Transformation Networks for Target-Oriented Sentiment Classification*

Xin Li[1], Lidong Bing[2], Wai Lam[1] and Bei Shi[1]
[1]Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Hong Kong
[2]Tencent AI Lab, Shenzhen, China
{lixin,wlam,bshi}@se.cuhk.edu.hk
lyndonbing@tencent.com

Abstract

Target-oriented sentiment classification aims at classifying sentiment polarities over individual opinion targets in a sentence. RNN with attention seems a good fit for the characteristics of this task, and indeed it achieves the state-of-the-art performance. After re-examining the drawbacks of attention mechanism and the obstacles that block CNN to perform well in this classification task, we propose a new model to overcome these issues. Instead of attention, our model employs a CNN layer to extract salient features from the transformed word representations originated from a bi-directional RNN layer. Between the two layers, we propose a component to generate target-specific representations of words in the sentence, meanwhile incorporate a mechanism for preserving the original contextual information from the RNN layer. Experiments show that our model achieves a new state-of-the-art performance on a few benchmarks.[1]

1 Introduction

Target-oriented (also mentioned as "target-level" or "aspect-level" in some works) sentiment classification aims to determine sentiment polarities over "opinion targets" that explicitly appear in the sentences (Liu, 2012). For example, in the sentence *"I am pleased with the fast **log on**, and the long **battery life**"*, the user mentions two targets "***log on***" and "***better life***", and expresses positive sentiments over them. The task is usually formulated as predicting a sentiment category for a (target, sentence) pair.

Recurrent Neural Networks (RNNs) with attention mechanism, firstly proposed in machine translation (Bahdanau et al., 2014), is the most commonly-used technique for this task. For example, Wang et al. (2016); Tang et al. (2016b); Yang et al. (2017); Liu and Zhang (2017); Ma et al. (2017) and Chen et al. (2017) employ attention to measure the semantic relatedness between each context word and the target, and then use the induced attention scores to aggregate contextual features for prediction. In these works, the attention weight based combination of word-level features for classification may introduce noise and downgrade the prediction accuracy. For example, in "*This **dish** is my favorite and I always get it and never get tired of it.*", these approaches tend to involve irrelevant words such as "never" and "tired" when they highlight the opinion modifier "favorite". To some extent, this drawback is rooted in the attention mechanism, as also observed in machine translation (Luong et al., 2015) and image captioning (Xu et al., 2015).

Another observation is that the sentiment of a target is usually determined by key phrases such as "is my favorite". By this token, Convolutional Neural Networks (CNNs)—whose capability for extracting the informative n-gram features (also called "active local features") as sentence representations has been verified in (Kim, 2014; Johnson and Zhang, 2015)— should be a suitable model for this classification problem. However, CNN likely fails in cases where a sentence expresses different sentiments over multiple targets, such as *"great **food** but the **service** was dreadful!"*. One reason is that CNN cannot fully explore the target information as done by RNN-based meth-

*The work was done when Xin Li was an intern at Tencent AI Lab. This project is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414).

[1]Our code is open-source and available at https://github.com/lixin4ever/TNet

ods (Tang et al., 2016a).[2] Moreover, it is hard for vanilla CNN to differentiate opinion words of multiple targets. Precisely, multiple active local features holding different sentiments (e.g., "great food" and "service was dreadful") may be captured for a single target, thus it will hinder the prediction.

We propose a new architecture, named Target-Specific **T**ransformation **Net**works (TNet), to solve the above issues in the task of target sentiment classification. TNet firstly encodes the context information into word embeddings and generates the contextualized word representations with LSTMs. To integrate the target information into the word representations, TNet introduces a novel Target-Specific Transformation (TST) component for generating the target-specific word representations. Contrary to the previous attention-based approaches which apply the same target representation to determine the attention scores of individual context words, TST firstly generates different representations of the target conditioned on individual context words, then it consolidates each context word with its tailor-made target representation to obtain the transformed word representation. Considering the context word *"long"* and the target "*battery life*" in the above example, TST firstly measures the associations between "long" and individual target words. Then it uses the association scores to generate the target representation conditioned on "long". After that, TST transforms the representation of "long" into its target-specific version with the new target representation. Note that "long" could also indicate a negative sentiment (say for "startup time"), and the above TST is able to differentiate them.

As the context information carried by the representations from the LSTM layer will be lost after the non-linear TST, we design a context-preserving mechanism to contextualize the generated target-specific word representations. Such mechanism also allows deep transformation structure to learn abstract features[3]. To help the CNN feature extractor locate sentiment indicators more accurately, we adopt a proximity strategy to scale the input of convolutional layer with positional relevance between a word and the target.

[2]One method could be concatenating the target representation with each word representation, but the effect as shown in (Wang et al., 2016) is limited.

[3]Abstract features usually refer to the features ultimately useful for the task (Bengio et al., 2013; LeCun et al., 2015).

