Learning Sentence Representations over Tree Structures for Target-dependent Classification

Junwen Duan, Xiao Ding, Ting Liu*
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, Harbin, China
{jwduan,xding,tliu}@ir.hit.edu.cn

Abstract

Target-dependent classification tasks, such as aspect-level sentiment analysis, perform fine-grained classifications towards specific targets. Semantic compositions over tree structures are promising for such tasks, as they can potentially capture long-distance interactions between targets and their contexts. However, previous work that operates on tree structures resorts to syntactic parsers or Treebank annotations, which are either subject to noise in informal texts or highly expensive to obtain. To address above issues, we propose a reinforcement learning based approach, which automatically induces target-specific sentence representations over tree structures. The underlying model is a RNN encoder-decoder that explores possible binary tree structures and a reward mechanism that encourages structures that improve performances on downstream tasks. We evaluate our approach on two benchmark tasks: firm-specific cumulative abnormal return prediction (based on formal news texts) and aspect-level sentiment analysis (based on informal social media texts). Experimental results show that our model gives superior performances compared to previous work that operates on parsed trees. Moreover, our approach gives some intuitions on how target-specific sentence representations can be achieved from its word constituents.

1 Introduction

We investigate target-dependent classification problem in this paper, with a special focus on the sentence level. Target-dependent classification aims to identify the fine-grained polarities of sentences towards specific targets, which is challenging but also important for deep text understanding. The definitions of polarity vary across different tasks, which can be *positive* or *negative* in aspect-level sentiment analysis, *favor* or *against* in stance detection, and *rise* or *drop* in financial news analysis towards the stock price movement of a particular firm.

Task	Example
Aspect-level Sentiment Analysis	The **food** is good but the **service** is dreadful.
Stance Detection	I don't care about **global climate change**.
Firm-specific Financial News Analysis	**Nike** sues **Wal-Mart** for Patent Infringement.

Table 1: Samples of target-dependent classification tasks. The targets of interest are in bold.

Table 1 gives examples of three target-dependent classification tasks. We can find that there can be multiple target mentions in the same text scope, which makes it challenging for generic sentence representation approaches. For the first example, a restaurant manager or a potential customer may be interested in both *food* and *service*; however, the sentiment polarities towards the two targets are different. Hence, it would be beneficial for such tasks to tailor the sentence representations with respect to particular targets.

Tree structures are promising for such tasks, as they can potentially capture long-distance dependencies between target words and their contexts (Li et al., 2015). Therefore, it is not surprising to find work that exploits the syntactically parsed trees for learning target-specific sentence representations. Dong et al. (2014) and Chang et al. (2016) adapted the word orders in a parsed tree, depending on their distances to the target entities. Nguyen et al. (2015) extended Dong et al. (2014) by combining the constituency tree and the dependency tree of a sentence. An important assumption of such work is that different tree structures lead to different semantic representations even for the same sentence. However, they all resort to ex-

* Corresponding author

Proceedings of NAACL-HLT 2018, pages 551–560
New Orleans, Louisiana, June 1 - 6, 2018.

ternal syntactic resources, such as parse trees or Treebank annotations (Marcus et al., 1993), which limits their broader applications. On the one hand, annotated data are highly expensive to produce; and informal texts, such those on the social media, remain a challenge for syntactic parsers (Kong et al., 2014). On the other hand, the tree structures in their pipeline-style architecture are fixed during training, which cascade errors to later representation learning stage.

A desirable solution would be to automatically and dynamically induce the tree structures for target-specific sentence representations. However, the challenge is that the absence of external supervisions makes it difficult to evaluate the quality of the tree structures and train the parameters. Inspired by Yogatama et al. (2016), we propose a reinforcement learning based approach that integrates target information and generates target-specific tree structures that benefit downstream classification tasks.

The underlying framework consists of two key components, a RNN encoder-decoder that explores possible binary tree structures according to a given target, and a tree-structured neural network that composes the input words into sentence representation based on the structure. The REINFORCE algorithm with the self-critic baseline (Rennie et al., 2016) is applied to update the parameters of the two components.

We evaluate our approach on two benchmark tasks: a firm-specific cumulative abnormal return prediction task (based on formal news texts) and an aspect-level sentiment analysis task (based on informal social media texts). Experimental results show that our approach achieves superior performances compared to baseline methods that operate on parsed trees. Moreover, our model sheds lights on understanding how sentences are composed from its word constituents towards specific targets.

