Abstract

Target-dependent sentiment classification remains a challenge: modeling the semantic relatedness of a target with its context words in a sentence. Different context words have different influences on determining the sentiment polarity of a sentence towards the target. Therefore, it is desirable to integrate the connections between target word and context words when building a learning system. In this paper, we develop two target dependent long short-term memory (LSTM) models, where target information is automatically taken into account. We evaluate our methods on a benchmark dataset from Twitter. Empirical results show that modeling sentence representation with standard LSTM does not perform well. Incorporating target information into LSTM can significantly boost the classification accuracy. The target-dependent LSTM models achieve state-of-the-art performances without using syntactic parser or external sentiment lexicons.

1 Introduction

Sentiment analysis, also known as opinion mining [Pang and Lee, 2008] [Liu, 2012], is a fundamental task in natural language processing and computational linguistics [Manning and Schütze, 1999; Jurafsky and Martin, 2000]. Sentiment analysis is crucial to understanding user generated text in social networks or product reviews, and has drawn a lot of attentions from both industry and academic communities. In this paper, we focus on target-dependent sentiment classification [Jiang et al., 2011; Dong et al., 2014; Vo and Zhang, 2015], which is a fundamental and extensively studied task in the field of sentiment analysis. Given a sentence and a target mention, the task calls for inferring the sentiment polarity (e.g., positive, negative, neutral) of the sentence towards the target. For example, let us consider the sentence: “I bought a new camera. The picture quality is amazing but the battery life is too short”. If the target string is picture quality, the expected sentiment polarity is “positive” as the sentence expresses a positive opinion towards picture quality. If we consider the target as battery life, the correct sentiment polarity should be “negative”.

Target-dependent sentiment classification is typically regarded as a kind of text classification problem in literature. Majority of existing studies build sentiment classifiers with supervised machine learning approach, such as feature based Support Vector Machine [Jiang et al., 2011] or neural network approaches [Dong et al., 2014; Vo and Zhang, 2015]. Despite the effectiveness of these approaches, we argue that target-dependent sentiment classification remains a challenge: how to effectively model the semantic relatedness of a target word with its context words in a sentence. One straightforward way to address this problem is to manually design a set of target-dependent features, and integrate them into existing feature-based SVM. However, feature engineering is labor intensive and the “sparse” and “discrete” features are clumsy in encoding side information like target-context relatedness. In addition, a person asked to do this task will naturally “look at” parts of relevant context words which are helpful to determine the sentiment polarity of a sentence towards the target. These motivate us to develop a powerful neural network approach, which is capable of learning continuous features (representations) without feature engineering and meanwhile capturing the intricate relatedness between target and context words.

In this paper, we present neural network models to deal with target-dependent sentiment classification. The approach is an extension on long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] by incorporating target information. Such target-dependent LSTM approach models the relatedness of a target word with its context words, and selects the relevant parts of contexts to infer the sentiment polarity towards the target. The model could be trained in an end-to-end way with standard backpropagation, where the loss function is cross-entropy error of supervised sentiment classification.

We apply the neural model to target-dependent sentiment classification on a benchmark dataset [Dong et al., 2014]. We compare with feature-based SVM [Jiang et al., 2011], adaptive recursive neural network [Dong et al., 2014] and lexicon-enhanced neural network [Vo and Zhang, 2015]. Empirical results show that the proposed approach without using syntactic parser or external sentiment lexicon obtains state-of-the-art classification accuracy. In addition, we find that modeling sentence with standard LSTM does not perform well...
on this target-dependent task. Integrating target information into LSTM could significantly improve the classification accuracy. The main contributions of this work are as follows:

- We develop target-dependent LSTM models for target-dependent sentiment classification, where the relatedness of target word with context words is encoded.
- Empirical results show that the neural model obtains state-of-the-art performance on a benchmark dataset. Incorporating target information into LSTM could improve the classification accuracy.

2 Related Work
We briefly review existing studies on target-dependent sentiment classification and neural network approaches for sentiment classification in this section.

