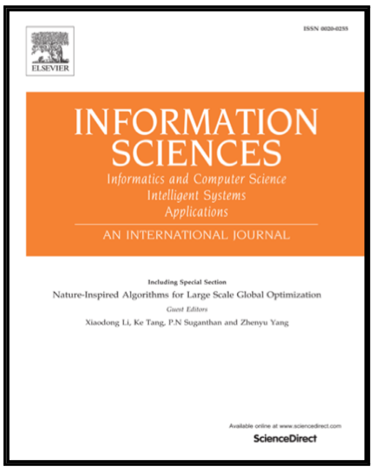


Target-Aware Convolutional Neural Network for Target-Level Sentiment Analysis

Dongmin Hyun, Chanyoung Park, Min-Chul Yang, Ilhyeon Song, Jung-Tae Lee, Hwanjo Yu

PII: S0020-0255(19)30287-7
DOI: https://doi.org/10.1016/j.ins.2019.03.076
Reference: INS 14416

To appear in: *Information Sciences*

Received date: 4 September 2018
Revised date: 27 March 2019
Accepted date: 28 March 2019

Target-Aware Convolutional Neural Network for Target-Level Sentiment Analysis

Dongmin Hyun[a], Chanyoung Park[b], Min-Chul Yang[c], Ilhyeon Song[c], Jung-Tae Lee[c], Hwanjo Yu[a,∗]

[a]*Dept. of Computer Science and Engineering, POSTECH, South Korea*
[b]*University of Illinois at Urbana-Champaign, USA*
[c]*NAVER Corporation, Seongnam, South Korea*

Abstract

Target-level sentiment analysis (TLSA) is a classification task to extract sentiments from targets in text. In this paper, we propose target-dependent convolutional neural network (TCNN) tailored to the task of TLSA. The TCNN leverages the distance information between the target word and its neighboring words to learn the importance of each word to the target. Experimental results show that the TCNN achieves state-of-the-art performance on both single- and multi-target datasets. Qualitative evaluations were conducted to demonstrate the limitations of previous TLSA methods and also to verify that distance information is crucial for TLSA. Furthermore, by exploiting a convolutional neural network (CNN), the TCNN trains six times faster per epoch than other baselines based on recurrent neural networks.

Keywords: sentiment anlaysis, target-level sentiment analysis (TLSA), word distance, deep learning, convolutional neural network (CNN)

1. Introduction

Sentiment analysis is a classification task that infers the sentiment polarity of a piece of text [23, 18, 13]. Sentiment analysis may be excuted at various levels such as document-level, sentence-level and target-level [14, 43, 34]. For example, sentence-level sentiment analysis captures a positive sentiment in the following sentence: “The steak

∗Corresponding author
Email address: dm.hyun@postech.ac.kr (Dongmin Hyun)

of course was great and the drink was made nice and strong" Whereas, the target-level sentiment analysis (TLSA) infers a positive sentiment from the target words "*steak*" and "*drink*".

Recently, TLSA has gained considerable attention from commerce and industry, because accurate sentiment analysis of customer text using specific product as targets improves the company's understanding of customer needs. This enables companies to provide services tailored to customer needs, which, in turn, leads to increased revenues. Past research on TLSA can be categorized into two streams according to how the target-relevant features are generated.

1) Generating features manually [40] using out-of-data information such as results from a syntactic parser [41].

2) Generating features automatically using deep learning-based methods such as recurrent neural network (RNN) [34, 46, 19, 44, 20].

Generating features using out-of-data information requires manual effort by domain-experts, and thus is very costly. Deep learning-based automatic feature extractors have been actively developed in various domains such as computer vision [17, 31, 5], natural language processing (NLP) [7, 33, 11], and sentiment analysis [35, 46, 19, 44, 20]. In particular, convolutional neural networks (CNN) have proven its effectiveness for document-level [12, 36, 1] and sentence-level sentiment analysis [14, 30, 26]. As for TLSA, which is the focus of this paper, RNNs have been widely used to extract features from text by considering the text as a sequence of words [34].

However, previous RNN-based methods have a limitation. An RNN ignores the *explicit distance information between each word and the target words*, despite the fact that the words closer to the target word are more likely to be related to the sentiment of the target. To focus on the target within a text, previous RNN-based TLSA methods [34, 46] split the text into three components:

1) Target words

2) Left-side text

3) Right-side text

Then RNNs are applied to each side of the target to extract target-relevant features. However, the RNN do not consider the vicinity of words. For example, they consider the text "Although the *space* is smaller than most, it is the best service you will find in even the largest restaurants". The sentiment on the target *space* is negative due to its small size, but this sentiment may be misjudged to be positive by RNNs, because the influence of the clue word "smaller" is dominated by the distant but more powerful word "best".

Consequently, attention mechanisms [2, 24, 15, 39] have been applied to RNNs to focus on words with a stronger influence on the sentiment of the target [46, 19, 44, 20]. The fundamental concept of the attention mechanism for TLSA is to enable neural networks to focus on words associated with the given target. In the previous example, the word "smaller" would be assigned a greater attention weighting if the target is "*space*" because they are semantically related. Conversely, "smaller" would receive a lower attention weighting if the target was "*service*". However, these attention-based methods are constrained by the inherent computation limitations of RNNs, which do not consider the explicit distance information of each word in the text in relation to the target word(s) when calculating attention weights.

We argue that CNNs are especially well-suited for TLSA because the convolution operation is followed by the max-pooling operation that extracts the most important n-gram features from a local context. This ameliorates the adverse effect of irrelevant words on a given target. However, CNNs have been primarily been exploited for the document-level [12, 4, 8] or sentence-level [14, 29] sentiment analysis. These conventional CNNs cannot be directly applied to TLSA because they overlook 1) the targets on which the sentiment polarity is judged, and 2) the position of the targets within the text.

This paper proposes novel CNN-based TLSA methods that consider the vicinity of words near the given target. Consider the previous sentence example. The proposed methods explicitly focus on words closer to the target such as "smaller", prioritizing closer words over distant but more powerful words such as "best". Such vicinity of words cannot be explicitly modeled in the conventional max-pooling scheme of CNNs. Thus, we propose two CNN-based methods to explicitly leverage the distance information between a target word and its neighboring words.

1) The first method selects the neighboring words of the target using a fixed-size "neighbor window".

2) The second method learns the importance of the neighboring words using distance embeddings [45]

To the best of our knowledge, this work is the first to propose distance-aware CNNs tailored to TLSA.

Experimental results show that our methods outperform existing state-of-the-art methods on single-target and multi-target datasets. Furthermore, qualitative evaluations were conducted to demonstrate the limitations of prior TLSA methods and verify that the distance information is crucial for TLSA. Finally, by exploiting CNNs, the TCNN trains six times faster per epoch than other RNN-based baselines because CNNs are easily parallelizable using GPU.

