Revising Attention with Position for Aspect-Level Sentiment Classification

Dong Wang[1,2], Tingwen Liu[1,2](✉), and Bin Wang[3]

[1] Institute of Information Engineering Chinese Academy of Sciences, Beijing, China
{wangdong,liutingwen}@iie.ac.cn

[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

[3] Xiaomi AI Lab, Beijing, China
wangbin11@xiaomi.com

Abstract. As a fine-grained classification task, aspect-level sentiment classification aims at determining the sentiment polarity given a particular target in a sentence. The key point of this task is to distinguish target-related words and target-unrelated words. To this end, attention mechanism is introduced into this task, which assigns high attention weights to target-related words and ignores target-unrelated words according to the semantic relationships between context words and target. However, existing work not explicitly take into account the position information of context words when calculating the attention weights. Actually, position information is very important for detecting the relevance of the word to target, where words that are closer to the target usually make a greater contribution for determining the sentiment polarity. In this work, we propose a novel approach to combine position information and attention mechanism. We get the position distribution according to the distances between context words and target, then leverage the position distribution to modify the attention weight distribution. In addition, considering that sentiment polarity is usually represented by a phrase, we use CNN for sentiment classification which can capture local n-gram features. We test our model on two public benchmark datasets from SemEval 2014, and the experimental results demonstrate the effectiveness of our approach.

Keywords: Aspect-level sentiment classification · Attention mechanism · Position information · CNN

1 Introduction

Recently, with the rise of the Internet, especially the e-commerce, aspect-level sentiment classification has attracted more and more attention. Unlike sentence-level sentiment classification [1], aspect-level sentiment classification [2] aiming at determine the sentiment polarity towards a particular target. For example, given a sentence "*The quality is good but the price is too high.*", the sentiment polarities for "*quality*" and "*price*" will be positive and negative respectively.

Early works focus on using traditional machine learning approaches, such as SVM [3]. However, this type of methods require manually constructed features that can reflect the relationship between context words and target, which is time consuming and labor intensive.

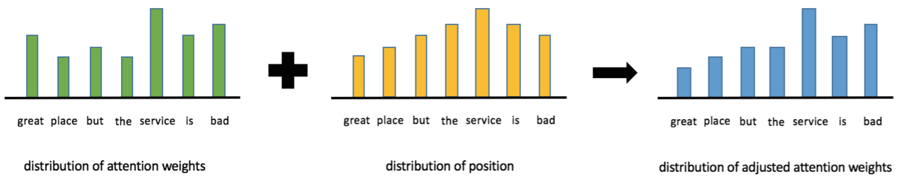

Fig. 1. The illustration of how position distribution adjusts attention weight distribution. The green, yellow and blue represent original attention weight distribution, position distribution, and adjusted attention weight distribution respectively. The target is "*service*". (Color figure online)

In recent years, neural networks have been widely used in natural language processing and show great power. Benefit from ability to model metanic relation, LSTM is particularly popular in aspect-level sentiment classification. Different from traditional machine learning methods, neural networks do not need manually constructed features.

The key point of aspect-level sentiment classification is to distinguish target-related words and target-unrelated words. For that, attention mechanism is introduced to assign different weights to words according to their semantic relationship with target. Existing work usually combine attention mechanism and LSTM, where LSTM encodes the sentence and the attention module assigns different weight to each word, then leverages the weighted representation of sentence to determine the sentiment polarity towards target, such as AE-LSTM [4], ATAE-LSTM [4], IAN [5] and EAM [6].

Intuitively, the closer a word is to target, the more important it is, and the higher its weight should be. For example, in sentence "*Great atmosphere, but the worst food.*", the sentiment polarity towards target "*food*" is negative and compared to words that are far away from target, the closer word "*worst*" play a more important role when determining the sentiment polarity, which is also in line with human cognition. However, the existing work either neglect the position information or don't explicitly model above phenomena. For example, PBAN [7] just appends position embedding after word embedding as input for model, EAM [6] just leverages the position information to select some keywords for attention module.