2.1 Target-Dependent Sentiment Classification
Target-dependent sentiment classification is typically regarded as a kind of text classification problem in literature. Therefore, standard text classification approach such as feature-based Support Vector Machine [Pang et al., 2002; Jiang et al., 2011] can be naturally employed to build a sentiment classifier. For example, Jiang et al. [2011] manually design target-independent features and target-dependent features with expert knowledge, syntactic parser and external resources. Despite the effectiveness of feature engineering, it is labor intensive and unable to discover the discriminative or explanatory factors of data. To handle this problem, some recent studies [Dong et al., 2014; Vo and Zhang, 2015] use neural network methods and encode each sentence in continuous and low-dimensional vector space without feature engineering. Dong et al. [2014] transfer a dependency tree of a sentence into a target-specific recursive structure, and used an Adaptive Recursive Neural Network to learn higher level representation. Vo and Zhang [2015] use rich features including sentiment-specific word embedding and sentiment lexicons.

Different from previous studies, we develop long short-term memory models in this work, which do not use dependency parsing results or external sentiment lexicons. The approach is capable of capturing the relatedness of target word with its context words when composing continuous representation of sentence.

2.2 Neural Network for Sentiment Classification
Neural network approaches have shown promising results on many sentiment analysis tasks like sentence/document-level sentiment classification [Socher et al., 2013b; Tang et al., 2015], opinion expression extraction [Irsoy and Cardie, 2014b], building sentiment lexicon [Tang et al., 2014a], open domain sentiment analysis [Zhang et al., 2015], fine-grained opinion mining [Liu et al., 2015], etc. The power of neural model lies in its ability in learning continuous text representation from data without any feature engineering. For sentence/document level sentiment classification, previous studies mostly have two steps. They first learn continuous word vector embeddings from data [Bengio et al., 2003; Mikolov et al., 2013; Pennington et al., 2014; Tang et al., 2014b]. Afterwards, semantic compositional approaches are used to compute the vector of a sentence/document from the vectors of its constituents based on the principle of compositionality [Fege, 1892]. Representative compositional approaches to learn sentence representation include recursive neural networks [Socher et al., 2013b; Irsoy and Cardie, 2014a], convolutional neural network [Kalchbrenner et al., 2014; Kim, 2014], long short-term memory [Li et al., 2015a] and tree-structured LSTM [Tai et al., 2015; Zhu et al., 2015]. There also exists some studies focusing on learning continuous representation of documents [Le and Mikolov, 2014; Tang et al., 2015; Bhatia et al., 2015].

3 The Approach
We describe the proposed approach for target-dependent sentiment classification in this section. We first present a basic long short-term memory (LSTM) approach, which models the semantic representation of a sentence without considering the target word being evaluated. Afterwards, we extend LSTM by considering the target word, obtaining the Target-Dependent Long Short-Term Memory (TD-LSTM) model. Finally, we extend TD-LSTM with target connection, where the semantic relatedness of target with its context words are incorporated.

3.1 Long Short-Term Memory (LSTM)
In this part, we describe a long short-term memory (LSTM) model for target-dependent sentiment classification. It is a basic version of our approach. In this setting, the target to be evaluated is ignored so that the task is considered in a target independent way.

We use LSTM as it is a state-of-the-art performer for semantic composition in the area of sentiment analysis [Li et al., 2015a; Tang et al., 2015]. It is capable of computing the representation of a longer expression (e.g. a sentence) from the representation of its children with multi levels of abstraction. The sentence representation can be naturally considered as the feature to predict the sentiment polarity of sentence.

Specifically, each word is represented as a low dimensional, continuous and real-valued vector, also known as word embedding [Bengio et al., 2003; Mikolov et al., 2013; Pennington et al., 2014; Tang et al., 2014b]. All the word vectors are stacked in a word embedding matrix $L_w \in \mathbb{R}^{d \times |V|}$, where $d$ is the dimension of word vector and $|V|$ is vocabulary size. In this work, we pre-train the values of word vectors from text corpus with embedding learning algorithms [Pennington et al., 2014; Tang et al., 2014b] to make better use of semantic and grammatical associations of words.

