Target Attention Network for Targeted Sentiment Analysis

Po-Chih Huang
Dept. of Computer Science
National Taiwan University
Taipei, Taiwan
brianhuang1019@gmail.com

Han-Hao Chen
Dept. of Computer Science
National Taiwan University
Taipei, Taiwan
m516825@gmail.com

Pu-Jen Cheng
Dept. of Computer Science
National Taiwan University
Taipei, Taiwan
pjcheng@csie.ntu.edu.tw

Abstract

We propose a novel target attention network (TAN) to identify the sentiment of opinion targets in a review or a twitter. Unlike previous works which represent the target in an averaging manner, we apply target attention to focus on more relevant parts of the target, which shows benefits for later sentiment classification. We also introduce a novel target-aware position embedding, which directly models the location relation between the target and its context to provide distinct information from semantic meanings for sequence encoder. Extensive experiments demonstrate the robustness of our findings. The experimental results show that our model consistently outperforms the state-of-the-art methods on four public benchmarks.

1 Introduction

Targeted sentiment analysis is an entity-level sentiment analysis, which aims to identify the sentiment polarity of specific opinion targets in a sentence. For example, in sentence *"The price is reasonable although the food quality is poor."*, the sentiment polarity of target *"price"* is positive, while the sentiment polarity of target *"food quality"* is negative. An important issue in targeted sentiment analysis is to model the complicated interaction between a target and its context, which can involve syntactic structures and semantic intentions such as negations, intensities, and even sarcasm. In the former example, context *"reasonable"* has higher distinguishability to identify the sentiment polarity for target *"price"*.

Some earlier works (Dong et al., 2014; Tang et al., 2016a; Zhang et al., 2016) use neural networks to encode a sequence. Such models may suffer poor performance when informative context words are far from the target, causing useful information to vanish after the long sequence encoding process. Some works (Tang et al., 2016b; Yang et al., 2017; Liu and Zhang, 2017; Ma et al., 2017; Chen et al., 2017) mitigate this problem by applying attention mechanism to focus on more related parts of the sequence. The common idea is to average embeddings over target words as the query, traverse its context with the query, and count attention weight for each context word.

The main flaw of these attention-based methods is the target representation issue. Since a target can contain multiple words, if you simply average all target words, meaningful words in a target may be diluted by a large number of meaningless words especially for longer targets, which may lead to poor attention results. We propose a novel target attention network to alleviate such problem. We introduce two types of target attention to integrate all words in a target in non-linear ways, rather than linear combination (average words) as previous works do. We believe that a deliberate representation of a target can boost performance.

We also find it beneficial to consider the positional information as input features. Even if conventional sequence encoders such as RNN can maintain the temporal order, the encoder knows nothing about the location of the target so the direct relation between the target and each context word is hard to model. We introduce a novel target-aware position embedding to represent the location relation between a target and its context. Combined with position embedding, the sequence encoder can consider more comprehensive features simultaneously during the encoding process.

We evaluate proposed approach on four public datasets. The first two are restaurant and laptop reviews from SemEval 2014 (Pontiki et al., 2014), the remaining two are collected from twitter by previous works (Dong et al., 2014; Zhang et al., 2016). Extrinsic comparisons with previous works demonstrate that our model outperforms all state-

of-the-art methods on all four datasets, no wonder the dataset is well-structured (reviews) or highly irregular (twitter). Intrinsic experiments show that our models with target attention are consistently superior than the ones without target attention, and the results of target attention can make a significant impact on later context attention. Experimental results also show that complex opinions for the target can be captured with position embedding.

2 Related Work

Targeted sentiment analysis is a fine-grained classification task. It requires different solutions from traditional sentiment analysis (Kim, 2014; Tang et al., 2015; Zhang et al., 2015; Yang et al., 2016) which predicts sentiment polarity for whole sentence or whole document.

The earliest tries on targeted sentiment analysis extract target-dependent features (Jiang et al., 2011; Kiritchenko et al., 2014; Wagner et al., 2014; Vo and Zhang, 2015), then use a classifier such as SVM to predict sentiment. These features need to be carefully designed by experts and exploit external resources like sentiment lexicons.

