Progressive Self-Supervised Attention Learning for Aspect-Level Sentiment Analysis

Jialong Tang[1,2,3*], **Ziyao Lu**[1*], **Jinsong Su**[1†], **Yubin Ge**[4], **Linfeng Song**[5],
Le Sun[2], **Jiebo Luo**[5]

[1]Xiamen University, Xiamen, China
[2]Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
[4]University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
[5]Department of Computer Science, University of Rochester, Rochester NY 14627, USA
`jialong2019@iscas.ac.cn, ziyaolu2018@stu.xmu.edu.cn`
`jssu@xmu.edu.cn`

Abstract

In aspect-level sentiment classification (ASC), it is prevalent to equip dominant neural models with attention mechanisms, for the sake of acquiring the importance of each context word on the given aspect. However, such a mechanism tends to excessively focus on a few frequent words with sentiment polarities, while ignoring infrequent ones. In this paper, we propose a progressive self-supervised attention learning approach for neural ASC models, which automatically mines useful attention supervision information from a training corpus to refine attention mechanisms. Specifically, we iteratively conduct sentiment predictions on all training instances. Particularly, at each iteration, the context word with the maximum attention weight is extracted as the one with active/misleading influence on the correct/incorrect prediction of every instance, and then the word itself is masked for subsequent iterations. Finally, we augment the conventional training objective with a regularization term, which enables ASC models to continue equally focusing on the extracted active context words while decreasing weights of those misleading ones. Experimental results on multiple datasets show that our proposed approach yields better attention mechanisms, leading to substantial improvements over the two state-of-the-art neural ASC models. Source code and trained models are available.[1]

1 Introduction

Aspect-level sentiment classification (ASC), as an indispensable task in sentiment analysis, aims at inferring the sentiment polarity of an input sentence in a certain aspect. In this regard, pre-

---

[*]Equal contribution
[†]Corresponding author
[1]https://github.com/DeepLearnXMU/PSSAttention

vious representative models are mostly discriminative classifiers based on manual feature engineering, such as Support Vector Machine (Kiritchenko et al., 2014; Wagner et al., 2014). Recently, with the rapid development of deep learning, dominant ASC models have evolved into neural network (NN) based models (Tang et al., 2016b; Wang et al., 2016; Tang et al., 2016a; Ma et al., 2017; Chen et al., 2017; Li et al., 2018; Wang et al., 2018), which are able to automatically learn the aspect-related semantic representation of an input sentence and thus exhibit better performance. Usually, these NN-based models are equipped with attention mechanisms to learn the importance of each context word towards a given aspect. It can not be denied that attention mechanisms play vital roles in neural ASC models.

However, the existing attention mechanism in ASC suffers from a major drawback. Specifically, it is prone to overly focus on a few frequent words with sentiment polarities and little attention is laid upon low-frequency ones. As a result, the performance of attentional neural ASC models is still far from satisfaction. We speculate that this is because there exist widely "*apparent patterns*" and "*inapparent patterns*". Here, "*apparent patterns*" are interpreted as high-frequency words with strong sentiment polarities and "*inapparent patterns*" are referred to as low-frequency ones in training data. As mentioned in (Li et al., 2018; Xu et al., 2018; Lin et al., 2017), NNs are easily affected by these two modes: "*apparent patterns*" tend to be overly learned while "*inapparent patterns*" often can not be fully learned.

Here we use sentences in Table 1 to explain this defect. In the first three training sentences, given the fact that the context word "*small*" occurs frequently with negative sentiment, the atten-

Type	Sentence	Ans./Pred.
Train	The **[place]** is small and crowded but the service is quick .	**Neg / —**
Train	The **[place]** is a bit too small for live music .	**Neg / —**
Train	The service is decent even when this small **[place]** is packed .	**Neg / —**
Test	At lunch time , the **[place]** is crowded .	**Neg / Pos**
Test	A small area makes for quiet **[place]** to study alone .	**Pos / Neg**

Table 1: The example of attention visualization for five sentences, where the first three are training instances and the last two are test ones. The bracketed bolded words are target aspects. Ans./Pred. = ground-truth/predicted sentiment label. Words are highlighted with different degrees according to attention weights.

tion mechanism pays more attention to it and directly relates the sentences containing it with negative sentiment. This inevitably causes another informative context word "*crowded*" to be partially neglected in spite of it also possesses negative sentiment. Consequently, a neural ASC model incorrectly predicts the sentiment of the last two test sentences: in the first test sentence, the neural ASC model fails to capture the negative sentiment implicated by "*crowded*"; while, in the second test sentence, the attention mechanism directly focuses on "*small*" though it is not related to the given aspect. Therefore, we believe that the attention mechanism for ASC still leaves tremendous room for improvement.

