Implicit Syntactic Features for Targeted Sentiment Analysis

Yuze Gao[†]**, Yue Zhang**[†] **and Tong Xiao**[‡]
Singapore University of Technology and Design[†]
Northeastern University[‡]
yuze_gao,yue_zhang@sutd.edu.sg
xiaotong@mail.neu.edu.cn

Abstract

Target-dependent sentiment analysis investigates the sentiment polarities on given target mentions from input texts. Different from sentence-level sentiment, it offers more fine-grained knowledge on each entity mention. While early work leveraged syntactic information, recent research has used neural representation learning to induce features automatically, thereby avoiding error propagation of syntactic parsers, which are particularly severe on social media texts.
We study a method to leverage syntactic information without explicitly building parser outputs, by training an encoder-decoder structure parser model on standard syntactic treebanks, and then leveraging its hidden encoder layers when analysing tweets. Such hidden vectors do not contain explicit syntactic outputs, yet encode rich syntactic features. We use them to augment the inputs to a baseline state-of-the-art target-dependent sentiment classifier, observing significant improvements on various benchmark datasets. We obtain the best accuracies on two different test sets for targeted sentiment.

1 Introduction

Target-dependent sentiment analysis investigates the problem of assigning sentiment polarity labels to a set of given target mentions in input sentences. Some example are shown in Table 1. For instance, given a sentence "I like [*Twitter*] better than [*Facebook*]", a target-specific sentiment model is expected to assign a positive (+) sentiment label on the target *Twitter* and a negative (−) sentiment label on the target *Facebook*.

"I like [***Twitter***]$_+$ better than *Facebook*"
"I like *Twitter* better than [***Facebook***]$_-$"
[***lindsay lohan***]$_0$ goes on yet another emo rant on her twitter.
Choose [***NBI***]$_+$ for insulation, home energy audits, housing repairing, air sealing, windows and doors,furnaces, air conditioners and en energy efficient appliances.

Table 1: Target-dependent sentiment analysis

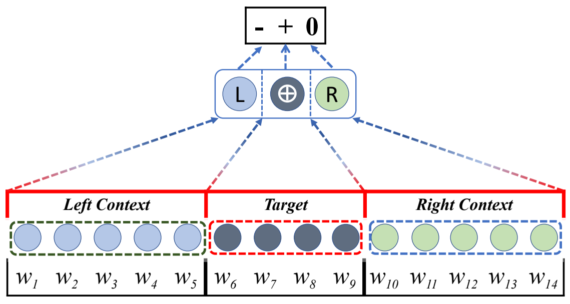
- + 0
L
⊕
R
Left Context
Target
Right Context
w_1 w_2 w_3 w_4 w_5
w_6 w_7 w_8 w_9
w_{10} w_{11} w_{12} w_{13} w_{14}

Figure 1: Sentence level context

The task has been addressed using neural network models, which learn target-specific representations of the input sentence. These representations are then used for predicting target-dependent sentiment polarities. In particular, Dong et al. (2014) derive the syntactic structure of input sentence using a dependency grammar, before transforming the tree structure to a target-centered form. A recursive neural network is used to transform the dependency syntax of a sentence into a target-specific vector for sentiment classification. More recently Vo and Zhang (2015) split the input sentence into three segments, with the target entity mention being in the center, and its left and right contexts surrounding it, as shown in Figure 1.

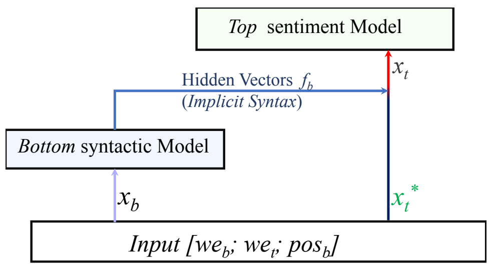

Figure 2: Model structure

Rich word embedding features are extracted from the target entity mention and its contexts, which are then used for classification by a linear SVM model. Without using syntactic information, this model gives better accuracies compared with the method of Dong et al. (2014).