Neural networks based methods attract attention for researchers because of the ability to extract features automatically and learn complex representation from data. AdaRNN (Dong et al., 2014) leverages dependency parsing tree and uses recursive neural networks for targeted sentiment analysis. TC-LSTM (Tang et al., 2016a) uses two separate LSTMs to encode left and right context. Since the sentiment may be dominated by either context, GRNN (Zhang et al., 2016) uses a LSTM-like gate mechanism to better concatenate both contexts.

However, the captured features are likely to be lost when the the informative word is far from target (Cho et al., 2014; Chen et al., 2017). Attention-based (Bahdanau et al., 2014; Hermann et al., 2015; Xu et al., 2015; Chorowski et al., 2015; Luong et al., 2015) and memory network (Weston et al., 2015; Sukhbaatar et al., 2015) based methods can utilize history information to focus on more related parts of the sequence. MemNet (Tang et al., 2016b) executes multiple hops on word embeddings, and focuses on different parts at each hop. AB-LSTM (Yang et al., 2017) uses dot product and bilinear term to count attention weight. BILSTM-ATT-G (Liu and Zhang, 2017) inherits the idea from GRNN, which uses gated mechanism to concatenate attention productions from both contexts. IAN (Ma et al., 2017) investigates the interactions between a target and its context, they use the average of the target and context to count attention weights for each other. RAM (Chen et al., 2017) is similar to MemNet, while the memory here is the hidden states of a LSTM, rather than word embeddings. They use a GRU to execute multiple hops in memory networks.

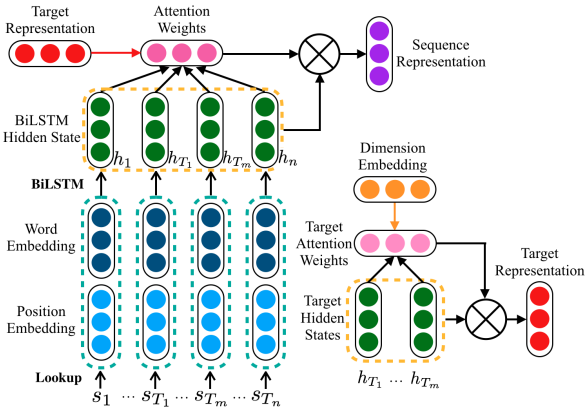

Figure 1: Calculating sequence representation in TAN. $\{s_1, ..., s_{T_1}, ..., s_{T_m}, ..., s_n\}$ is the input sequence with n words in length, $\{s_{T_1}, ..., s_{T_m}\}$ is the target with m words. $\{h_1, ..., h_{T_1}, ..., h_{T_m}, ..., h_n\}$ are hidden states of BiLSTM. We show FM style target attention here.

3 Target Attention Network

In this section, we first give an overview of the task. Afterwards, we will describe our lookup methods for input features, sequence encoding process, and how to apply attention on the target and sequence. Finally, we will show the training and inference processes. The overall architecture of our system is shown in Figure 1.

Let's assume the current input sequence is $seq = \{s_1, ..., s_{T_1-1}, s_{T_1}, ..., s_{T_m}, s_{T_m+1}, ..., s_n\}$, meaning the input sequence has n words in length, and the target $T = \{s_{T_1}, ..., s_{T_m}\}$ has m words in length. The goal of the task is to predict the sentiment polarity of the target T toward sequence seq. For example, in the sentence *"The price is reasonable although the food quality is poor."*, target *"price"* has positive polarity, while the polarity of target *"food quality"* is negative. Other than the methods mentioned in the related work, we extract novel target-aware input features from seq, and utilize novel target attention networks to focus on important parts of target T.