In summary, our contributions are as follows:

• TNet adapts CNN to handle target-level sentiment classification, and its performance dominates the state-of-the-art models on benchmark datasets.

• A novel Target-Specific Transformation component is proposed to better integrate target information into the word representations.

• A context-preserving mechanism is designed to forward the context information into a deep transformation architecture, thus, the model can learn more abstract contextualized word features from deeper networks.

2 Model Description

Given a target-sentence pair $(\mathbf{w}^\tau, \mathbf{w})$, where $\mathbf{w}^\tau = \{w_1^\tau, w_2^\tau, ..., w_m^\tau\}$ is a sub-sequence of $\mathbf{w} = \{w_1, w_2, ..., w_n\}$, and the corresponding word embeddings $\mathbf{x}^\tau = \{x_1^\tau, x_2^\tau, ..., x_m^\tau\}$ and $\mathbf{x} = \{x_1, x_2, ..., x_n\}$, the aim of target sentiment classification is to predict the sentiment polarity $y \in \{P, N, O\}$ of the sentence $\mathbf{w}$ over the target $\mathbf{w}^\tau$, where P, N and O denote "positive", "negative" and "neutral" sentiments respectively.

The architecture of the proposed Target-Specific Transformation Networks (TNet) is shown in Fig. 1. The bottom layer is a BiLSTM which transforms the input $\mathbf{x} = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^{n \times \dim_w}$ into the contextualized word representations $\mathbf{h}^{(0)} = \{h_1^{(0)}, h_2^{(0)}, ..., h_n^{(0)}\} \in \mathbb{R}^{n \times 2\dim_h}$ (i.e. hidden states of BiLSTM), where $\dim_w$ and $\dim_h$ denote the dimensions of the word embeddings and the hidden representations respectively. The middle part, the core part of our TNet, consists of L Context-Preserving Transformation (CPT) layers. The CPT layer incorporates the target information into the word representations via a novel Target-Specific Transformation (TST) component. CPT also contains a context-preserving mechanism, resembling identity mapping (He et al., 2016a,b) and highway connection (Srivastava et al., 2015a,b), allows preserving the context information and learning more abstract word-level features using a deep network. The top most part is a position-aware convolutional layer which first encodes positional relevance between a word and a target, and then extracts informative features for classification.

2.1 Bi-directional LSTM Layer

As observed in Lai et al. (2015), combining contextual information with word embeddings is an

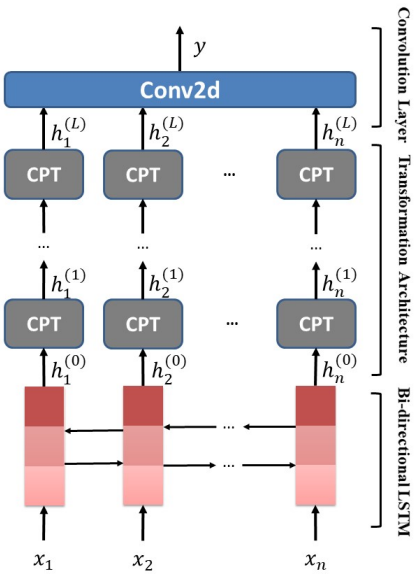

Figure 1: Architecture of TNet.

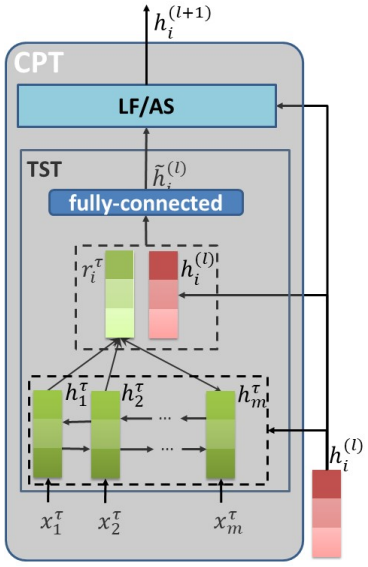

Figure 2: Details of a CPT module.

effective way to represent a word in convolution-based architectures. TNet also employs a BiLSTM to accumulate the context information for each word of the input sentence, i.e., the bottom part in Fig. 1. For simplicity and space issue, we denote the operation of an LSTM unit on x_i as $\text{LSTM}(x_i)$. Thus, the contextualized word representation $h_i^{(0)} \in \mathbb{R}^{2\text{dim}_h}$ is obtained as follows:

$$h_i^{(0)} = [\overrightarrow{\text{LSTM}}(x_i); \overleftarrow{\text{LSTM}}(x_i)], i \in [1, n]. \quad (1)$$

2.2 Context-Preserving Transformation

The above word-level representation has not considered the target information yet. Traditional attention-based approaches keep the word-level features static and aggregate them with weights as the final sentence representation. In contrast, as shown in the middle part in Fig. 1, we introduce multiple CPT layers and the detail of a single CPT is shown in Fig. 2. In each CPT layer, a tailor-made TST component that aims at better consolidating word representation and target representation is proposed. Moreover, we design a context-preserving mechanism enabling the learning of target-specific word representations in a deep neural architecture.