Our contributions are summarized as follow:

- We designed novel distance-aware CNNs that are specifically tailored to the task of TLSA. We address the limitations of previous RNN-based methods by explicitly leveraging the distance information between a target word and its neighboring words.
- Experimental results demonstrate that the proposed methods outperform existing state-of-the-art TLSA methods on both single- and multi-target datasets. Qualitative analyses highlight the merit of our CNN-based methods in comparison with feature engineering-based and RNN-based baselines.
- The training times of RNN- and CNN-based methods were also compared. The results prove that the proposed CNN-based methods train substantially faster than the RNN-based methods owing to the easily-parallelizable architecture of CNNs.

2. Background

TLSA is a classification task that judges the sentiment polarity of a target in text. The term “target” in TLSA is often confused with the term “aspect”, as in aspect-level sentiment analysis (ALSA). The major difference between TLSA and ALSA lies in the presence of the target or aspect in the text. Unlike the target, an aspect is not explicitly provided in the text. For example, in the text “The screen is bright and clear, and the operating system is friendly to a novice”, the aspects of interest are the “quality of screen” and “user-friendliness”, and the targets are “screen” and “operating system”. Thus, targets are also distinguished from aspects by the fact that the position of the target in the text is known a priori in TLSA. However, note that models for ALSA have been applied to TLSA by treating targets as aspects [43]. The rest of Section 2 examines methods related to TLSA.

2.1. Related Work

Various machine learning methods have been applied to TLSA. Vo & Zhang [40] defined target-specific features using multiple pre-trained word embedding vectors, and used a support vector machine (SVM) as a classifier. Wang et al. [41] leveraged a syntactic parser to extract dependent words related to a given target with an SVM. However, these feature-engineering approaches usually require domain experts to identify informative features for different data. Additionally, using out-of-data information such as the results of a syntactic parser is also constrained because the parser can be language dependent [41].

Recently, deep learning, especially RNN-based approaches, have grown popular owing to the ability to automatically learn features from data. Tang et al. [34] considered surrounding features of a given target by combining two long short-term memory (LSTM) RNNs, each of which takes as inputs the left- and right-side text based on the position of the target in the text. Wang et al. [43] incorporated target-embedding vectors in an LSTM-based attention method to focus on target-dependent words in a given text. Zhang et al. [46] used gate units on a bidirectional gated RNN to learn the contribution of the left- and right-side text and of the target itself on the sentiment polarity of the target. Ma et al. [20] modeled interactions between a target and the given text by using the attention mechanism on LSTM outputs. Yang et al. [44] applied the attention mechanism to a bidirectional LSTM BiLSTM to learn the importance of each word in relation to the target. Liu & Zhang [19] applied gate units to the attention layer, followed by bidirectional LSTMs, to learn the contributions of the whole text, the left-side text and the right-side text on the sentiment polarity of the target. Tay et al. [37] extended the method of Wang et al. [43] by reducing the trainable parameters to simplify the method and improve efficiency. Tay et al. [38] incorporated a sentiment-lexicon dictionary to explicitly model positive and negative words with an LSTM-based attention mechanism. Ma et al. [21] exploited commonsense knowledge with an LSTM-based method with a hierarchical attention mechanism for better understanding on a given text. Wang et al. [42] emphasized the importance of linguistic hints such as explicit sentiment words and swift words (e.g., “but”). With the linguistic hints, they proposed a joint method which integrated deep learning models with the linguistic hints. Performance of all these methods is constrained by their lack of consideration of the explicit distance information for each word in the text in relation to the target.

Thus, we propose using a CNN to alleviate the performance limitations of existing RNN-based methods. A CNN can ameliorate the adverse effects of irrelevant words on a given target by focusing on local n-grams and adopting the max-pooling scheme. However, CNNs have not yet been actively explored for TLSA, despite the fact that CNNs have become popular in document-level [12, 4, 8] and sentence-level [14, 10, 26, 29] sentiment analysis. This prpularity stems from the CNNs’ comparable performance of various NLP tasks combined with shorter training time compared to RNN variants.

In SemEval[1]-2017 task 4, Rosenthal et al. [27] explored sentiment analysis in Twitter. Specifically, subtasks B and C focused on TLSA. The winning solution, BB_twtr [6], adopted distant training using noisy tweets and supervised training using a ensemble model composed of ten CNNs and ten LSTMs. DataStories [3] was runner-up solution that adopted multiple BILSTMs with an attention mechanism and then trained word-

[1] SemEval is an international forum for natural-language shared tasks. We only mention the task that was held in 2017 since SemEval does not hold sentiment analysis in 2018 and 2019.

embedding vectors using 330 million English tweets created over the course of 4 years. However, these methods cannot be directly compared to the proposed method because they were specifically designed to win a challenge by adopting various empirical approaches, such as ensemble learning, distant training and unsupervised training.

3. Model Architecture

Conventional CNNs for sentence-level sentiment analysis utilize max-pooling layers to extract the most important features from text. This enables the method to focus on parts that are influential to the sentiment of a sentence. However, such method cannot be directly applied to TLSA because it overlooks two factors: the targets on which the sentiment polarity is to be judged and their positions in the text. Regarding the above constraints, the key concept of our approach is that a vanilla CNN is tailored to determine the distance information between words in the text and the target. We designed two approaches:

1) Hard scheme: Provide vicinity words near the target within a specific distance to the CNN.

2) Soft scheme: Specify the word distance between the target for each word.

To this end, we propose a simple baseline method (TCNN-concat) which ignores distance information and two novel CNN-based methods (TCNN-hard for the hard scheme and TCNN-soft for the soft scheme) which are tailored to TLSA using distance information. As depicted in (Figure 1a) the TCNN-hard (Figure 1a) extracts target-relevant features from a distance-aware window called the *neighbor window*. In the previous example, if the size of neighbor window is set to "2", then the words in the neighbor window will be "Although the *space* is smaller" where "*space*" is the target. In contrast, the TCNN-soft ,shown in Figure 1b, masks input words with their distances from the target. Thus, the distance of the target to itself is zero, the distance of the closest words "the" and "is" to 1 and the next closest words "Although" and "smaller" have a distance of 2. Note that the importance of distances in the TCNN-soft is learned from the underlying data. In next subsections, we describe a conventional CNN and then detail the propose CNN-based methods .

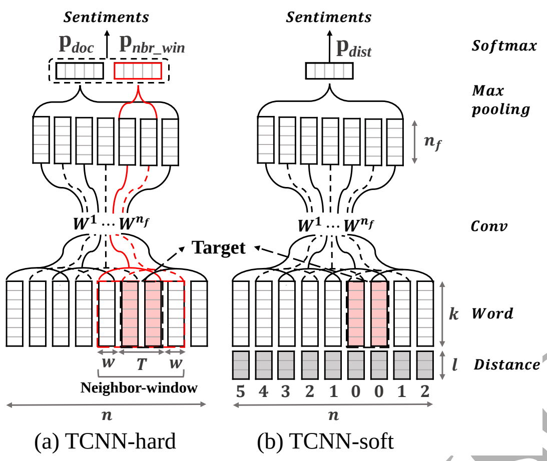


Figure 1: Architecture of our proposed methods. Red parts indicate target-relevant components.