Since syntactic parsing of tweets can be inaccurate due to intrinsic noise in their writing style, most subsequent work followed Vo and Zhang (2015), avoiding the use of syntactic information explicitly. Zhang et al. (2016) applied a bidirectional Gated RNN to learn a dense representation of the input sentence, and then use a three-way gated network structure to integrate target entity mention and its left and right contexts. The final resulting representation is used for softmax classification. Tang et al. (2015) also use a RNN (LSTM) to represent the input sentence, yet directly integrating the target embedding to each hidden state for deriving a target-specific vector, which is used for sentiment classification. Liu and Zhang (2017) extended both Zhang et al. (2016) and Tang et al. (2015) by introducing the attention mechanism, obtaining the best accuracies on both datasets so far.

Intuitively, syntactic information should be useful for sentiment analysis given a target, since target-related semantic information such as predicate-argument structure information is contained in syntactic structures. The main issue of Dong et al. (2014)'s method is that *explicit* syntactic structures are inaccurate and noisy. We try to avoid this issue by using *implicit* syntactic information, by integrating the hidden feature layers of a state-of-the-art neural dependency parsing as features to the state-of-the-art targeted sentiment classification models of Liu and Zhang (2017), using neural stacking (Zhang and Weiss 2016; Chen et al. 2016). The main structure of our model is shown in Figure 2.

We choose the parser of Dozat and Manning (2016) as our syntactic model, which gives the best results on a WSJ benchmark by using multi-layer LSTMs to encode rich input information. The structure of the model is shown in Figure 4, which first learns a vector form of each input word (**W** and **T**), and then uses a simple bi-affine attention mechanism to find word-word relations. The feature vectors (**A** and **D**) thus contain rich syntactic information about each word, yet do not explicitly specify the syntactic structure of the sentence. Hence, using them as features gives our model more syntactic background of the sentence, yet without suffering from error propagation.

Results on both the dataset of Zhang et al. (2016) and the dataset of Tang et al. (2015) show that syntactic information is highly useful for improving the accuracies of target-dependent sentiment analysis. Our final models give the best reported results on both datasets. The source code is released at `https://github.com/CooDL/TSSSF`.

2 Model

As shown in Figure 2, our neural stacking model consists of two brief components: a bottom level syntactic model for obtaining the syntactic information and a top level sentiment model for target-dependent sentiment classification.

2.1 Input representation

Given an input sentence, we first obtain its word representations. In particular, we train two separate word embedding sets for the bottom level syntactic model and top level sentiment model, respectively, denoted as we_b and we_t, respectively. This is because our syntactic parser is trained on news data, while our sentiment classification is trained on Twitter data.

In addition, for the bottom syntactic model, we also use optionally part-of-speech tag embeddings pos_b, which are randomly initialised and learned during the training of the model. Formally, given a word w, the representation for the bottom level model is:

$$x_b = we_b \oplus pos_b,$$

and the input form of the top level model is

$$x_t = x_t^* \oplus f_b = we_t \oplus Bottom(x_b)$$

Here we use $Bottom(x_b)$ indicate the bottom level syntax model.

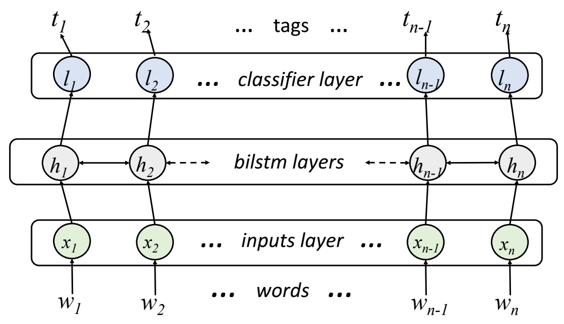

Figure 3: POS-tagging model

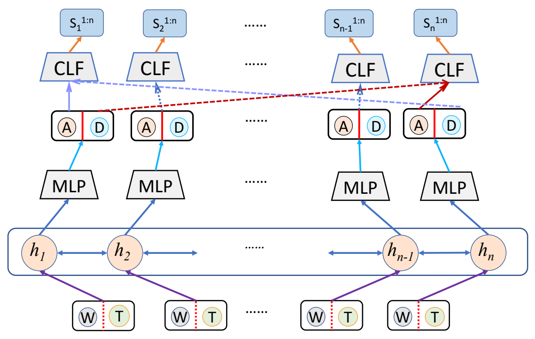

Figure 4: Dependency parsing model

2.2 Syntactic Sub Models

The twitter data suffer poor accuracies by syntax parsers in contrast with news data such as PTB. Directly using explicit twitter syntax features has an error propagation problem. We use a pre-trained syntax model to turn raw word embeddings into implicit syntactic features. Both a POS model and a dependency model are used to utilize syntax features.