2.2.1 Target-Specific Transformation

TST component is depicted with the TST block in Fig. 2. The first task of TST is to generate the representation of the target. Previous methods (Chen et al., 2017; Liu and Zhang, 2017) average the embeddings of the target words as the target representation. This strategy may be inappropriate in some cases because different target words usually do not contribute equally. For example, in the target "*amd turin processor*", the word "processor" is more important than "amd" and "turin", because the sentiment is usually conveyed over the phrase head, i.e.,"processor", but seldom over modifiers (such as brand name "amd"). Ma et al. (2017) attempted to overcome this issue by measuring the importance score between each target word representation and the averaged sentence vector. However, it may be ineffective for sentences expressing multiple sentiments (e.g., *"Air has higher resolution but the fonts are small."*), because taking the average tends to neutralize different sentiments.

We propose to dynamically compute the importance of target words based on each sentence word rather than the whole sentence. We first employ another BiLSTM to obtain the target word representations $\mathbf{h}^\tau \in \mathbb{R}^{m \times 2\text{dim}_h}$:

$$h_j^\tau = [\overrightarrow{\text{LSTM}}(x_j^\tau); \overleftarrow{\text{LSTM}}(x_j^\tau)], j \in [1, m]. \quad (2)$$

Then, we dynamically associate them with each word w_i in the sentence to tailor-make target representation r_i^τ at the time step i:

$$r_i^\tau = \sum_{j=1}^{m} h_j^\tau * \mathcal{F}(h_i^{(l)}, h_j^\tau), \quad (3)$$

where the function $\mathcal{F}$ measures the relatedness between the j-th target word representation h_j^τ and

the i-th word-level representation $h_i^{(l)}$:

$$\mathcal{F}(h_i^{(l)}, h_j^{\tau}) = \frac{\exp\left(h_i^{(l)\top} h_j^{\tau}\right)}{\sum_{k=1}^{m} \exp\left(h_i^{(l)\top} h_k^{\tau}\right)}. \quad (4)$$

Finally, the concatenation of r_i^{τ} and $h_i^{(l)}$ is fed into a fully-connected layer to obtain the i-th target-specific word representation $\tilde{h}_i^{(l)}$:

$$\tilde{h}_i^{(l)} = g(W^{\tau}[h_i^{(l)} : r_i^{\tau}] + b^{\tau}), \quad (5)$$

where $g(*)$ is a non-linear activation function and ":" denotes vector concatenation. W^{τ} and b^{τ} are the weights of the layer.

2.2.2 Context-Preserving Mechanism

After the non-linear TST (see Eq. 5), the context information captured with contextualized representations from the BiLSTM layer will be lost since the mean and the variance of the features within the feature vector will be changed. To take advantage of the context information, which has been proved to be useful in (Lai et al., 2015), we investigate two strategies: Lossless Forwarding (LF) and Adaptive Scaling (AS), to pass the context information to each following layer, as depicted by the block "**LF/AS**" in Fig. 2. Accordingly, the model variants are named **TNet-LF** and **TNet-AS**.

Lossless Forwarding. This strategy preserves context information by directly feeding the features before the transformation to the next layer. Specifically, the input $h_i^{(l+1)}$ of the $(l+1)$-th CPT layer is formulated as:

$$h_i^{(l+1)} = h_i^{(l)} + \tilde{h}_i^{(l)}, i \in [1, n], l \in [0, L], \quad (6)$$

where $h_i^{(l)}$ is the input of the l-th layer and $\tilde{h}_i^{(l)}$ is the output of TST in this layer. We unfold the recursive form of Eq. 6 as follows:

$$h_i^{(l+1)} = h_i^{(0)} + \text{TST}(h_i^{(0)}) + \cdots + \text{TST}(h_i^{(l)}). \quad (7)$$

Here, we denote $\tilde{h}_i^{(l)}$ as $\text{TST}(h_i^{(l)})$. From Eq. 7, we can see that the output of each layer will contain the contextualized word representations (i.e., $h_i^{(0)}$), thus, the context information is encoded into the transformed features. We call this strategy "Lossless Forwarding" because the contextualized representations and the transformed representations (i.e., $\text{TST}(h_i^{(l)})$) are kept unchanged during the feature combination.

