ISSN 1870-4069

Hidden Recursive Neural Network for Sentence Classification

Minglei Li[1], Qin Lu[1],
Yunfei Long[1], Lin Gui[2]

[1] The Hong Kong Polytechnic University,
Department of Computing,
Hong Kong

[2] Harbin Institute of Technology,
Laboratory of Network Oriented Intelligent Computation,
China

{csmli, csluqin, csylong}@comp.polyu.edu.hk,
guilin.nlp@gmail.com

Abstract. Recursive Neural Network has been successfully used in sentence-level sentiment analysis for language compositionality based on structured parsing trees. Later on, several modified versions are proposed. These models either treat word vectors as model parameters or employ pre-trained word vectors as input. The former has the advantage of learning task specific word vectors but has much larger parameter size. The later has the advantage of using the encoded semantic information in the vectors and has much smaller parameter size but the general word vectors may be not task-specific. In this work, we propose a hidden recursive neural network (HRNN) which can take the advantages of both learning word vectors and using pre-trained word vectors. This model takes the pre-trained word vectors as the input and adds one hidden layer to extract task-specific representation. Then the recursive composition process is performed in the hidden space. We perform extensive experiments on several sentence classification tasks and results show that our proposed model outperforms both methods and the other baselines, which indicates the effectiveness of our proposed model.

Keywords: Recursive neural network, word vectors, recurrent neural networks.

1 Introduction

Sentence classification requires appropriate feature representations. Traditional methods are mainly based on manually defined features such as bag-of-words, sentiment lexicon, n-grams, etc [17,31]. In recent years, word vector representation, obtained through neural network as a dense and low dimensional vector in an unsupervised way, shows promising result on many tasks of natural language processing [3,14]. The word vectors can encode semantic information.

Inspired by the word vector representation, different methods are proposed to infer dense vector representation of longer text, such as phrases and sentences, and then apply it to task specific sentence-level classification problems such as the sentence-level sentiment analysis. Recursive neural network (RNN) shows promising result on inferring semantic representation of longer text units based on structured parsing trees [26].

This model computes phrase and sentence representation recursively in a bottom-up approach based on syntactic parsing trees. Several modified versions are proposed based on the original RNN, such as the Matrix-vector Recursive Neural Network [25], the Recursive Neural Tensor Network [28], the Adaptive Recursive Neural Network (AdaRNN) [5], the deep recursive neural network (DRNN) [10], the tagging-specific recursive neural network [23], and the Gated Recursive Neural Network (GRNN) [2], etc.

Some models, such as the RNN, MV-RNN, RNTN and AdaRNN, treat the word vectors as model parameters and the word vectors are learned during training the model so that the learned word vectors are more task specific. However, this will make the model parameters increase linearly with the vocabulary size, which will need much larger training data size. That is why the vector dimension is set to about 25 in the original RNN, MV-RNN and RNTN, much smaller compared to the dimension of the commonly used pre-trained word vectors.

In contrast, some models, such as the DRNN and GRNN, use pre-trained word vectors as input, which can reduce the model parameters and this also helps to make full use of the encoded semantic information in the word vectors. However, pre-trained word vectors are designed for general semantic representation, which cannot contain sufficient task specific knowledge compared to learning the word vectors as model parameters.

Intuitively, if the semantic meaning of a word is fully encoded into a dense vector, we should be able to infer the task specific subspace representation, and then perform the recursive composition in this subspace. The hierarchical deep learning model has been shown to be able to extract higher level abstract representation [13]. Based on the above analysis, we propose a new RNN model to take the advantages of both learning the word vectors and using pre-trained word vectors.

This model employs the pre-trained word vectors as input to make use of the semantic information encoded in the vectors and adds one hidden layer beyond the input layer to extract task specific information. Then, the recursive composition is performed in the hidden space. We perform extensive experiments on several sentence classification tasks and the results indicate that our proposed model outperforms both the RNN model that learns the word vectors and the RNN model that simply employs pre-trained word vectors as input. Our model also outperforms other baselines on most of the datasets.

The rest of the paper is organized as follows. Section 2 introduces related work on sentence level classification problem, especially for the sentence representation learning. Section 3 goes into the details of the original RNN model, and based on this, we introduce the details of our model in section 4. Section 5 shows the experiments and result analysis. Section 6 gives the conclusion and future work.