3.1. CNN

A CNN designed for sentence-level sentiment analysis [14] takes an input text of n-words, which is represented as concatenated word vectors $\mathbf{X} \in \mathbb{R}^{n\times k}$ where n is the number of words in the input text and k is the size of the embedding dimension for each word. The CNN performs convolution operations on $\mathbf{X}$ using convolution filters to extract local features in the text. More precisely, by applying the f-th convolution filter $\mathbf{W}^f \in \mathbb{R}^{j\times k}$ with the window size j on a sub-text containing j words $\mathbf{X}_{((i:i+j-1),:)} \in \mathbb{R}^{j\times k}$, we obtain a feature

$$c_i^f = \sigma(\mathbf{W}^f * \mathbf{X}_{((i:i+j-1),:)} + b^f) \in \mathbb{R}, \tag{1}$$

where * is the convolution operator, $\mathbf{X}_{((i:i+j-1),:)}$ is the concatenated word vectors from i-th word to $(i + j - 1)$-th word, $b^f \in \mathbb{R}$ is the bias for $\mathbf{W}^f$, and σ is a non-linear function such as hyperbolic tangent (tanh) and rectified linear unit (ReLU) [22]. By applying the f-th convolution filter on the entire text $\mathbf{X}$, we obtain a feature map

$$\mathbf{c}^f = [c_1^f, c_2^f, ..., c_i^f, ..., c_{n-j+1}^f] \in \mathbb{R}^{n-j+1}, \tag{2}$$

At the penultimate layer, the max-pooling scheme is used to identify the most important feature in the feature map. i.e., $\hat{c}^f = max(\mathbf{c}^f)$. We use multiple convolution filters to capture various aspects of features. The features generated from n_f convolution filters followed by the max-pooling layer form a feature vector

$$\mathbf{x} = [\hat{c}^1, \hat{c}^2, ..., \hat{c}^j, ..., \hat{c}^{n_f}] \in \mathbb{R}^{n_f}, \tag{3}$$

where n_f is the number of filters. To predict the sentiment polarity of the input text s, the feature vector **x** is fed into a fully-connected softmax layer,

$$\mathbf{s} = softmax(\mathbf{W}^T\mathbf{x} + b), \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{n_f \times 3}$ and $b \in \mathbb{R}^3$ are a weight matrix and a bias for the fully-connected softmax layer in which the number "3" indicates the number of sentiment categories. For regularization, we employ dropout [32] on both the input layer and the penultimate layer.

3.2. *TCNN-concat*

The conventional CNN does not consider the target and its position in text. As a straightforward approach, we inject the target information into the CNN by concatenating the target word vectors and the embedding of the input text. More precisely, the target-concatenated text $\mathbf{X}^{t.concat} \in \mathbb{R}^{(n+T)\times k}$ is formed by concatenating the input words $\mathbf{X} \in \mathbb{R}^{n\times k}$ and target words $\mathbf{X}_{((t_1:t_T),:)} \in \mathbb{R}^{T\times k}$ as follows:

$$\mathbf{X}^{t.concat} = [\mathbf{X}; \mathbf{X}_{((t_1:t_T),:)}] \in \mathbb{R}^{(n+T)\times k}, \tag{5}$$

where the operation ';' is the concatenation operation, t_1 and t_T are the start and end position of the target words in the input text, respectively, and T is the length of the target words[2]. The target-concatenated text $\mathbf{X}^{t.concat}$ is fed into a CNN to extract both target-level and text-level features. The structure of the following layers is identical to the model described in Section 3.1. We refer to this method as a target-dependent CNN using concatenation (TCNN-concat), and consider it a naive baseline method of applying a CNN in TLSA.

[2]Note that a single target may contain multiple words. For instance, $T = 2$ for a target word *"baseball cap"*.

However, the sentiment polarity on a target is more likely to be dependent on its neighboring words because words relevant to the target are often located near the target itself. Therefore, we designed two novel CNN-based methods called TCNN-hard and TCNN-soft, to explicitly model the vicinity of words to the target.

3.3. TCNN-hard

The TCNN-hard method, seen in Figure 1a, extracts the target-level features by applying the CNN to the neighboring words of a target. The neighboring words are determined by a fixed-size "*neighbor window*"[3], assuming that the window contains clue words for judging the sentiment on the target. A "neighbor window" of size w contains $(2w + T)$ words including w neighboring words to the left and right of the target and T target word(s). Formally, given the entire text $\mathbf{X} \in \mathbb{R}^{n \times k}$, the text in the "neighbor window" is equivalent to the sub-text $\mathbf{X}_{((t_1 - w:t_T + w),:)} \in \mathbb{R}^{(2w+T)\times k}$ from which the CNN extracts target-level features. The final feature can be represented as

$$\mathbf{x}^{hard} = [\mathbf{x}; \mathbf{x}^{sub}] \in \mathbb{R}^{2 * n_f}, \tag{6}$$

where $\mathbf{x}^{sub} \in \mathbb{R}^{n_f}$ is the feature extracted from the "neighbor window" $\mathbf{X}_{((t_1 - w:t_T + w),:)}$ using the CNN. The final feature $\mathbf{x}^{hard}$ is fed to the fully-connected softmax layer to predict target-level sentiments.

Note that given a document with multiple targets, sentiments on these targets tend to be unanimous[4], which implies that the overall sentiment on the document should not be neglected. To model this observation, we also extract document-level features from the entire text $\mathbf{X}$. The target-level and document-level features are concatenated and fed into the softmax layer to classify the target-level sentiments. Note that we also show the performance of a method named TCNN-hard^{nbr} that only adopts the "neighbor window" to demonstrate that we need both the document-level and target-level features to correctly predict sentiments on targets.

[3]We introduce the term "*neighbor window*" to differentiate from the convolution filter of CNN.

[4]In the multi-target dataset used in our experiments, half of the documents have an unanimous sentiment of the targets within each document, that is positive, neutral or negative.

3.4. TCNN-soft

The TCNN-hard method extracts target-level and text-level features by using a fixed-size "neighbor window", which does not consider the interactions between the target and other words outside the "neighbor window". Therefore, instead of fixing the size of the "neighbor window", we give each word the distance information calculated based on the distance from the target. More precisely, each word vector is concatenated with a distance embedding vector corresponding to its distance from the target. For example, in the text "It is the best *service* you will find in even the largest restaurants", assuming that the distance is zero for the target *service*, the relative distance of the word "best" is -1, and +1 for "you". In this work, however, we adopt the absolute distance to reduce the number of model parameters, which helps to prevent overfitting. Thus, the distance for "best" and "you" is set to 1.