2 Problem Definition

We formalize the problem of learning sentence representations for target-dependent classification tasks as constructing and semantically composing the target-specific binary syntactic trees of sentences. The input of the model is a tuple $(\mathbf{x}, x_{target}, c_{target})$, in which $\mathbf{x}$ is a sentence of n words $\{x_1, x_2, \cdots, x_n\}$; x_{target} is the target of interest mentioned in the sentence and c_{target} is the polarity regarding the target. For sentence $\mathbf{x}$, we can construct a valid binary syntactic tree by n SHIFT and $n-1$ REDUCE transitions $\mathbf{a} = \{a_0, a_1, \cdots, a_{2n-1}\}$, in which $a_t \in \{\text{SHIFT}, \text{REDUCE}\}$ specifies the transition taken at step t. The SHIFT transition adds a leaf node to the tree while the REDUCE transition combines two leaf nodes to form a parent node.

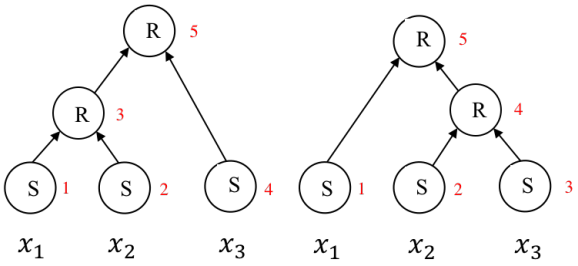

Figure 1: For input sequence $\{x_1, x_2, x_3\}$, the shift-reduce orders can be {S,S,R,S,R} and {S,S,S,R,R}, where **S** stands for SHIFT and **R** stands for REDUCE.

Figure 1 illustrates two examples on how can we construct a binary tree by only using SHIFT and REDUCE transitions and how can we obtain different binary trees by varying the SHIFT-REDUCE transition orders.

We design a transition generator $\mathbf{G}$ (Section 3.1) for generating transition orders $\mathbf{a}$, $\mathbf{G}(\mathbf{x}, x_{target}) \rightarrow \mathbf{a}$ and a composition function $\mathbf{C}$ (Section 3.2) that composes sentence $\mathbf{x}$ following the transition orders $\mathbf{a}$ into sentence representation $\mathbf{s}$, $\mathbf{C}(\mathbf{a}, \mathbf{x}) \rightarrow \mathbf{s}$.

Our ultimate goal is to use the sentence representation $\mathbf{s}$ for target-dependent classification. The objective is thus to minimize the negative log-likelihood Eq 1 with L2 norm, in which θ denotes all the parameters of our model.

$$J(\theta) = -\log P(c_{target}|\mathbf{s}; \theta) + \lambda||\theta||_2 \quad (1)$$

3 Model

The architecture of our proposed approach is illustrated in Figure 2, which is made up of two main components, a transition generator $\mathbf{G}$ and a composition function $\mathbf{C}$. The transition generator is a RNN encoder-decoder that generates discrete target-specific SHIFT-REDUCE transition orders, given a sentence and the target of interest. The composition function is a tree-structured neural network that semantically composes the word constituents following the transition orders. The main challenges for such a framework are two-fold. On

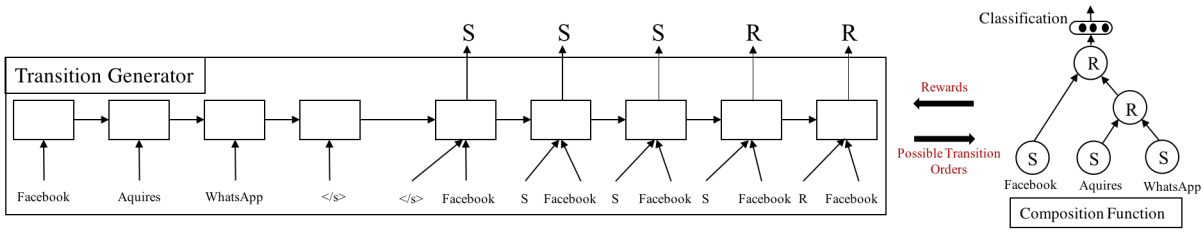

Figure 2: The framework of our proposed method. The left side is a variant of standard encoder-decoder that generates discrete SHIFT-REDUCE transition orders. It considers the target information at decoding. The right side is a composition function that semantically composes word representations into sentence representation following the transition orders. REINFORCE with the self-critical baseline is applied to reward the generated structures and update the parameters.

the one hand, the transition generator is fully unsupervised as we do not resort to external syntactic resources. On the other hand, the transitions generated at each step are discrete, making it difficult to train and propagate errors to update the model parameters. We give details of the two components and how we address the challenges in this section.