Adaptive Scaling. Lossless Forwarding introduces the context information by directly adding back the contextualized features to the transformed features, which raises a question: Can the weights of the input and the transformed features be adjusted dynamically? With this motivation, we propose another strategy, named "Adaptive Scaling". Similar to the gate mechanism in RNN variants (Jozefowicz et al., 2015), Adaptive Scaling introduces a gating function to control the passed proportions of the transformed features and the input features. The gate $\mathbf{t}^{(l)}$ as follows:

$$t_i^{(l)} = \sigma(W_{trans} h_i^{(l)} + b_{trans}), \quad (8)$$

where $t_i^{(l)}$ is the gate for the i-th input of the l-th CPT layer, and σ is the *sigmoid* activation function. Then we perform convex combination of $h_i^{(l)}$ and $\tilde{h}_i^{(l)}$ based on the gate:

$$h_i^{(l+1)} = t_i^{(l)} \odot \tilde{h}_i^{(l)} + (1 - t_i^{(l)}) \odot h_i^{(l)}. \quad (9)$$

Here, $\odot$ denotes element-wise multiplication. The non-recursive form of this equation is as follows (for clarity, we ignore the subscripts):

$$h^{(l+1)} = [\prod_{k=0}^{l}(1 - t^{(k)})] \odot h^{(0)}$$
$$+[t^{(0)} \prod_{k=1}^{l}(1 - t^{(k)})] \odot \text{TST}(h^{(0)}) + \cdots$$
$$+t^{(l-1)}(1 - t^{(l)}) \odot \text{TST}(h^{(l-1)}) + t^{(l)} \odot \text{TST}(h^{(l)}).$$

Thus, the context information is integrated in each upper layer and the proportions of the contextualized representations and the transformed representations are controlled by the computed gates in different transformation layers.

2.3 Convolutional Feature Extractor

Recall that the second issue that blocks CNN to perform well is that vanilla CNN may associate a target with unrelated general opinion words which are frequently used as modifiers for different targets across domains. For example, "***service***" in *"Great food but the service is dreadful"* may be associated with both "great" and "dreadful". To solve it, we adopt a proximity strategy, which is observed effective in (Chen et al., 2017; Li and Lam, 2017). The idea is a closer opinion word is more likely to be the actual modifier of the target.

		# Positive	# Negative	# Neutral
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 1: Statistics of datasets.

Specifically, we first calculate the position relevance v_i between the i-th word and the target[4]:

$$v_i = \begin{cases} 1 - \frac{(k+m-i)}{C} & i < k+m \\ 1 - \frac{i-k}{C} & k+m \leq i \leq n \\ 0 & i > n \end{cases} \quad (10)$$

where k is the index of the first target word, C is a pre-specified constant, and m is the length of the target $\mathbf{w}^\tau$. Then, we use v to help CNN locate the correct opinion w.r.t. the given target:

$$\hat{h}_i^{(l)} = h_i^{(l)} * v_i, i \in [1, n], l \in [1, L]. \quad (11)$$

Based on Eq. 10 and Eq. 11, the words close to the target will be highlighted and those far away will be downgraded. v is also applied on the intermediate output to introduce the position information into each CPT layer. Then we feed the weighted $\mathbf{h}^{(L)}$ to the convolutional layer, i.e., the top-most layer in Fig. 1, to generate the feature map $\mathbf{c} \in \mathbb{R}^{n-s+1}$ as follows:

$$c_i = \text{ReLU}(\boldsymbol{w}_{conv}^\top \mathbf{h}_{i:i+s-1}^{(L)} + b_{conv}), \quad (12)$$

where $\mathbf{h}_{i:i+s-1}^{(L)} \in \mathbb{R}^{s \cdot \dim_h}$ is the concatenated vector of $\hat{h}_i^{(L)}, \cdots, \hat{h}_{i+s-1}^{(L)}$, and s is the kernel size. $\boldsymbol{w}_{conv} \in \mathbb{R}^{s \cdot \dim_h}$ and $b_{conv} \in \mathbb{R}$ are learnable weights of the convolutional kernel. To capture the most informative features, we apply max pooling (Kim, 2014) and obtain the sentence representation $z \in \mathbb{R}^{n_k}$ by employing n_k kernels:

$$z = [\max(\mathbf{c}_1), \cdots, \max(\mathbf{c}_{n_k})]^\top. \quad (13)$$

Finally, we pass z to a fully connected layer for sentiment prediction:

$$p(y|\mathbf{w}^\tau, \mathbf{w}) = \text{Softmax}(W_f z + b_f). \quad (14)$$

where W_f and b_f are learnable parameters.

[4]As we perform sentence padding, it is possible that the index i is larger than the actual length n of the sentence.

3 Experiments

3.1 Experimental Setup

As shown in Table 1, we evaluate the proposed TNet on three benchmark datasets: LAPTOP and REST are from SemEval ABSA challenge (Pontiki et al., 2014), containing user reviews in laptop domain and restaurant domain respectively. We also remove a few examples having the "conflict label" as done in (Chen et al., 2017); TWITTER is built by Dong et al. (2014), containing twitter posts. All tokens are lowercased without removal of stop words, symbols or digits, and sentences are zero-padded to the length of the longest sentence in the dataset. Evaluation metrics are Accuracy and Macro-Averaged F1 where the latter is more appropriate for datasets with unbalanced classes. We also conduct pairwise t-test on both Accuracy and Macro-Averaged F1 to verify if the improvements over the compared models are reliable.