Formally, i-th word vector $\mathbf{X}_{(i,:)} \in \mathbb{R}^k$ and its distance embedding vector $\mathbf{D}_{(dist(i),:)} \in \mathbf{D} \in \mathbb{R}^{M\times l}$ are concatenated to form a distance-augmented input, where $\mathbf{D}_{(dist(i),:)}$ is a vector of l dimensions, $dist(i)$ computes the absolute distance of i-th word in the text from the given target, M[5] is the maximum value of $dist(i)$ for all i in the corpus, and l is the dimension size of the distance embedding vector. For example, if the distance between the target and the i-th word is 2, then $\mathbf{X}_{(i,:)}$ is augmented with $\mathbf{D}_{(2,:)} \in \mathbb{R}^l$ to form $\mathbf{X}^{d.concat}_{(i,:)} = [\mathbf{X}_{(i,:)}; \mathbf{D}_{(2,:)}] \in \mathbb{R}^{(k+l)}$. The distance embedding-augmented entire input $\mathbf{X}^{d.concat} \in \mathbb{R}^{n\times(k+l)}$ is fed into a CNN to extract features to judge the target-level sentiments. The distance-embedding vectors are randomly initialized before training and then updated during the training process. This method is named TCNN-soft, and Figure 1b shows its architecture.

3.5. Training

The proposed methods are trained by optimizing the cross-entropy loss:

$$loss = -\sum_{i}^{n}\sum_{j}^{C} y_{ij} log(\hat{y}_{ij}) + \lambda \left\|\Theta\right\|^2 \tag{7}$$

[5] In our experiments, *M* is 39 and 37 on single- and multi-target datasets, respectively.

where n is the number of data, C is the number of sentiment classes, y is true sentiment, $\hat{y}$ is the model prediction, and Θ is the parameter set. The regularizer L_2 is applied on Θ to prevent overfitting and Adam [16] is used as an optimizer.

4. Material and methods

Section 4 describes comprehensive experiments conducted to answer the following questions:

Q.1 How do our proposed methods perform compared with other state-of-the-art baselines?

Q.2 Why do our baselines show poor performance? Does TCNN-soft actually focus on the neighboring words of the target by leveraging the distance embeddings?

Q.3 How fast is the training time of the TCNN compared to RNN-based baselines?

Q.4 How do the newly introduced hyperparameters w and l affect the model performance?

Q.5 How does the performance of methods differ in relation to text length?

4.1. Experimental settings

4.1.1. Dataset

We use two real-world datasets to validate the performance of the proposed methods. The single-target dataset [9] contains (text, target, sentiment) triplets, where only one target appears in each text. The multi-target dataset [41] also contains (text, target, sentiment) triplets, but multiple targets appear in each text, i.e., 3.09 targets per text. Both datasets are in English and were collected from Twitter. For each dataset, we randomly sample 10% of training data as validation data while considering the sentiment class distributions as listed in Table 1. Note that the numbers in the multi-target data denote the number of targets.

Table 1: Statistics of datasets.

	Single-target		**Multi-target**	
	Train	Test	Train	Test
Positive	1,561	173	1,466	399
Neutral	3,127	346	3,722	985
Negative	1,560	173	4,724	1,291
Total	6,248	692	9,912	2,675

4.1.2. Evaluation measure

We employ two metrics used in prior work [9, 34], i.e., accuracy and macro-F1, to evaluate the TLSA methods. The macro-F1 metric measures how well the predictions are balanced throughout different sentiment classes.

4.1.3. Comparison methods

We compare our proposed methods with existing state-of-the-art methods from both of the TLSA streams:

1) Generating feature manually using out-of-data information.

2) Generating features automatically using RNN.

Properties of the methods are summarized in Table 2. Note that *target-position information* refers to the position of given target in text and *explicit-distance information* refers to the distance between context words and given target.

The following existing methods were evaluated:

- **ATAE-LSTM** [43]: This method is designed for ALSA, and it captures the relation between a given target and context words by concatenating them for attention mechanism.
- **AF-LSTM (CONV)** [37]: This method is an extension of ATAE-LSTM [43], which allow the model to exploit more sophisticated interactions, not just the concatenation of a given target and context words.

Table 2: Properties of methods being compared.

Methods	TLSA?	Deep-learning based?	Target-position information?	Explicit-distance information?
ATAE-LSTM		✓		
AF-LSTM (CONV)		✓		
TD-LSTM	✓	✓	✓	
AB-LSTM	✓	✓		
TDparse	✓		✓	
BILSTM-ATT-G	✓	✓	✓	
TCNN-hard	✓	✓	✓	✓
TCNN-soft	✓	✓	✓	✓

- **TD-LSTM** [34]: This method extracts target-relevant features from left-side and right-side texts of a given target using two LSTMs.
- **AB-LSTM** [44]: This method applies the attention mechanism to a bidirectional LSTM to capture relations between a given target and context words.
- **TDparse** [41]: This method is the state-of-the-art method for TLSA in the multi-target dataset. It extracts target-dependent words using a syntactic parser.
- **BILSTM-ATT-G** [19]: This method applies the attention mechanism on bidirectional LSTMs with gate units. It is the state-of-the-art method for TLSA in the single-target dataset.

4.1.4. Hyperparameters

Every hyperparameter for each method is tuned on the validation set by grid search. We run every experiment ten times and reported the average results. We use 0.5 as the dropout probability, 32 as mini-batch size, and 0.01 as λ. The best performing method-specific parameters are listed in Table 3.

4.1.5. Word embedding

We use pretrained word-embedding vectors trained using Glove [25] with 200 dimensions as done by our baselines [34, 19].

Table 3: Hyperparameters of proposed methods.

Model	Dataset	Filter size	n_f	w	l
TCNN-hard	Single-target	5,6	256	2	-
	Multi-target	2,3	128	3	-
TCNN-soft	Single-target	3,4	256	-	32
	Multi-target	2,3	256	-	32

4.1.6. Experiment environment

Experiments are conducted on a Linux machine with an Intel Core i7-2600, Nvidia TITAN X GPU and 16 GB RAM. Experiments on the baselines, excluding the TD-LSTM are conducted using the authors' code. We implement the TD-LSTM method in our framework to directly compare the training time with our methods.

5. Results and discussion

5.1. Performance comparison (Q.1)

To demonstrate the effectiveness of our proposed methods, we compare our methods to existing state-of-the-art TLSA methods. Table 4 summarizes the results. We have the following observations.

5.1.1. Comparisons among baselines

1) We observe that ATAE-LSTM, which is an ALSA method, shows poor performance on both datasets. As an extension of ATAE-LSTM, AF-LSTM (CONV), shows inferior performance on the single-target dataset. However, this implies that it is crucial to model the sophisticated interaction between a given target and context words, and that they can be valid baselines for the methods designed for TLSA.