To this end, we propose a novel approach to explicitly model above phenomena. On one hand, we also use the standard attention mechanism to calculate the weight of each word and get the attention weight distribution. On the other hand, we use the distances between context words and target to get the position distribution. Then, we calculate the difference between attention weight distribution and position distribution, and add it to the classification loss as a penalty.

During the training process, model will adjust the original attention weight distribution to reduce the difference between it and the position distribution, and finally reaches a balance between classification error and distribution difference. The Fig. 1 shows the above process. Actually, we explore multiple ways to calculate the position distribution and the difference between position distribution and attention weight distribution, which will be detailed in the model section.

Considering that sentiment polarity is usually represented by a phrase, we used CNN to replace LSTM to capture local n-gram feature. Similar to textCNN [1], we also apply the CNN on word embeddings. The difference is that we will first adjust the word embeddings with the adjusted attention weights to eliminate the information of target-unrelated words, then use CNN to get the final sentiment polarity.

The main contributions of our work include: (1) We propose a novel approach to explicitly use position information: leverage the position distribution to adjust the attention weight distribution. (2) We explore a variety of ways to utilize position information and introduce CNN to replace LSTM for capturing local n-gram feature more effectively. (3) Our approach achieves comparable performance on two public benchmark datasets *Restaurants* and *Laptops* from SemEval 2014 [8].

2 Related Work

Early works focus on leveraging classification algorithms on manually built features that reflect some relationship between context words and target, such as SVM [3] and MaxEnt-LDA [9]. However, manually built features are based on heuristic rules or external resources, such as dependency tree and sentiment lexicon which focus on structural information of sentence but do not contain the deep semantic information.

Recently, neural networks have attracted more and more attention and benefit from ability to capture long distance dependencies of sentence, LSTM is widely used in aspect-level sentiment classification. TD-LSTM [10] leverage two LSTMs to code the left context and right context with respect to target for classification.

Later, attention mechanism was introduced to assign different weights to target-related words and target-unrelated words and the weighted hidden states of LSTM will be used for sentiment classification. Existing works focus on how to design effective attention mechanism. AE-LSTM and ATAE-LSTM [4] is the earlier work which just simply calculate the attention weights with standard attention mechanism. IAN [5] learn target-aware context representation and context-aware target representation with attention mechanism before final classification.

SA-LSTM-P [11] leverages CRF to make the calculation of attention no longer independent of each other. BILSTM-ATT-G [12] divides the sentence into two parts with respect to the target and determine how much information from each part should be preserved with attention mechanism. However, all above methods ignore the position information. EAM [6] and PBAN [7] take into account the position information where EAM leverages the position information to select keywords for calculating attention ignoring other words and PBAN appends position embedding after word embedding as the input of model. In addition, in relation extraction task, PaNSM [13] takes position embedding into the calculation formula of semantic attention. But, both of three approaches above do not explicitly model how position information guide the generation of attention weights.

Also, memory network, first proposed by Facebook in 2014 [14], is introduced to aspect-level sentiment classification, such as MemNet [15], which use the target-specific representation repeatedly to retrieve from memory, and then updates the target-specific representation with the retrieved information. Finally, the target-specific representation will be used for the sentiment classification. The memory of MemNet consists of word embeddings of context words, which will be adjusted according to the distances between context words and target.

