Dependency-tree Based Convolutional Neural Networks for Aspect Term Extraction

Hai Ye[†], Zichao Yan[†], Zhunchen Luo[‡⋆⋆] and Wenhan Chao[†]

[†]School of Computer Science and Engineering, Beihang University, Beijing, China
[‡]China Defense Science and Technology Information Center, Beijing, China
{yehai, yanzichao, chaowenhan}@buaa.edu.cn, zhunchenluo@gmail.com

Abstract. Aspect term extraction is one of the fundamental subtasks in aspect-based sentiment analysis. Previous work has shown that sentences' dependency information is critical and has been widely used for opinion mining. With recent success of deep learning in natural language processing (NLP), recurrent neural network (RNN) has been proposed for aspect term extraction and shows the superiority over feature-rich CRFs based models. However, because RNN is a sequential model, it can not effectively capture tree-based dependency information of sentences thus limiting its practicability. In order to effectively exploit sentences' dependency information and leverage the effectiveness of deep learning, we propose a novel dependency-tree based convolutional stacked neural network (DTBCSNN) for aspect term extraction, in which tree-based convolution is introduced over sentences' dependency parse trees to capture syntactic features. Our model is an end-to-end deep learning based model and it does not need any human-crafted features. Furthermore, our model is flexible to incorporate extra linguistic features to further boost the model performance. To substantiate, results from experiments on SemEval2014 Task4 datasets (reviews on restaurant and laptop domain) show that our model achieves outstanding performance and outperforms the RNN and CRF baselines.

Keywords: Aspect term extraction, Dependency information, Tree-based convolution, Deep learning

1 Introduction

Aspect-based sentiment analysis (or opinion mining) aims to identify the opinions in a given document. To achieve this goal, six subtasks should be considered and aspect term extraction is one of the important subtasks [1]. Aspect terms are attributes (or properties) of the entity that opinion expresses on. For example, given the product review "I love the way the entire suite of software works together", the aspect term is "suite of software".

⋆⋆ Corresponding author.

The task of aspect term extraction is usually regarded as a sequence labeling problem, in which each word in sentence is labeled by conventionally used BIO tagging scheme. In this paper, we also regard aspect term extraction as a sequence labeling problem. Conditional random fields (CRFs) and its variants like semi-CRFs have been successfully applied to this problem. However, these CRFs based models are feature-rich models which need much human-crafted feature engineering effort to work well.

In recent years, deep learning has become the popular and effective method to deal with the tasks in computer vision (CV) and natural language processing (NLP). [2] first applied deep learning to NLP tasks including part-of-speech tagging, chunking, etc. Meanwhile, deep recurrent neural network (RNN[1]) based models have been proposed for aspect term extraction [3]. Unlike the CRFs based models, these RNN based models do not need any manually features. The experimental results on SemEval2014 Task4 datasets show the superiority of RNN based models over traditional CRFs based models.

Previous work has shown that leveraging syntactic features is helpful for opinion mining. Dependency parse tree is one of the important syntactic features and has been widely applied to aspect-based sentiment analysis [4–8]. [4] applied dependency path features to opinion target extraction. Recently, [8] employed an unsupervised method to incorporate dependency context features into embeddings for aspect term extraction. These works manifest that leveraging dependency information of sentences may be helpful and necessary for aspect term extraction.

Although these RNN based models mentioned above can solve the shortcomings that CRFs based models have, they can not make full use of dependency information of sentences that is critical for opinion mining. Because RNN belongs to sequential models, it codes words one by one along the sentence and can only capture the linear context features, thus ignoring tree-based syntactic features over a long path. This drawback may limit its practicability for aspect term extraction.

So, in order to exploit sentences' dependency information and leverage the effectiveness of deep learning for aspect term extraction, we propose a novel dependency-tree based convolutional stacked neural network (DTBCSNN) to extract aspect terms without any human-crafted feature engineering effort. DTBCSNN consists of three main parts: a dependency-tree based convolutional layer (DTBCL), a stacked neural network (SNN[2]) and an inference layer. DTBCL is applied to effectively capture the sentences' dependency information and its core notion is tree-based convolution. **Tree-based convolution** has been explored in a lot of works [9–11]. It can effectively exploit sentences' syntactic features over the parse trees, capturing the relations between words in a long distance. So we adopt tree-based convolution to exploit dependency information of sentences. Specifically, DTBCL first does convolution operation over the fixed-

[1] In this paper, RNN refers to recurrent neural network.