One potential solution to the above-mentioned issue is supervised attention, which, however, is supposed to be manually annotated, requiring labor-intense work. In this paper, we propose a novel progressive self-supervised attention learning approach for neural ASC models. Our method is able to automatically and incrementally mine attention supervision information from a training corpus, which can be exploited to guide the training of attention mechanisms in ASC models. The basic idea behind our approach roots in the following fact: the context word with the maximum attention weight has the greatest impact on the sentiment prediction of an input sentence. Thus, such a context word of a correctly predicted training instance should be taken into consideration during the model training. In contrast, the context word in an incorrectly predicted training instance ought to be ignored. To this end, we iteratively conduct sentiment predictions on all training instances. Particularly, at each iteration, we extract the context word with the maximum attention weight from each training instance to form attention supervision information, which can be used

to guide the training of attention mechanism: in the case of correct prediction, we will remain this word to be considered; otherwise, the attention weight of this word is expected to be decreased. Then, we mask all extracted context words of each training instance so far and then refollow the above process to discover more supervision information for attention mechanisms. Finally, we augment the standard training objective with a regularizer, which enforces attention distributions of these mined context words to be consistent with their expected distributions.

Our main contributions are three-fold: (1) Through in-depth analysis, we point out the existing drawback of the attention mechanism for ASC. (2) We propose a novel incremental approach to automatically extract attention supervision information for neural ASC models. To the best of our knowledge, our work is the first attempt to explore automatic attention supervision information mining for ASC. (3) We apply our approach to two dominant neural ASC models: Memory Network (**MN**) (Tang et al., 2016b; Wang et al., 2018) and Transformation Network (**TNet**) (Li et al., 2018). Experimental results on several benchmark datasets demonstrate the effectiveness of our approach.

2 Background

In this section, we give brief introductions to MN and TNet, which both achieve satisfying performance and thus are chosen as the foundations of our work. Here we introduce some notations to facilitate subsequent descriptions: $x = (x_1, x_2, ..., x_N)$ is the input sentence, $t = (t_1, t_2, ..., t_T)$ is the given target aspect, y, $y_p \in \{$Positive, Negative, Neutral$\}$ denote the ground-truth and the predicted sentiment, respectively.
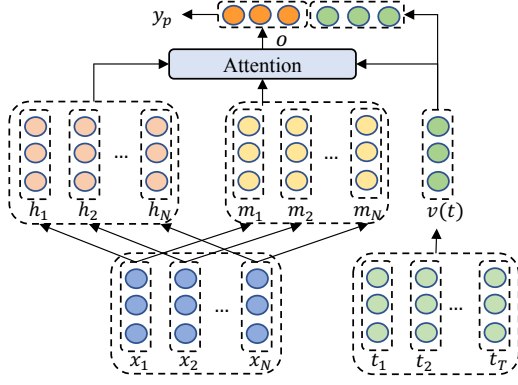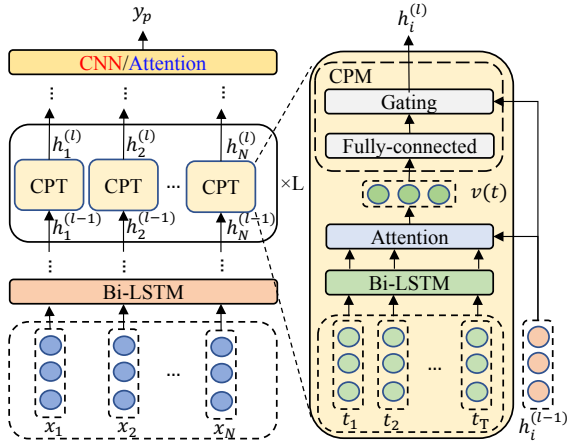
Figure 1: The framework architecture of MN.

Figure 2: The framework architecture of TNet/TNet-ATT. Note that TNet-ATT is the variant of TNet replacing CNN with an attention mechanism.

MN (Tang et al., 2016b; Wang et al., 2018). The framework illustration of MN is given in Figure 1. We first introduce an aspect embedding matrix converting each target aspect word t_j into a vector representation, and then define the final vector representation $v(t)$ of t as the averaged aspect embedding of its words. Meanwhile, another embedding matrix is used to project each context word x_i to the continuous space stored in memory, denoted by m_i. Then, an internal attention mechanism is applied to generate the aspect-related semantic representation o of the sentence x: $o = \sum_i softmax(v_t^T M m_i)h_i$, where M is an attention matrix and h_i is the final semantic representation of x_i, induced from a context word embedding matrix. Finally, we use a fully connected output layer to conduct classification based on o and $v(t)$.