3.1 Target-Aware Input

Word Embedding. When dealing with natural language processing tasks, it is helpful to use a low-dimensional and continuous real-valued vector to represent each word (Mikolov et al., 2013; Pennington et al., 2014). Let $W^e \in \mathbb{R}^{d_w \times |V|}$ be the word embedding lookup table, where d_w is the dimension of a word vector, and $|V|$ is the vocabulary size. For each input sequence $\{s_1, ..., s_{T_1-1}, s_{T_1}, ..., s_{T_m}, s_{T_m+1}, ..., s_n\}$, we lookup W^e, and return a word vector sequence $\{w_1, ..., w_{T_1-1}, w_{T_1}, ..., w_{Tm}, w_{T_m+1}, ..., w_n\}$, where w_i is the word vector of s_i, and $w_i \in \mathbb{R}^{d_w}$. If the index of s_i in the vocabulary is idx_i, we can make an one-hot vector e_i of size $|V|$ which has value 1 at index idx_i and 0's at other indices. The lookup method for s_i is through a matrix multiplication:

$$w_i = W^e e_i \in \mathbb{R}^{d_w}$$

Position Embedding. Word embedding is good but sometimes not enough for our task. If we only consider word embedding in our model, the model will never know where exactly the target T is located. One trivial solution is to add a special *begin of target* token before s_{T_1} and an *end of target* token after s_{T_m}. While doing this, models still know nothing about the position of the target until it read the special tokens. Our method is to maintain another position embedding lookup table $P^e \in \mathbb{R}^{d_p \times 2|L|}$, where d_p is the dimension of a position vector, and $|L|$ is maximum length of all input sequences. Just like what we do with word embedding, for each input sequence $\{s_1, ..., s_{T_1-1}, s_{T_1}, ..., s_{T_m}, s_{T_m+1}, ..., s_n\}$, we lookup P^e, and return a position vector sequence $\{p_1, ..., p_{T_1-1}, p_{T_1}, ..., p_{T_m}, p_{T_m+1}, ..., p_n\}$ where p_i is the position vector of w_i, and $p_i \in \mathbb{R}^{2*d_p}$. The purpose of position embedding is to represent the location relation between s_i and target T. That is to count the relative location loc_{i1} from s_i to s_{T_1} and another relative location loc_{i2} from s_i to s_{T_m} by:

$$loc_{i1} = i - T_1$$

$$loc_{i2} = i - T_m$$

Each relative location can be mapped into an independent column of P^e. We count the indices of the relative locations loc_{i1} and loc_{i2}, respectively, by:

$$idx_{i1} = loc_{i1} + |L|$$

$$idx_{i2} = loc_{i2} + |L|$$

To do the lookup process, we will make two one-hot vectors of size $2 * |L|$. The first vector e_{i1} has value 1 at index idx_{i1} and 0's at other indices. The second vector e_{i2} has value 1 at index idx_{i2} and 0's at other indices. The lookup method for s_i is:

$$p_{i1} = P^e e_{i1} \in \mathbb{R}^{d_p}$$

$$p_{i2} = P^e e_{i2} \in \mathbb{R}^{d_p}$$

$$p_i = [p_{i1}; p_{i2}] \in \mathbb{R}^{2*d_p}$$

Notation $[\cdot;\cdot]$ means to concatenate two vectors.

Let's do some summary, for an input sequence $seq = \{s_1, ..., s_{T_1-1}, s_{T_1}, ..., s_{T_m}, s_{T_m+1}, ..., s_n\}$, we will lookup W^e and P^e, and return a vector sequence $\{v_1, ..., v_{T_1-1}, v_{T_1}, ..., v_{T_m}, v_{T_m+1}, ..., v_n\}$ where v_i is the concatenation of word embedding and position embedding:

$$v_i = [w_i; p_i] \in \mathbb{R}^{d_w+2*d_p}$$

So v_i can contain semantic information and target-aware information simultaneously. We let W^e and P^e be learned in the training process.

3.2 Sequence Attention Network

For many sequence modeling tasks, it is beneficial to have access to future as well as past context (Karpathy et al., 2015; Zhai et al., 2016). One solution is the bidirectional mechanism (Schuster and Paliwal, 1997; Graves and Schmidhuber, 2005; Graves et al., 2013). Bidirectional LSTM extends standard LSTM by introducing a backward LSTM, where its input sequence is in reversed order $seq_R = \{s_n, ..., s_1\}$. We adopt bidirectional LSTM in our work. For the i^{th} word in the input sequence, we count its forward hidden state $\overrightarrow{h_i}$ by standard LSTM, and its backward hidden state $\overleftarrow{h_i}$ by backward LSTM. The final hidden state h_i is the concatenation of the forward and the backward hidden states:

$$h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}] \in \mathbb{R}^B$$

After counting all hidden states from BiLSTM $H = \{h_1, ..., h_{T_1-1}, h_{T_1}, ..., h_{T_m}, h_{T_m+1}, ..., h_n\}$, we want to calculate a fixed size sequence representation r. Attention network is to give each candidate a weight that tells how important it is to a given condition, and count the weighted sum of the candidates. Here, vectors in H are our candidates, target T is our condition, and the weighted

sum of each vector in H is our sequence representation r. With attention, target T can focus on more important parts of the context. For an input sequence *"the food is good, but the service is bad"*, target *"food"* and target *"service"* will focus on different parts of the sequence, and generate different sequence representation r. We adopt the concatenation manner (Bahdanau et al., 2014) to count the attention scores, which considers both candidates and the given condition by:

$$\beta_i = U_a^T tanh(W_a[h_i; t_r] + b_a),$$

where h_i is the hidden state of the i^{th} word, β_i is its attention score, and U_a, W_a and b_a are attention parameters to learn. t_r is the target representation. We will describe how to get t_r with target attention networks in Section 3.3. After counting $\{\beta_1, ..., \beta_n\}$, we use a $softmax$ function to make them sum of 1:

$$\alpha_i = \frac{exp(\beta_i)}{\sum_{j=1}^{n} exp(\beta_j)}$$

Lastly, we count the weighted sum of H by:

$$s_H = \sum_{i=1}^{n} \alpha_i h_i \in \mathbb{R}^B$$

It is beneficial to split the sequence according to the target T in targeted sentiment analysis (Vo and Zhang, 2015; Tang et al., 2016a; Zhang et al., 2016; Liu and Zhang, 2017). We split H to get left context $H_L = \{h_1, ..., h_{T_1-1}\}$ and right context $H_R = \{h_{T_m+1}, ..., h_n\}$. H, H_L, and H_R will have its own attention network and parameters. We can get three weighted sum s_H, s_{H_L}, and s_{H_R} from these attention networks, and the sequence representation r is the concatenation of them:

$$r = [s_H; s_{H_L}; s_{H_R}]$$

3.3 Target Attention Network

We will describe how to get the target representation t_r here. We have hidden states of target words $H_T = \{h_{T_1}, ..., h_{T_m}\} \in \mathbb{R}^{B \times m}$ produced from BiLSTM, and want to get a fixed size t_r from H_T, where target length m varies from sequence to sequence. Unlike the models mentioned in related work which use the average of H_T as t_r, we let the model to focus on more important parts of target T by applying attention on H_T. We introduce two different kinds of target attention network.

Fully-Connected Layers (FC). First type of the target attention is to extract feature from H_T itself. We count the attention weight vector through two fully-connected layers with $tanh$ non-linear activation function between them, and H_T as input:

$$\beta = W_2^T tanh(W_1 H_T + b_1) + b_2,$$

where $\beta \in \mathbb{R}^m$ is the attention weight vector, $W_1 \in \mathbb{R}^{l \times B}, b_1 \in \mathbb{R}^l, W_2 \in \mathbb{R}^l, b_2 \in \mathbb{R}$ are the parameters to learn, and l is the output size of the first layer. Then we can count the normalized attention weights $\alpha \in \mathbb{R}^m$ by a $softmax$ function:

$$\alpha_i = \frac{exp(\beta_i)}{\sum_{j=1}^{m} exp(\beta_j)}$$

The target representation t_r can be counted by:

$$t_r = H_T \alpha \in \mathbb{R}^B$$

Factorization Machine (FM). Second type of the target attention utilizes the factorization machine. FM is a popular method in recommender systems (Rendle, 2010). It extracts interactions between variables using factorized parameters. For an input vector $x \in \mathbb{R}^d$, FM not only applies linear regression to x but models the relationship between each dimension in x. The formula of FM of degree $deg = 2$ is:

$$FM(x) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \langle v_i, v_j \rangle x_i x_j,$$