We use LSTM to compute the vector of a sentence from the vectors of words it contains, an illustration of the model is shown in Figure 1. LSTM is a kind of recurrent neural network (RNN), which is capable of mapping vectors of words with variable length to a fixed-length vector by recursively transforming current word vector $w_t$ with the output vector of the previous step $h_{t-1}$. The transition function of standard RNN is a linear layer followed by a pointwise non-linear layer such as hyperbolic tangent function ($\tanh$).

\[
h_t = \tanh(W \cdot [h_{t-1}; w_t] + b)
\] (1)
where $W \in \mathbb{R}^{d \times 2d}$, $b \in \mathbb{R}^{d}$, $d$ is dimension of word vector. However, standard RNN suffers the problem of gradient vanishing or exploding [Bengio et al., 1994; Hochreiter and Schmidhuber, 1997], where gradients may grow or decay exponentially over long sequences. Many researchers use a more sophisticated and powerful LSTM cell as the transition function, so that long-distance semantic correlations in a sequence could be better modeled. Compared with standard RNN, LSTM cell contains three additional neural gates: an input gate, a forget gate and an output gate. These gates adaptively remember input vector, forget previous history and adaptively remember input vector, forget previous history and generate output vector [Hochreiter and Schmidhuber, 1997]. LSTM cell is calculated as follows.

$$i_t = \sigma(W_i \cdot [h_{t-1}; w_t] + b_i) \quad (2)$$

$$f_t = \sigma(W_f \cdot [h_{t-1}; w_t] + b_f) \quad (3)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}; w_t] + b_o) \quad (4)$$

$$g_t = \tanh(W_g \cdot [h_{t-1}; w_t] + b_g) \quad (5)$$

$$c_t = i_t \odot g_t + f_t \odot c_{t-1} \quad (6)$$

$$h_t = o_t \odot \tanh(c_t) \quad (7)$$

where $\odot$ stands for element-wise multiplication, $\sigma$ is sigmoid function, $W_i, b_i, W_f, b_f, W_o, b_o$ are the parameters of input, forget and output gates.

After calculating the hidden vector of each position, we regard the last hidden vector as the sentence representation [Li et al., 2015a; Tang et al., 2015]. We feed it to a linear layer whose output length is class number, and add a softmax layer to output the probability of classifying the sentence as positive, negative or neutral. Softmax function is calculated as follows, where $C$ is the number of sentiment categories.

$$softmax_x = \frac{\exp(x_i)}{\sum_{i'=1}^{C} \exp(x_{i'})} \quad (8)$$

### 3.2 Target-Dependent LSTM (TD-LSTM)

The aforementioned LSTM model solves target-dependent sentiment classification in a target-independent way. That is to say, the feature representation used for sentiment classification remains the same without considering the target words. Let us again take “I bought a new camera. The picture quality is amazing but the battery life is too short” as an example. The representations of this sentence with regard to picture quality and battery life are identical. This is evidently problematic as the sentiment polarity labels towards these two targets are different.

To take into account of the target information, we make a slight modification on the aforementioned LSTM model and introduce a target-dependent LSTM (TD-LSTM) in this subsection. The basic idea is to model the preceding and following contexts surrounding the target string, so that contexts in both directions could be used as feature representations for sentiment classification. We believe that capturing such target-dependent context information could improve the accuracy of target-dependent sentiment classification.

Specifically, we use two LSTM neural networks, a left one LSTM$_L$ and a right one LSTM$_R$, to model the preceding and
We represent target
representation, \{w_1, w_2, ..., w_l\} are preceding context words, \{w_r, ..., w_{n-1}, w_n\} are following context words.

following contexts respectively. An illustration of the model is shown in Figure 1. The input of LSTM_L is the preceding contexts plus target string, and the input of LSTM_R is the following contexts plus target string. We run LSTM_L from left to right, and run LSTM_R from right to left. We favor this strategy as we believe that regarding target string as the last unit could better utilize the semantics of target string when using the composed representation for sentiment classification. Afterwards, we concatenate the last hidden vectors of LSTM_L and LSTM_R, and feed them to a softmax layer to classify the sentiment polarity label. One could also try averaging or summing the last hidden vectors of LSTM_L and LSTM_R as alternatives.

3.3 Target-Connection LSTM (TC-LSTM)

Compared with LSTM model, target-dependent LSTM (TD-LSTM) could make better use of the target information. However, we think TD-LSTM is still not good enough because it does not capture the interactions between target word and its contexts. Furthermore, a person asked to do target-dependent sentiment classification will select the relevant context words which are helpful to determine the sentiment polarity of a sentence towards the target.