2.2.1 POS Model

We employ a simplified bi-directional LSTM (BiLSTM) POS-tagging model (see Figure 3), trained on PTB3 (Toutanova et al. 2003; Labeau et al. 2015). As for every sentence sequence $w_1, w_2, ..., w_n$, its corresponding word embedding sequence $x_1, x_2, ..., x_n$, we integrate its word embedding into a k_1-layer BiLSTM:

$$\begin{aligned} S' &= [h_1, h_2, ..., h_n] \\ &= \text{BiLSTM}([x_1, x_2, ..., x_n])^{k_1}, \end{aligned} \tag{1}$$

where S' is the k_1-layer BiLSTM hidden state output. A classifier is then used to weight the hidden state of each word in S' and derive the label. Here W_1 is the weight matrix and b is the bias:

$$Labels = \text{Classifier}(W_1 S' + b), \tag{2}$$

The BiLSTM hidden layer $h_1, h_2, ..., h_n$ and the result of $W_1 S' + b$ (the labels' logits) will act as our implicit syntactic features.

2.2.2 Dependency Model

In this model, we use a dependency parser to replace the POS-tagging model in Section 2.2.1. In particular, the model of Dozat and Manning (2016) is used, which fuses several BiLSTM layers to encode the input sentence before doing bi-affine attention to learn dependency arcs betwtween different words.

Two different dependency models are trained: one being a POS$\oplus$ dependency with bottom input $we_b \oplus pos_b$, one being no-POS dependency model with just word embedding we_b. , given a sentence sequence $w_1, w_2, ..., w_n$, it integrates the word embedding $we_b(\boldsymbol{W})$ and POS-tag embedding $pos_b(\boldsymbol{T})$ into a k_2-layer BiLSTM and generate the LSTM states S' of the words in sentence $\boldsymbol{S}$, here $x_i = we_b^i \oplus pos_b^i$, $h_i = \overleftarrow{h_i} \oplus \overrightarrow{h_i}$,

$$\begin{aligned} S' &= [h_1, h_2, ..., h_n] \\ &= \text{BiLSTM}([x_1, x_2, ..., x_n])^{k_2}, \end{aligned} \tag{3}$$

MLP (Multilayer Perceptron) layers are used to reduce the dimension size and build features from the BiLSTM state output S'. Here it gives four kind features: $head_{arc}$, $head_{dep}$, rel_{arc}, rel_{dep}:

$$\begin{aligned} &head_{arc}, head_{dep}, rel_{arc}, rel_{dep} \\ &\quad = \text{MLP}([h_1, h_2, ..., h_n])^{k_3}, \end{aligned} \tag{4}$$

Based on the features, a bi-affine classifier gives every word in the sentence S a corresponding dependency *head* using the feature $head_{arc}$(**A**) and $head_{dep}$(**D**). We obtain the *head* relation set $S^{'}_{head} = \{head_i^j, i, j \in [1, n]\}$:

$$head_i^j = \text{Classifier}(head_{arc}^i, head_{dep}^j) \tag{5}$$

Another bi-affine classifier is used to classify the dependency relation based on the feature $head_{arc}$(**A**), $head_{dep}$(**D**) and $head_i^j$, and we obtain the *rel* relation label set $S^{'}_{rel} = \{rel_i^j, i, j \in [1, n]\}$:

$$rel_i^j = \text{Classifier}(head_{arc}^i, head_{dep}^j, head_i^j), \tag{6}$$

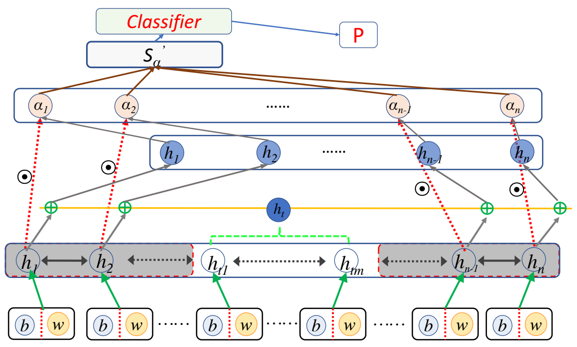

Figure 5: Target-dependent sentiment analysis with attention, shadow parts donate the attention part in a sentence

Using the two classifiers, we obtain the dependency root and relation between every two words in the sentence S. We pre-train the dependency parser model with 4 bi-directional LSTM layers and 2 layers of MLP, and use its intermediate output (the MLP output vector) as implicit syntactic feature inputs to the top sentiment model.