3.1 Transition Generator

The basic idea of the transition generator is to generate different transition orders given different targets. We propose using the RNN encoder-decoder framework (Cho et al., 2014), which has shown capacity in shift-reduce parsing (Vinyals et al., 2015; Liu and Zhang, 2017b). A standard RNN encoder-decoder contains two recurrent neural networks, one for encoding a sequence of variable-length into a vector representation and the other for decoding the representation back into another variable-length sequence.

Encoder We employ a standard Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) as our encoder. Given the input sentence $\{x_1, x_2, \cdots, x_n\}$, we first obtain their word vectors $\{\overrightarrow{e}(x_1), \overrightarrow{e}(x_2), \cdots, \overrightarrow{e}(x_n)\}$ by looking them up from a pre-trained embedding matrix $\overrightarrow{e}$. We reverse the input sentence and feed their word embeddings sequentially to the LSTM. The hidden states of each token $\{h_1, h_2, \cdots, h_n\}$ are kept for the decoding stage. The hidden state and cell state of the last LSTM unit are used as the initial states for decoder.

Decoder Following Bahdanau et al. (2014), we use an attention-based decoder. The decoder aligns with all the encoder hidden states at each step of decoding to obtain a context vector c_t, such that each input words show different weights at decoding. We denote the hidden states of our decoder as $\{d_1, d_2, \cdots, d_{2n-1}\}$. The attention score over each of the encoder hidden state h_i is computed by:

$$u_t^i = d_{t-1} \odot h_i \tag{2}$$

$$a_t^i = \frac{\exp(u_t^i)}{\sum_{i'} \exp(u_t^{i'})} \tag{3}$$

$$c_t = \sum_{i=1}^{n} a_t^i \cdot h_i, \tag{4}$$

in which $\odot$ denotes element-wise dot product; a_t^i is the normalized attention score and the context vector c_t is a weighted sum of all the encoder hidden states.

To enable the target of interest to influence the decoding process, we enrich the input of the decoder by concatenating the target entity. The hidden state of the decoder at time t is obtained by:

$$d_t = \text{LSTM}(c_t \oplus \overrightarrow{e}(a_{t-1}) \oplus \overrightarrow{e}(x_{target}), d_{t-1}), \tag{5}$$

in which $\oplus$ denotes concatenation operation; x_{target} is the embedding of the target entity; $\overrightarrow{e}(a_{t-1})$ is the embedding of the last decoded transition and c_t is the context vector.

Decoding In a supervised RNN decoder setting, the goal of each step is to estimate the conditional probability

$$P(a_t|a_{1:t-1}, c_t, d_t) = g(a_{t-1}, c_t, d_t), \tag{6}$$

in which $a_{1:t-1}$ are previously decoded transitions, c_t is the context vector, d_t is current decoder hidden state and g is non-linear network. $P(a_t|a_{1:t-1}, c_t, d_t\}$ is a distribution over the transition space $\{\text{SHIFT}, \text{REDUCE}\}$. By comparing

the decoded outputs with the ground-truth labels, the prediction errors can back-propagate to update parameters of the encoder-decoder network.

However, it is no more applicable in our settings, as we do not have any explicit supervisions from external syntactic resources. To make training the transition generator possible, we resort to a reinforcement learning framework, obtaining the transitions by sampling from a policy network. We represent the current state $\mathbb{S}_t$ by concatenating $\overrightarrow{e}(a_{t-1}), \overrightarrow{e}(x_{target}), c_t, d_t$.

$$\mathbb{S}_t = [\overrightarrow{e}(a_{t-1}) \oplus \overrightarrow{e}(x_{target}) \oplus c_t \oplus d_t] \quad (7)$$

The policy network $\pi(a_t|\mathbb{S}_t)$ is defined by Eq 8,

$$\pi(a_t|\mathbb{S}_t) \propto \exp(g(\mathbb{S}_t)), \quad (8)$$

in which g is a one-layer non-linear feed-forward neural network. We decode the transition a_t by sampling from the distributions given by the policy network.

3.2 Composition Function

When a valid binary tree of a sentence is generated, we use the composition function to obtain the representation following the transition orders. We maintain two data structures at composition; a buffer that stores words yet to be processed and a stack that stores the partially completed subtrees. Initially, the stack is empty, and the buffer stores all the words in the sentence. The operations specified by SHIFT and REDUCE are as follows.