where the learned parameters are bias $w_0 \in \mathbb{R}$, linear regression weight vector $w \in \mathbb{R}^d$, and the dimension embedding lookup table $V \in \mathbb{R}^{k \times d}$. A column v_i within V describes the i^{th} variable with k factors. Notation $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k:

$$\langle v_i, v_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$$

To get v_i, we lookup V with an one-hot vector $e_i \in \mathbb{R}^d$, which has value 1 at index i and 0's at other indices:

$$v_i = V e_i \in \mathbb{R}^k$$

We use the prediction $FM(\cdot)$ as our target attention weights, so each target hidden state h_{T_i} will count its weight by:

$$\beta_{T_i} = FM(h_{T_i})$$

Then a $softmax$ function is used to make the summation of attention weights equal to 1:

$$\alpha_{T_i} = \frac{exp(\beta_{T_i})}{\sum_{j=1}^{m} exp(\beta_{T_j})}$$

Finally, the target representation t_r is counted by:

$$t_r = \sum_{i=1}^{m} \alpha_{T_i} h_{T_i} \in \mathbb{R}^B$$

3.4 Model Training and Inferring

In Section 3.2, we describe the method to get sequence representation r. We use a $softmax$ classifier to predict probability distribution $p \in \mathbb{R}^C$ of C sentiment categories. The classifier takes r as input, and W_p and b_p are parameters to learn:

$$p = softmax(W_p^T r + b_p) \in \mathbb{R}^C$$

The model is trained in a supervised manner by minimizing the cross entropy between prediction and true label, whose loss function is given below:

$$L(\theta) = -\sum_{(x,y)\in D} \sum_{c \in C} y^c log(p^c),$$

where θ are all learned parameters in the model, D means all training instances, each x contains an input sequence seq and a target T to count p, and each $y \in \mathbb{R}^C$ is an one-hot vector where index of true sentiment class is 1. During inference, we choose the index with maximum value in p as predicted result $\hat{y}$:

$$\hat{y} = \arg\max p$$

4 Experiments

4.1 Experimental Setting

We conduct experiments on four public datasets, as shown in Table 1. The first two are from SemEval 2014 (Pontiki et al., 2014), containing reviews of restaurant domain (Restaurant) and laptop domain (Laptop), which are widely used in previous works. The third one is a collection of tweets (Twitter), collected by Dong (2014). The last one is also a twitter collection (Z-Dataset), collected by Zhang (2016), which includes the MPQA corpus and the Mitchell's (2013) corpus.

Same as previous works, we remove a few examples having the "conflict" sentiment class for the first two datasets. We use 300-dimensional word vectors[1] pretrained by GloVe (Pennington et al., 2014) as the initial word embedding for the first two datasets, and 200-dimensional word vectors[2] for the last two datasets. We compare our model with the following methods:

Dataset		#Positive	#Neutral	#Negative
Restaurant	Train	2164	633	805
	Test	728	196	196
Laptop	Train	987	460	866
	Test	341	169	128
Twitter	Train	1561	3127	1560
	Test	173	346	173
Z-Dataset	Train	2416	4689	2384
	Dev	255	509	272
	Test	294	581	295

Table 1: Details of the experimental datasets

Feature Engineering Based. Featured-SVM (Kiritchenko et al., 2014) extracts target features, surface features, lexicon features and parsing features. A SVM classifier is used to classify the sentiment polarity. Target-dep (Vo and Zhang, 2015) uses five pooling functions to extract features for a given target. They firstly split a sentence into three sections, including the target, its left contexts, and its right contexts.

Neural Network Based. AdaRNN (Dong et al., 2014) leverages the results from dependency parsing trees. Recursive neural networks are used for the final prediction. TC-LSTM (Tang et al., 2016a) uses a forward LSTM and a backward LSTM to encode the information. They concatenate the last hidden states of both LSTMs for the sentiment prediction. GRNN (Zhang et al., 2016) also encodes the input sequence with a bidirectional recurrent neural network. They use a gated mechanism to concatenate the target reprsentation and the representation of its context.