[2] In order to simplify the description of our model, we define the several hidden neural networks being stacked together as SNN.

depth subtrees of a parsed dependency tree. Then the output hidden features from DTBCL are propagated to the SNN to learn tag score distributions for tags. The inference layer is to find tag path with highest scores based on the learned tag score distributions. Though our model is an end-to-end model, it is flexible and can incorporate extra linguistic features to further boost the model performance. We conduct experiments on SemEval2014 Task4 datasets and the experimental results show the superiority of our model over the RNN and CRF baselines.

To sum up, our contributions in this paper can be encapsulated as follows:

• Novel tree-based convolution combined with a neural network is introduced to effectively leverage sentences' dependency information critical but not fully exploited by previous deep learning based models for aspect term extraction.

• Our model is an end-to-end deep learning based model that does not need any manually features. Furthermore, our model is flexible enough to incorporate linguistic features to boost model performance.

• We conduct extensive experiments to evaluate the model sensitivities to architectures, adding linguistic features and word embeddings.

2 Related Work

Among previous work on aspect terms or opinion targets extraction, there are typical methods worth to mention. [12] applied part-of-speech tagging parser to label words and phrases to extract hot (frequent) features for mining customer reviews. Then [13] used association mining method to extract product features. Following up, [14] proposed to use human-defined opinion word seeds and rules from dependency parsing to extract opinion targets iteratively. This kind of problem could also be regarded as a sequence labeling problem and then a classifier is applied. Hidden Markov Models (HMMs) [15] and conditional random fields (CRFs) [4, 16] are usually the chosen ideal models and the wining systems [17, 18] in SemEval2014 Task4 datasets are CRFs based models. Topic model techniques can also be applied to this kind of problem using Latent Dirichlet Allocation (LDA) [19, 20].

The topic of sentiment analysis has been explored by deep learning in recent years and has witnessed state-of-the-art performance in this domain [22, 23]. There are also some work on aspect term or opinion expression extraction using deep learning models. [21] originally combined the deep recurrent neural networks (RNN) and pre-trained word embeddings for opinion expression extraction. Afterwards, motivated by this work, [3] proposed a similar recurrent neural network model and push it further, a set of different types of RNN models were explored. These proposed RNN models outperform traditional feature-rich CRF model. However, recurrent neural network can not effectively capture tree-based syntactic information. As a result, RNN model may not so well fit the aspect term extraction problem.

Tree-based convolution has been studied by [9], which aims to capture sentences' syntactic features to solve certain problems where syntactic information

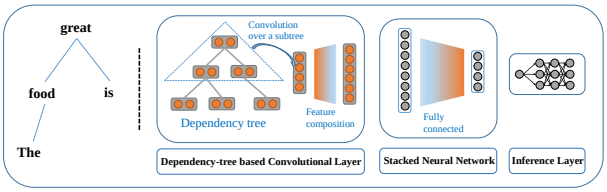

Fig. 1. Left: Example of a dependency parse tree for sentence "The food is great". Right: Illustration of the overall architecture of the Dependency-tree based Convolutional Stacked Neural Network (DTBCSNN). The main components of DTBCSNN are dependency-tree based convolutional layer, stacked neural network and inference layer.

is needed. Tree-based convolution achieved extraordinary performance on sentiment analysis and question classification [9], which manifests the effectiveness of this kind of approach. Furthermore, tree-based convolution is successfully applied to programming language processing [10].

Though inspired by [9], our model is totally different from theirs as follows: 1) We deal with different problems. Ours is aspect term extraction but theirs are sentiment polarity analysis and question classification. 2) We use different model architectures. To fit the problem, we connect the output features of DTBCL over a fixed-size window instead of applying a pooling layer. And we also combine DTBCL with SNN. Similar to [2], we use an inference layer instead of softmax layer [3] for the task of choosing labels, so we apply a stacked neural network after DTBCL to learn tag score distributions for inference layer. We will detail our model in the following sessions.

3 Dependency-tree Based Convolutional Stacked Neural Networks

Overall, our model can be divided into three major components: 1) dependency-tree based convolutional layer; 2) stacked neural network; 3) inference layer. Figure 1 right shows the overall architecture of DTBCSNN. Following, we will give out detail discussions about our model.