TNet is compared with the following methods.

- **SVM** (Kiritchenko et al., 2014): It is a traditional support vector machine based model with extensive feature engineering;
- **AdaRNN** (Dong et al., 2014): It learns the sentence representation toward target for sentiment prediction via semantic composition over dependency tree;
- **AE-LSTM**, and **ATAE-LSTM** (Wang et al., 2016): AE-LSTM is a simple LSTM model incorporating the target embedding as input, while ATAE-LSTM extends AE-LSTM with attention;
- **IAN** (Ma et al., 2017): IAN employs two LSTMs to learn the representations of the context and the target phrase interactively;
- **CNN-ASP**: It is a CNN-based model implemented by us which directly concatenates target representation to each word embedding;
- **TD-LSTM** (Tang et al., 2016a): It employs two LSTMs to model the left and right contexts of the target separately, then performs predictions based on concatenated context representations;
- **MemNet** (Tang et al., 2016b): It applies attention mechanism over the word embeddings multiple times and predicts sentiments

Hyper-params	TNet-LF			TNet-AS		
	LAPTOP	REST	TWITTER	LAPTOP	REST	TWITTER
$\dim_w$	300			300		
$\dim_h$	50			50		
dropout rates (p_{lstm}, p_{sent})	(0.3, 0.3)			(0.3, 0.3)		
L	2			2		
batch size	64	25	64	64	32	64
s	3			3		
n_k	50			100		
C	40.0			30.0		

Table 2: Settings of hyper-parameters.

based on the top-most sentence representations;

- **BILSTM-ATT-G** (Liu and Zhang, 2017): It models left and right contexts using two attention-based LSTMs and introduces gates to measure the importance of left context, right context, and the entire sentence for the prediction;

- **RAM** (Chen et al., 2017): RAM is a multi-layer architecture where each layer consists of attention-based aggregation of word features and a GRU cell to learn the sentence representation.

We run the released codes of TD-LSTM and BILSTM-ATT-G to generate results, since their papers only reported results on TWITTER. We also rerun MemNet on our datasets and evaluate it with both accuracy and Macro-Averaged F1.[5]

We use pre-trained GloVe vectors (Pennington et al., 2014) to initialize the word embeddings and the dimension is 300 (i.e., $\dim_w = 300$). For out-of-vocabulary words, we randomly sample their embeddings from the uniform distribution $\mathcal{U}(-0.25, 0.25)$, as done in (Kim, 2014). We only use one convolutional kernel size because it was observed that CNN with single optimal kernel size is comparable with CNN having multiple kernel sizes on small datasets (Zhang and Wallace, 2017). To alleviate overfitting, we apply dropout on the input word embeddings of the LSTM and the ultimate sentence representation z. All weight matrices are initialized with the uniform distribution $\mathcal{U}(-0.01, 0.01)$ and the biases are initialized as zeros. The training objective is cross-entropy, and Adam (Kingma and Ba, 2015) is adopted as the optimizer by following the learning rate and the decay rates in the original paper.

The hyper-parameters of TNet-LF and TNet-AS are listed in Table 2. Specifically, all hyper-parameters are tuned on 20% randomly held-out training data and the hyper-parameter collection producing the highest accuracy score is used for testing. Our model has comparable number of parameters compared to traditional LSTM-based models as we reuse parameters in the transformation layers and BiLSTM.[6]

3.2 Main Results

As shown in Table 3, both TNet-LF and TNet-AS consistently achieve the best performance on all datasets, which verifies the efficacy of our whole TNet model. Moreover, TNet can perform well for different kinds of user generated content, such as product reviews with relatively formal sentences in LAPTOP and REST, and tweets with more ungrammatical sentences in TWITTER. The reason is the CNN-based feature extractor arms TNet with more power to extract accurate features from ungrammatical sentences. Indeed, we can also observe that another CNN-based baseline, i.e., CNN-ASP implemented by us, also obtains good results on TWITTER.

On the other hand, the performance of those comparison methods is mostly unstable. For the tweet in TWITTER, the competitive BILSTM-ATT-G and RAM cannot perform as effective as they do for the reviews in LAPTOP and REST, due to the fact that they are heavily rooted in LSTMs and the ungrammatical sentences hinder their ca-

[5]The codes of TD-LSTM/MemNet and BILSTM-ATT-G are available at: http://ir.hit.edu.cn/~dytang and http://leoncrashcode.github.io. Note that MemNet was only evaluated with accuracy.

[6]All experiments are conducted on a single NVIDIA GTX 1080. The prediction cost of a sentence is about 2 ms.