Based on the consideration mentioned above, we go one step further and develop a target-connection long short-term memory (TC-LSTM). This model extends TD-LSTM by incorporating an target connection component, which explicitly utilizes the connections between target word and each context word when composing the representation of a sentence.

An overview of TC-LSTM is illustrated in Figure 2. The input of TC-LSTM is a sentence consisting of \( n \) words \( \{w_1, w_2, ..., w_n\} \) and a target string \( t \) occurs in the sentence. We represent target \( t \) as \( \{w_{t+1}, w_{t+2}...w_{r-1}\} \) because a target could be a word sequence of variable length, such as “google” or “harry potter”. When processing a sentence, we split it into three components: target words, preceding context words and following context words. We obtain target vector \( v_{\text{target}} \) by averaging the vectors of words it contains, which has been proven to be simple and effective in representing named entities [Socher et al., 2013; Sun et al., 2015]. When compute the hidden vectors of preceding and following context words, we use two separate long short-term memory models, which are similar with the strategy used in TD-LSTM. The difference is that in TC-LSTM the input at each position is the concatenation of word embedding and target vector \( v_{\text{target}} \), while in TD-LSTM the input at each position only includes the embedding of current word. We believe that TC-LSTM could make better use of the connection between target and each context word when building the representation of a sentence.

3.4 Model Training

We train LSTM, TD-LSTM and TC-LSTM in an end-to-end way in a supervised learning framework. The loss function is the cross-entropy error of sentiment classification.

\[
\text{loss} = - \sum_{s \in S} \sum_{c=1}^{C} P_c^g(s) \cdot \log(P_c(s))
\]

where \( S \) is the training data, \( C \) is the number of sentiment categories, \( s \) means a sentence, \( P_c(s) \) is the probability of predicting \( s \) as class \( c \) given by the softmax layer, \( P_c^g(s) \) indicates whether class \( c \) is the correct sentiment category, whose value is 1 or 0. We take the derivative of loss function through back-propagation with respect to all parameters, and update parameters with stochastic gradient descent.

4 Experiment

We apply the proposed method to target-dependent sentiment classification to evaluate its effectiveness. We describe experimental setting and empirical results in this section.

4.1 Experimental Settings

We conduct experiment in a supervised setting on a benchmark dataset [Dong et al., 2014]. Each instance in the training/test set has a manually labeled sentiment polarity. Training set contains 6,248 sentences and test set has 692 sentences. The percentages of positive, negative and neutral in training and test sets are both 25%, 25%, 50%. We train the model on training set, and evaluate the performance on test set. Evaluation metrics are accuracy and macro-F1 score.
over positive, negative and neutral categories [Manning and Schütze, 1999; Jurafsky and Martin, 2000].

4.2 Comparison to Other Methods

We compare with several baseline methods, including:

In SVM-indep, SVM classifier is built with target-dependent features, such as unigram, bigram, punctuations, emoticons, hashtags, the numbers of positive or negative words in General Inquirer sentiment lexicon. In SVM-dep, target-dependent features [Jiang et al., 2011] are also concatenated as the feature representation.

In Recursive NN, standard Recursive neural network is used for feature learning over a transferred target-dependent dependency tree [Dong et al., 2014]. AdaRNN-w/oE, AdaRNN-w/E and AdaRNN-comb are different variations of adaptive recursive neural network [Dong et al., 2014], whose composition functions are adaptively selected according to the inputs.

In Target-dep, SVM classifier is built on rich target-independent and target-dependent features [Vo and Zhang, 2015]. In Target-dep⁺, sentiment lexicon features are further incorporated.