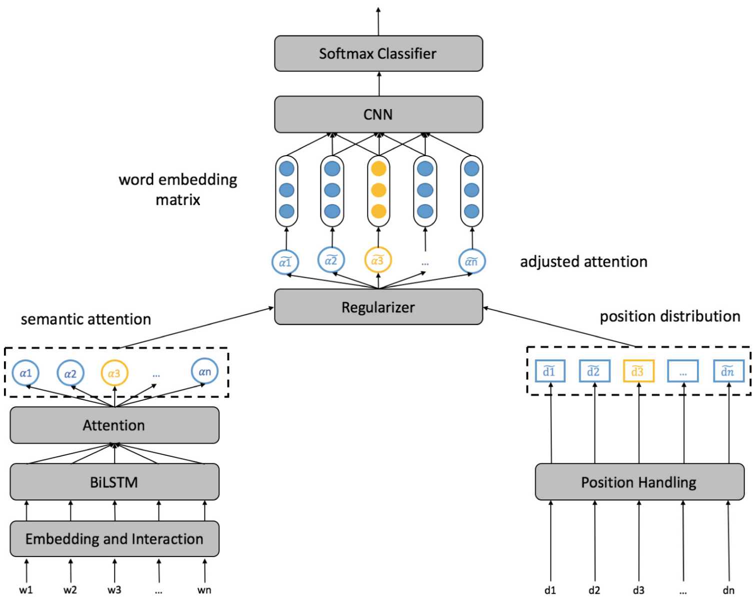

Fig. 2. An overview of our model. The lower left part is the module for calculating of attention, and the lower right part is the module for calculating of distance. The upper part is the classification module.

3 Our Method

In this section, we will introduce our model and give details about the internal structure and function of each module. Figure 2 is the overview of our model.

3.1 Problem Definition

Given a sentence s of length n, and a target t that appears in sentence s, aspect-level sentiment classification aims to assign sentiment polarity to target t.

We use w_i to indicate the i-th word, use x_i, x_t to indicate the embedding of i-th word and target respectively, h_i to indicate the hidden state of i-th word, h_t to indicate the hidden state of target, d_i to indicate the distance between w_i and target.

3.2 Embedding and Interaction

In this layer, we map each word to sparse vector representation for neural network. On one hand, we obtain the word embedding of each word by looking up the embedding table. On the other hand, we also use element-wise multiplication to get interaction between context word and target word [16] to capture richer information, then append it to word embedding vector:

$$x_i^c = [x_i; x_i \odot x_t], \quad i \in [1, n]. \tag{1}$$

The new representation x_i^c will be the input of BiLSTM.

3.3 BiLSTM

In this layer, we use BiLSTM to code the sentence which read the s from w_1 to w_n and w_n to w_1 at the same time so that the representation of each word can include both the past and future information:

$$\overrightarrow{h}_i = \overrightarrow{\text{LSTM}}(x_i^c), \quad i \in [1, n], \tag{2}$$

$$\overleftarrow{h}_i = \overleftarrow{\text{LSTM}}(x_i^c), \quad i \in [1, n]. \tag{3}$$

Then we concatenate the forward hidden state and backward hidden state at each time step to get the representation of every word:

$$h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]. \tag{4}$$

3.4 Attention

At this layer, we use attention mechanism to calculate the weights of context words towards target. We use hidden states from BiLSTM to represent each word and adopt the standard attention mechanism [17] to calculate the weight of each word:

$$att_i = \sigma(v^T \tanh(W_a[h_i; h_t])), \tag{5}$$

where W, v are trainable parameters, σ is sigmoid activate function which scales a scalar to 0-1, att is the attention weight vector.

3.5 CNN

In this layer, we use CNN to extract local n-gram features of sentence for sentiment classification. Firstly, we multiply word embeddings with attention weights obtained in the last layer, to erase the information of noise words:

$$\tilde{x}_i = x_i * att_i, \tag{6}$$

where x_i indicates the original word embedding.

After getting the weighted word embedding matrix, we use multi-sizes filters to get multiple features:

$$c_i = \text{ReLU}(W_c * \tilde{x}_{i:i+s} + b_c). \tag{7}$$

We use multiple kernels to capture as much as possible features and apply max pooling to capture the most crucial features:

$$z = [\max(c_1), \max(c_2), ..., \max(c_n)], \tag{8}$$

where z is the final representation of the sentence s given target t.