TNet (Li et al., 2018). Figure 2 provides the framework illustrations of TNet, which mainly consists of three components:

(1) The bottom layer is a Bi-LSTM that transforms the input x into the contextualized word representations $h^{(0)}(x)=(h_1^{(0)}, h_2^{(0)}, ..., h_N^{(0)})$ (i.e. hidden states of Bi-LSTM). (2) The middle part, as the core of the whole model, contains L layers of Context-Preserving Transformation (CPT), where word representations are updated as $h^{(l+1)}(x)=$CPT$(h^{(l)}(x))$. The key operation of CPT layers is Target-Specific Transformation. It contains another Bi-LSTM for generating $v(t)$ via an attention mechanism, and then incorporates $v(t)$ into the word representations. Besides, CPT layers are also equipped with a Context-Preserving Mechanism (CPM) to preserve the context information and learn more abstract word-level features. In the end, we obtain the word-level semantic representations $h(x)=(h_1,h_2...,h_N)$, with $h_i=h_i^{(L)}$. (3) The topmost part is a CNN layer used to produce the aspect-related sentence representation o for the sentiment classification.

In this work, we consider another alternative to the original TNet, which replaces its topmost CNN with an attention mechanism to produce the aspect-related sentence representation as o=Atten($h(x)$, $v(t)$). In Section 4, we will investigate the performance of the original TNet and its variant equipped with an attention mechanism, denoted by **TNet-ATT**.

Training Objective. Both of the above-mentioned models take the negative log-likelihood of the gold-truth sentiment tags as their training objectives:

$$J(D;\theta) = - \sum_{(x,t,y)\in D} J(x,t,y;\theta)$$
$$= \sum_{(x,t,y)\in D} d(y) \cdot \log d(x,t;\theta), \quad (1)$$

where D is the training corpus, $d(y)$ is the one-hot vector of y, $d(x,t;\theta)$ is the model-predicted sentiment distribution for the pair (x,t), and $\cdot$ denotes the dot product of two vectors.

3 Our Approach

In this section, we first describe the basic intuition behind our approach and then provide its details. Finally, we elaborate how to incorporate the mined supervision information for attention mechanisms into neural ASC models. It is noteworthy that our method is only applied to the training optimization

of neural ASC models, without any impact on the model testing.

3.1 Basic Intuition

The basic intuition of our approach stems from the following fact: in attentional ASC models, the importance of each context word on the given aspect mainly depends on its attention weight. Thus, the context word with the maximum attention weight has the most important impact on the sentiment prediction of the input sentence. Therefore, for a training sentence, if the prediction of ASC model is correct, we believe that it is reasonable to continue focusing on this context word. Conversely, the attention weight of this context word should be decreased.

However, as previously mentioned, the context word with the maximum attention weight is often the one with strong sentiment polarity. It usually occurs frequently in the training corpus and thus tends to be overly considered during model training. This simultaneously leads to the insufficient learning of other context words, especially low-frequency ones with sentiment polarities. To address this problem, one intuitive and feasible method is to first shield the influence of this most important context word before reinvestigating effects of remaining context words of the training instance. In that case, other low-frequency context words with sentiment polarities can be discovered according to their attention weights.

3.2 Details of Our Approach

Based on the above analysis, we propose a novel incremental approach to automatically mine influential context words from training instances, which can be then exploited as attention supervision information for neural ASC models.

As shown in Algorithm 1, we first use the initial training corpus D to conduct model training, and then obtain the initial model parameters $\theta^{(0)}$ (**Line 1**). Then, we continue training the model for K iterations, where influential context words of all training instances can be iteratively extracted (**Lines 6-25**). During this process, for each training instance (x, t, y), we introduce two word sets initialized as $\emptyset$ (**Lines 2-5**) to record its extracted context words: (1) $s_a(x)$ consists of context words with **a**ctive effects on the sentiment prediction of x. Each word of $s_a(x)$ will be encouraged to remain considered in the refined model training, and (2) $s_m(x)$ contains context words with **m**isleading

---

Algorithm 1 : Neural ASC Model Training with Automatically Mined Attention Supervision Information.