The neural models developed in this paper are abbreviated as LSTM, TD-LSTM and TC-LSTM, which are described in the previous section. We use 100-dimensional Glove vectors learned from Twitter, randomize the parameters with uniform distribution $U(-0.003, 0.003)$, set the clipping threshold of softmax layer as 200 and set learning rate as 0.01.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Macro-F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-indep</td>
<td>0.627</td>
<td>0.602</td>
</tr>
<tr>
<td>SVM-dep</td>
<td>0.634</td>
<td>0.633</td>
</tr>
<tr>
<td>Recursive NN</td>
<td>0.630</td>
<td>0.628</td>
</tr>
<tr>
<td>AdaRNN-w/oE</td>
<td>0.649</td>
<td>0.644</td>
</tr>
<tr>
<td>AdaRNN-w/E</td>
<td>0.658</td>
<td>0.655</td>
</tr>
<tr>
<td>AdaRNN-comb</td>
<td>0.663</td>
<td>0.659</td>
</tr>
<tr>
<td>Target-dep</td>
<td>0.697</td>
<td>0.680</td>
</tr>
<tr>
<td>Target-dep⁺</td>
<td>0.711</td>
<td>0.699</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.665</td>
<td>0.647</td>
</tr>
<tr>
<td>TD-LSTM</td>
<td>0.708</td>
<td>0.690</td>
</tr>
<tr>
<td>TC-LSTM</td>
<td><strong>0.715</strong></td>
<td>0.695</td>
</tr>
</tbody>
</table>

Table 1: Comparison of different methods on target-dependent sentiment classification. Evaluation metrics are accuracy and macro-F1. Best scores are in bold.

Experimental results of baseline models and our methods are given in Table 1. Comparing between SVM-indep and SVM-dep, we can find that incorporating target information can improve the classification accuracy of a basic SVM classifier. AdaRNN performs better than feature based SVM by making use of dependency parsing information and tree-structured semantic composition. We can find that target-dep is a strong performer even without using lexicon features. It benefits from rich automatic features generated from word embeddings.

Among LSTM based models described in this paper, the basic LSTM approach performs worst. This is not surprising because this task requires understanding target-dependent text semantics, while the basic LSTM model does not capture any target information so that it predicts the same result for different targets in a sentence. TD-LSTM obtains a big improvement over LSTM when target signals are taken into consideration. This result demonstrates the importance of target information for target-dependent sentiment classification. By incorporating target-connection mechanism, TC-LSTM obtains the best performances and outperforms all baseline methods in term of classification accuracy.

Comparing between Target-dep⁺ and Target-dep, we find that sentiment lexicon feature could further improve the classification accuracy. Our final model TC-LSTM without using sentiment lexicon information performs comparably with Target-dep⁺. We believe that incorporation lexicon information in TC-LSTM could get further improvement. We leave this as a potential future work.

4.3 Effects of Word Embeddings

It is well accepted that a good word embedding is crucial to composing a powerful text representation at higher level. We therefore study the effects of different word embeddings on LSTM, TD-LSTM and TC-LSTM in this part. Since the benchmark dataset from [Dong et al., 2014] comes from Twitter, we compare between sentiment-specific word embedding (SSWE) [Tang et al., 2014b] and Glove vectors [Pennington et al., 2014]. All these word vectors are 50-dimensional and learned from Twitter. SSWEh, SSWEr and SSWEu are different embedding learning algorithms introduced in [Tang et al., 2014b]. SSWEh and SSWE learn word embeddings by only using sentiment of sentences. SSWEu takes into account of sentiment of sentences and contexts of words simultaneously.

From Figure 3 we can find that SSWEh and SSWEr perform worse than SSWEu, which is consistent with the results reported on target-independent sentiment classification of tweets [Tang et al., 2014b]. This shows the importance

SSWE vectors are publicly available at http://ir.hit.edu.cn/~dytang
Glove vectors are publicly available at http://nlp.stanford.edu/projects/glove/
of context information for word embedding learning as both SSWEₐ and SSWEᵣ do not encode any word contexts. Glove and SSWEᵣ perform comparably, which indicates the importance of global context for estimating a good word representation. In addition, the target connection model TC-LSTM performs best when considering a specific word embedding.

<table>
<thead>
<tr>
<th></th>
<th>50dms</th>
<th>100dms</th>
<th>200dms</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>27</td>
<td>95</td>
<td>329</td>
</tr>
<tr>
<td>LSTM-TD</td>
<td>20</td>
<td>93</td>
<td>274</td>
</tr>
<tr>
<td>LSTM-TC</td>
<td>65</td>
<td>280</td>
<td>1,165</td>
</tr>
</tbody>
</table>

Table 2: Time cost of each model with 50dms, 100dms and 200dms Glove vectors. Each value means how many seconds cost in each training iteration.