	Models	LAPTOP		REST		TWITTER	
		ACC	Macro-F1	ACC	Macro-F1	ACC	Macro-F1
Baselines	SVM	70.49♮	-	80.16♮	-	63.40*	63.30*
	AdaRNN	-	-	-	-	66.30♮	65.90♮
	AE-LSTM	68.90♮	-	76.60♮	-	-	-
	ATAE-LSTM	68.70♮	-	77.20♮	-	-	-
	IAN	72.10♮	-	78.60♮	-	-	-
	CNN-ASP	72.46	65.31	77.82	65.11	73.27	71.77
	TD-LSTM	71.83	68.43	78.00	66.73	66.62	64.01
	MemNet	70.33	64.09	78.16	65.83	68.50	66.91
	BILSTM-ATT-G	74.37	69.90	80.38	70.78	72.70	70.84
	RAM	75.01	70.51	79.79	68.86	71.88	70.33
CPT Alternatives	LSTM-ATT-CNN	73.37	68.03	78.95	68.71	70.09	67.68
	LSTM-FC-CNN-LF	75.59	70.60	80.41	70.23	73.70	72.82
	LSTM-FC-CNN-AS	75.78	70.72	80.23	70.06	74.28	72.60
Ablated TNet	TNet w/o transformation	73.30	68.25	78.90	65.86	72.10	70.57
	TNet w/o context	73.91	68.87	80.07	69.01	74.51	73.05
	TNet-LF w/o position	75.13	70.63	79.86	69.69	73.83	72.49
	TNet-AS w/o position	75.27	70.03	79.79	69.78	73.84	72.47
TNet variants	TNet-LF	76.01†,‡	71.47†,‡	**80.79**†,‡	70.84‡	74.68†,‡	73.36†,‡
	TNet-AS	**76.54**†,‡	**71.75**†,‡	80.69†,‡	**71.27**†,‡	**74.97**†,‡	**73.60**†,‡

Table 3: Experimental results (%). The results with symbol"♮" are retrieved from the original papers, and those starred (∗) one are from Dong et al. (2014). The marker † refers to p-value < 0.01 when comparing with BILSTM-ATT-G, while the marker ‡ refers to p-value < 0.01 when comparing with RAM.

pability in capturing the context features. Another difficulty caused by the ungrammatical sentences is that the dependency parsing might be error-prone, which will affect those methods such as AdaRNN using dependency information.

From the above observations and analysis, some takeaway message for the task of target sentiment classification could be:

- LSTM-based models relying on sequential information can perform well for formal sentences by capturing more useful context features;
- For ungrammatical text, CNN-based models may have some advantages because CNN aims to extract the most informative n-gram features and is thus less sensitive to informal texts without strong sequential patterns.

3.3 Performance of Ablated TNet

To investigate the impact of each component such as deep transformation, context-preserving mechanism, and positional relevance, we perform comparison between the full TNet models and its ablations (the third group in Table 3). After removing the deep transformation (i.e., the techniques introduced in Section 2.2), both TNet-LF and TNet-AS are reduced to TNet w/o transformation (where position relevance is kept), and their results in both accuracy and F1 measure are incomparable with those of TNet. It shows that the integration of target information into the word-level representations is crucial for good performance.

Comparing the results of TNet and TNet w/o context (where TST and position relevance are kept), we observe that the performance of TNet w/o context drops significantly on LAPTOP and REST[7], while on TWITTER, TNet w/o context performs very competitive (p-values with TNet-LF and TNet-AS are 0.066 and 0.053 respectively for Accuracy). Again, we could attribute this phenomenon to the ungrammatical user generated content of twitter, because the context-preserving component becomes less important for such data. TNet w/o context performs consistently better than TNet w/o transformation, which verifies the efficacy of the target specific transformation (TST), before applying context-preserving.

As for the position information, we conduct statistical t-test between TNet-LF/AS and TNet-LF/AS w/o position together with performance comparison. All of the produced p-values are less than 0.05, suggesting that the improvements brought in by position information are significant.

[7]Without specification, the significance level is set to 0.05.

3.4 CPT versus Alternatives

The next interesting question is what if we replace the transformation module (i.e., the CPT layers in Fig.1) of TNet with other commonly-used components? We investigate two alternatives: attention mechanism and fully-connected (FC) layer, resulting in three pipelines as shown in the second group of Table 3 (position relevance is kept for them).

LSTM-ATT-CNN applies attention as the alternative[8], and it does not need the context-preserving mechanism. It performs unexceptionally worse than the TNet variants. We are surprised that LSTM-ATT-CNN is even worse than TNet w/o transformation (a pipeline simply removing the transformation module) on TWITTER. More concretely, applying attention results in negative effect on TWITTER, which is consistent with the observation that all those attention-based state-of-the-art methods (i.e., TD-LSTM, MemNet, BILSTM-ATT-G, and RAM) cannot perform well on TWITTER.

LSTM-FC-CNN-LF and LSTM-FC-CNN-AS are built by applying FC layer to replace TST and keeping the context-preserving mechanism (i.e., LF and AS). Specifically, the concatenation of word representation and the averaged target vector is fed to the FC layer to obtain target-specific features. Note that LSTM-FC-CNN-LF/AS are equivalent to TNet-LF/AS when processing single-word targets (see Eq. 3). They obtain competitive results on all datasets: comparable with or better than the state-of-the-art methods. The TNet variants can still outperform LSTM-FC-CNN-LF/AS with significant gaps, e.g., on LAPTOP and REST, the accuracy gaps between TNet-LF and LSTM-FC-CNN-LF are 0.42% ($p < 0.03$) and 0.38% ($p < 0.04$) respectively.