Input: D: the initial training corpus;
    θ^{init}: the initial model parameters;
    ϵ_α: the entropy threshold of attention weight distribution;
    K: the maximum number of training iterations;
1: $\theta^{(0)} \leftarrow \textbf{\textit{Train}}(D; \theta^{init})$
2: **for** $(x, t, y) \in D$ **do**
3:     $s_a(x) \leftarrow \emptyset$
4:     $s_m(x) \leftarrow \emptyset$
5: **end for**
6: **for** $k = 1, 2..., K$ **do**
7:     $D^{(k)} \leftarrow \emptyset$
8:     **for** $(x, t, y) \in D$ **do**
9:        $v(t) \leftarrow \textbf{\textit{GenAspectRep}}(t, \theta^{(k-1)})$
10:       $x' \leftarrow \textbf{\textit{MaskWord}}(x, s_a(x), s_m(x))$
11:       $h(x') \leftarrow \textbf{\textit{GenWordRep}}(x', v(t), \theta^{(k-1)})$
12:       $y_p, \alpha(x') \leftarrow \textbf{\textit{SentiPred}}(h(x'), v(t), \theta^{(k-1)})$
13:       $E(\alpha(x')) \leftarrow \textbf{\textit{CalcEntropy}}(\alpha(x'))$
14:       **if** $E(\alpha(x')) < \epsilon_\alpha$ **then**
15:          $m \leftarrow argmax_{1 \le i \le N} \; \alpha(x'_i)$
16:          **if** $y_p == y$ **then**
17:            $s_a(x) \leftarrow s_a(x) \cup \{x'_m\}$
18:          **else**
19:            $s_m(x) \leftarrow s_m(x) \cup \{x'_m\}$
20:          **end if**
21:       **end if**
22:       $D^{(k)} \leftarrow D^{(k)} \cup (x', t, y)$
23:     **end for**
24:     $\theta^{(k)} \leftarrow \textbf{\textit{Train}}(D^{(k)}; \theta^{(k-1)})$
25: **end for**
26: $D_s \leftarrow \emptyset$
27: **for** $(x, t, y) \in D$ **do**
28:     $D_s \leftarrow D_s \cup (x, t, y, s_a(x), s_m(x))$
29: **end for**
30: $\theta \leftarrow \textbf{\textit{Train}}(D_s)$
Return: θ;

---

effects, whose attention weights are expected to be decreased. Specifically, at the k-th training iteration, we adopt the following steps to deal with (x, t, y):

In **Step 1**, we first apply the model parameters $\theta^{(k-1)}$ of the previous iteration to generate the aspect representation $v(t)$ (**Line 9**). Importantly, according to $s_a(x)$ and $s_m(x)$, we then mask all previously extracted context words of x to create a new sentence x', where each masked word is replaced with a special token "$\langle mask \rangle$" (**Line 10**). In this way, the effects of these context words will be shielded during the sentiment prediction of x', and thus other context words can be potentially extracted from x'. Finally, we generate the word representations $h(x')=\{h(x'_i)\}_{i=1}^{N}$ (**Line 11**).

In **Step 2**, on the basis of $v(t)$ and $h(x')$, we

Iter	Sentence	Ans./Pred.	$E(\alpha(x'))$	x'_m
1	The **[place]** is small and crowded but the service is quick .	Neg / Neg	2.38	*small*
2	The **[place]** is $\langle mask \rangle$ and crowded but the service is quick .	Neg / Neg	2.59	*crowded*
3	The **[place]** is $\langle mask \rangle$ and $\langle mask \rangle$ but the service is quick .	Neg / Pos	2.66	*quick*
4	The **[place]** is $\langle mask \rangle$ and $\langle mask \rangle$ but the service is $\langle mask \rangle$.	Neg / Neg	3.07	—

Table 2: The example of mining influential context words from the first training sentence in Table 1. $E(\alpha(x'))$ denotes the entropy of the attention weight distribution $\alpha(x')$, ϵ_α is entropy threshold set as 3.0, and x'_m indicates the context word with the maximum attention weight. Note that all extracted words are replaced with "$\langle mask \rangle$" and their background colors are labeled as white.

leverage $\theta^{(k-1)}$ to predict the sentiment of x' as y_p (**Line 12**), where the word-level attention weight distribution $\alpha(x') = \{\alpha(x'_1), \alpha(x'_2), ..., \alpha(x'_N)\}$ subjecting to $\sum_{i=1}^N \alpha(x'_i) = 1$ is induced.