Attention-Based. MemNet (Tang et al., 2016b) applies multiple time attentions on the word embedding. The output of the last attention is used for the sentiment prediction. AB-LSTM (Yang et al., 2017) compares two types of attention scores: dot products and bilinear terms. BILSTM-ATT-G (Liu and Zhang, 2017) applies attention to the left context and the right context according to a target. They combine the outputs of multiple attentions with a gated mechanism similar to GRNN. IAN (Ma et al., 2017) averages a target and its context to count the attention weights for each other. RAM (Chen et al., 2017) borrows the idea from Mem-

[1] glove.42B.300d
[2] glove.twitter.27B

	Restaurant		Laptop		Twitter		Z-dataset	
Model	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Featured-SVM	0.8016	-	0.7049	-	0.6340†	0.6330†	-	-
AdaRNN	-	-	-	-	0.6630	0.6590	-	-
Target-dep	-	-	-	-	0.7110	0.6990	-	-
TC-LSTM	0.7800†	0.6673†	0.7183†	0.6843†	0.7150	0.6950	-	-
MemNet	0.8095	-	0.7237	-	0.6850†	0.6691†	-	-
GRNN	-	-	-	-	-	-	0.7196	0.6955
AB-LSTM	-	-	-	-	0.7260	0.7220	-	-
BILSTM-ATT-G	-	-	-	-	0.7360	0.7210	0.7500	0.7230
IAN	0.7860	-	0.7210	-	-	-	-	-
RAM	0.8023	0.7080	0.7449	0.7135	0.6936	0.6730	-	-
TAN-FC	0.8214	0.7486	0.7492	0.7098	0.7399	0.7274	0.7573	0.7350
TAN-FC-PE	**0.8277**	**0.7509**	0.7570	0.7158	0.7471	0.7312	0.7556	0.7322
TAN-FM	0.8214	0.7444	0.7540	0.7137	**0.7486**	**0.7360**	0.7564	0.7320
TAN-FM-PE	0.8223	0.7499	**0.7586**	**0.7193**	0.7399	0.7283	**0.7641**	**0.7399**

Table 2: Final results of proposed target attention networks (TAN). FC and FM are two types of target attention. Postfix 'PE' means to consider position embedding as input features. Compared results are extracted from corresponding papers, the ones with '†' are not reported in original papers but reproduced by Chen (2017).

Net. The difference is they apply multiple time attentions on the hidden states of a BiLSTM, rather than apply attention on the word embedding.

4.2 Evaluations & Results

We use two metrics to evaluate our performance. The first metric is the accuracy, which has been reported in all previous works, directly telling how well the model does. The other metric is the macro-F1 score, since all four experimental datasets are imbalanced, macro-F1 score provides another point of view to show how well the model does among all classes. We report the performance of the proposed target attention network (TAN), including two types of target attention: **FC** (TAN-FC) and **FM** (TAN-FM), described in Section 3.3. We also report the results considering position embedding as input features (with postfix 'PE') and the ones which only use word embedding as input features (without postfix 'PE'). The results are shown in Table 2. Our approach outperforms all compared methods on all four datasets.

Let's take a look at the results. Feature engineering based methods such as Featured-SVM and Target-dep are mostly not comparable to neural network based methods. These methods depend on external resources like sentiment lexicons, which may lower its flexibility for languages with less resources. AdaRNN faces the same problem since their results are highly depending on the performance of parsing trees.

TC-LSTM and GRNN use neural networks to encode sequence. Nevertheless, the absence of attention mechanism makes their performances inferior to attention-based methods. MemNet applies multiple time attentions, but the temporal information is weaker as they doesn't use any sequence encoder. RAM not only applies attention multiple times but uses a BiLSTM as the sequence encoder. Their performance is the best in the compared methods. While they average target as the initial attention query, which may dilute the informative words among target. Our target attention networks alleviate this problem, which makes our approachs surpass theirs. We have about 2% accuracy gain and 4% F1 score gain on the restaurant reviews, 1% accuracy gain and 0.5% F1 score gain on the laptop reviews, and 5% accuracy gain and 6% F1 score gain on the twitter dataset.

The other methods with attention mechanism exist the same problem of the target representation. AB-LSTM and BILSTM-ATT-G both use the mean of the target as the query of sequence attention. IAN uses both the mean of a target and its context to count attention weights, which may encounter more information dilution. Our approach outperforms these methods on all datasets, reaching the latest state-of-the-art on these benchmarks.