3.1 Dependency-tree Based Convolutional Layer for Incorporating Dependency Information

Dependency-tree based convolutional layer (DTBCL) is the core component of our model, which aims to capture sentences' dependency information.

To apply dependency-tree based convolution, a sentence should firstly be converted into a dependency parse tree. Each of its nodes represents the original word of the sentence and is initialized by pre-trained word embeddings. Non-leaf nodes can be seen as parent nodes governing a set of child nodes, which have the relationship with their corresponding child nodes called parent-child relation (PCR). In this paper, we regard the different relationships between parent nodes

and child nodes like *nsubj*, *nmod*, *conj*, etc. as one shared relation PCR. Every word in the sentence can be seen as a parent node connecting with its child nodes. Figure 1 left shows an example of dependency parse tree of sentence "The food is great". We can see from the parse tree that word "great" has child nodes "food" and "is" and, word "food" has word "The" as its child node. Then we use a two-layer fixed-depth feature detector to slide over the sentences' dependency parse trees to capture features which are each corresponding to the words in the sentence.

Suppose that we are given a subtree: a parent node p with its child nodes c_1,c_2,...,c_n, the output feature y of the subtree can be calculated by the function:

$$y = G(W_p \cdot p + \sum_{i=1}^{n} W_c \cdot c_i + b) \tag{1}$$

where W_p, $W_c \in R^{N_f \times N_{embed}}$, W_p is the weight matrix for parent nodes and W_c is the weight matrix for child nodes based on the relation PCR; b is the bias; p, $c_i \in R^{N_{embed}}$, $b \in R^{N_f}$. (N_f is the dimension of output feature y; N_{embed} is the word embedding dimension; n represents the number of child nodes of parent p.) $G(\cdot)$ is an activation function and we use the *ReLU* function in this paper.

We take the parse tree from Figure 1 left for example. The features for each word can be calculated by function (1) as follows:

$$y_{The} = G(W_p \cdot W_{The} + b) \qquad y_{food} = G(W_p \cdot W_{food} + W_c \cdot W_{The} + b)$$
$$y_{is} = G(W_p \cdot W_{is} + b) \qquad y_{great} = G(W_p \cdot W_{great} + W_c \cdot W_{food} + W_c \cdot W_{is} + b)$$

where W_{word} ($word \in \{The, food, is, great\}$) represents pre-trained word embedding. After applying function (1) to every subtree, we can get output features $\{f\} \in R^{N_f}$ one-one corresponding to the words in the given sentence. Considering that the word tagging is influenced by its neighboring words, we further aggregate the features over a fixed-size window to get the compositional features $\{f_1\}$. Specifically,

$$f_{1,i} = \begin{pmatrix} f_{\lfloor i - N_{win/2} \rfloor} \\ \vdots \\ f_{\lfloor i + N_{win/2} \rfloor} \end{pmatrix} \tag{2}$$

where $f_{1,i} \in \{f_1\}$; $\{f_1\} \in R^{N_{f_1}}$, $N_{f_1} = N_{win} \cdot N_f$. ($N_{win}$ is the size of window.) The features with indexes exceeding the boundary of the sentence are padded with zero vectors.

3.2 Stacked Neural Networks for Tag Score Distributions Learning

After getting the features that leverage the sentences' dependency information, we propagate these features to the stacked neural network (SNN). SNN is applied to learn tag score distributions for inference layer.

A SNN with L layers can be seen as a composition function $H_\theta(\cdot)$ with parameters θ:

$$H_\theta(\cdot) = H_\theta^L(H_\theta^{L-1}(...H_\theta^1...)) \tag{3}$$

where H_θ^l is defined for layer l ($1 \leq l \leq L$). In this paper, we apply two hidden layers as our stacked neural networks. The architecture of two hidden layers has been applied in [2] for tag score distributions learning. Motivated by this, we adopt two hidden layers for this paper.