2) The performance of BILSTM-ATT-G is generally superior to that of TD-LSTM and AB-LSTM, which emphasizes the importance of the neural network architecture design.

Table 4: Performance comparisons on both datasets ($*$: Taken from their paper[6]. † indicates that the result of paired t-test is significant at $p < 0.05$.

Model	Single-target		Multi-target	
	Accuracy	Macro-F1	Accuracy	Macro-F1
TD-LSTM	$0.686_{\pm 0.0140}$	$0.662_{\pm 0.0144}$	$0.552_{\pm 0.0057}$	$0.479_{\pm 0.0092}$
ATAE-LSTM	$0.679_{\pm 0.0054}$	$0.655_{\pm 0.0082}$	$0.564_{\pm 0.0054}$	$0.465_{\pm 0.0078}$
AB-LSTM	$0.668_{\pm 0.0079}$	$0.638_{\pm 0.0127}$	$0.578_{\pm 0.0057}$	$0.432_{\pm 0.0025}$
AF-LSTM (CONV)	$0.682_{\pm 0.0041}$	$0.630_{\pm 0.0041}$	$0.576_{\pm 0.0054}$	$0.476_{\pm 0.0108}$
TDparse	$0.715_{\pm 0.0077}$	$0.690_{\pm 0.0101}$	0.568^*	0.455^*
BILSTM-ATT-G	$0.719_{\pm 0.0079}$	$0.704_{\pm 0.0082}$	$0.573_{\pm 0.0094}$	$0.476_{\pm 0.0126}$
LSTM	$0.675_{\pm 0.0061}$	$0.651_{\pm 0.0069}$	$0.543_{\pm 0.0073}$	$0.407_{\pm 0.0223}$
LSTM-soft	$0.694_{\pm 0.0062}$	$0.671_{\pm 0.0072}$	$0.574_{\pm 0.0071}$	$0.472_{\pm 0.0206}$
CNN	$0.706_{\pm 0.0034}$	$0.690_{\pm 0.0040}$	$0.534_{\pm 0.0058}$	$0.436_{\pm 0.0082}$
TCNN-concat	$0.691_{\pm 0.0085}$	$0.675_{\pm 0.0065}$	$0.570_{\pm 0.0037}$	$0.471_{\pm 0.0069}$
TCNN-hard^{nbr}	$0.700_{\pm 0.0056}$	$0.683_{\pm 0.0049}$	$0.575_{\pm 0.0049}$	$0.484_{\pm 0.0076}$
TCNN-hard	$\mathbf{0.730}^{\dagger}{}_{\pm 0.0056}$	$\mathbf{0.714}^{\dagger}{}_{\pm 0.0044}$	$0.583^{\dagger}{}_{\pm 0.0043}$	$\mathbf{0.493}^{\dagger}{}_{\pm 0.0053}$
TCNN-soft	$0.714_{\pm 0.0042}$	$0.698_{\pm 0.0036}$	$0.579_{\pm 0.0035}$	$0.491_{\pm 0.0073}$
TCNN-hard+soft	$0.725_{\pm 0.0086}$	$0.707_{\pm 0.0089}$	$\mathbf{0.586}_{\pm 0.0036}$	$0.491_{\pm 0.0115}$

3) BILSTM-ATT-G consistently outperforms TDparse. This verifies the effectiveness of deep learning methods as automatic target-level feature extractors for TLSA.

5.1.2. *Analysis of CNN-based baselines*

1) We observe that the conventional CNN usually had the worst performance among the various CNN-based methods. This verifies that the conventional CNN should be redesigned to incorporate the distance information related to targets.

2) The TCNN-concat performs worse than the distance-aware methods, i.e., TCNN-hard^{nbr}, TCNN-hard and TCNN-soft. This implies that the distance information

[6]Since the parsing result for the multi-target dataset is not publicly available, we ran the syntactic parser ourselves and evaluated TDparse on the returned parsing result. However, we obtained performance lower than the originally reported results, so we report here the numbers in the original paper.

Table 5: Performance comparison on tweets that contain more than one type of sentiment. † indicates that the result of paired t-test is significant against BILSTM-ATT-G at $p < 0.05$.

Model	Accuracy	Macro-F1
BILSTM-ATT-G	$0.508_{\pm 0.0120}$	$0.406_{\pm 0.0193}$
TCNN-hard	**0.524**† $_{\pm 0.0092}$	**0.420**† $_{\pm 0.0120}$
TCNN-soft	0.520† $_{\pm 0.0118}$	$0.416_{\pm 0.0082}$

of each word regarding the target is beneficial for TLSA.

3) The TCNN-hard^{nbr} performs worse than TCNN-hard and TCNN-soft. This implies that the overall sentiment of the input text should not be overlooked when judging the sentiment polarity of a target.

4) It is worth noting that CNN shows relatively high performance on the single-target dataset, whereas TCNN-hard^{nbr} performs relatively well on the multi-target dataset. We posit that because the target-relevant words are relatively farther spread out in single-target sentences compared to multi-target cases, the CNN can capture somewhat relevant features even without the distance information. Conversely, as target-relevant words tend to be located near the target for the multi-target case, TCNN-hard^{nbr} performs relatively well in multi-target scenarios. We will discuss about this observation in more detail in Section 5.2.

5.1.3. Analysis of our proposed methods

1) Both the TCNN-hard and TCNN-soft outperform the existing state-of-the-art TLSA methods. This verifies that our proposed methods successfully model the distance information of each word in the text in relation to the target word(s), by redesigning the CNN architecture to focus on the target. Note that all the improvements of our best performing method TCNN-hard are statistically significant with $p < 0.05$.

2) Table 5 lists the evaluation results for multi-target datasets whose tweets contain

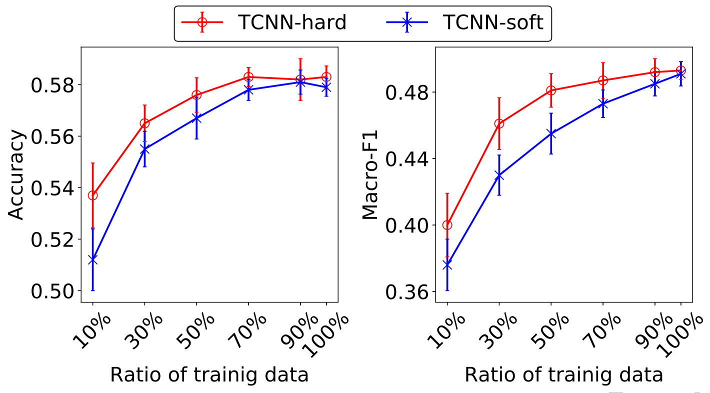


Figure 2: Performance comparisons on multi-target dataset between TCNN-hard and TCNN-soft over various ratios of training data.

more than one type of sentiment[7], a more challenging task than when multiple targets in a tweet share the same sentiment. We observe that our proposed methods outperform BILSTM-ATT-G, indicating the superiority of our methods.