3.5 Impact of CPT Layer Number

As our TNet involves multiple CPT layers, we investigate the effect of the layer number L. Specifically, we conduct experiments on the held-out training data of LAPTOP and vary L from 2 to 10, increased by 2. The cases L=1 and L=15 are also included. The results are illustrated in Figure 3. We can see that both TNet-LF and TNet-AS achieve the best results when L=2. While increasing L, the performance is basically becoming worse. For large L, the performance of TNet-AS generally becomes more sensitive, it is probably because AS involves extra parameters (see Eq 9) that increase the training difficulty.

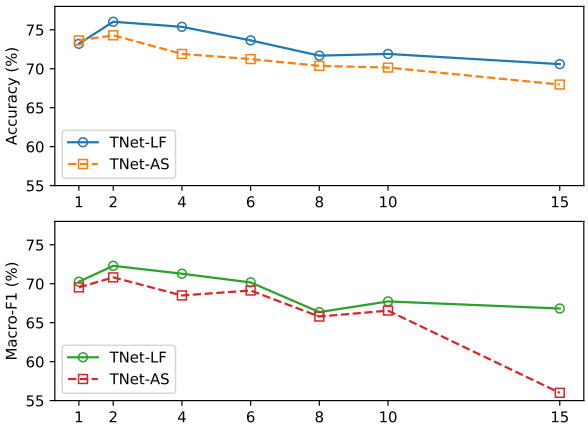

Figure 3: Effect of L.

3.6 Case Study

Table 4 shows some sample cases. The input targets are wrapped in the brackets with true labels given as subscripts. The notations P, N and O in the table represent positive, negative and neutral respectively. For each sentence, we underline the target with a particular color, and the text of its corresponding most informative n-gram feature[9] captured by TNet-AS (TNet-LF captures very similar features) is in the same color (so color printing is preferred). For example, for the target "resolution" in the first sentence, the captured feature is "Air has higher". Note that as discussed above, the CNN layer of TNet captures such features with the size-three kernels, so that the features are trigrams. Each of the last features of the second and seventh sentences contains a padding token, which is not shown.

Our TNet variants can predict target sentiment more accurately than RAM and BILSTM-ATT-G in the transitional sentences such as the first sentence by capturing correct trigram features. For the third sentence, its second and third most informative trigrams are "100% . PAD" and "' s not", being used together with "features make up", our models can make correct predictions. Moreover, TNet can still make correct prediction when the explicit opinion is target-specific. For example,

[8]We tried different attention mechanisms and report the best one here, namely, *dot* attention (Luong et al., 2015).

[9]For each convolutional filter, only one n-gram feature in the feature map will be kept after the max pooling. Among those from different filters, the n-gram with the highest frequency will be regarded as the most informative n-gram w.r.t. the given target.

Sentence	BILSTM-ATT-G	RAM	TNet-LF	TNet-AS
1. Air has higher **[resolution]**$_{\text{P}}$ but the **[fonts]**$_{\text{N}}$ are small .	(N$^{\text{✗}}$, N)	(N$^{\text{✗}}$, N)	(P, N)	(P, N)
2. Great **[food]**$_{\text{P}}$ but the **[service]**$_{\text{N}}$ is dreadful .	(P, N)	(P, N)	(P, N)	(P, N)
3. Sure it ' s not light and slim but the **[features]**$_{\text{P}}$ make up for it 100% .	N$^{\text{✗}}$	N$^{\text{✗}}$	P	P
4. Not only did they have amazing , **[sandwiches]**$_{\text{P}}$, **[soup]**$_{\text{P}}$, **[pizza]**$_{\text{P}}$ etc , but their **[homemade sorbets]**$_{\text{P}}$ are out of this world !	(P, O$^{\text{✗}}$, O$^{\text{✗}}$, P)	(P, P, O$^{\text{✗}}$, P)	(P, P, P, P)	(P, P, P, P)
5. **[startup times]**$_{\text{N}}$ are incredibly long : over two minutes .	P$^{\text{✗}}$	P$^{\text{✗}}$	N	N
6. I am pleased with the fast **[log on]**$_{\text{P}}$, speedy **[wifi connection]**$_{\text{P}}$ and the long **[battery life]**$_{\text{P}}$ ($>$ 6 hrs) .	(P, P, P)	(P, P, P)	(P, P, P)	(P, P, P)
7. The **[staff]**$_{\text{N}}$ should be a bit more friendly .	P$^{\text{✗}}$	P$^{\text{✗}}$	P$^{\text{✗}}$	P$^{\text{✗}}$

Table 4: Example predictions, color printing is preferred. The input targets are wrapped in brackets with the true labels given as subscripts. ✗ indicates incorrect prediction.