In **Step 3**, we use the entropy $E(\alpha(x'))$ to measure the variance of $\alpha(x')$ (**Line 13**), which contributes to determine the existence of an influential context word for the sentiment prediction of x',

$$E(\alpha(x')) = -\sum_{i=1}^N \alpha(x'_i) \log(\alpha(x'_i)). \quad (2)$$

If $E(\alpha(x'))$ is less than a threshold ϵ_α (**Line 14**), we believe that there exists at least one context word with great effect on the sentiment prediction of x'. Hence, we extract the context word x'_m with the maximum attention weight (**Line 15-20**) that will be exploited as attention supervision information to refine the model training. Specifically, we adopt two strategies to deal with x'_m according to different prediction results on x': if the prediction is correct, we wish to continue focusing on x'_m and add it into $s_a(x)$ (**Lines 16-17**); otherwise, we expect to decrease the attention weight of x'_m and thus include it into $s_m(x)$ (**Lines 18-19**).

In **Step 4**, we combine x', t and y as a triple, and merge it with the collected ones to form a new training corpus $D^{(k)}$ (**Line 22**). Then, we leverage $D^{(k)}$ to continue updating model parameters for the next iteration (**Line 24**). In doing so, we make our model adaptive to discover more influential context words.

Through K iterations of the above steps, we manage to extract influential context words of all training instances. Table 2 illustrates the context word mining process of the first sentence shown in Table 1. In this example, we iteratively extract three context words in turn: "*small*", "*crowded*" and "*quick*". The former two words are included in $s_a(x)$, while the last one is contained in $s_m(x)$.

Finally, the extracted context words of each training instance will be included into D, forming a final training corpus D_s with attention supervision information (**Lines 26-29**), which will be used to carry out the last model training (**Line 30**). The details will be provided in the next subsection.

3.3 Model Training with Attention Supervision Information

To exploit the above extracted context words to refine the training of attention mechanisms for ASC models, we propose a soft attention regularizer $\triangle(\alpha(s_a(x) \cup s_m(x)), \hat{\alpha}(s_a(x) \cup s_m(x)); \theta)$ to jointly minimize the standard training objective, where $\alpha(*)$ and $\hat{\alpha}(*)$ denotes the model-induced and expected attention weight distributions of words in $s_a(x) \cup s_m(x)$, respectively. More specifically, $\triangle(\alpha(*), \hat{\alpha}(*); \theta)$ is an *Euclidean Distance* style loss that penalizes the disagreement between $\alpha(*)$ and $\hat{\alpha}(*)$.

As previously analyzed, we expect to equally continue focusing on the context words of $s_a(x)$ during the final model training. To this end, we set their expected attention weights to the same value $\frac{1}{|s_a(x)|}$. By doing so, the weights of words extracted first will be reduced, and those of words extracted later will be increased, avoiding the over-fitting of high-frequency context words with sentiment polarities and the under-fitting of low-frequency ones. On the other hand, for the words in $s_m(x)$ with misleading effects on the sentiment prediction of x, we want to reduce their effects and thus directly set their expected weights as 0. Back to the sentence shown in Table 2, both "*small*" and "*crowded*"$\in s_a(x)$ are assigned the same expected weight 0.5, and the expected weight of "*quick*"$\in s_m(x)$ is 0.

Finally, our objective function on the training corpus D_s with attention supervision information

Domain	Dataset	#Pos	#Neg	#Neu
LAPTOP	Train	980	858	454
	Test	340	128	171
REST	Train	2159	800	632
	Test	730	195	196
TWITTER	Train	1567	1563	3127
	Test	174	174	346

Table 3: Datasets in our experiments. **#Pos**, **#Neg** and **#Neu** denotes the number of instances with Positive, Negative and Neutral sentiment, respectively.

becomes

$$J_s(D_s; \theta) = - \sum_{(x,t,y) \in D_s} \{ J(x, t, y; \theta) + \quad (3)$$

$$\gamma \triangle (\alpha(s_a(x) \cup s_m(x)), \hat{\alpha}(s_a(x) \cup s_m(x)); \theta) \},$$

where $J(x, t, y; \theta)$ is the conventional training objective defined in Equation 1, and $\gamma > 0$ is a hyper-parameter that balances the preference between the conventional loss function and the regularization term. In addition to the utilization of attention supervision information, our method has a further advantage: it is easier to address the vanishing gradient problem by adding such information into the intermediate layers of the entire network (Szegedy et al., 2015), because the supervision of $\hat{\alpha}(*)$ is closer to $\alpha(*)$ than y.

4 Experiments

Datasets. We applied the proposed approach into MN (Tang et al., 2016b; Wang et al., 2018) and TNet-ATT (Li et al., 2018) (see Section 2), and conducted experiments on three benchmark datasets: LAPTOP, REST (Pontiki et al., 2014) and TWITTER (Dong et al., 2014). In our datasets, the target aspect of each sentence has been provided. Besides, we removed a few instances with conflict sentiment labels as implemented in (Chen et al., 2017). The statistics of the final datasets are listed in Table 3.