2 Related Work

All classification models at the sentence level are based on sentence representation. Traditional representation methods are mainly based on manual features, such as n-grams, word lexicon, POS tags or manual defined rules [24,17]. These methods treat a word as a symbol and cannot consider the relationship between words, such as antonyms and synonyms. They also fail to consider the word order information. The word vector (also called word embedding) can encode the semantic information of a word into a low-dimensional and dense vector, such as it can encode the following relationships between the corresponding word vectors: *"king"-"queen"="man"-"woman"* or *"China"-"Beijing"="France"-"Paris"* [14]. Such word representation can be used to learn higher level representations such as a phrase or a sentence.

Previous methods to infer phrase level representation from word representation include vector average, addition, and element-wise multiplication [15,16]. Baroni also considers POS categories of words that a noun is used as a vector whereas adjectives, adverbs, and verbs form a matrix to modify the properties of the noun [1]. Recently, neural network based models are proposed. For example, the recurrent neural network is used to infer sentence representations for sentiment classification [11].

The convolutional neural network is employed for sentence classification [12]. This model does not consider the syntactic structure information of a sentence. Based on the structured parsing tree, the recursive neural network (RNN for short) model represents a tree node as a vector and the parent node is computed from the child nodes through a matrix composition function [26]. The Matrix-Vector Recursive Neural Network (MV-RNN) modifies the RNN by representing a node through a vector and a matrix so that it can consider the relation between the child nodes. In Recursive Neural Tensor Network (RNTN), the composition function is a global tensor instead of a matrix [28].

The Adaptive Recursive Neural Network (AdaRNN) employs n composition function during the recursive process and the parent node's representation is the weighted addition of the n composition functions [5]. This model treats the word vectors as parameters without employing the pre-trained word vectors. Distinguishing the phrase types, the Phrase Recursive Neural Network (PRNN) uses two sets of composition functions for the inter-phrases and outer-phrases respectively [19]. This model takes pre-trained word vectors as input without learning the word vectors.

Taking the syntactic information into consideration, Qian [23] proposes the POS tagging-specific recursive neural network that learns different composition functions for different kinds of POS tagging and also learns tagging-specific embedding to be concatenated after the word embedding during the composition process. This model also employs pre-trained word vectors as input. A gated recursive neural network is proposed that employs a full binary tree structure to control the composition process through two kinds of gates [2].

3 The Recursive Neural Network Model

The original RNN is used for sentence-level sentiment analysis task, however, it can be used in any sentence-level classification task.

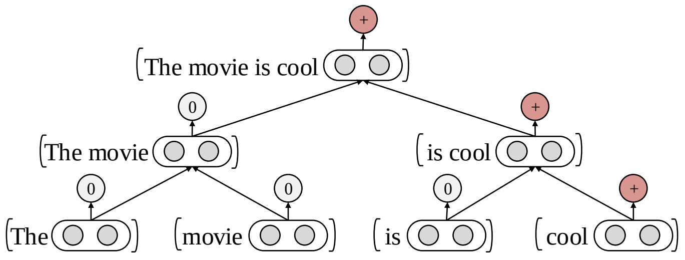


Fig. 1. RNN for sentiment analysis. It is trained based on the constraint of the sentiment label of every node. The word vectors are treated as model parameters and are learned simultaneously during training the model.

For easy discussion, we will take the sentiment analysis as a classification example to explain the model. The structure of RNN [26] for sentiment analysis is shown in Figure 1. Given a structured representation of a sentence, such as the syntactic parsing tree, RNN computes the node's representation in a bottom-up style. To be more specific, given the vector representations of the left child $\boldsymbol{v}_l$ and the right child $\boldsymbol{v}_r$, the representation of its parent $\boldsymbol{v}_p$, is computed using the following formula:

$$\boldsymbol{v}_p = \sigma\left(W\begin{bmatrix}\boldsymbol{v}_l\\ \boldsymbol{v}_r\end{bmatrix} + \boldsymbol{b}\right), \tag{1}$$

where $W \in \mathbb{R}^{d\times 2d}$ and $b \in \mathbb{R}^{d\times 1}$ are the composition parameters shared by all the nodes; $\boldsymbol{v}_l$ and $\boldsymbol{v}_r \in \mathbb{R}^{d\times 1}$; d is the dimension of the nodes' vectors; σ is the activation function. In the original RNN, the vector of the leaf node is treated as model parameters and is also learned during training the model. So the leaf nodes' vectors consist of a parameter matrix $L \in \mathbb{R}^{|V|\times d}$ where $|V|$ is the vocabulary size of the training data. After obtaining the node's vector representation, the node's label is predicted through softmax function:

$$\boldsymbol{y}_i = g\left(W_s\boldsymbol{v}_i + \boldsymbol{b}_s\right), \tag{2}$$

where $\boldsymbol{y}_i \in \mathbb{R}^K$ is the output class vector and K is the class number; g is the softmax function; $W_s \in \mathbb{R}^{K\times d}$ and $\boldsymbol{b}_s \in \mathbb{R}^{K\times 1}$ are the weight matrix and bias, respectively. The original model predicts every internal node's label to constrain the recursive process. The loss function consists of cross entropy and L2 regularization:

$$J(\theta) = -\sum_i^m\sum_j^K t_j log(y_j) + \lambda\left\|\theta\right\|^2, \tag{3}$$

where t_j and y_j are the gold label and predicted label respectively; K is the class number; m is the training sample number; $\theta = \{W, W_s, b, b_s, L\}$ is the set of model parameters; λ is the regularization parameter.

The original RNN model treat the word vectors as model parameters, which can learn more task-specific word vectors.

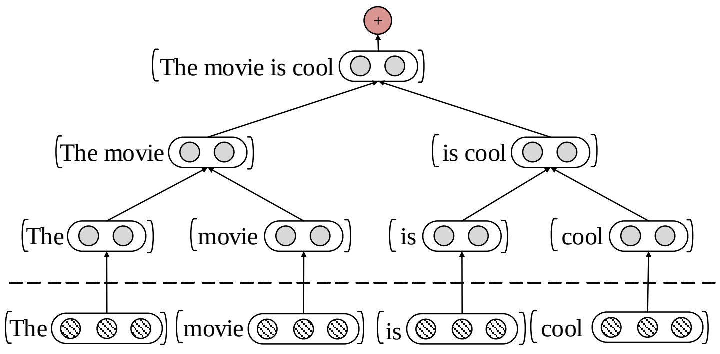


Fig. 2. HRNN for sentiment analysis. Pre-trained word vectors (below the dotted line) serves as the input and one hidden layer is added(above the dotted line) as the representation of the leaf nodes.

However, this will make the parameters space increase linearly with the vocabulary size. The pre-trained word vectors, such as the public available word2vec [14] or Glove [22], have also been used as input vector of the leaf nodes in the latter work [10,29,2]. This can make use of the encoded semantic information in the word vectors but the pre-trained word vectors are designed for the general word meaning, which can not be task-specific.

4 Our Proposed Model

To be able to employ the advantage of learning task-specific representation and the pre-train word vectors, we propose our model as shown in Figure 2. Compared to RNN, we employ the pre-trained word vectors as input and add one hidden layer to extract the task-specific representation, which can not only make use of the semantic information encoded in the word vectors, but also learn the task-specific representation without increasing the model parameters.

For this reason, we refer to our proposed model as the Hidden Recursive Neural Network (HRNN). In the HRNN model, the representation of a leaf node j is calculated by:

$$\boldsymbol{v}_j = \sigma\left(W_I \boldsymbol{v}_j^I + \boldsymbol{b}_I\right), \tag{4}$$

where $\boldsymbol{v}_j^I$ is the pre-trained word vector of word j, W_I and $\boldsymbol{b}_I$ are the model parameters shared between the input layer and the hidden layer to extract task-specific information, and σ is the activation function. The representations of all non-leaf nodes are calculated using Formula 1.

In addition, during the training process, the recursive process of the original RNN is constrained by the sentiment label of the internal nodes because it predicts every node's label. This limits the model's applications because it can only be trained on the corpus of the Stanford Sentiment Treebank (SST) [28], which is the only fully annotated corpus at every phrase.

This makes it difficult to be applied to other sentence classification tasks as annotating every node in the trees is very labor intensive. To extend this model to other tasks, we remove the label constraint of the internal nodes and only predict the label of the root node.

Intuitively, if the pre-trained word vectors have encoded the general semantic meaning and the recursive process can infer the higher phrases' representation, the constraint of task-specific label (such as sentiment label) can make the higher phrases' representation deviated from the original semantic meaning and this may have adverse effect on the final sentence representation because the deviation may accumulate during the recursive process.