The normal dependency syntax model shares the same network frame with the no-POS dependency model. They have slight differences in the classifier. Both models are end-to-end denpendency parsers with different initial inputs. We choose the same output (Bi-LSTM hidden vector and MLP vector) of the two models as implicit syntactic features.

2.3 Target-dependent Sentiment Model

We use the attention-based model of Liu and Zhang (2017) as our top level model. The overall structure is shown in Figure 5. Given a sentence, it first uses several BiLSTM layers to learn its syntactic features, and then an attention layer is used to select the relative wegihts of the words according to the target entity over the untargeted words in the whole sentence (Bahdanau et al. 2014; Yang et al. 2016). In particular, for a target word, it applies the target word hidden vector to find a weight forevery word (except the target words) in the sentence (see Figure 5). The model also uses a BiLSTM to represent the feature layer from bottom syntactic model $f_b(\boldsymbol{b})$ and the word embedding $we_t(\boldsymbol{w})$ of a word sequence $w_1, w_2, ..., w_n$ as the hidden vector of each word.

$$[h_1, h_2, ..., h_n] = \text{BiLSTM}([r_1, r_2, ..., r_n])^{k_4}, \quad (7)$$

where $r_i = f_b^i \oplus we_t^i$ and k_4 is the BiLSTM layer number.

The target phrase words $h_{t_1}, h_{t_2}, ..., h_{t_m}$ are represented as one vector h_t($h_t \notin [h_1, h_n]$). It is the average of the target phrase words hidden vectors, $h_t = \frac{1}{m}\sum_{i=1}^{m} h_{t_i}$.

We build a vanilla attention model by calculating a weight value α_i for each word in the sentence. The sentence S then can be represented as follows:

$$\begin{aligned} S_\alpha' &= \text{Attention}([h_1, h_2, ..., h_n], h_t) \\ &= \sum_{i=1}^{n} \alpha_i h_i, \end{aligned} \quad (8)$$

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

The weight scores β_i are calculated by using target representation h_t and each word hidden vector representation in the sentence,

$$\beta_i = U^T \tanh(W_2 \cdot [h_i : h_t] + b_1), \quad (9)$$

The sentence representation S_α' is used to predict the probability vector P sentiment labels on target by:

$$P = \text{Classifier}(W_3 \cdot S_\alpha' + b_2), \quad (10)$$

2.4 Training

Our training procedure consists of two steps, one being to pre-train the bottom syntactic models, the other being to apply the pre-trained bottom syntactic model and train the top sentiment analysis model.

All models are trained by minimizing the sum of cross-entropy loss and a $L2$ regularization loss of all trainable weights ΔW.

$$loss = \frac{1}{n}\sum_{i}^{n} \sigma(y_i, y_i') + \frac{\lambda}{2}||\Delta W||^2, \quad (11)$$

The model feature inputs (word embeddings, POS-tag embeddings) are the sum of a trainable embedding and a pre-trained (or learned) embedding. All the weight matrix will be initialized with an orthogonal loss less than $1e^{-6}$.

We choose different intermediate outputs of different bottom level syntax models. For POS-tagging model, we use the BiLSTM hidden output (lm_{pos}) and POS-tags vector before softmax (lt_{pos}). For the dependency sub model, we utilize

Bottom Syntactic Model	
LSTM Size(d_{blstm})	300
MLP Size (d_{mlp})	100
LSTM Layers(POS Model) (k_1)	2
LSTM Layers(Dep. Model) (k_2)	4
LSTM Dropout Rate (dr_{blstm})	0.6
MLP Layers (k_3)	2
MLP Dropout Rate (dr_{mlp})	0.67
Batch Size(b_b)	1000
Word Embeddings (d_{bw})	100
POS Embeddings (d_{pos})	100
Top Sentiment Model	
LSTM Size(d_{tlstm})	200
LSTM Layers(k_4)	1
Word Embedding(d_{tw})	200
Batch Size(b_t)	200
LSTM Dropout Rate(dr_{tlstm})	0.5
Same Parameters	
Word Minimum Occurance	3
Learning Rate(lr)	0.02
Learning Rate Decay Rate(lr_{speed})	0.75
Decay Steps($lr_{distance}$)	1500
Random Seed	1314
Train Iterations	30000

Table 2: Hyper-parameters values

the last BiLSTM layer hidden feature(lm_{dep}) and the MLP layer output (mlp_{dep}) optionally.

