Feature Optimization for Constituent Parsing via Neural Networks

Zhiguo Wang
IBM Watson
1101 Kitchawan
Yorktown Heights, NY, USA
zhigwang@us.ibm.com

Haitao Mi
IBM Watson
1101 Kitchawan
Yorktown Heights, NY, USA
hmi@us.ibm.com

Nianwen Xue
Brandeis University
415 South St
Waltham, MA, USA
xuen@brandeis.edu

Abstract

The performance of discriminative constituent parsing relies crucially on feature engineering, and effective features usually have to be carefully selected through a painful manual process. In this paper, we propose to automatically learn a set of effective features via neural networks. Specifically, we build a feedforward neural network model, which takes as input a few primitive units (words, POS tags and certain contextual tokens) from the local context, induces the feature representation in the hidden layer and makes parsing predictions in the output layer. The network simultaneously learns the feature representation and the prediction model parameters using a back propagation algorithm. By pre-training the model on a large amount of automatically parsed data, and then fine-tuning on the manually annotated Treebank data, our parser achieves the highest F_1 score at 86.6% on Chinese Treebank 5.1, and a competitive F_1 score at 90.7% on English Treebank. More importantly, our parser generalizes well on cross-domain test sets, where we significantly outperform Berkeley parser by 3.4 points on average for Chinese and 2.5 points for English.

1 Introduction

Constituent parsing seeks to uncover the phrase structure representation of sentences that can be used in a variety of natural language applications such as machine translation, information extraction and question answering (Jurafsky and Martin, 2008). One of the major challenges for this task is that constituent parsers require an inference algorithm of high computational complexity in order to search over their large structural space, which makes it very hard to efficiently train discriminative models. So, for a long time, the task was mainly solved with generative models (Collins, 1999; Charniak, 2000; Petrov et al., 2006). In the last few years, however, with the use of effective parsing strategies, approximate inference algorithms, and more efficient training methods, discriminative models began to surpass the generative models (Carreras et al., 2008; Zhu et al., 2013; Wang and Xue, 2014).

Just like other NLP tasks, the performance of discriminative constituent parsing crucially relies on feature engineering. If the feature set is too small, it might underfit the model and leads to low performance. On the other hand, too many features may result in an overfitting problem. Usually, an effective set of features have to be designed manually and selected through repeated experiments (Sagae and Lavie, 2005; Wang et al., 2006; Zhang and Clark, 2009). Not only does this procedure require a lot of expertise, but it is also tedious and time-consuming. Even after this painstaking process, it is still hard to say whether the selected feature set is complete or optimal to obtain the best possible results. A more desirable alternative is to learn features automatically with machine learning algorithms. Lei et al. (2014) proposed to learn features by representing the cross-products of some primitive units with low-rank tensors for dependency parsing. However, to achieve competitive performance, they had to combine the learned features with the traditional hand-crafted features. For constituent parsing, Henderson (2003) employed a recurrent neural network to induce features from an unbounded parsing history. However, the final performance was below the state of the art.

In this work, we design a much simpler neural network to automatically induce features from just the local context for constituent parsing. Concretely, we choose the shift-reduce parsing strategy to build the constituent structure of a sentence, and train a feedforward neural network model to jointly learn feature representations and make parsing predictions. The input layer of the network takes as input a few primitive units (words, POS tags and certain contextual tokens) from the local context, the hidden layer aims to induce a distributed feature representation by combining all the primitive units with different weights, and the output layer attempts to make parsing predictions based on the feature representation. During the training process, the model simultaneously learns the feature representation and prediction model parameters using a backpropagation algorithm. Theoretically, the learned feature representation is optimal (or at least locally optimal) for the parsing predictions. In practice, however, our model does not work well if it is only trained on the manually annotated Treebank data sets. However, when pre-trained on a large amount of automatically parsed data and then fine-tuned on the Treebank data sets, our model achieves a fairly large improvement in performance. We evaluated our model on both Chinese and English. On standard data sets, our model reaches $F_1 = 86.6\%$ for Chinese and outperforms all the state-of-the-art systems, and for English our final performance is $F_1 = 90.7\%$ and this result surpasses that of all the previous neural network based models and is comparable to the state-of-the-art systems. On cross-domain data sets, our model outperforms the Berkeley Parser[1] by 3.4 percentage points for Chinese and 2.5 percentage points for English.