"long" in the fifth sentence is negative for "startup time", while it could be positive for other targets such as "battery life" in the sixth sentence. The sentiment of target-specific opinion word is conditioned on the given target. Our TNet variants, armed with the word-level feature transformation w.r.t. the target, is capable of handling such case.

We also find that all these models cannot give correct prediction for the last sentence, a commonly used subjunctive style. In this case, the difficulty of prediction does not come from the detection of explicit opinion words but the inference based on implicit semantics, which is still quite challenging for neural network models.

4 Related Work

Apart from sentence level sentiment classification (Kim, 2014; Shi et al., 2018), aspect/target level sentiment classification is also an important research topic in the field of sentiment analysis. The early methods mostly adopted supervised learning approach with extensive hand-coded features (Blair-Goldensohn et al., 2008; Titov and McDonald, 2008; Yu et al., 2011; Jiang et al., 2011; Kiritchenko et al., 2014; Wagner et al., 2014; Vo and Zhang, 2015), and they fail to model the semantic relatedness between a target and its context which is critical for target sentiment analysis. Dong et al. (2014) incorporate the target information into the feature learning using dependency trees. As observed in previous works, the performance heavily relies on the quality of dependency parsing. Tang et al. (2016a) propose to split the context into two parts and associate target with contextual features separately. Similar to (Tang et al., 2016a), Zhang et al. (2016) develop a three-way gated neural network to model the interaction between the target and its surrounding contexts. Despite the advantages of jointly modeling target and context, they are not capable of capturing long-range information when some critical context information is far from the target. To overcome this limitation, researchers bring in the attention mechanism to model target-context association (Tang et al., 2016a,b; Wang et al., 2016; Yang et al., 2017; Liu and Zhang, 2017; Ma et al., 2017; Chen et al., 2017; Zhang et al., 2017; Tay et al., 2017). Compared with these methods, our TNet avoids using attention for feature extraction so as to alleviate the attended noise.

5 Conclusions

We re-examine the drawbacks of attention mechanism for target sentiment classification, and also investigate the obstacles that hinder CNN-based models to perform well for this task. Our TNet model is carefully designed to solve these issues. Specifically, we propose target specific transformation component to better integrate target information into the word representation. Moreover, we employ CNN as the feature extractor for this classification problem, and rely on the context-preserving and position relevance mechanisms to maintain the advantages of previous LSTM-based models. The performance of TNet consistently dominates previous state-of-the-art methods on different types of data. The ablation studies show the efficacy of its different modules, and thus verify the rationality of TNet's architecture.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. In *Proceedings of ICLR*.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A Reis, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *WWW workshop on NLP in the information explosion era*, pages 339–348.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of EMNLP*, pages 463–472.

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 *Proceedings of ACL*, pages 49–54.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity mappings in deep residual networks. In *Proceedings of ECCV*, pages 630–645.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of ACL*, pages 151–160.

Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Proceedings of NIPS*, pages 919–927.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of ICML*, pages 2342–2350.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, pages 1746–1751.

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

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of SemEval*, pages 437–442.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of AAAI*, volume 333, pages 2267–2273.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Xin Li and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *Proceedings of EMNLP*, pages 2876–2882.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Proceedings of EACL*, pages 572–577.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*, pages 1412–1421.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of IJCAI*, pages 4068–4074.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of SemEval*, pages 27–35.

Bei Shi, Zihao Fu, Lidong Bing, and Wai Lam. 2018. Learning domain-sensitive and sentiment-aware word embeddings. In *Proceedings of ACL*.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015a. Training very deep networks. In *Proceedings of NIPS*, pages 2377–2385.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015b. Highway networks. *arXiv preprint arXiv:1505.00387*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of EMNLP*, pages 214–224.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. *arXiv preprint arXiv:1712.05403*.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of WWW*, pages 111–120.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of IJCAI*, pages 1347–1353.

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 *Proceedings of SemEval*, pages 223–229.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of EMNLP*, pages 606–615.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML*, pages 2048–2057.

Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. 2017. Attention based lstm for target dependent sentiment classification. In *Proceedings of AAAI*, pages 5013–5014.

Jianxing Yu, Zheng-Jun Zha, Meng Wang, and Tat-Seng Chua. 2011. Aspect ranking: identifying important product aspects from online consumer reviews. In *Proceedings of ACL*, pages 1496–1505.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Proceedings of AAAI*, pages 3087–3093.

Ye Zhang and Byron Wallace. 2017. A sensitivity analysis of (and practitioners guide to) convolutional neural networks for sentence classification. In *Proceedings of IJCNLP*, pages 253–263.

Yue Zhang, Zhenghua Li, Jun Lang, Qingrong Xia, and Min Zhang. 2017. Dependency parsing with partial annotations: An empirical comparison. In *Proceedings of IJCNLP*, pages 49–58.