3.6 Softmax Classifier

In order to get the final sentiment polarity, we need to feed the sentence representation to a multi-classes classifier:

$$o = W_f * z + b_f. \tag{9}$$

The probability that a sentence's sentiment belongs to k-th class is calculated as follows, where C is the total number of classes:

$$p_k = \frac{exp(o_k)}{\sum_{i=1}^{C} exp(o_i)}. \tag{10}$$

The label with highest probability will be the predicated sentiment polarity of sentence given target t.

3.7 Position Handling and Regularizer

Although attention mechanism can distinguish the importance of words to a certain extent, it is semantic based which neglects the position information of words. In this layer, we will take into account position information and leverage the position distribution of context words to adjust the above attention weight distribution.

To this end, we explore two ways to compute the position distribution and three ways to measure the difference between the attention weight distribution and position distribution.

Calculating Position Distribution. In this part, we will introduce two ways to get position distributions of context words. Firstly, we calculate the distance d_i between i-th word and target t:

$$d_i = \begin{cases} \mid i - t_s \mid, & i < t_s, \\ 0, & t_s \leq i \leq t_e, \\ \mid i - t_e \mid, & i > t_e, \end{cases} \tag{11}$$

where t_s indicates the start index of target and t_e indicates the end index of target.

Normalized Position Distribution (N). We map d_i to between 0 and 1, in the same interval as attention weight:

$$\tilde{d_i} = 1 - \frac{d_i}{n}, \tag{12}$$

where the closer word is to target, the larger $\tilde{d_i}$.

Gaussian Position Distribution (G). We convert the original distance sequence into a gaussian distribution [18], where $\tilde{d_i}$ is also between 0 and 1:

$$\tilde{d_i} = exp(-\frac{d_i^2}{2\sigma^2}), \tag{13}$$

where σ is the standard deviation which is set as $\sigma = \frac{D}{2}$ and D is the window size.

Calculating Difference Between Position Distribution and Attention Weight Distribution. In this part, we will introduce three ways to calculate the difference between position distribution $\tilde{d}$ and attention weight distribution att.

Absolute Difference (AD). We use element-wise subtraction to get the absolute difference sequence between position distribution and attention weight distribution. Then we calculate the average of the sequence as the final absolute difference:

$$\text{AD}(\tilde{d}, att) = \frac{\sum_{i=1}^{n} |\tilde{d_i} - att_i|}{n}. \tag{14}$$

Kullback-Leibler Divergence (KL). Firstly, we use softmax function to convert $\tilde{d}$ and att into probability distribution with sum of 1, and then calculate the Kullback-Leibler divergence between them:

$$att^p = \text{softmax}(att), \tag{15}$$

$$\tilde{d}^p = \text{softmax}(\tilde{d}), \tag{16}$$

$$\text{KL}(att^p \mid\mid \tilde{d}^p) = \sum_{i=1}^{n} att_i^p * \log \frac{att_i^p}{\tilde{d_i}^p}. \tag{17}$$

Jensen-Shannon Divergence (JS). Just like the above method, except that we calculate the Jensen-Shannon divergence between position distribution and attention weight distribution:

$$\mathrm{JS}(att^p \mid\mid \tilde{d}^p) = \frac{1}{2}\mathrm{KL}(att^p \mid\mid \frac{att^p + \tilde{d}^p}{2}) + \frac{1}{2}\mathrm{KL}(\tilde{d}^p \mid\mid \frac{att^p + \tilde{d}^p}{2}). \tag{18}$$

We use $\mathrm{Diff}(\tilde{d}, att)$ to indicate the difference between position distribution and attention weight distribution which is calculated like above, and add it to classification loss as a regularizer. Our final loss is defined as follows:

$$loss = \frac{1}{N}\,[\sum_{i=1}^{N} -y_i \log p(y_i) + \beta \mathrm{Diff}(\tilde{d}, att)], \tag{19}$$

where β is the coefficient of regularizer.

4 Experiments

4.1 Datasets

We perform experiments on SemEval 2014 [8] which includes two datasets: *Laptop* and *Restaurant*. Each sample in the datasets consists of a sentence, a target and the corresponding sentiment polarity. Table 1 shows the statistic results of the datasets.