If this intuition is true, annotating all the phrases in the parsing tree is not worthy. We will perform experiments to test the effect of this simple modification. The output of the root node is calculated using softmax:

$$\boldsymbol{y} = g(W_s \boldsymbol{v}_{\text{root}} + \boldsymbol{b}_s), \tag{5}$$

where $\boldsymbol{v}_{\text{root}}$ is the root node's vector representation and g is the softmax function. The loss function is the same as Formula 3 and $\theta = \{W_I, W, W_s, b_I, b, b_s\}$.

5 Performance Evaluation

We conduct two experiments to test the effect of our two modifications. The first experiment is designed to investigate the effect of removing the label constraint of the internal nodes. The second experiment is designed to test the effectiveness of our proposed model that employs the pre-trained word vectors and adds one hidden layer.

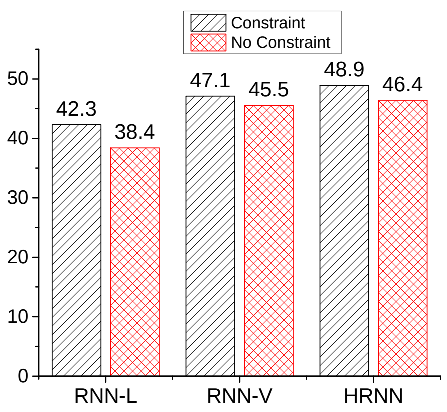
Parameter Setting. For the activation function, we use the rectifier linear activation $f(x) = max\{0, x\}$, which shows better result than Sigmoid function in previous work [10]. For model training, we use back propagation [7] through the stochastic gradient descent algorithm of mini-batch AdaGrad [6].

The original learning rate for the hyper-parameter is set to 0.01 and the regularization parameter λ is set to 0.0001. The mini-batch size is set to 50 and maximum epoch number is set to 100. For the word vectors, we test a number of pre-trained word vectors such as the 300-dimensional Google word vector[1] trained on Google News dataset [14], and Glove with different dimensions trained on different datasets [22].

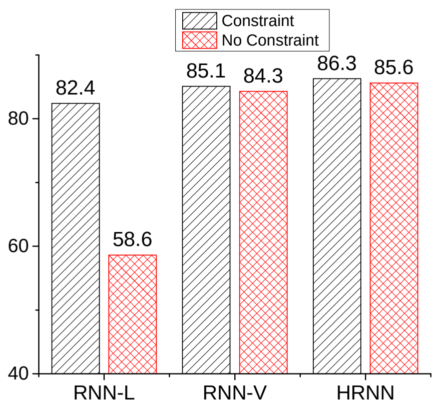
The 300-dimensional Glove840B trained on 840 billion tokens of common crawl data[2] achieves the best result, so we choose this in the following experiments. For unknown words, we randomly initialize them. No fine-tuning is performed on the pre-trained vectors like the work in [12] because we assume that the pre-trained word vectors represent the general word meanings and the hidden layer is responsible for sentiment specific information extraction.

[1]code.google.com/p/word2vec/

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



(a) Result for the fine-grained task.



(b) Result for the binary task.

Fig. 3. The performance (accuracy) of RNN under different settings with or without the label constraint. **RNN-L**: The RNN model that learns the word vectors. **RNN-V**: The RNN model that uses the pre-trained word vectors as input. **HRNN**: Our proposed model. The black box (Constraint) is the result with label constraint and the red box (No Constraint) is the result without label constraint.

5.1 Experiment 1

This experiment aims to test the effect of removing the label constraint of the internal nodes. We test the performance of the RNN-based models with or without the label constraint.

The tested models include the RNN model that treats word vectors as model parameters (denoted as RNN-L, where L means learning the word vectors), the RNN model with the pre-trained word vectors as input (denoted as RNN-V where V means employing pre-trained vectors as input), and the proposed HRNN model that employs pre-trained word vectors as input and adds one hidden layer.

The word vector dimension of RNN-L is set to 25. The hidden dimension of HRNN is set to 100.

Since only the Stanford Sentiment Treebank (SST) [28][3] is annotated at every node, we perform the experiment on this dataset. The SST comes from critic reviews in Rotten Tomatoes and every sentence is parsed by the Stanford Parser[4].