- For a SHIFT transition, the buffer pops the topmost word out and pushes it to the top of the stack.
- For a REDUCE transition, the topmost two elements of the stack are popped out and composed. Their compositions are then pushed back to the stack.

To produce a valid binary tree, we follow Yogatama et al. (2016) to disallow SHIFT transition when the buffer is empty and forbid REDUCE transition when the stack has no more than two elements.

We use a tree-LSTM (Tai et al., 2015) to semantically compose the top two elements of the stack. Initially, the hidden state h_t and the cell state s_t of leaf nodes are given by another LSTM. The tree-LSTM works as follows,

$$\begin{bmatrix} i_t \\ f_t^l \\ f_t^r \\ o_t \\ g_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \cdot W \cdot \begin{bmatrix} h_t^l \\ h_t^r \end{bmatrix} \quad (9)$$

$$s_t = f_t^l \odot s_t^l + f_t^r \odot s_t^r + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(\hat{s}_t),$$

in which $\odot$ denotes element-wise dot product; i_t and o_t are the input and output gate, respectively; f_t^l and f_t^r are the left and right forget gates; $h_t^l, h_t^r, s_t^l, s_t^r$ are the hidden and cell states of the left and right nodes in the subtree. The hidden state of the topmost node is used as the representation for the input sentence.

3.3 Training with REINFORCE

The goal for training is to optimize the parameters of the transition generator θ_G and the composition function θ_C. It is easy to optimize θ_C, the output of which is directly connected to the classifier, the classification loss can back-propagate to update its parameters.

However, the transitions sampled from the policy network $\pi(\mathbf{a}|\mathbb{S})$ are discrete, which makes θ_G no more differentiable to our objective. A possible solution is to maximize the expected reward $\mathbb{E}_{p(\mathbf{a};\theta_G)}R(\mathbf{a})$. As we are in a reinforcement learning setting, we can immediately receive a reward $\mathrm{R}(\mathbf{a})$ for transitions $\mathbf{a} = \{a_1, a_2, \cdots, a_t\}$ at the end of the classification. The reward is defined as the logarithm of classification probability for the right label c_{target}, $R(\mathbf{a}) = \log P(c_{target}|C(\mathbf{a}, \mathbf{x}))$.

However, it is computationally intractable to compute $\mathbb{E}_{p(\mathbf{a};\theta_G)}R(\mathbf{a})$, as the number of possible transition orders $\mathbf{a}$ is exponentially large. To address this, we use the REINFORCE algorithm to approximate the gradients by running M examples.

$$\bigtriangledown_{\theta_G} J(\theta_G) \approx -\frac{1}{M}\sum_{m=1}^{M}[\bigtriangledown_{\theta_G} \log p(\mathbf{a}) R^m(\mathbf{a})] \quad (10)$$

The $\bigtriangledown_{\theta_G} \log p(\mathbf{a})$ can be used to update θ_G.

REINFORCE algorithm is non-biased but may have high variance. To reduce the variance, a widely used trick is to subtract a baseline from the reward. It has been theoretically proven that

any baselines that do not depend on the actions are applicable. In this paper, we follow Rennie et al. (2016) to apply a self-critical baseline to the rewards. Rather than estimating a baseline reward, the self-critical method uses the outputs given by the test-time inference algorithm as the baselines. This can thus alleviate the over-fitting problem on test dataset.

At inference, we use a greedy decoding strategy by selecting the most probable transitions given by the policy network (Eq 8).

$$\hat{a}_i = \arg\max_{a_i} \pi(a_i|\mathbb{S}_t) \tag{11}$$

The self-critical baseline reward is $R(\hat{\mathbf{a}}) = \log P(c_{target}|C(\hat{\mathbf{a}}, \mathbf{x}))$, the formula to update θ_G become Eq 12.

$$\begin{aligned} &\nabla_{\theta_G} J(\theta_G) \approx \\ &-\frac{1}{M}\sum_{m=1}^{M}[\nabla_{\theta_G} \log p(\mathbf{a})(R^m(\mathbf{a}) - R^m(\hat{\mathbf{a}}))] \end{aligned} \tag{12}$$

4 Experiments and Results

The proposed approach is evaluated on two aspect-level tasks: (1) firm-oriented cumulative abnormal return prediction on formal financial news texts and (2) aspect-level sentiment analysis on informal social media texts.