Contrast Models. We referred to our two enhanced ASC models as **MN(+AS)** and **TNet-ATT(+AS)**, and compared them with **MN**, **TNet**, and **TNet-ATT**. Note our models require additional $K+1$-iteration training, therefore, we also compared them with the above models with additional $K+1$-iteration training, which are denoted as **MN(+KT)**, **TNet(+KT)** and **TNet-ATT(+KT)**. Moreover, to investigate effects of different kinds of attention supervision information, we
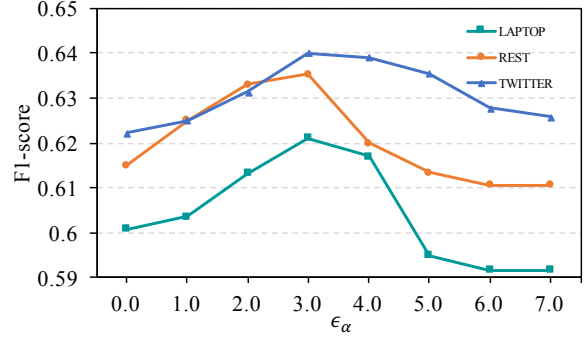

Figure 3: Effects of ϵ_α on the validation sets using MN(+AS).
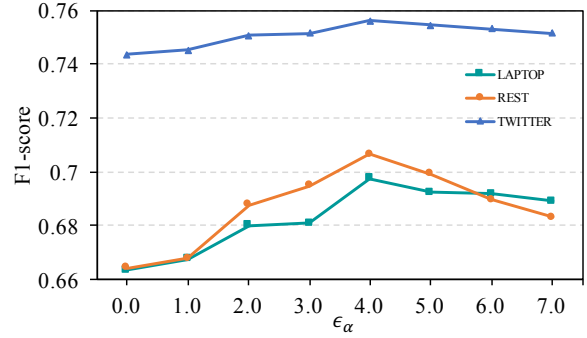

Figure 4: Effects of ϵ_α on the validation sets using TNet-ATT(+AS).

also listed the performance of **MN(+AS_a)** and **MN(+AS_m)**, which only leverage context words of $s_a(x)$ and $s_m(x)$, respectively, and the same for **TNet-ATT(+AS_a)** and **TNet-ATT(+AS_m)**.

Training Details. We used pre-trained *GloVe* vectors (Pennington et al., 2014) to initialize the word embeddings with vector dimension 300. For out-of-vocabulary words, we randomly sampled their embeddings from the uniform distribution [-0.25, 0.25], as implemented in (Kim, 2014). Besides, we initialized the other model parameters uniformly between [-0.01, 0.01]. To alleviate overfitting, we employed *dropout* strategy (Hinton et al., 2012) on the input word embeddings of the LSTM and the ultimate aspect-related sentence representation. *Adam* (Kingma and Ba, 2015) was adopted as the optimizer with the learning rate 0.001.

When implementing our approach, we empirically set the maximum iteration number K as 5, γ in Equation 3 as 0.1 on LAPTOP data set, 0.5 on REST data set and 0.1 on TWITTER data set, respectively. All hyper-parameters were tuned on 20% randomly held-out training data. Finally, we used F1-Macro and accuracy as our evaluation

Model	LAPTOP		REST		TWITTER	
	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy
MN (Wang et al., 2018)	62.89	68.90	64.34	75.30	—	—
MN	63.28	68.97	65.88	77.32	66.17	67.71
MN(+KT)	63.31	68.95	65.86	77.33	66.18	67.78
MN(+AS$_m$)	64.37	69.69	68.40	78.13	67.20	68.90
MN(+AS$_a$)	64.61	69.95	68.59	78.23	67.47	69.17
MN(+AS)	**65.24****	**70.53****	**69.15****	**78.75***	**67.88****	**69.64****
TNet (Li et al., 2018)	71.75	76.54	71.27	80.69	73.60	74.97
TNet	71.82	76.12	71.70	80.35	76.82	77.60
TNet(+KT)	71.74	76.44	71.36	80.59	76.78	77.54
TNet-ATT	71.21	76.06	71.15	80.32	76.53	77.46
TNet-ATT(+KT)	71.44	76.06	71.01	80.50	76.58	77.46
TNet-ATT(+AS$_m$)	72.39	76.89	72.04	80.96	77.42	78.08
TNet-ATT(+AS$_a$)	73.30	77.34	72.67	81.33	77.63	78.47
TNet-ATT(+AS)	**73.84****	**77.62****	**72.90****	**81.53***	**77.72****	**78.61***

Table 4: Experimental results on various datasets. We directly cited the best experimental results of MN and TNet reported in (Wang et al., 2018; Li et al., 2018). ** and * means significant at $p < 0.01$ and $p < 0.05$ over the baselines (MN, TNet) on each test set, respectively. Here we conducted 1,000 bootstrap tests (Koehn, 2004) to measure the significance in metric score differences.

measures.

