2001. *Long Short-Term Memory in Recurrent Neural Networks*. Ph.D. thesis, Ph. D. thesis, Ecole Polytechnique Federale de Lausanne.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12.

Alex Graves. 2010. Rnnlib: A recurrent neural network library for sequence learning problems. `http://sourceforge.net/projects/rnnl`.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. 2013a. Unsupervised sentiment analysis with emotional signals. In *Proceedings of the 22nd international conference on World Wide Web*, pages 607–618. International World Wide Web Conferences Steering Committee.

Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. 2013b. Exploiting social relations for sentiment analysis in microblogging. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 537–546. ACM.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 720–728.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, pages 489–493. Association for Computational Linguistics.

Hsiang Hui Lek and Danny CC Poo. 2013. Aspect-based twitter sentiment classification. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, pages 366–373. IEEE.

Kun-Lin Liu, Wu-Jun Li, and Minyi Guo. 2012. Emoticon smoothed language models for twitter sentiment analysis. In *AAAI*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Preslav Nakov, Zornitsa Kozareva, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval*, volume 13.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318.

Hassan Saif, Yulan He, and Harith Alani. 2012a. Alleviating data sparsity for twitter sentiment analysis. *Making Sense of Microposts (# MSM2012)*.

Hassan Saif, Yulan He, and Harith Alani. 2012b. Semantic sentiment analysis of twitter. In *The Semantic Web–ISWC 2012*, pages 508–524. Springer.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 53–63. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. 2011. User-level sentiment analysis incorporating social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1397–1405. ACM.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring sentiment in social media: Bootstrapping subjectivity clues from multilingual twitter streams. In *Association for Computational Linguistics (ACL)*.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 172–182. Association for Computational Linguistics.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.