3 Experiments

We evaluate the performances of our model and compare them with state-of-the-art results using two standard datasets for target-dependent sentiment (Zhang et al., 2016; Tang et al., 2015). The PTB3 dataset is used to pre-train our bottom level syntax models.

3.1 Data

We conduct experiments on two datasets, one being the training/dev/test dataset of Zhang et al. (2016) (Z-Set), which consists of the MPQA corpus[1] and Mitchell et al. (2013)'s corpus[2], the other being the dataset of the benchmark training/test dataset of **?** (T-Set), we label these datasets' POS-tags with the open parser tools ZPar (Zhang and Clark, 2011). Two sets of word embedding are used in this experiment: The GloVe[3] (Pennington et al., 2014) twitter embedding (100 dimensions) for the bottom model, and the GloVe twitter word embedding (200 dimensions) for the top target-dependent sentiment analysis model. Also, due to lack of syntactically labelled twitter data, we used the PTB3 dataset to pre-train our bottom models. We follow the standard splits of PTB3, using 2-21 as the bottom model training data, section 22 for the development set and 23 as the test set.

[1]http://mpqa.cs.pitt.edu/corpora/mpqa corpus/
[2]http://www.m-mitchell.com/code/index.html
[3]https://nlp.stanford.edu/projects/glove/

		Total	Pos	Neg	Neu
T-set	Train	6248	1561	1560	3127
	Test	692	173	173	346
Z-set	Train	9489	2416	2384	4689
	Dev	1036	255	272	509
	Test	1170	294	295	581

Table 3: Sentiment Distribution

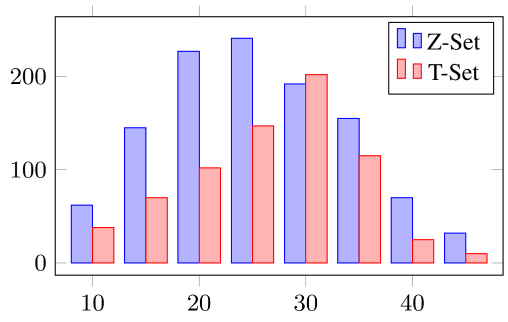

Figure 6: Test-set length distribution

We calculate statistics on sentiment polorities and lengths for both datasets. Table 3 shows the same percentage of three sentiment labels and Figure 6 shows length distribution on the test sets.

3.2 Trainning Settings

First, we use the PTB3 dataset with the standard split method pre-train the POS syntax model and dependency syntax model with the hyper-parameters listed in Table 2. A best model on the devset is saved for the neural stacking bottom syntax model.

Once obtaining the pre-trained bottom syntax model, we build the top sentiment model based on intermediate output syntax model features f_b and top word embedding we_t.

3.3 Hyper-parameters

Embedding Size: Our embedding is a superposition of a trainable and a pre-trained word embedding. We fixed the word embedding dimension of we_b and we_t to 100 and 200, respectively to match two pre-trained GloVe word embeddings set from Pennington et al. (2014).

Models	Acc.(%)	F1(%)	UAS	LAS
POS-tagging	92.4	91.6	/	/
Normal Dep.	/	/	95.6	93.8
No-POS Dep.	/	/	94.3	92.7

Table 4: Results for Syntactic Sub Model on PTB3 development set.

Dropout Rate: Dropout wrappers are applied to both the bottom level syntax model and top level sentiment model to avoid overfitting and learn better features. For the bottom syntax model, we use the PTB3 dataset to pre-train and tune hyper-parameters. A dropout rate of $\xi = 0.6$ for the BiLSTM layer and a softmax classifier layer to classify the learned features from hidden BiLSTM vector are used, respectively. Dropout rates of $\xi = 0.6$ and $\xi = 0.67$ are applied to every second BiLSTM layer and MLP layer, respectively, in the dependency model. We gain the best results (see Table 4) of different bottom syntax models on the PTB3 dataset.

For the top sentiment model, we use the model with only top word embedding inputs as our baseline. Here, the bottom syntactic features f_b are pre-processed with a dropout wrapper of $\phi = 0.5$ before being concatenated to the top model word embedding we_t, which is also wrapped with a dropout of $\varphi = 0.8$ for training models.