4.3 Effects of Target Attention Network

As mentioned before, we believe that simply average target words may hurt performance because informative words will be overwhelmed by a large number of meaningless words, especially for longer targets. We want to know exactly how target attention network affects performance when target length varies. We use Z-dataset here because it contains more longer targets than other datasets. To show the pure effects of target attention network, we remove position embedding in

this experiment. We only report the performance of FM type target attention here for succinctness.

In Figure 2, we show the accuracy for the model with and without target attention. It is straightforward to see that with target attention (green line), we can always get equal or better accuracy than the one without target attention (yellow line). One interesting thing is that the accuracy gain with target attention (red dashed line) has an increasing trend. Specifically, the performance gap between the model with target attention and the one without target attention becomes larger as the target length grows. This finding shows that simply average target words can result in poor performance when facing longer targets. Target attention networks can alleviate this problem by emphasizing informative parts in the target.

Figure 2: Accuracy w/ & w/o target attention. Models with target attention consistently outperform the ones without target attention, especially for longer targets.

Next, we show how target attention influences the result of subsequent sequence attention. We pick a testing example from the restaurant reviews, as illustrated in Figure 3. The input sequence is *"The fish is fresh but the variety of fish is nothing out of ordinary"*, the target is *"variety of fish"*, and its sentiment polarity is negative. Colors in the figure represent the weights of sequence attention. Words with darker color means the model pays more attention on them. The target attention may attend on different parts of the target. The curves on the target *"variety of fish"* represent the intensity of target attention weights. In the first row, word *"fish"* of the target gets the most of the target attention weight, then sequence attention will focus on *"fresh"* as its most important context, while *"fresh"* can lead to the wrong answer. In the second row, word *"variety"* of the target gets the most of the target attention weight, the later sequence attention can capture more related context *"nothing out of ordinary"*, and make the correct prediction. The attention results of sequence can change significantly when focusing on different parts of the target, which verifies the necessity and the importance of the target attention.

4.4 Effects of Position Embedding

To demonstrate the effects of the position embedding, we remove target attention networks in this experiment. Figure 4 is a testing example picked from the restaurant reviews. The input sequence is *"the food was definitely good, but when all was said and done, i just couldn't justify it for the price"*, the target is *"price"*, and the sentiment polarity of *"price"* is negative. Again, colors in the figure represent the sequence attenton weights of each word. The first row is the model without position embedding, we can see it spends half of the attention weights on *"definitely good"*. However, *"definitely good"* can lead to wrong sentiment polarity for the target *"price"*. Although it puts 0.2 weight on *"but"*, which tries to twist the semantic intent, the intensity is not enough so the prediction is still incorrect. Now we look at the second row, with position embedding, our model can capture more relative parts *"could n't justify"*, which actually determines the sentiment ploarity of the target *"price"*. An interesting finding is that attention-based models tend to put most of the weights on obvious sentiment words such as good and bad. This phenomenon can sometimes hurt the performance when contradictory sentiment words appear at the same time (e.g. *"The food is good but the service is bad"*). Position embedding provides distinct information besides sentiments, so our model can consider both features and extract more complicated opinion for targets, just as discriminative phrase *"could n't justify"* is captured in our case.

4.5 Ablation Study

To further demonstrate the adventages of the proposed methods, we execute ablation study. We experiment on all four datasets, with removements of target attention or position embedding or both of them. We list the results of FM type target attention here for comparison. Table 3 shows the accuracy results and table 4 show the macro-F1 results.

Figure 3: Attend on different target words will lead to different sequence attention result and prediction. If target attention attends on word *"fish"* of the target, sequence attention will focus on context *"fresh"*. If target attention attends on word *"variety"* of the target, sequence attention will focus on context *"nothing out of ordinary"*.