Table 1. Statistic results of *Laptop* and *Restaurant*.

	Set	Total	Positive	Negative	Neutral
Laptop	Train	2328	994	870	464
	Test	638	341	128	169
Restaurant	Train	3608	2164	807	637
	Test	1120	728	196	196

4.2 Parameters Initializing

We use 300 dimensions pre-trained Glove vector [19] to initialize our word embedding, which will be tuned during training. The dimension of hidden state of BiLSTMs is set to 100 and learning rate is set to 0.0005. We use three convolutional kernel sizes 3, 4, 5 with 128 filters in each size. The weights of neural network are all uniformly initialized between -1 and 1 while the bias are set to 0. The coefficient of regularizer is tuned between 0 and 4, where we first find the maximum value and then fine-tune it from 0 to maximum. We set the dropout of input and output to 0.2 and 0.5 respectively by grid searching from 0 to 1. We use Adam [20] as our training method.

4.3 Model Comparisons

In order to inspect the performance of our model, we compare our model with some classical models, such as SVM, AE-LSTM, ATAE-LSTM, TD-LSTM, IAN, EAM, MemNet, PBAN. Also, we perform some ablation experiments to show the effect of individual modules. We report the accuracy and Macro-F1 on *Laptop* and *Restaurant* datasets.

– SVM [3]: This approach leverages traditional machine learning algorithm SVM with feature engineering to classify the sentiment polarity.
– AE-LSTM & ATAE-LSTM [4]: This approach employs two LSTMs to model the left and right contexts of the target separately, then performs predictions based on concatenated context representations.
– TD-LSTM [10]: This approach employs two LSTMs to model the left and right contexts of the target separately, then performs predictions based on concatenated context representations.
– IAN [5]: This approach leverages two LSTMs to model context and aspect target separately. Then both context and aspect target learn their representation from their interaction.
– EAM [6]: This approach models each aspect target as a mixture of K aspect embeddings and selectively focuses on a small subset of context words according to the position information on the dependency tree of sentence.

Table 2. Experimental results of our proposed model and compared baseline models. Models with * indicate that position information is taken into account.

	Models	Laptop		Restaurant	
		ACC	Macro-F1	ACC	Macro-F1
Baselines	SVM	70.49	–	80.16	–
	AE-LSTM	68.90	–	76.60	–
	ATAE-LSTM	68.70	–	77.20	–
	TD-LSTM	71.83	68.43	78.00	66.73
	IAN	72.10	–	78.60	–
	EAM*	71.94	69.23	80.63	71.32
	MemNet*	72.21	–	80.95	–
	PBAN*	74.12	–	81.16	–
LAC-Pos variants	LAC-Pos-N-AD	74.92	70.67	81.25	71.64
	LAC-Pos-N-KL	**75.08**	70.73	80.98	71.71
	LAC-Pos-N-JS	74.76	70.21	81.25	71.73
	LAC-Pos-G-AD	74.76	70.50	81.07	70.76
	LAC-Pos-G-KL	74.61	70.30	**81.34**	**71.91**
	LAC-Pos-G-JS	74.76	**70.76**	80.89	70.65

- MemNet [15]: This approach applies attention mechanism over the word embeddings multiple times and predicts sentiments based on the top-most sentence representations. And it also leverages the position information to adjust the word embeddings in memory.
- PBAN [7]: This approach append position embedding after word embedding and mutually models the relation between aspect term and sentence by employing bidirectional attention mechanism.

4.4 Experimental Results

Table 2 shows the experimental results of our proposed model and other baseline models. Since adding position information, we call our proposed basic model as LAC-Pos, which means LSTM-Attention-CNN-Position. According to the method of calculating the position distribution and calculating the difference between position distribution and attention weight distribution, we add suffix after LAC-Pos to represent our full model. For example, LAC-Pos-N-AD means basic LAC-Pos equipped with normalized position distribution and using absolute difference to calculate the difference between position distribution and attention weight distribution.