Then, every phrase in the parsing trees is annotated with polarity through the crowdsourcing platform of Amazon Mechanical Turk. There are two tasks for this dataset. The first is binary sentiment classification on positive/negative. The second is more fine-grained classification on five classes: very negative, negative, neutral, positive, and very positive. We use the standard train/dev/test splits of 6920/872/1821 for the binary classification and 8544/4404/2210 for the fine-grained classification. The result is shown in Figure 3. As shown in the figure, for different models, the result with the label constraint all outperforms the result without the label constraint, which

[3]http://nlp.stanford.edu/sentiment/treebank.html

[4]http://nlp.stanford.edu/software/lex-parser.shtml

Table 1. The parameter size of different models. d is the dimension of node's vector. For RNN-L, $d = 25$, for RNN-V, $d = 300$, for HRNN, $d = 100$. $d_v = 300$ is the dimension of pre-trained word vectors. $|V| = 18K$ is the vocabulary size (We use the vocabulary of the fine-grained task as an example).

Model	Model Size	#of parameters
RNN-L	$\|V\| \times d + 2d \times d$	451K
RNN-V	$2d \times d$	180K
HRNN	$d_v \times d + 2d \times d$	50K

indicates that the label constraint of the internal nodes is beneficial for training the RNN model. However, obtaining such training data is labor intensive. Comparing between the RNN-L, RNN-V and HRNN shows that learning the vectors achieves the worst result, this may result from the much larger parameter size.

RNN with pre-trained word vectors (RNN-V) performs better because of the pre-trained word vectors. HRNN further outperforms RNN-V, which indicates the effectiveness of the added hidden layer. We will conduct more experiments to validate this. From this experiment, we can conclude that the label constraint is beneficial for the RNN model and our model performs better than the original RNN model.

The detail of the different models' parameters is shown in Table 1. In this table, the *Model Size* is the model parameter numbers. Since the softmax weights and bias of different models are the same, we just ignore this part and only calculate the composition and word vector part. It shows that the HRNN model has much fewer parameters than the RNN-L and RNN-V.

5.2 Experiment 2

The second experiment aims to test the effectiveness of our proposed model on other sentence level classification problems. The evaluation datasets include:

1. MPQA [32]:Binary classification of short texts with positive and negative[5].
2. Subj [20]: Classifying sentence as subjective or objective[6].
3. MR [21]:Binary classification of movie reviews with positive and negative[7].
4. CR [9]: Binary classification of customer review with positive and negative[8].

The detailed statistic information of these datasets is shown in Table 2. Since all these datasets only have the labels at the sentence level, all the RNN-based models are trained without the label constraint of the internal nodes. To be noted that the best hidden dimension d varies for different datasets and we only report the best result under the best hidden dimension. The best d is 100 for MPQA, 90 for MR, 100 for Subj and 90 for CR. HRNN is compared with the following baselines:

[5]Opinion polarity detection subtask of the MPQA dataset
[6]www.cs.cornell.edu/people/pabo/movie-review-data/
[7]www.cs.cornell.edu/people/pabo/movie-review-data/
[8]www.cs.uic.edu/ liub/FBS/sentiment-analysis.html

Table 2. Statistics of the datasets. K is the class number. l is the sentence average length of the dataset. N is the total sample number. $|V|$ is the vocabulary size. $|V_{pre}|$ is the number of words present in the pre-trained word vectors. $Test$ is the training the test data size (CV means 10-fold cross validation).

Dataset	K	l	N	$\|V\|$	$\|V_pre\|$	$Test$
MPQA	2	3	10606	6225	6205	CV
Subj	2	23	10000	22361	20898	CV
MR	2	20	10662	19994	18632	CV
CR	2	19	3771	5624	5481	CV