Training: We tune the hyper-parameters of the bottom syntax model on the PTB3 development set and top sentiment on the Z-Set development set. Words that occur less than a minimum amount of 3 times are treated as unknown words. Standard SGD with a decaying learning rate ($2e^{-2}$) is used for optimization, where the decay rate (0.75) is used to reduce the learning rate after each training iteration step ($lr_{distance}$).

$$lr_{new} = lr \cdot (lr_{speed})^{total_{steps}/lr_{distance}}, \quad (12)$$

There are several hyper-parameters in our models. We tune all the model hyper-parameters on the dev set with grid-search. With a learning rate of $\varphi = 2e^{-2}$, we did a large parameter iteration on learning rate decay steps $lr_{distance}$, decay rate lr_{speed}, batch size ($b_b \& b_t$) and dropout. The batch size ($b_b \& b_t$) has a great impact on model weights gradient and training speeds, and we choose a balanced point of 200 and 1000 for top and bottom model respectively. The decaying learning rate can also help in avoiding early overfitting and large weights optimization. The details of other hyper-parameters are listed in Table 2.

Models	Acc.(%)
Baseline	73.24
$+lm_{pos}$	73.53
$+lt_{pos}$	73.34
$+lm_{pos}\&ltpos$	73.81
$+mlp_{dep}$	74.23
$+lm_{dep}$	73.96
$+lm_{dep}\&mlp_{dep}$	74.59

Table 5: Dev set accuracies for sentiment sub model

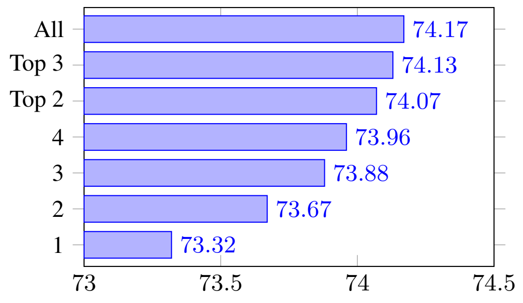

Table 6: Dev Results on BiLSTM feature layers

3.4 Development Experiments

Syntactic features: We measure the efficience of different syntax features; the results are listed in Table 5. Within syntactic features, the baseline system (our implementation of Liu and Zhang (2017)) gives an accuracy of 73.24%. With only POS features, the accuracies can reach 74.23%, which is significantly ($p < 0.01$ by T-test) higher. With dependency information, the accuracy further rises to 74.59%, which is significant improved by 1.4 points to the baseline. This shows that syntactic information is indeed useful for target-dependent sentiment classification.

BiLSTM Layers: We also concatenate the hidden BiLSTM vector from different layers to construct a fast forword feature network to build feature from the dependency model.

$$lm_{dep} = \text{MLP}(\text{CONCAT}(lm_{dep}[1:n])), \quad (13)$$

here, MLP is used to reduce the concatenated lm_{dep} dimensions and ($1 <= n <= 4$). A dropout wrapper of $\phi = 0.6$ is applied for the concatenated LSTM vectors $lm_{dep}[1:n]$.

The results of fast forwards features from different LSTM layer are shown in Tabel 6. Here

	Acc.(%)		F1(%)	
Models	Z_{set}	T_{set}	Z_{set}	T_{set}
Jiang et al. (2011)	/	63.4	/	63.3
Dong et al. (2014)	/	66.3	/	65.9
Vo and Zhang (2015)	69.6	71.1	65.6	69.9
Tang et al. (2015)	/	71.5	/	69.5
Zhang et al. (2016)	71.9	72.0	69.6	70.9
Liu and Zhang (2017)	73.5	72.4	70.6	70.5
Baseline	*73.0*	*71.7*	*70.2*	*70.1*
$+\ lm_{pos}$ [a]	73.5	72.4	71.2	70.4
$+\ lt_{pos}$ [b]	73.2	72.0	70.8	70.2
$+\ lm_{pos}\&lt_{pos}$ [c]	**73.9**	**72.5**	**71.4**	**70.7**
$+\ lm_{dep}$	73.5	72.2	70.7	70.6
$+\ mlp_{dep}$	74.0	72.6	71.3	70.9
$+\ lm_{dep}\&mlp_{dep}$	**74.1**	**72.7**	**71.7**	**71.3**
$+\ lm^*_{dep}$ [d]	73.3	72.4	70.9	70.5
$+\ mlp^*_{dep}$ [e]	74.2	72.8	71.3	70.5
$+\ lm^*_{dep}\&mlp^*_{dep}$ [f]	**74.3**	**72.8**	**71.8**	**71.4**