4.1 Firm-specific cumulative abnormal return prediction

Firm-specific Cumulative Abnormal Return (CAR) prediction task (Chang et al., 2016) studies the impact of new information towards a specific firm. Multiple firms may be involved in the same new event, however, the event can present different impacts to these firms. Conceptually, **Abnormal Return** is the difference between the actual return of a stock and its expected return. The expected return can be approximated by daily indexes, such as S&P 500 index. For example, if a stock is expected to rise by 5%, but on the event day, it rises by 2%, although it gives a positive return, the abnormal return is -3%. **Cumulative Abnormal Return** is the accumulated abnormal return in an event window, which is usually triggered by new events. We use a three-day window (-1, 0, 1), denoted as CAR_3, with event day centering at day 0. We predict whether an event has positive or negative impact to the cumulative abnormal return of a given firm.

	Training	Development	Test
$+CAR_3$	7167	354	728
$-CAR_3$	7102	387	731
Total	14269	741	1459
Firms	1216	302	424

Table 2: Number of CAR_3 in the datasets

4.1.1 Data

We use the same news dataset as Chang et al. (2016), which are abstracts extracted from the Reuters news dataset released by Ding et al. (2014; 2015; 2016). Compared to the full texts of news documents, abstracts are supposed to be more informative and less noisy. Ding et al. (2014) show that modeling abstracts alone can achieve comparable or even better performances compared to full texts in stock market prediction. To better interpret our approach, we only extract event days with a single news document, which covers over 70% cases in the dataset. This final dataset yields a total of 16469 instances, including 1291 firms, of which 10% are reserved for validation, and 20% are used for testing. The numbers of positive and negative CAR_3 examples and number of firms in the subsets are listed in Table 2.

4.1.2 Baseline

To evaluate the performance of our approach on formal news texts, we compare with state-of-the-art target-independent and target-dependent baselines. Among the baselines, *Sentiment-based* and *Bi-LSTM* are target-independent, which learn generic representations for sentences, while *Bi-LSTM + Attention* and *TGT-CTX-TLSTM* are target-dependent.

Sentiment-Based Sentiments among breaking news, earning reports and online message boards, are found to be correlated with market volatility (Schumaker and Chen, 2009; Das and Chen, 2007). We adopt lexicon-based sentiment analysis as our baseline, using the sentiment lexicons released by Loughran and McDonald (2011). We follow the prior literature (Mayew and Venkatachalam, 2012) and use the count of positive words, negative words, the differences between positives and negatives, and their length-normalized values as our feature vectors.

Bi-LSTM We stack a forward and a backward LSTM to capture the contextual representations for the sentence. The last hidden states of both

Parameters	Value
word dimension	200
LSTM hidden dimension	200
dropout probablity	0.5
batch size	64
initial learning rate	0.0005

Table 3: Hyper-parameters for firm-oriented cumulative abnormal return task

directions are concatenated and then used for classification.

Bi-LSTM + Attention We extend vanilla Bi-LSTM by adding an attention mechanism over the hidden states. We concatenated the hidden states $\hat{h}_t = \{h_t^l, h_t^r\}$ of each input token x_t, the target representation $\overrightarrow{e}_{target}$ is adopted to weigh each of the hidden states.

$$u_t = v^\top \tanh(W_1 \overrightarrow{e}(x_{target}) + W_2 h_t + b) \quad (13)$$

$$a_t = softmax(u_t) \quad (14)$$

$$d_t = \sum a_t h_t \quad (15)$$

TGT-CTX-TLSTM The method of Chang et al. (2016), which we follow and is used as our main baseline. It is a hybrid model which integrates both sequential information and syntactic parse tree information. As the first step, the abstract is parsed with an external syntactic parser to obtain the dependency relations between the words. The parse tree are then adapted and binarized depending on their distances to targets in the dependency graph. A tree-structured Long Short-Term Memory Network (Tai et al., 2015) is then applied to learn a vector representation of the binarized tree structure.

4.1.3 Parameters & Metrics

The hyper-parameters used in this paper are listed in Table 3. We pretrain word vectors with the Word2Vec (Mikolov et al., 2013) tool on the news dataset released by Ding et al. (2014), which are fine-tuned during training. The embeddings of target firms are obtained by averaging their words of constituents.

We use macro-F1 to evaluate the performance on both positive and negative classes.