For each hidden feature $f_{1,i} \epsilon \{f_1\}$, we apply SNN to learn the tag score distributions. Concretely, the score distribution $f_{s,i}$ for $f_{1,i}$ can be calculated by function (4):

$$f_{s,i} = W^{(2)} \cdot f_{2,i} + b^{(2)} = W^{(2)} \cdot g(W^{(1)} \cdot f_{1,i} + b^{(1)}) + b^{(2)} \tag{4}$$

where ($W^{(1)} \epsilon R^{N_{f_2} \times N_{f_1}}$, $b^{(1)} \epsilon R^{N_{f_2}}$) and ($W^{(2)} \epsilon R^{N_{f_s} \times N_{f_2}}$,$b^{(2)} \epsilon R^{N_{f_s}}$) are the parameter matrixs and bias for the first hidden layer and second hidden layer respectively. (N_{f_1} and N_{f_2} are the output vector dimensions of the first hidden layer and second hidden layer respectively; N_{f_s} is the size of tag score distribution vector and also is the number of tags.) $g(\cdot)$ is an activation function and we use *Sigmoid* function in this paper. After applying SNN to each hidden feature $f_{1,i} \epsilon \{f_1\}$, we get the tag score distributions $\{f_s\} \epsilon R^{N_{f_s}}$.

3.3 Inference Layer

After SNN, we can get score distributions over tags for every word in a sentence. The inference layer is used to find a specific tag path with a highest score representing the most possible BIO labels for the sentence. More explanation about the physical meanings of inference layer can be found in [2].

Suppose we are given the tag score distributions $\{f_s\} \epsilon R^{N_{f_s}}$ for the sentence with size n and learned transition matrix $A \epsilon R^{N_{f_s} \times (N_{f_s}+1)}$, we can get the tag path $[t_1 : t_n]^*$ by function (5):

$$[t_1 : t_n]^* = argmax_{[t_1:t_n]} \sum_{i=1}^{n} A_{t_{i-1},t_i} + f_{s,t_i} \tag{5}$$

where t_i is the tag for the word in the i_{th} position of the sentence; A_{t_{i-1},t_i} represents the transition possibility from tag t_{i-1} to tag t_i; f_{s,t_i} is the score for tag t_i in i_{th} position. We use the *Viterbi* algorithm to solve function (5).

3.4 Training Method

As discussed above, we get the parameters of our model: $\Theta = \{W_p, W_c, b, W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, A\}$. The object function is:

$$\sum_{\forall d \epsilon S, d \to t} log\ p(t|d,\theta) \tag{6}$$

where S is training set; t is the golden tag path for $d \epsilon S$; $log\ p(t|d,\theta) = log\ exp\{s(d,t,\theta)\} / \sum_{\forall t'} exp\{s(d,t',\theta)\} = s(d,t,\theta) - log \sum_{\forall t'} exp\{s(d,t',\theta)\}$, which is a sentence-level likelihood ($s(d,t',\theta)$ is the score for tag path t')(Because

Table 1. Statistics of SemEval2014 Datasets. #S means "sentence" and #T means "aspect term".

	Restaurant		**Laptop**		**Total**
	Training	Test	Training	Test	
#S	3,041	800	3,045	800	7,686
#T	3,693	1,134	2,358	654	7,839

of space limitation, we can not discuss sentence-level likelihood in detail but more explanation about it can be found in [2]). We use a variation of stochastic gradient descent called AdaGrad and backpropagation algorithm to update parameters Θ.

4 Boosting the Model by Adding Linguistic Features

Though our model DTBCSNN is an end-to-end deep learning based model without any human-crafted features, it can easily incorporate linguistic features to further boost model performance. Our model does not need to change the dimensions of word embeddings or stacked neural network settings, only requiring to append the linguistic feature vectors to the output feature vectors of the first hidden layer and learn extra parameters for linguistic features. The number of linguistic features that may be useful for aspect term extraction such as POS tags, sentiment lexicon is large. In this paper, we choose POS tags and chunk information [3] as the linguistic features.

5 Experiments

5.1 Experimental Settings

Pre-processing We use the common BIO coding method to label our dataset, in which "B" represents "beginning of aspect term", "I" represents "inside of aspect term" and "O" is for "outside of aspect term". The predicted segmentation with "B" at the beginning followed by "O" is regarded as an aspect term.

Datasets In this paper, we adopt the datasets from SemEval2014 Task4[3] (reviews on restaurant and laptop domain) whose specific description is shown in Table 1. For each domain, we only use the training set to train our model and then apply the test set to evaluate the trained model. In all of the experiments, we train and test our model in a unified manner.

Evaluation Metrics Exact evaluation metric is applied in our paper. This means that only the predicted aspect term whose boundary matches the golden boundary can be seen a right one. We use F1 scores to evaluate the model performance.