4.1 Effects of ϵ_α

ϵ_α is a very important hyper-parameter that controls the iteration number of mining attention supervision information (see **Line 14** of Algorithm 1). Thus, in this group of experiments, we varied ϵ_α from 1.0 to 7.0 with an increment of 1 each time, so as to investigate its effects on the performance of our models on the validation sets.

Figure 3 and 4 show the experimental results of different models. Specifically, MN(+AS) with ϵ_α=3.0 achieves the best performance, meanwhile, the optimal performance of TNet-ATT(+AS) is obtained when ϵ_α=4.0. We observe the increase of ϵ_α does not lead to further improvements, which may be due to more noisy extracted context words. Because of these results, we set ϵ_α for MN(+AS) and TNet-ATT(+AS) as 3.0 and 4.0 in the following experiments, respectively.

4.2 Overall Results

Table 4 provides all the experimental results. To enhance the persuasiveness of our experimental results, we also displayed the previously reported scores of MN (Wang et al., 2018) and TNet (Li et al., 2018) on the same data set. According to the experimental results, we can come to the following conclusions:

First, both of our reimplemented MN and TNet are comparable to their original models reported in (Wang et al., 2018; Li et al., 2018). These results show that our reimplemented baselines are competitive. When we replace the CNN of TNet with an attention mechanism, TNet-ATT is slightly inferior to TNet. Moreover, when we perform additional K+1-iteration of training on these models, their performance has not changed significantly, suggesting simply increasing training time is unable to enhance the performance of the neural ASC models.

Second, when we apply the proposed approach into both MN and TNet-ATT, the context words in $s_a(x)$ are more effective than those in $s_m(x)$. This is because the proportion of correctly predicted training instances is larger than that of incorrectly ones. Besides, the performance gap between MN(+AS$_a$) and MN(+AS$_m$) is larger than that between two variants of TNet-ATT. One underlying reason is that the performance of TNet-ATT is better than MN, which enables TNet-ATT to produce more correctly predicted training instances. This in turn brings more attention supervision to TNet-ATT than MN.

Finally, when we use both kinds of attention supervision information, no matter for which metric, MN(+AS) remarkably outperforms MN on all test sets. Although our TNet-ATT is slightly in-

Model	Sentence	Ans./Pred.
TNet-ATT	The **[folding chair]** i was seated at was uncomfortable .	**Neg / Neu**
TNet-ATT(+AS)	The **[folding chair]** i was seated at was uncomfortable .	**Neg / Neg**
TNet-ATT	The **[food]** did take a few extra minutes ... the cute waiters ...	**Neu / Pos**
TNet-ATT(+AS)	The **[food]** did take a few extra minutes ... the cute waiters ...	**Neu / Neu**

Table 5: Two test cases predicted by TNet-ATT and TNet-ATT(+AS).

ferior to TNet, TNet-ATT(+AS) still significantly surpasses both TNet and TNet-ATT. These results strongly demonstrate the effectiveness and generality of our approach.

4.3 Case Study

In order to know how our method improves neural ASC models, we deeply analyze attention results of TNet-ATT and TNet-ATT(+AS). It has been found that our proposed approach can solve the above-mentioned two issues well.

Table 5 provides two test cases. TNet-ATT incorrectly predicts the sentiment of the first test sentence as neutral. This is because the context word "*uncomfortable*" only appears in two training instances with negative polarities, which distracts attention from it. When using our approach, the average attention weight of "*uncomfortable*" is increased to 2.6 times than that of baseline in these two instances. Thus, TNet-ATT(+AS) is capable of assigning a greater attention weight $(0.0056 \rightarrow 0.2940)$ to this context word, leading to the correct prediction of the first test sentence. For the second test sentence, since the context word "*cute*" occurs in training instances mostly with positive polarity, TNet-ATT directly focuses on this word and then incorrectly predicts the sentence sentiment as positive. Adopting our method, attention weights of "*cute*" in training instances with neural or negative polarity are significantly decreased. Specifically, in these instances, the average weight of "*cute*" is reduced to 0.07 times of the original. Hence, TNet-ATT(+AS) assigns a smaller weight $(0.1090 \rightarrow 0.0062)$ to "*cute*" and achieves the correct sentiment prediction.