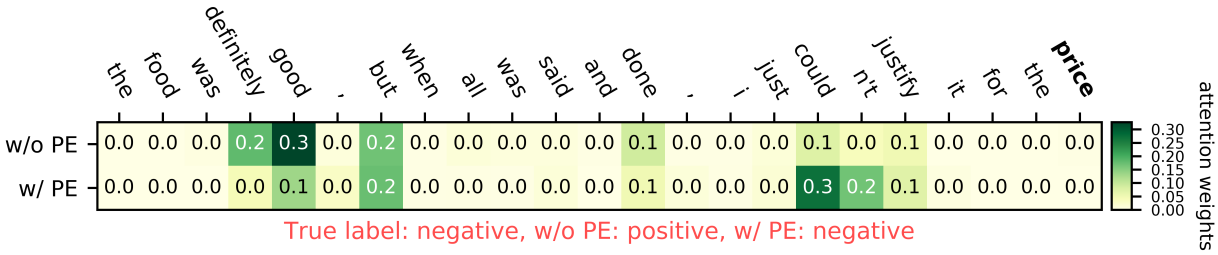

Figure 4: Attention weights w/ & w/o position embedding (PE). Without position embedding, the sequence attention will focus on context *"definitely good"* which leads to wrong prediction. The model with position embedding can extract more complicated opinion for the target, just as *"could n't justify"* in our case.

Accuracy		w/o PE	w/ PE
Restaurant	w/o TA	0.7959	0.8071
	w/ TA	0.8214	0.8223
Laptop	w/o TA	0.7116	0.7288
	w/ TA	0.7540	0.7586
Twitter	w/o TA	0.7153	0.7168
	w/ TA	0.7486	0.7399
Z-Dataset	w/o TA	0.7291	0.7419
	w/ TA	0.7564	0.7641

Table 3: Accuracy of ablation study. TA means target attention, PE means position embedding. We list the results of FM type target attention for comparison.

Macro-F1		w/o PE	w/ PE
Restaurant	w/o TA	0.7025	0.7245
	w/ TA	0.7444	0.7499
Laptop	w/o TA	0.6745	0.6837
	w/ TA	0.7137	0.7193
Twitter	w/o TA	0.6965	0.7004
	w/ TA	0.7360	0.7283
Z-Dataset	w/o TA	0.7005	0.7184
	w/ TA	0.7320	0.7399

Table 4: Macro-F1 scores of ablation study. TA means target attention, PE means position embedding. We list the results of FM type target attention for comparison.

There is about 2.5% to 4% accuracy gain and a 3% to 4% F1 score gain for target attention, which is the main reason why our work outperforms previous works. Position embedding can further improve performance by about 0.5% to 1%. Position embedding can obtain higher performance gain of 1% to 2% when target attention is absent.

5 Conclusions

We propose a novel target attention network to concentrate on important parts of the target. Experiments show that target attention can boost performance especially for longer targets, and target attention has a great effect on later context attention. We also introduce a novel target-aware position embedding to model the location relation between the target and its context. Position embedding provides distinct information from semantic meanings, which helps our model to extract more complicated opinion for targets. Our target attention network outperforms all state-of-the-art on four public benchmarks, the experimental results demonstrate our findings. The concepts of molding target words non-linearly and representing the location relation between a target and its context can easily apply to other entity-based tasks, such as relation classification, which is a potential direction for future works.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of*

the 3rd International Conference on Learning Representations.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoderdecoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*.

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 the 52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Alex Graves, Abdel rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. Association for Computational Linguistics.

Alex Graves and Jurgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 151–160.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. In *CoRR*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442.

Jiangming Liu and Yue Zhang. 2017. 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, pages 572–577.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv:1301.3781. Version 3.

Margaret Mitchell, Jacqueline Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1654.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Maria Pontiki, Dimitrios Galanis, John Pavlopoulos, Haris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Steffen Rendle. 2010. Factorization machines. In *ICDM '10 Proceedings of the 2010 IEEE International Conference on Data Mining*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. In *IEEE TRANSACTIONS ON SIGNAL PROCESSING*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of 26th International Conference on Computational Linguistics*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*.

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 the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 223–229.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the 3th International Conference on Learning Representations*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*.

Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. 2017. Attention-based lstm for target-dependent sentiment classification. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL 16*.

Shuangfei Zhai, Keng hao Chang, Ruofei Zhang, and Zhongfei (Mark) Zhang. 2016. Deepintent: Learning attentions for online advertising with recurrent neural networks. In *KDD 16*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*.