Table 7: Test set results with different syntactic features, the features with * means they are built from the no-POS dependency syntax model

		Pos	Neg	Neu
Z-Set	Baseline	61.64	69.83	78.67
	$POS_{[c]}$	61.43	70.17	78.97
	$DEP_{[f]}$	61.14	71.14	79.63
T-Set	Baseline	62.57	69.36	75.70
	$POS_{[c]}$	61.84	69.41	77.62
	$DEP_{[f]}$	62.74	70.31	78.42

Table 8: F1 values(%) of each polarity on test set of Z-Set, T-Set, the $POS_{[c]}$ and $DEP_{[f]}$ indicate the features listed in Table 7

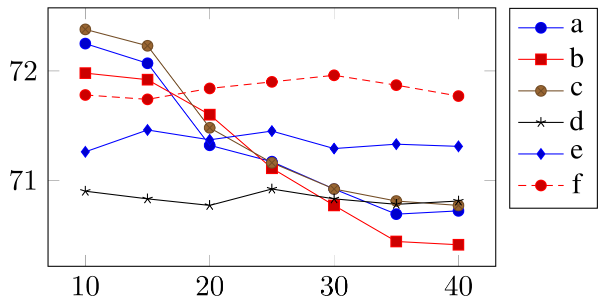

Figure 7: Test-set Accuracy against sentence length (Z-Set), a,b,c,d,e,f indicate the features listed in Table 7, respectively

we refer to the *first* BiLSTM layer as 1, and the *last* BiLSTM layer as 4. *Top 2* indicates the *layer3 & layer4*. Without fast forward connections, the results are 73.24%. With setting 1 to 4, the accuracies increase from 73.24% to 73.32%, 73.67%, 73.88% and 73.96%, respectively. Finally, the best results are obtained with 74.17%. We thus use the settings *layer4* for final tests, for a nice balance of efficiency and accuracy.

3.5 Results

We conduct final tests on the test set of Z-Set and T-Set, respectively investigating two questions. First, we verify whether this kind implicit features enhance the accuracy of twitter target-dependent sentiment analysis. Second, we measure how syntax affect target-dependent sentiment analysis results.

First, we compare the effects of different features on target target-dependent sentiment analysis. We take the top model with only word embedding inputs as our baseline system. The results are listed in Table 7. We can see that the syntactic features contribute to enhancing the accuracy of target-dependent sentiment analysis. Compared with our baseline on both test-set, we obtain an increase of Acc. by 1.3 points ($p < 0.01$) on Z-Set and 1 point ($p < 0.05$) on T-Set. For the POS-tagging model, the lm_{pos} feature provides more information than the lt_{pos} feature, and the lt_{pos} has little impact in their combination case.

The dependency model features work better than the POS-tag features. lm_{dep} is weaker than the mlp_{dep} feature, since mlp_{dep} feature contains more learned and special features, which provide the model with sentence level dependency structure.

Second, we separately test the effect of features made with respect to different sentence lengths and sentiment polorities. As two datasets have different max sentence lengths (Z-set 84 words, T-set 44 words), we focus on the length range [10,40] and treat the sentence with length 10- and 40+ as 10 and 40, respectively. The results are listed in Table 8, Figure 7 (Here we use the test set of Zhang et al. (2016)). The POS-tags features (a,b,c in Table 7) have advantages in short sentence (10-15 words), it gains a significant higher than the dependency features. In contrast, the dependency features (d,e,f in Table 7) show larger contribution on longer sentence (30-40 words).

Finally, our model gives a F1 score of 71.8% and 71.4% on both test sets, respectively, which are the best reported results so far.

3.6 Analysis

The results show that features have different contributions to enhance the accuracy of targeted sentiment classification. The bottom syntax model output contains different syntactic information. Using them as features do contribution to the top model gain the information about sentence structure or word interrelation.

The POS-tagging model features perform well on short sentences. We believe that a POS-tagging model feature vector contains relation between a present word and its POS context words. This matches its adjacent words, helping model gain local phrase-level structure information. For example, if a word has a VB tag and its adjacent words are RB and NN, a tighter relation will be generated between VB and NN.