4.1.4 Test Results

The macro-F1 scores of our method and baselines are presented in Table 4. Sentiment-based method

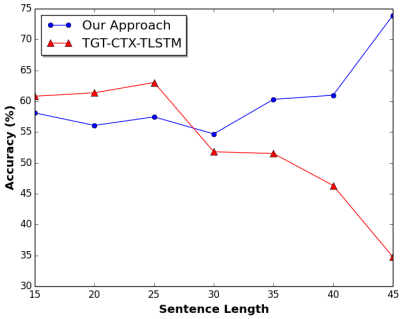

Figure 3: Accuracy with respect to sentence length.

gives the highest F1 score on the positive class. However, its performance is not consistent on the negative class, which suggests that it tends to misclassify the sentence as positive. Bi-LSTM + Attention outperforms the vanilla one without attention and is much robust in both positive and negative analysis. Our approach achieves an overall Macro-F1 of 58.2%, with an F1 score of 57.2% and 59.2% on positive and negative classes, respectively. Compared to the state-of-the-art model that exploits automatically parsed structures, we obtain an over 2% absolute gains without using explicit supervisions in learning the structures.

Method	**Class**	**F1-score**
Sentiment-based	+CAR$_3$	**0.597**
	-CAR$_3$	0.476
	Macro	0.536
Bi-LSTM	+CAR$_3$	0.557
	-CAR$_3$	0.490
	Macro	0.523
Bi-LSTM + Attention	+CAR$_3$	0.575
	-CAR$_3$	0.523
	Macro	0.549
TD-CTX-TLSTM	+CAR$_3$	0.552
	-CAR$_3$	0.570
	Macro	0.561
Our Approach	+CAR$_3$	0.572
	-CAR$_3$	**0.592**
	Macro	**0.582**

Table 4: Results for cumulative abnormal return prediction task

4.1.5 Accuracy Versus Sentence Length

Longer sentences are much more challenging for syntactic parsers. To gain insights on the performances of our approach on long sentences, we further inspect the accuracies with regards to different sentence lengths. As shown in Figure 3, we compare with structure-dependent baseline TGT-CTX-TLSTM. We divide the sentences into seven bins, each of which contains sentences with length $[5 *$

$i, 5 * (i + 1)]$. TGT-CTX-TLSTM gives higher accuracies over sentences with shorter lengths, while the accuracies decline sharply over sentences with lengths of over 30. Our approach is more consistent on both long and short sentences. As the sentence length grows, the accuracy our model gradually increases, showing its robustness and effectiveness across sentences of variable lengths.

4.2 Aspect-level Sentiment Analysis

To verify our proposed approach on informal social media texts, we apply it to aspect-level sentiment analysis on tweets. Aspect-level sentiment analysis aims to identify sentiment polarities towards specific targets mentioned in a sentence. Target-specific sentence representations can be naturally applied to this task.

Dataset	#Target	#Positive	#Negative	#Neutral
Training	6248	1568	1560	3127
Testing	692	173	173	346

Table 5: Statistics of aspect-level sentiment analysis datasets

4.2.1 Dataset

We apply our model to a benchmark aspect-level sentiment analysis dataset used in previous work (Dong et al., 2014). The statistics of the dataset are shown in Table 5. The target entities and corresponding ground-truth labels are annotated. The labels belong to one of {positive, neutral, negative}, thus the task is a three-way classification.

4.2.2 Baselines

We compare our approach with feature-based and neural-based models.

Jiang et al. (2011) They extract rich target-dependent and target-independent lexical and syntactic features for classification.

Dong et al. (2014) They adapt the parse tree of a sentence concerning the target with predefined rules and use recursive neural network (Socher et al., 2013) to learn a target-specific sentence representation.

4.2.3 Parameters & Metrics

The parameter settings are listed in Table 6. We use 100-dimension GloVe vectors which are pre-trained on a large Twitter Corpus (Pennington et al., 2014) and fine-tuned during training.

Parameters	Value
word dimension	100
LSTM hidden dimension	100
dropout probablity	0.5
batch size	32
initial learning rate	0.0005

Table 6: Hyper-parameters for aspect-level sentiment analysis

The commonly-used metrics classification accuracy and macro-F1 are adopted to evaluate the performances.