5 Related Work

Recently, neural models have been shown to be successful on ASC. For example, due to its multiple advantages, such as being simpler and faster, MNs with attention mechanisms (Tang et al., 2016b; Wang et al., 2018) have been widely used.

Another prevailing neural model is LSTM that also involves an attention mechanism to explicitly capture the importance of each context word (Wang et al., 2016). Overall, attention mechanisms play crucial roles in all these models.

Following this trend, researchers have resorted to more sophisticated attention mechanisms to refine neural ASC models. Chen et al., (2017) proposed a multiple-attention mechanism to capture sentiment features separated by a long distance, so that it is more robust against irrelevant information. An interactive attention network has been designed by Ma et al., (2017) for ASC, where two attention networks were introduced to model the target and context interactively. Liu et al., (2017) proposed to leverage multiple attentions for ASC: one obtained from the left context and the other one acquired from the right context of a given aspect. Very recently, transformation-based model has also been explored for ASC (Li et al., 2018), and the attention mechanism is replaced by CNN.

Different from these work, our work is in line with the studies of introducing attention supervision to refine the attention mechanism, which have become hot research topics in several NN-based NLP tasks, such as event detection (Liu et al., 2017), machine translation (Liu et al., 2016), and police killing detection (Nguyen and Nguyen, 2018). However, such supervised attention acquisition is labor-intense. Therefore, we mainly commits to automatic mining supervision information for attention mechanisms of neural ASC models. Theoretically, our approach is orthogonal to these models, and we leave the adaptation of our approach into these models as future work.

Our work is inspired by two recent models: one is (Wei et al., 2017) proposed to progressively mine discriminative object regions using classification networks to address the weakly-supervised semantic segmentation problems, and the other one is (Xu et al., 2018) where a dropout method integrating with global information is presented to

encourage the model to mine inapparent features or patterns for text classification. To the best of our knowledge, our work is the first one to explore automatic mining of attention supervision information for ASC.

6 Conclusion and Future Work

In this paper, we have explored how to automatically mine supervision information for attention mechanisms of neural ASC models. Through in-depth analyses, we first point out the defect of the attention mechanism for ASC: a few frequent words with sentiment polarities are tend to be over-learned, while those with low frequency often lack sufficient learning. Then, we propose a novel approach to automatically and incrementally mine attention supervision information for neural ASC models. These mined information can be further used to refine the model training via a regularization term. To verify the effectiveness of our approach, we apply our approach into two dominant neural ASC models, where experimental results demonstrate our method significantly improves the performance of these two models.

Our method is general for attention mechanisms. Thus, we plan to extend our approach to other neural NLP tasks with attention mechanisms, such as neural document classification (Yang et al., 2016) and neural machine translation (Zhang et al., 2018).

Acknowledgments

References

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *EMNLP*.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *Computer Science*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *SemEval*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *ACL*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dolla r. 2017. Focal loss for dense object detection. In *ICCV*.

Lemao Liu, Masao Utiyama, Andrew M. Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *COLING*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *ACL*.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *IJCAI*.

Minh Nguyen and Thien Nguyen. 2018. Who is killed by police: Introducing supervised attention for hierarchical lstms. In *COLING*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *COLING*.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *EMNLP*.

Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. DCU: aspect-based polarity classification for semeval task 4. In *SemEval*.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018. Target-sensitive memory networks for aspect sentiment classification. In *ACL*.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *EMNLP*.

Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. 2017. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *CVPR*.

Hengru Xu, Shen Li, Renfen Hu, Si Li, and Sheng Gao. 2018. From random to supervised: A novel dropout mechanism integrated with global information. In *CONLL*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Neural machine translation with deep attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yue Zhang and Jiangming Liu. 2017. Attention modeling for targeted sentiment. In *EACL*.

Progressive Self-Supervised Attention Learning for Aspect-Level Sentiment Analysis

Tang, Jialong; Lu, Ziyao; Su, Jinsong; Ge, Yubin; Song, Linfeng; Sun, Le; Luo, Jiebo

---

01   mingchen yuan   Page 1

19/9/2019 2:41

adj. 支配的，统治的；占优势的；显性的
n. 显性

---

02   mingchen yuan   Page 1

19/9/2019 2:45

adj. 普遍的, 盛行的, 流行的

---

03   mingchen yuan   Page 1

19/9/2019 2:45

---

04   mingchen yuan   Page 1

19/9/2019 2:41

---

05   mingchen yuan   Page 1

19/9/2019 2:41