3) Notably, TCNN-hard consistently outperforms TCNN-soft as shown in Table 4, despite the fact that TCNN-soft is an enhanced version that considers the interactions between the target and all other words in the sentence. We attribute such performance to the fact that the size of our benchmark datasets are relatively small while the number of parameters increases in TCNN-soft, which give rise to overfitting in the TCNN-soft method.

To empirically validate this phenomenon, we trained both methods on varied ratios of training data. Figure 2 graphs the results. We observe that as the ratio of training data grows larger, the performance gap between TCNN-hard and TCNN-soft decreases. This implies that TCNN-soft perform better on larger datasets. Note that the single-target dataset used in our experiments is a widely used benchmark dataset that was employed in all our TLSA baselines. Moreover, the multi-target dataset is the largest multi-target dataset publicly available.

[7]The multi-target dataset contains 43% of such tweets.

4) The TCNN-hard+soft combines both the TCNN-hard and TCNN-soft, training jointly by concatenating their penultimate layers, which is fed into the final softmax layer. We observe that TCNN-hard+soft tends to perform worse than TCNN-hard . We conjecture that the pool performance is the result of redundant incorporation of distance information.

5) Finally, we apply the distance-embedding of the TCNN-soft on an LSTM to verify which deep-learning architecture is more appropriate for modeling distance information. In the Table 4, LSTM refers a vanilla LSTM that takes only words as input and LSTM-soft refers a LSTM that takes the distance-augmented input used for TCNN-soft. The LSTM-soft outperforms the vanilla LSTM on both datasets, which means the distance information is also effective for use of RNN-based approaches. However, LSTM-soft performs worse than TCNN-soft on both datasets, which confirms that CNNs are better suited for modeling distance information than RNNs.

5.2. *Qualitative analysis (Q.2)*

In this subsection, we conduct extensive qualitative analyses to answer the following questions:

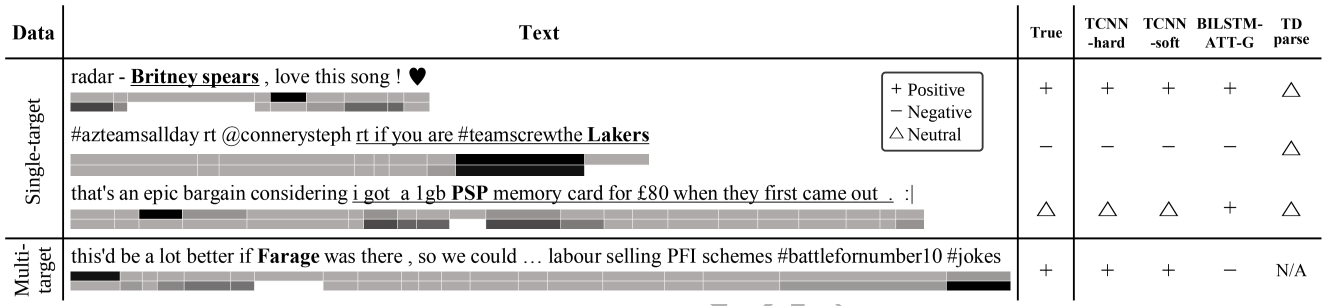
1) Why do the state-of-the-art baselines, TDparse and BILSTM-ATT-G, perform poorly in TLSA

2) Is the distance information actually reflected in TCNN-soft?

Note that in Figure 3, we show dependent words parsed by TDparse and visualize attention weights learned by BILSTM-ATT-G.

Recall that TDparse uses a dependency parser to first extract target-dependent words that are assumed to be relevant to the target, and later uses the extracted words as inputs to an SVM classifier. Figure 3 shows examples of classification results.

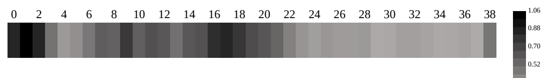
In the first example, the parser fails to extract any words relevant to the target *Britney Spears*. Thus, TDparse fails to predict the correct sentiment label. In the second and third examples, although the parser correctly extracts target-relevant words,



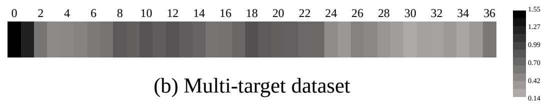
Data	Text	True	TCNN -hard	TCNN -soft	BILSTM-ATT-G	TD parse
Single-target	radar - **Britney spears** , love this song ! ♥	+	+	+	+	△
	#azteamsallday rt @connerysteph rt if you are #teamscrewthe **Lakers**	−	−	−	−	△
	that's an epic bargain considering i got a 1gb **PSP** memory card for £80 when they first came out . :|	△	△	△	+	△
Multi-target	this'd be a lot better if **Farage** was there , so we could … labour selling PFI schemes #battlefornumber10 #jokes	+	+	+	−	N/A

+ Positive
− Negative
△ Neutral

Figure 3: Examples of classification results. The target is in bold and the dependent words parsed by TDparse are underlined. Heat maps represent attentions learned by BILSTM-ATT-G (top: the attention weights learned from the entire text, bottom: the attention weights learned from left- and right-side texts.).



(a) Single-target dataset



(b) Multi-target dataset

Figure 4: Heat map of L2-norm of distance embedding vectors on both datasets.

TDparse provides an incorrect prediction on the second example. This is because the negative neighboring word "#teamscrewthe" of the target "*Lakers*" is not present in the pretrained vocabularies set [25], and thus its embedding vector is randomly initialized while training the SVM[8] of TDparse. In contrast, deep learning based methods such as BILSTM-ATT-G and TCNN-hard provide correct predictions because the word "#teamscrewthe" is updated during the model training.

In the third and fourth examples, we observe that BILSTM-ATT-G provides incorrect predictions. We attribute this poor performance to the fact that RNN-based methods overlook the distance information between each word in the text and the target. Recall that BILSTM-ATT-G exploits gate units to balance the contributions among the attentive words trained from the entire text, the left-side of target text and the right-side of target text. Take the third case as an example. From the entire text, the model assigns the word "epic" the highest attention. From the left-side text and the right-side text of "*PSP*", the words "memory" and "card" received the highest attention. Since the word "epic" received the highest attention weight, it is considered as the most important word, which eventually results in an incorrect prediction of the target-level sentiment regarding "epic". Similar explanations can be applied to the fourth case.

In contrast, our proposed methods overcome the limitations of TDparse and BILSTM-ATT-G by updating word vectors during the model training and focusing on neighboring words based on their distance from the target. Figure 4 shows the visualization of the trained distance embedding vectors. The numbers above the bars indicate the absolute distance from the target, where 0 denotes the target. The color depth indicates the

[8]Every word vector is fixed during the training of SVM.