1. **RNN-V**: The original RNN model with pre-trained word vectors as input and without hidden layer.
2. **RNN-L**: The original RNN model that treats the word vectors as model parameters.
3. **CNN-static** [12]: Using convolutional neural network that employs pre-trained word vectors as input and the word vectors are fixed without fine-tuning.
4. **CNN-non-static** [12]: Using convolutional neural network that employs pre-trained word vectors as input and the word vectors are fine-tuned during training the model.
5. **CNN-multichannel** [12]: Same architecture as CNN-non-static but with two sets of pre-trained word vectors.
6. **RAE** [27]: Recursive autoencoders with pre-trained word vectors trained from Wikipedia.
7. **CCAE** [8]: Combining the power of recursive, vector-based models with the linguistic intuition of the CCG formalism.
8. **Sent-Parser** [4]: Sentiment analysis specific parser that directly analyzes the sentiment structure of a sentence.
9. **NBSVM** and **MNB** [31]: Naive Bayes SVM and Multinomial Naive Bayes with the variant of uni-bigrams.
10. **G-Dropout** and **F-Dropout** [30]: Gaussian dropout and fast dropout.
11. **Tree-CRF** [18]: A dependency tree based method for sentiment classification of subjective sentences using conditional random fields with hidden variables.
12. **CRF-PR** [33]: Encoding the intuitive lexical and discourse knowledge as expressive constraints and integrating them into the learning of the conditional random filed model via posterior regularization.

The results are shown in Table 3 and as shown in the first three rows, the RNN-V performs much better than the RNN-L on all the five datasets, which again indicates that employing pre-trained word vectors is better than learning the word vectors. The proposed HRNN outperforms the RNN-V and RNN-L models on all the five

Table 3. Performance (accuracy) of different models on different datasets. The top three results in each dataset are marked as bold. The symbol "-" means that no result is reported by the author on the corresponding dataset.

Model	MPQA	MR	Subj	CR
HRNN	**90.1**	**81.2**	**93.8**	**84.9**
RNN-V	88.7	80.4	92.8	83.4
RNN-L	85.4	73.1	90.7	74.6
CNN-static	**89.6**	81.0	93.0	**84.7**
CNN-non-static	**89.5**	**81.5**	93.4	84.3
CNN-multichannel	89.4	**81.1**	93.2	**85.0**
RAE	86.4	77.7	-	-
CCAE	87.2	77.8	-	-
Sent-Parser	86.3	79.5	-	-
NBSVM	86.3	79.4	93.2	81.8
MNB	86.3	79.0	**93.6**	80.0
G-dropout	86.1	79.0	93.4	82.1
F-dropout	86.3	79.1	**93.6**	81.9
Tree-CRF	86.1	77.3	-	81.4
CRF-PR	-	-	-	82.4

datasets, which again indicates the effectiveness of the added hidden layer. Comparing the RNN-based models with the CNN-based models shows that if the RNN model simply employs the pre-trained word vectors as input, its performance is worse than the CNN-based models on all the datasets.

However, after adding one hidden layer, our HRNN model performs slightly better than all the other baselines on the MPQA and Subj datasets and also achieves comparable results with the best performance on the MR and CR datasets.

The advantage of our model is more obvious on the MPQA dataset. This may result from the fact that the average sentence length in MPQA is much shorter than that in other datasets (shown in Table 2) and the syntactic parsing result on MPQA is more accurate than that on other datasets.

In conclusion, as shown in experiment 1 and experiment 2, by simply adding one hidden layer on the pre-trained word vectors, our model can improve the performance of the original RNN model.

6 Conclusion and Future Work

In this paper, we propose the hidden recursive neural network (HRNN) that can leverage the advantages of both using pre-trained word vectors and learning the word vectors, which are two common strategies used in the recursive neural network. Experiments on several sentence classification tasks show that using pre-trained word vectors is better than learning the word vectors and our model performs better than both of them.

We also investigate the effect of the label constraint of the internal nodes during training the RNN model and the experiment shows that the label constraint is better than no label constraint.

Since the performance of the RNN-based model relies on the pre-trained word vectors, we will investigate better ways to learn more expressive word vectors in the future. In addition, we will also investigate the effect of the syntactic parsing trees on the performance of the RNN-based model.