Phrase-level structure contributes to short sentences, but can be ambiguous for long sentence. Even though a RNN can learn some sentence-level information, with the increasing of the sentence length, this local benefit can decrease gradually. This can be the reason of the result in Figure 7 where the F1 value of the POS-tagging model drops as the sentence length increase.

The stable performance of the dependency model in Figure 7 suggests that the overall sentence structure and local phrase-level structure can be both provided by the dependency model features. The more nonlocal sentence structure can help the model grasp the sentence sentiment easier. It has a slightly weakened in the overall structures of longer sentence.

The benefits from semantic features is structural and non-sentiment related. Though POS-tag information can generate dependency relations, we use the PTB3 data to pre-train the bottom level models, where noise may weaken the advantages. In contrast, the dependency model contains more detailed information, and is useful for PTB-like formal data. The effect can be discounted on twitter data. The results from Table 8 show that the F1 values show no significant variation on different sentiment polarities.

3.7 Attention values

We compared both types of features with the baseline on the attention values and structural relation between words (Figure 8). The relation is computed by the top model LSTM hidden vector under feature $[f]$ in Table 7. The grey level corresponds to their attention values. Darker colors mean closer attention. Here the baseline is the top model with only word embedding inputs. Figure 8(a) is a short sentence (12 words). We can see that the different features do not affect the sentence structure significantly. The POS-tagging model features focus on its adjacent and related words, such as the word '*emo*', which has a tight relation with the adjacent word '*rant*' and its adjunct word '*another*'. When the sentence length increases, the difference between POS and DEP becomes obvious. In Figure 8(b), the DEP has more related darker grey words attention compared to a normal word in the sentence (20+ words, we here hide some words due to limited space). For the phrase '*en engery efficient appliances*', for example, the POS features give shallow local relations, but deep remote semantic relations are given by the DEP features, such as the nominal modifier word '*Choose*' and its paralleling structure word '*insulation*'.

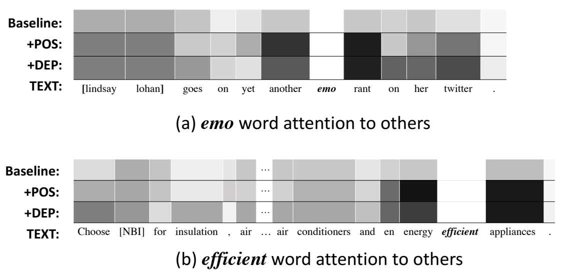

Figure 8: Word attention under implicit syntactic features, darker grayscale means closer attention

4 Conclusion

We investigated the use of implicit syntactic features for improving target-dependent sentiment analysis, by using hidden word representations of a state-of-the-art parsing to augment the input of a state-of-the-art target-dependent sentiment classifier. Neural stacking is used, where the parser is first trained using news article data, and then fine-tuned during the training of the sentiment classification system. In this way, our method leverages syntactic information, which is intuitively useful for target-dependent sentiment analysis, yet does not suffer from error propagations of using explicit syntactic parsing output features. Results on two target-dependent sentiment datasets show that our use of syntax can significantly enhance the accuracies of the baseline model, and our final model outperforms existing methods that use explicit syntactic features and without syntactic features, giving the best accuracies on both datasets.

Acknowledgement

We thank the anonymous reviewers for their detailed and constructive comments. Yue Zhang is the corresponding author.

References

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

Hongshen Chen, Yue Zhang, and Qun Liu. 2016. Neural network for heterogeneous annotations pages 731–741. https://www.aclweb.org/anthology/D/D16/D16-1070.pdf.

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.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734* .

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.

Matthieu Labeau, Kevin Löser, Alexandre Allauzen, and Rue John von Neumann. 2015. Non-lexical neural architecture for fine-grained pos tagging. In *EMNLP*. pages 232–237.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. *EACL 2017* page 572.

Margaret Mitchell, Jacqueline Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *EMNLP 2013*. pages 1643–1654.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Effective lstms for target-dependent sentiment classification. In *COLING*. pages 3298–3307.

Kristina Toutanova, Mark Mitchell, and Christopher D Manning. 2003. Optimizing local probability models for statistical parsing. *Lecture notes in computer science* pages 409–420.

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

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL-HLT*. pages 1480–1489.

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

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. *arXiv preprint arXiv:1603.06598* .

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics* 37(1):105–151.