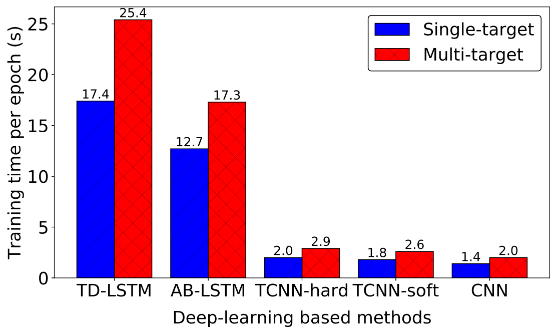


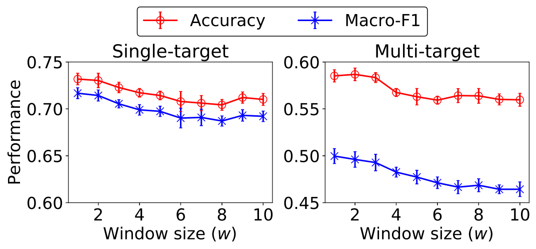
Figure 5: Training time per epoch.

L2-norm of each distance embedding vector, the darker the larger. We observe that for both datasets, words located closer to the target are more important than words farther away from the target, which empirically verifies the initial motivation for this work. Note that in the single-target case (Figure 4a), even words that are relatively far from the target also tend to be assigned high importance values, unlike in the multi-target case (Figure 4b). We hypothesize that when writing a sentence about multiple targets, authors tend to write clue words pertaining each target near the target itself, whereas in case of single-target, clue words tend to be spread throughout the sentence. For this reason, our method also focuses on words that are relatively far from the target during the training process for the single-target dataset.

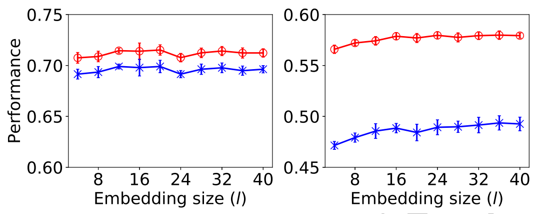
5.3. Training time analysis (Q.3)

In this section, we analyze the training time of deep learning based methods including our proposed methods (Figure 5). Deep learning based methods generally outperform TDparse which uses an SVM with manually generated features as inputs, while requiring many more parameters to tune and a large volume of data to train. This implies that the training time of deep learning cannot be neglected. For fair comparisons, we fix the mini-batch size to 32 for all methods being compared. Note that BILSTM-ATT-G is excluded because their code cannot run on GPU[9].

[9]As an indirect comparison, TD-LSTM consistently trains faster than BILSTM-ATT-G whose architecture subsumes that of TD-LSTM.



(a) Results on various ws. (TCNN-hard)



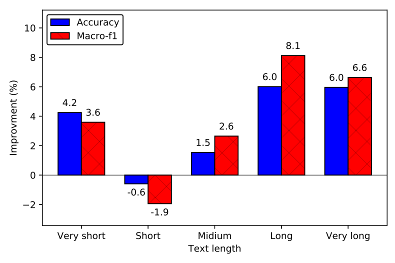
(b) Results on various ls. (TCNN-soft)

Figure 6: Analysis on hyperparameters.

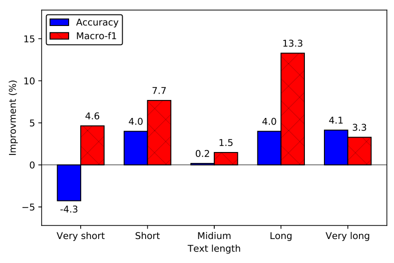
For both datasets, our proposed methods require substantially shorter training time than the RNN-based baselines because the CNN is easily parallelizable on GPU, unlike the RNN. More precisely, the training time for TCNN-hard is six times faster per epoch compared to that of AB-LSTM on both datasets. This demonstrates that our proposed methods not only improve performance but also reduce the training time.

5.4. *Hyperparameter analysis (Q.4)*

Our proposed methods introduce two new hyperparameters: the size w of the "neighbor window" in the TCNN-hard, and the dimension size l of the distance embedding in the TCNN-soft. The TCNN-hard performs better with a smaller w than with larger values of w on both datasets (Figure 6a), which implies that lower numbers words close to the target are likely to be relevant to the target. Notably, the performance on the multi-target dataset drops more rapidly than on the single-target as w gets larger. This add more support our observation that target-relevant words tend to be located nearer to their respective targets in the multi-target dataset, and thus larger w degrades the performance.



(a) Single-target dataset



(b) Multi-target dataset

Figure 7: Performance improvement of TCNN-hard over BILSTM-ATT-G on various text lengths

Figure 6b shows the model performance over different dimension sizes of distance embedding. The performance tends to improve as l increases on the multi-target dataset (Figure 6b), whereas not much performance difference occurs over various ls on the single-target dataset. This confirms our analysis in Section 5.2 that in the single-target scenarios, clue words regarding the target are likely to be spread throughout a sentence, and thus the distance information is less critical in single-target cases than in the multi-target cases. Therefore, as long as the distance information is considered in the form of the distance embedding, the model performance is not greatly affected by its dimension size l.

5.5. *Performance comparison over text length (Q.5)*

We investigate the performance of methods over various text lengths to better understand the behavior of our proposed method. Figure 7 illustrates the results. We first divide test data of each dataset into five subsets based on their text length: *very short, short, medium, long,* and *very long* (Table 6). Then, we evaluate the performance of TCNN-hard and BILSTM-ATT-G on the subsets, while training the models with the original training data for each dataset. For a more intuitive understanding, we will now examine the performance improvement of TCNN-hard over BILSTM-ATT-G.

Observe that BILSTM-ATT-G focuses on specific text lengths so that the improvement of TCNN-hard over BILSTM-ATT-G is relatively small for some text lengths.

Table 6: Statistics of test data over text lengths. x indicates a text length and Avg. is an average number of words.

Dataset	Info.	Very short	Short	Medium	Long	Very long	Avg.
Single-target	Length	$x < 10$	$10 \leq x < 16$	$16 \leq x < 23$	$23 \leq x < 28$	$x \geq 28$	19.5
	Ratio	9.8%	19.2%	36.4%	21.6%	13.0%	
Multi-target	Length	$x < 17$	$17 \leq x < 22$	$22 \leq x < 27$	$27 \leq x < 29$	$x \geq 29$	23.9
	Ratio	8.2%	16.1%	44.7%	19.3%	11.7%	

However, TCNN-hard does demonstrate improvement over BILSTM-ATT-G at most text lengths. More specifically, TCNN-hard achieves its most substantial improvements on long and very long texts from both datasets than on short text. This means TCNN-hard effectively filters target-irrelevant words from long texts by using distance information. With this observation, we can infer that the explicit-distance information is an important feature to find target-relevant words in texts for successful TLSA.