References

1. Baroni, M., Zamparelli, R.: Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 1183–1193 (2010)
2. Chen, X., Qiu, X., Zhu, C., Wu, S., Huang, X. J.: Sentence modeling with gated recursive neural network. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 793–798 (2015) doi: 10.18653/v1/D15-1092
3. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research, vol. 12, pp. 2493–2537 (2011)
4. Dong, L., Wei, F., Liu, S., Zhou, M., Xu, K.: A statistical parsing framework for sentiment classification. Computational Linguistics, vol. 41, no. 2, pp. 293–336 (2015) doi: 10.1162/COLI_a_00221
5. Dong, L., Wei, F., Zhou, M., Xu, K.: Adaptive multi-compositionality for recursive neural network models. IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 24, no. 3, pp. 422–431 (2014) doi: 10.1109/TASLP.2015.2509257
6. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, vol. 12, no. 7, pp. 2121–2159 (2011)
7. Goller, C., Kuchler, A.: Learning task-dependent distributed representations by backpropagation through structure. In: Proceedings of International Conference on Neural Networks (ICNN'96), vol. 1, pp. 347–352 (1996) doi: 10.1109/ICNN.1996.548916
8. Hermann, K. M., Blunsom, P.: The role of syntax in vector space models of compositional semantics. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 894–904 (2013)
9. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 168–177 (2004) doi: 10.1145/1014052.1014073
10. Irsoy, O., Cardie, C.: Deep recursive neural networks for compositionality in language. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, pp. 2096–2104 (2014)
11. Irsoy, O., Cardie, C.: Opinion mining with deep recurrent neural networks. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 720–728 (2014) doi: 10.3115/v1/D14-1080
12. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751 (2014) doi: 10.3115/v1/D14-1181
13. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature, vol. 521, pp. 436–444 (2015) doi: 10.1038/nature14539
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, vol. 2, pp. 3111–3119 (2013)

15. Mitchell, J., Lapata, M.: Vector-based models of semantic composition. In: Proceedings of ACL-08: HLT, pp. 236–244 (2008) https://aclanthology.org/P08-1028.pdf
16. Mitchell, J., Lapata, M.: Composition in distributional models of semantics. Cognitive Science, vol. 34, no. 8, pp. 1388–1429 (2010) doi: 10.1111/j.1551-6709.2010.01106.x
17. Mohammad, S.: Portable features for classifying emotional text. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 587–591 (2012)
18. Nakagawa, T., Inui, K., Kurohashi, S.: Dependency tree-based sentiment classification using CRFs with hidden variables. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 786–794 (2010)
19. Nguyen, T. H., Shirai, K.: PhraseRNN: Phrase recursive neural network for aspect-based sentiment analysis. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (2015) doi: 10.18653/v1/D15-1298
20. Pang, B., Lee, L.: A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (2004) doi: 10.3115/1218955.1218990
21. Pang, B., Lee, L.: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pp. 115–124 (2005) doi: 10.3115/1219840.1219855
22. Pennington, J., Socher, R., Manning, C. D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014) doi: 10.3115/v1/D14-1162
23. Qian, Q., Tian, B., Huang, M., Liu, Y., Zhu, X., Zhu, X.: Learning tag embeddings and tag-specific composition functions in recursive neural network. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1365–1374 (2015) doi: 10.3115/v1/P15-1132
24. Silva, J., Coheur, L., Mendes, A. C., Wichert, A.: From symbolic to sub-symbolic information in question classification. Artificial Intelligence Review, vol. 35, no. 2, pp. 137–154 (2011) doi: 10.1007/s10462-010-9188-4
25. Socher, R., Huval, B., Manning, C. D., Ng, A. Y.: Semantic compositionality through recursive matrix-vector spaces. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1201–1211 (2012)
26. Socher, R., Lin, C. C., Manning, C., Ng, A. Y.: Parsing natural scenes and natural language with recursive neural networks. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, pp. 129–136 (2011)
27. Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., Manning, C. D.: Semi-supervised recursive autoencoders for predicting sentiment distributions. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pp. 151–161 (2011)
28. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642 (2013)
29. Tai, K. S., Socher, R., Manning, C. D.: Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (2015) doi: 10.3115/v1/P15-1150

30. Wang, S., Manning, C.: Fast dropout training. In: Proceedings of the 30th International Conference on Machine Learning, vol. 28, pp. 118–126 (2013)
31. Wang, S., Manning, C. D.: Baselines and bigrams: Simple, good sentiment and topic classification. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pp. 90–94 (2012)
32. Wiebe, J., Wilson, T., Cardie, C.: Annotating expressions of opinions and emotions in language. Language Resources and Evaluation, vol. 39, pp. 165–210 (2005) doi: 10.1007/s10579-005-7880-9
33. Yang, B., Cardie, C.: Context-aware learning for sentence-level sentiment analysis with posterior regularization. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp. 325–335 (2014) http://aclweb.org/anthology/P14-1031