We explore a variety of methods to leverage position information which are called LAC-Pos Variants. From Table 2 we can observe that our approach outperforms all the compared approaches on both *Laptop* and *Restaurant*. And the performances among all the LAC-Pos variants are very close which demonstrates that our approach is robust, generalized and not limited to specific methods of calculating position distribution and calculating the difference. The reason that results of our full models vary in different distance measures may be that the calculating methods vary in different distance measures, which will give different results even given two same probability distributions.

Table 3. Experimental results of ablated LAC-Pos.

Ablated LAC-Pos	Laptops		Restaurants	
	ACC	Macro-F1	ACC	Macro-F1
LPC-N	72.26	67.26	79.73	69.50
LPC-G	71.79	67.36	79.73	69.59
LAC	73.35	68.53	80.54	71.24

IAN, EAM, MemNet and PBAN, with sophisticated attention modules, outperform than TD-LSTM on both *Laptop* and *Restaurant*, which shows the attention mechanism can figure out target-related words effectively. In addition, EAM, MemNet and PBAN all take into account the position information, although the methods of using position information are different, and they all outperform than IAN, which indicates the position information is helpful for this task.

Although PBAN has achieved good performance, it is still not as good as our model, especially on *Laptop*. The reason may be that position embedding does indicate the distance between context words and target, but it does not explicitly requires higher weights for words that are closer to target. In contrast, our model adds the difference between position distribution and attention weight distribution to final loss as penalty, which is a clear and strong signal: the closer the word is to the target, the higher the weight.

Furthermore, we observe that after adding the position information, the promotion on *Laptop* is greater than *Restaurant*. The reason may be the data in *Laptop* contain a large number of exclusive noun which are rare words, and it makes difficulties to calculate the accurate attention weights, so that the attention mechanism can not distinguish target-related words and target-unrelated words well, so the position information will be very helpful for adjusting attention weights.

4.5 Ablation Experiments

To demonstrate the effectiveness of our approach, we set up three ablated models which use either the attention weight or position information as features. LPC-N and LPC-G mean we abandon attention mechanism and only use normalized position distribution and gaussian position distribution to adjust word embeddings respectively. LAC means we use only attention weights to adjust word embedding.

We conduct the statistical t-test between our full models and LAC. From Table 3 we can observe that after abandoning the position information, the performance of LAC drops dramatically. And all of the produced p-values are less than 0.05, which indicates the improvements brought in by position information are significant. Further more, we can see that if we only use the position information without the attention module, the performances of LPC-N and LPC-G are worse than LAC, which proves the correctness of our strategy that combining the attention mechanism and position information.

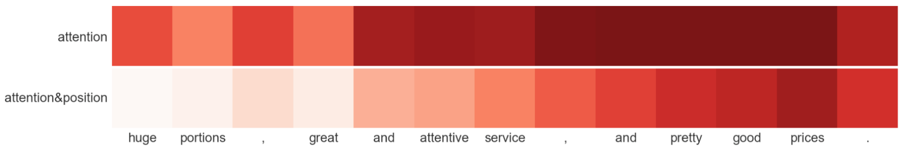

Fig. 3. Illustration of attention where horizontal axis is the sentence and the vertical axis is the method. The upper heat map is the attention distribution of LAC and the lower heat map is the attention distribution of LAC-Pos-N-AD, where the targets are *prices*.

Even if we abandon position information, LAC still outperforms than most compared models, which shows the power of CNN capturing local features. When we only use position information, LPC-N and LPC-G can still get pretty good results, which shows that position information is actually helpful to distinguish target-related words and target-unrelated words.

4.6 Case Study

In this part, we take a case to show the effectiveness of our method. Figure 3 shows two heat maps of attention weights from LAC and LAC-Pos-N-AD respectively. The attention weights are both transformed into between 0 and 1 which indicates how much information should be preserved for every word. The weight of each word is visualized by the color depth where the redder the color, the greater the attention weight.