Model	Acc	F1
Jiang et al.(2011)	63.4	63.3
Dong et al.(2014)	66.3	65.9
Our Method	**68.2**	**66.3**

Table 7: Final results on aspect-level sentiment analysis task

4.2.4 Final Results

The final results on aspect-level sentiment analysis task are shown in Table 7. Dong et al. (2014) are used as our main baseline, as they build target-specific sentence representation over adapted tree structures. Neural-based models outperform Jiang et al. (2011), which did a lot of feature engineerings, showing the effectiveness of automatically induced features. Our approach gives superior performances compared to Dong et al. (2014), which operates on parsed trees. We achieve 68.2% classification accuracy and 66.3 macro-F1. We do not rely on a preprocessing syntactic parser as the first step to obtain the tree structures. On the one hand, social media texts are informal and extremely noisy, which remains a challenge for syntactic parsers. The pipeline-style architecture of Dong et al. (2014) cascades parse errors to later stages, which will hurt the performances on downstream tasks. On the other hand, the adapted tree structures in Dong et al. (2014), while in our approach, the tree structures are also tuned dynamically during training, so as to find the optimal structures that would benefit downstream classification tasks.

4.3 Case Study

To gain further insights on the induced structures, we inspect the shift-reduce trees our approach generated in this section. We present two examples that our model gives high confidences in Figure 4. For the sentence *"Nike NKE.N has sued Wal-Mart WMT.N, saying the world 's largest retailer*

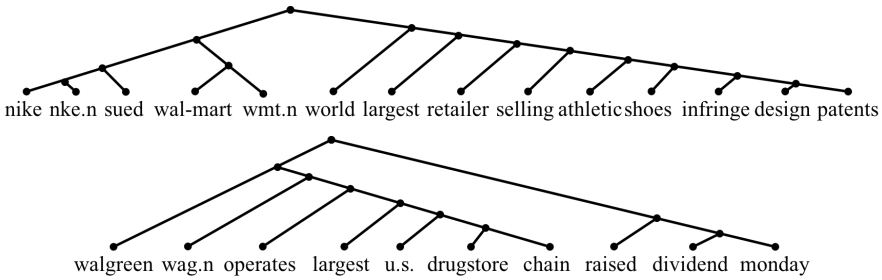

Figure 4: Two tree structures generated by our model. We removed stop words and punctuations. The upper tree structure is for the sentence *"Nike NKE.N has sued Wal-Mart WMT.N, saying the world 's largest retailer is selling athletic shoes that infringe on its design patents"* and the bottom one is for the sentence *"Walgreen WAG.N , which operates the largest U.S. drugstore chain , raised its dividend on Monday."*

is selling athletic shoes that infringe on its design patents", the core part *"Nike sued Wal-mart"* and the rest of the sentence are in two separate subtrees, which reduces potentially information loss about the key event when composing them into sentence representation. Similarly, for the sentence *"Walgreen WAG.N , which operates the largest U.S. drugstore chain , raised its dividend on Monday."*, the model learns to make the target *"Walgreen"* and key event *"raised its dividend on Monday"* close to each other in the tree, although there are sequentially many words in between. These are good examples given by our model, we also find a lot of highly left- or right-biased tree structures. Intuitively, the completely left- and right-biased tree structures are equivalent to forward and backward sequential structures, respectively.

5 Related Work

Our model is related to the following research areas, each having tremendous literatures.

5.1 Target-specific Sentence Representation

It is beneficial for numerous tasks, such as aspect-level sentiment analysis and stance detection, to have the sentence representations being tailored to specific targets. Early approaches rely on feature engineering by extracting target-dependent features (Jiang et al., 2011), while recent work mainly focuses on semantic compositions over the vector space with deep neural models. Depending on how they model the target and context, we further classify related work into three categories.

The first category relies on syntactic parse trees. Dong et al. (2014) are among the first to exploit tree structures, in which they adapt the parse trees based on the dependency relations between the words and the target, and then use a recursive neural network to learn the sentence representations. Similarly, Chang et al. (2016) explore a hybrid model that considers both sequential and structural information of a sentence. Nguyen et al. (2015) extend Dong et al. (2014) by combining the constituency tree and the dependency tree of a sentence. The performances of their methods highly rely on external parsers, which is subject to noise in informal social media texts.

The second category models the interactions between the target and its left context and right context. Vo and Zhang (2015) split a sentence into three parts and use pooling function to automatic inducing features for a given target. Similar to Vo and Zhang (2015), Zhang et al. (2016) exploit the gates instead of pooling functions to control the information flow of contexts. Tang et al. (2015) model by concatenating the word embeddings and target entity embeddings and use two LSTMs to encode left- and right contexts. Liu et al. (2017a) propose to use the attention mechanism to assign different weights to the left and right context depending on the target.