Model Settings We firstly use pre-trained word embeddings to initialize the word vectors. Considering that the performance of model can benefit from

[3] http://alt.qcri.org/semeval2014/task4/index.php?id=data-and-tools

the word embeddings trained by the same domain corpora, we train word embeddings with gensim[4] on Yelp Challenge dataset[5] for restaurant domain and on Amazon dataset[6] for laptop domain. We select all restaurant reviews from Yelp datasets and only the electronic reviews from Amazon datasets. We tuned word embedding dimension in $\{100, 150, 200, 250, 300, 350, 400\}$ and determined the size of 300 for the two domains. The window size is 3 tuned from $\{1, 3, 5\}$. For laptop domain, N_f is 300 tuned from $\{200, 250, 300, 350, 400\}$ and N_{f_2} is 250 tuned from $\{200, 250, 300, 350\}$; for restaurant domain, N_f is 250 and N_{f_2} is 200 tuned the same way as laptop domain. During the process, Stanford Parser[24] is used to get dependency parse trees for sentences.

We combine mini-batch AdaGrad and early stopping to train our model. The mini-batch size is 25 tuned from $\{10, 15, 20, 25, 30\}$. In order to use the method of early stopping, 10% data from the training set are randomly selected as validation set and the rest of them as training set. When each epoch training is finished, we use validation set to evaluate the model, to keep whichever parameters that entail the best performance on the validation set. By the way the early stopping steps are set to 10 tuned from $\{5, 10, 15, 20\}$, meaning that if the performance fails to exceed the best result over 10 times, the training will be stopped. After going through the steps mentioned above, it is time to use test set to evaluate the trained model on the best parameters.

Adding Linguistic Features We use POS tags and chunk information as the extra linguistic features. POS tags contains four types including *noun*, *adjective*, *verb* and *adverb*, while chunk contains five classes: *NP*, *VP*, *PP*, *ADJP* and *ADVP*. These are all coded as binary features.

5.2 Baseline Methods

We compare our model with the following baselines:

• **CRF** A linear-chain CRF with commonly used linguistic features including current word, context information, POS tag, positions, stylistics and prefixes and suffixes between one to four characters [3].

• **Elman-RNN** An Elman type recurrent neural network on the top of word embeddings proposed by [3]. Elman-RNN contains a lookup-table layer, a hidden layer and an output layer.

• **Elman-RNN+F** The above Elman-RNN adding the same linguistic features as ours.

• **LSTM-RNN** An LSTM network which is the another type of recurrent neural network proposed by [3]. It shares a same architecture with Elman-RNN.

• **LSTM-RNN+F** The above LSTM network adding the same linguistic features as ours.

• **HIS_RD** The top system [18] on the laptop domain for SemEval2014 Task4. It is a CRF based model leveraging lexical features, syntactic information, etc.

[4] https://pypi.python.org/pypi/gensim
[5] http://www.yelp.com/dataset challenge
[6] http://jmcauley.ucsd.edu/data/amazon/links.html

Table 2. Model comparison results in terms of F1 scores (%).

Models	Restaurant	Laptop
CRF	77.28	68.66
Elman-RNN	80.37	74.43
Elman-RNN+F	81.66	74.25
LSTM-RNN	79.79	73.52
LSTM-RNN+F	81.37	75.00
HIS_RD	79.62	74.55
DLIREC	**84.01**	73.78
DTBCSNN	82.26	74.70
DTBCSNN+F	83.97	**75.66**

• **DLIREC** The winning system [17] on the restaurant domain for SemEval2014 Task4. It is another CRF based model and it considers various lexical, syntactic, semantic features of the sentences.

We mainly compare our model with the RNN based models proposed by [3]. In their paper, they have proposed numerous types of RNN based models including Jordan-RNN, Elman-RNN, LSTM, Bi-Elman-RNN, Bi-LSTM, etc. Their experimental results show that Elman-RNN and LSTM achieve higher performance over other models, so we adopt Elman-RNN and LSTM as the baselines ignoring the other inferior models.