As shown in the heat map of LAC, the attention-based model not only give high weights to the real target-related words *pretty good*, but also those target-unrelated words such as *huge, great* and *attentive*, which is also a common problem of the current attention mechanism.

However, after adding position information, our LAC-Pos-N-AD still pays attention to the target-related words but ignore those target-unrelated words according to the position information. Therefore, the position distribution can better guide the generation of attention weights.

5 Conclusions

In this paper, we analyze the importance of position information in aspect-level sentiment classification and propose a novel approach to combine position information and attention mechanism: leverage the position distribution to modify the attention weight distribution. Then we use the adjusted attention weights to adjust the word embeddings and apply CNN on the weighted embedding matrix to capture the local n-gram features for sentiment classification. We perform experiments on two datasets of SemEval 2014 and the experimental results show the effectiveness of our model.

References

1. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014). https://doi.org/10.3115/v1/D14-1181
2. Liu, B.: Sentiment analysis and opinion mining. Synth. Lect. Hum. Lang. Technol. **5**(1), 1–167 (2012). https://doi.org/10.2200/S00416ED1V01Y201204HLT016
3. Jiang, L., Yu, M., Zhou, M., Liu, X., Zhao, T.: Target-dependent twitter sentiment classification. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pp. 151–160. Association for Computational Linguistics (2011)
4. Wang, Y., Huang, M., Zhao, L., et al.: Attention-based LSTM for aspect-level sentiment classification. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 606–615 (2016). https://doi.org/10.18653/v1/D16-1058

5. Ma, D., Li, S., Zhang, X., Wang, H.: Interactive attention networks for aspect-level sentiment classification. arXiv preprint arXiv:1709.00893 (2017). https://doi.org/10.24963/ijcai.2017/568
6. He, R., Lee, W.S., Ng, H.T., Dahlmeier, D.: Effective attention modeling for aspect-level sentiment classification. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 1121–1131 (2018)
7. Gu, S., Zhang, L., Hou, Y., Song, Y.: A position-aware bidirectional attention network for aspect-level sentiment analysis. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 774–784 (2018)
8. Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., Manandhar, S.: Semeval-2014 task 4: aspect based sentiment analysis. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pp. 27–35 (2014). https://doi.org/10.3115/v1/S14-2004
9. Zhao, W.X., Jiang, J., Yan, H., Li, X.: Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 56–65. Association for Computational Linguistics (2010)
10. Tang, D., Qin, B., Feng, X., Liu, T.: Effective LSTMs for target-dependent sentiment classification. arXiv preprint arXiv:1512.01100 (2015)
11. Wang, B., Lu, W.: Learning latent opinions for aspect-level sentiment classification. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
12. Liu, J., Zhang, Y.: Attention modeling for targeted sentiment. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, vol. 2, pp. 572–577 (2017). https://doi.org/10.18653/v1/E17-2091
13. Zhang, Y., Zhong, V., Chen, D., Angeli, G., Manning, C.D.: Position-aware attention and supervised data improve slot filling. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (2017). https://doi.org/10.18653/v1/D17-1004
14. Weston, J., Chopra, S., Bordes, A.: Memory networks. arXiv preprint arXiv:1410.3916 (2014)
15. Tang, D., Qin, B., Liu, T.: Aspect level sentiment classification with deep memory network. arXiv preprint arXiv:1605.08900 (2016). https://doi.org/10.18653/v1/D16-1021
16. Yang, J., Yang, R., Wang, C., Xie, J.: Multi-entity aspect-based sentiment analysis with context, entity and aspect memory. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018b). https://doi.org/10.1145/3321125
17. Luong, M.-T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015). https://doi.org/10.18653/v1/D15-1166
18. Yang, B., Tu, Z., Wong, D.F., Meng, F., Chao, L.S., Zhang, T.: Modeling localness for self-attention networks. arXiv preprint arXiv:1810.10182 (2018a). https://doi.org/10.18653/v1/D18-1475
19. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014). https://doi.org/10.3115/v1/D14-1162
20. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)