The third category controls the information flow from the target to the sentence representation. Augenstein et al. (2016) use conditional encoding to encode the target and use it as the initial states for the sentence representation.

Our method belongs to the first category that exploits tree structures. The main difference is we do not use external supervision from dependency parser or treebank annotations.

5.2 Neural-based Syntactic Constituency Parsing

Our work is related to syntactic constituency parsing as we build the tree structure in a transition

manner. Syntactic constituency parsing is a fundamental task in natural language processing, which uses phrase structure to organize words into nested constituents. Early approaches rely on probabilistic context-free grammars or transition-based models with rich features (Collins, 1997; Klein and Manning, 2003). Recently, recursive neural network (Socher et al., 2013) and neural-based transition model (Liu and Zhang, 2009) are also applied, which achieve competitive or even better performances compared to traditional state-of-the-art approaches that rely on hand-crafted features. Vinyals et al. (2015), from which we get inspirations, use the RNN Encoder-Decoder to encode the sentence and generate its corresponding full parse tree. Bowman et al. (2016) propose a Stack SPINN framework that integrates parsing and interpreting the sentence in a hybrid model. Yogatama et al. (2016) extend their model by using reinforcement learning to build the tree structures that can improve performances of end tasks.

We differ from the aforementioned approaches in two aspects. First, we do not use any explicit supervisions to guide the decoder. The parameters of our framework are optimized by the objective of end tasks. Another difference is that we learn target-specific instead of general-purpose sentence representations.

6 Conclusion

In this paper, we propose a framework that automatically induces target-specific sentence representations over tree structures without recourse to external syntactic resources. Experimental results on formal and informal texts showed that our approach is both robust and effective compared to previous work that operates on parsed trees. Moreover, the approach gives intuitions on how sentence structures are composed from their word constituents concerning a specific target.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and suggestions to help improve this paper. This work was partly supported by the National Key Basic Research Program of China via grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via grant 61472107 and 61702137.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance Detection with Bidirectional Conditional Encoding pages 876–885. http://arxiv.org/abs/1606.05464.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A Fast Unified Model for Parsing and Sentence Understanding https://doi.org/10.18653/v1/P16-1139.

Ching-Yun Chang, Yue Zhang, Zhiyang Teng Teng, Zahn Bozanic, and Bin Ke. 2016. Measuring the information content of financial news. In *26th Coling*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 16–23.

Sanjiv R Das and Mike Y Chen. 2007. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science* 53(9):1375–1388.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2014. Using structured events to predict stock price movement: An empirical investigation. In *EMNLP*. pages 1415–1425.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *IJCAI*. pages 2327–2333.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2016. Knowledge-driven event embedding for stock prediction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 2133–2142.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL (2)*. pages 49–54.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

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: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 151–160.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 423–430.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1001–1012.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185* .

Jiangming Liu and Yue Zhang. 2009. Shift-Reduce Constituent Parsing with Neural Lookahead Features 1.

Jiangming Liu and Yue Zhang. 2017a. Attention Modeling for Targeted Sentiment. *Short Papers* 2:572–577. https://www.aclweb.org/anthology/E/E17/E17-2091.pdf.

Jiangming Liu and Yue Zhang. 2017b. Encoder-decoder shift-reduce syntactic parsing. *arXiv preprint arXiv:1706.07905* .

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance* 66(1):35–65.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

William J Mayew and Mohan Venkatachalam. 2012. The power of voice: Managerial affective states and future firm performance. *The Journal of Finance* 67(1):1–43.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Thien Hai Nguyen. 2015. PhraseRNN : Phrase Recursive Neural Network for Aspect-based Sentiment Analysis (September):2509–2514.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563* .

Robert P Schumaker and Hsinchun Chen. 2009. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *TOIS* 27(2):12.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*. Citeseer, volume 1631, page 1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Effective lstms for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100* .

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Duy-tin Vo and Yue Zhang. 2015. Target-Dependent Twitter Sentiment Classification with Rich Automatic Features (Ijcai):1347–1353.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to Compose Words into Sentences with Reinforcement Learning pages 1–10. https://doi.org/10.1051/0004-6361/201527329.

M Zhang, Y Zhang, and DT Vo. 2016. Gated Neural Networks for Targeted Sentiment Analysis. *30th Conference on Artificial Intelligence (AAAI 2016)* pages 3087–3093. http://zhangmeishan.github.io/targeted-sentiment.pdf.