6. Conclusion

In this paper, we propose novel CNN-based methods tailored to TLSA. Our proposed methods address the constrains of manually generating the features and, more importantly, the inherent limitation of RNN-based methods which fail to model the explicit-distance information for each word in the text from the target words.

Experimental results demonstrate that our proposed methods outperform the state-of-the-art baselines on both single-target and multi-target datasets. Our comprehensive qualitative analyses empirically explain the poor performance of our baselines and the superior performance of our proposed methods. Furthermore, our proposed methods train six times faster than the RNN-based baselines, indicating grate potential for practical applications.

We have two directions of future work to improve the proposed distance-aware CNNs. One direction is to apply a distance-guided attention mechanism to the proposed CNN to visualize the importance of each word while considering its distance from given target. This attention mechanism can give the proposed CNN the interpretability for words in each text. We also expect the methods to focus on important words more accurately. The second research direction is to design an advanced architecture using a capsule network [28] that is proposed to enhance the max-pooling operation of CNN. The limitation of the max-pooling is the loss of positional information after the operation, which can prevent our proposed methods from capturing positional relations between words, e.g., the order of the words, after the max-pooling operation. Therefore, we expect the capsule network will enhance the performance of our proposed methods.

7. Acknowledgements

This research was supported by the Ministry of Science and ICT (MSIT), Korea, under the Information and Communication Technology (ICT) Consilience Creative program (IITP-2019-2011-1-00783) and (IITP-2018-0-00584) supervised by the Institute for Information & communications Technology Planning & Evaluation (IITP), and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the MSIT (NRF-2017M3C4A7063570) and (NRF-2016R1E1A1A01942642).

References

References

[1] Amplayo, R. K., Kim, J., Sung, S., & Hwang, S.-w. (2018). Cold-start aware user and product attention for sentiment classification. *arXiv preprint arXiv:1806.05507*, .

[2] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, .

[3] Baziotis, C., Pelekis, N., & Doulkeridis, C. (2017). Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 747–754).

[4] Bhatia, P., Ji, Y., & Eisenstein, J. (2015). Better document-level sentiment analysis from rst discourse parsing. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (pp. 2212–2218).

[5] Cai, Z., Fan, Q., Feris, R. S., & Vasconcelos, N. (2016). A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision* (pp. 354–370). Springer.

[6] Cliche, M. (2017). Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 573–580).

[7] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*, 2493–2537.

[8] Dahou, A., Xiong, S., Zhou, J., Haddoud, M. H., & Duan, P. (2016). Word embeddings and convolutional neural network for arabic sentiment classification. In *COLING* (pp. 2418–2427).

[9] Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M., & Xu, K. (2014). Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL (2)* (pp. 49–54).

[10] Dos Santos, C. N., & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *COLING* (pp. 69–78).

[11] Hu, B., Lu, Z., Li, H., & Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems* (pp. 2042–2050).

[12] Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, .

[13] Kim, S.-M., & Hovy, E. (2004). Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics* (p. 1367). Association for Computational Linguistics.

[14] Kim, Y. (2014). Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, (pp. 1746–1751).

[15] Kim, Y., Denton, C., Hoang, L., & Rush, A. M. (2017). Structured attention networks. *arXiv preprint arXiv:1702.00887*, .

[16] Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, .

[17] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

[18] Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, *5*, 1–167.

[19] Liu, J., & Zhang, Y. (2017). Attention modeling for targeted sentiment. *EACL 2017*, (p. 572).

[20] Ma, D., Li, S., Zhang, X., & Wang, H. (2017). Interactive attention networks for aspect-level sentiment classification. *arXiv preprint arXiv:1709.00893*, .

[21] Ma, Y., Peng, H., & Cambria, E. (2018). Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *Proceedings of AAAI*.

[22] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807–814).

[23] Pang, B., Lee, L. et al. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, *2*, 1–135.

[24] Parikh, A. P., Täckström, O., Das, D., & Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, .

[25] Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). URL: `http://www.aclweb.org/anthology/D14-1162`.

[26] Ren, Y., Zhang, Y., Zhang, M., & Ji, D. (2016). Context-sensitive twitter sentiment classification using neural network. In *AAAI* (pp. 215–221).

[27] Rosenthal, S., Farra, N., & Nakov, P. (2017). Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 502–518).

[28] Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems* (pp. 3856–3866).

[29] Segura-Bedmar, I., Quirós, A., & Martínez, P. (2017). Exploring convolutional neural networks for sentiment analysis of spanish tweets. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers* (pp. 1014–1022). volume 1.

[30] Severyn, A., & Moschitti, A. (2015). Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 959–962). ACM.

[31] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, .

[32] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

[33] Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, .

[34] Tang, D., Qin, B., Feng, X., & Liu, T. (2016). Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics* (pp. 3298–3307).

[35] Tang, D., Qin, B., & Liu, T. (2015). Deep learning for sentiment analysis: successful approaches and future challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *5*, 292–303.

[36] Tang, D., Qin, B., Liu, T., & Yang, Y. (2015). User modeling with neural network for review rating prediction. In *IJCAI* (pp. 1340–1346).

[37] Tay, Y., Luu, A. T., & Hui, S. C. (2018). Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. In *Proceedings of AAAI*.

[38] Tay, Y., Luu, A. T., Hui, S. C., & Su, J. (2018). Attentive gated lexicon reader with contrastive contextual co-attention for sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 3443–3453).

[39] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998–6008).

[40] Vo, D.-T., & Zhang, Y. (2015). Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI* (pp. 1347–1353).

[41] Wang, B., Liakata, M., Zubiaga, A., & Procter, R. (2017). Tdparse: Multi-target-specific sentiment recognition on twitter. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers* (pp. 483–493). volume 1.

[42] Wang, Y., Chen, Q., Liu, X., Ahmed, M., Li, Z., Pan, W., & Liu, H. (2018). Senhint: A joint framework for aspect-level sentiment analysis by deep neural networks and linguistic hints. In *Companion of the The Web Conference 2018 on The Web Conference 2018* (pp. 207–210). International World Wide Web Conferences Steering Committee.

[43] Wang, Y., Huang, M., Zhao, L., & Zhu, X. (2016). Attention-based lstm for aspect-level sentiment classification. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (pp. 606–615).

[44] Yang, M., Tu, W., Wang, J., Xu, F., & Chen, X. (2017). Attention based lstm for target dependent sentiment classification. In *AAAI* (pp. 5013–5014).

[45] Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J. et al. (2014). Relation classification via convolutional deep neural network. In *COLING* (pp. 2335–2344).

[46] Zhang, M., Zhang, Y., & Vo, D.-T. (2016). Gated neural networks for targeted sentiment analysis. In *AAAI* (pp. 3087–3093).