We compare between Glove vectors with different dimensions (50/100/200). Classification accuracy and time cost are given in Figure 3 and Table 2 respectively. We can find that 100-dimensional word vectors perform better than 50-dimensional word vectors, while 200-dimensional word vectors do not show significant improvements. Furthermore, TD-LSTM and LSTM have similar time cost, while TD-LSTM gets higher classification accuracy as target information is incorporated. TC-LSTM performs slightly better than TD-LSTM while at the cost of longer training time because the parameter number of TC-LSTM is larger.

4.4 Case Study

In this section, we explore to what extent the target-dependent LSTM models including TD-LSTM and TC-LSTM improve the performance of a basic LSTM model.

Table 3: Examples drawn from the test set whose polarity labels are incorrectly inferred by LSTM but correctly predicted by both TD-LSTM and TC-LSTM. For each example, target words are in **bold**, “gold” is the ground truth and “LSTM” means the predicted sentiment label from LSTM model.

<table>
<thead>
<tr>
<th>Example</th>
<th>gold</th>
<th>LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>i hate my ipod</em> look at my last tweet before the argh one that’s for you</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td><em>okay sooo ... uhhhhm .... what is going on with lindsay lohan</em>’s face? boring day at the office = perez and tomorrow overload. not good</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td><em>i heard ShannonBrown did his thing in the lakers game!! got ta love him</em></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><em>Hey google, thanks for all these great Labs features on Chromium, but how about ”Create Application Shortcut”??</em></td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

In Table 3 we list some examples whose polarity labels are incorrectly inferred by LSTM but correctly predicted by both TD-LSTM and TC-LSTM. We observe that LSTM model prefers to assigning the polarity of the entire sentence while ignoring the target to be evaluated. TD-LSTM and TC-LSTM could take into account of target information to some extend. For example, in the 2nd example the opinion holder expresses a negative opinion about his work, but holds a neutral sentiment towards the target “lindsay lohan”. In the last example, the whole sentence expresses a neutral sentiment while it holds a positive opinion towards “google”.

We analyse the error cases that both TD-LSTM and TC-LSTM cannot well handle, and find that 85.4% of the misclassified examples relate to neutral category. The positive instances are rarely misclassified as negative, and vice versa. A example of errors is: "freaky friday on television reminding me to think wtf happened to lindsay lohan, she was such a terrific actress , + my huge crush on haley hudson.", which is incorrectly predicted as positive towards target “lindsay lohan” in both TD-LSTM and TC-LSTM.

4.5 Discussion

In order to capture the semantic relatedness between target and context words, we extend TD-LSTM by adding a target connection component. One could also try other extensions to capture the connection between target and context words. For example, we also tried an attention-based LSTM model, which is inspired by the recent success of attention-based neural network in machine translation [Bahdanau et al., 2015] and document encoding [Li et al., 2015b]. We implement the soft-attention mechanism [Bahdanau et al., 2015] to enhance TD-LSTM. We incorporate two attention layers for preceding LSTM and following LSTM, respectively. The output vector for each attention layer is the weighted average among hidden vectors of LSTM, where the weight of each hidden vector is calculated with a feedforward neural network. The outputs of preceding and following attention models are concatenated and fed to softmax for sentiment classification. However, we cannot obtain better result with such an attention model. The accuracy of this attention model is slightly lower than the standard LSTM model (around 65%), which means that the attention component has a negative impact on the model. A potential reason might be that the attention based LSTM has larger number of parameters, which cannot be easily optimized with the small number of corpus.

5 Conclusion

We develop target-specific long short term memory models for target-dependent sentiment classification. The approach captures the connection between target word and its contexts when generating the representation of a sentence. We train the model in an end-to-end way on a benchmark dataset, and show that incorporating target information could boost the performance of a long short-term memory model. The target-dependent LSTM model obtains state-of-the-art classification accuracy.

References


[Li et al., 2015a] Jiwei Li, Dan Jurafsky, and Eudard Hovy. When are tree structures necessary for deep learning of representations? *EMNLP*, 2015.