5.3 Final Results and Analysis

Table 2 shows the results of the mentioned baselines and our model. We use the same linguistic features which are used in [3]. In [3], they used different dimensional word embeddings for a specific model to evaluate performance, so we report the best result for a specific model from their paper. The followings are the analysis and conclusions:

• Comparing to the traditional linear CRF model, the deep learning based models (DTBCSNN and RNNs) apparently achieve much better performance especially in the laptop domain. This demonstrates the effectiveness of deep learning and its superiority over linear CRF.

• Our model is more outstanding than RNN based models. Quantitatively, from results on the Table 2, DTBCSNN can achieve a result of 82.26% in restaurant domain and outperforms all of the RNNs based models even including those adding extra linguistic features. In laptop domain, DTBCSNN can achieve 74.70% which is better than HIS_RD and almost every RNNs based models except LSTM-RNN+F. These results indicate the limits of the RNNs based models and the effectiveness of our model by leveraging syntactic features for aspect term extraction.

• Adding linguistic features is ascertained to boost our model, as DTBCSNN+F achieves 83.97% exceeding DTBCSNN by 1.71% in the restaurant domain and exceeds 0.96% than DTBCSNN in the laptop domain. Comparing

Table 3. Effect of DTBCL in terms of F1 scores (%).

Models	Restaurant	Laptop
SNN	80.42	70.59
DTBCSNN	82.26	74.70

Table 4. Effect of adding linguistic features in terms of F1 scores (%).

Models	Restaurant	Laptop
DTBCSNN	82.26	74.70
DTBCSNN+POS	82.60	75.43
DTBCSNN+chunk	82.53	75.08
DTBCSNN+F	83.97	75.66

with the wining systems HIS_RD and DLIREC, DTBCSNN+F can achieve a performance close to DLIREC which is slightly lower by 0.04% in the restaurant domain but DTBCSNN and DTBCSNN+F all outperform HIS_RD in the laptop domain.

5.4 Development Experiments for Model Analysis

In this part, we do extensive experiments to evaluate the model architecture, with added linguistic features and word embeddings.

Effect of DTBCL DTBCL refers to the first part of our model that is dependency-tree based convolutional layer. We do experiment excluding DTBCL to see how much influence it can engender to the model. We use DTBCSNN as the baseline. After dropping DTBCL, the rest of the model is a stacked neural network (SNN). The result is in the Table 3, from which, we can see that the performance is damaged after dropping DTBCL on both dataset domain. The score on restaurant domain is reduced by 1.84% and 4.11% on laptop domain, which demonstrates the importance and effectiveness of tree-based convolution to capture syntactic features for aspect term extraction in our model.

Effect of Adding Linguistic Features We divide the two linguistic features mentioned above into isolated ones and add each feature to DTBCSNN to see the performances. The results are in Table 4, from which we can find out that the performance of DTBCSNN is improved after adding one certain linguistic feature on both domains and adding all the linguistic features is better than adding only one single feature. Based on the results, we can safely conclude that 1) our model can easily incorporate linguistic features to improve the model performance; 2) we can explore more linguistic features and then combine them to further boost our model.

Effect of Word Embedding Dimensions In an attempt to evaluate the model sensitivity to the dimension of word embeddings, we vary the dimension from 50 to 400 with interval as 50. Restaurant domain uses word embeddings trained on Yelp dataset and laptop domain applies word embeddings trained on Amazon dataset. After conducting the experiment on DTBCSNN model, results are shown in Figure 2, which tells us that the highest score is achieved at around dimension 300 and that after dimension 150, the performance does not vary so much. The results verify that DTBCSNN is not so sensitive to the word embedding dimensions on the condition that the dimension is in the appropriate range e.g. from 150 to 400.

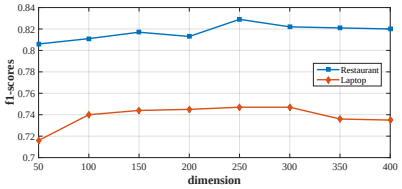

Fig. 2. Effect of word embedding dimensions.

Table 5. Effect of word embedding types in terms of F1 scores (%).

Type	Restaurant	Type	Laptop
Yelp	82.26	Amazon	74.70
Google	81.40	Google	72.63

Effect of Word Embedding Types In order to see the effect of word embedding types to the model, we use word embeddings trained on Yelp datasets (YWE), word embeddings trained on Amazon datasets (AWE) and Google word embeddings[7] (GWE) to do the experiment with DTBCSNN. Specifically, we test the model with YWE and GWE on restaurant domain and with AWE and GWE on laptop domain. The results are shown in Table 5. From the results, we can see that the performance of the model declines when using GWE as the word embeddings. On restaurant domain, the performance reduces by 0.86% compared to YWE and on laptop domain it drops by 2.07% compared to AWE, which tells us that using word embeddings trained by the same domain corpora is necessary for the problem.

6 Conclusion

In this paper, we propose a novel dependency-tree based convolutional stacked neural network, aiming to leverage dependency information of sentences and the effectiveness of deep learning for aspect term extraction. We apply tree-based convolution to capture dependency information for this problem. Our model does not need any manually features and is flexible to incorporate extra linguistic features to further boost model performance. The results of experiments on datasets show the superiority of our model to baselines.

7 Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 61602490) and the National High-tech Research and Development Program (863 Program) (No. 2014AA015105). Thanks for the anonymous reviewers for their valuable comments.

[7] https://code.google.com/archive/p/word2vec/

References

1. Liu, B., Hu, M., Cheng, J.: Opinion observer: analyzing and comparing opinions on the web. In: WWW, 342-351 (2005)
2. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P. P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research 12:2493-2537 (2011)
3. Liu, P., Joty, S. R., Meng, H. M.: Fine-grained opinion mining with recurrent neural networks and word embeddings. In: EMNLP, 1433-1443 (2015)
4. Jakob, N., Gurevych, I.: Extracting opinion targets in a single and cross-domain setting with conditional random fields. In: EMNLP, 1035-1045 (2010)
5. Johansson, R., Moschitti, A.: Syntactic and semantic structure for opinion expression detection. In: CoNLL, 67-76 (2010)
6. Johansson, R., Moschitti, A.: Extracting opinion expressions and their polarities-exploration of pipelines and joint models. In: ACL, 101-106 (2011)
7. Li, F., Han, C., Huang, M., Zhu, X., Xia, Y., Zhang, S., Yu, H.: Structure-aware review mining and summarization. In: COLING, 653-661 (2010)
8. Yin, Y.,Wei, F., Dong, L., Xu, K., Zhang, M., Zhou, M.: Unsupervised word and dependency path embeddings for aspect term extraction. In: IJCAI, 2979-2985 (2016)
9. Mou, L., Peng, H., Li, G., Xu, Y., Zhang, L., Jin, Z.: Discriminative neural sentence modeling by treebased convolution. In: EMNLP, 2315-2325 (2015)
10. Mou, L., Li, G., Zhang, L., Wang, T., Jin, Z.: Convolutional neural networks over tree structures for programming language processing. In: AAAI, 1287-1293 (2016)
11. Ma, M., Huang, L., Zhou, B., Xiang, B.: Dependency-based convolutional neural networks for sentence embedding. In: ACL, 174-179 (2015)
12. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: KDD, 168-177 (2004)
13. Liu, B., Hu, M., Cheng, J.: Opinion observer: analyzing and comparing opinions on the web. In: WWW, 342-351 (2005)
14. Qiu, G., Liu, B., Bu, J., Chen, C.: Opinion word expansion and target extraction through double propagation. Computational Linguistics 37(1):9-27 (2011)
15. Jin,W., Ho, H. H.: A novel lexicalized hmm-based learning framework for web opinion mining. In: ICML, 465-472. ACM (2009)
16. Yang, B., Cardie, C.: Extracting opinion expressions with semi-markov conditional random fields. In: EMNLP, 1335-1345 (2012)
17. Toh, Z., Wang, W.: Dlirec: Aspect term extraction and term polarity classification system. In: SemEval, 235-240 (2014)
18. Chernyshevich, M.: Ihs r&d belarus: Cross-domain extraction of product features using crf. In: SemEval, 309-313 (2014)
19. Titov, I., McDonald, R. T.: Modeling online reviews with multi-grain topic models. In: WWW, 111-120 (2008)
20. Moghaddam, S., Ester, M.: On the design of LDA models for aspect-based opinion mining. In: CIKM, 803-812 (2012)
21. Irsoy, O., Cardie, C.: Opinion mining with deep recurrent neural networks. In: EMNLP, 720-728 (2014)
22. Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: EMNLP, 1631-1642 (2013)
23. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP, 1746-1751 (2014)
24. Klein, D., Manning, C. D.: Accurate unlexicalized parsing. In: ACL, 423-430 (2003)