The remainder of this paper is organized as follows: Section 2 introduces the shift-reduce constituent parsing approach. Section 3 describes our feature optimization model and some parameter estimation techniques. We discuss and analyze our experimental results in Section 4. Section 5 discusses related work. Finally, we conclude this paper in Section 6.

2 Shift-Reduce Constituent Parsing

Shift-reduce constituent parsing utilizes a series of shift-reduce decisions to construct syntactic trees. Formally, the shift-reduce system is a quadruple $C = (S, T, s_0, S_t)$, where S is a set of parser *states* (sometimes called *configurations*), T is a finite set of *actions*, s_0 is an initialization function

[1]https://code.google.com/p/berkeleyparser/

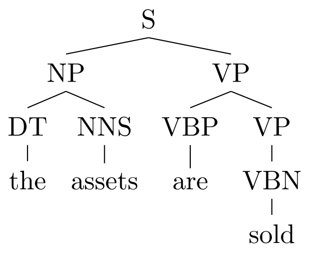

Figure 1: An example of constituent tree.

to map each input sentence into a unique *initial state*, and $S_t \in S$ is a set of *terminal states*. Each action $t \in T$ is a transition function that maps a state into a new state. A parser state $s \in S$ is defined as a tuple $s = (\sigma, \beta)$, where σ is a *stack* which is maintained to hold partial subtrees that are already constructed, and β is a *queue* which is used for storing remaining unprocessed words. In particular, the initial state has an empty stack σ and a queue β containing the entire input sentence, and the terminal states have an empty queue β and a stack σ containing only one complete parse tree. The task of parsing is to scan the input sentence from left to right and perform a sequence of shift-reduce actions to transform the initial state into a terminal state.

In order to jointly assign POS tags and construct a constituent structure for an input sentence, we define the following actions for the action set T, following Wang and Xue (2014):

- SHIFT-X (`sh-x`): remove the first word from β, assign a POS tag X to the word and push it onto the top of σ;
- REDUCE-UNARY-X (`ru-x`): pop the top subtree from σ, construct a new unary node labeled with X for the subtree, then push the new subtree back onto σ. The head of the new subtree is inherited from its child;
- REDUCE-BINARY-{L/R}-X (`rl`/`rr-x`): pop the top two subtrees from σ, combine them into a new tree with a node labeled with X, then push the new subtree back onto σ. The left (L) and right (R) versions of the action indicate whether the head of the new subtree is inherited from its left or right child.

With these actions, our parser can process trees with unary and binary branches easily. For example, in Figure 1, for the sentence "the assets are sold", our parser can construct the parse tree by performing the action sequence {`sh-DT, sh-NNS, rr-NP, sh-VBP, sh-VBN, ru-VP, rr-VP, rr-S`}. To process multi-branch trees, we employ binarization and debinarization processes described in Zhang and Clark (2009) to transform multi-branch trees into binary trees and restore the generated binary trees back to their original forms. For inference, we employ the beam search decoding algorithm (Zhang and Clark, 2009) to balance the tradeoff between accuracy and efficiency.

3 Feature Optimization Model

3.1 Model

To determine which action $t \in T$ should be performed at a given state $s \in S$, we need a model to score each possible $\langle s, t\rangle$ combination. In previous approaches (Sagae and Lavie, 2005; Wang et al., 2006; Zhang and Clark, 2009), the model is usually defined as a linear model $Score(s,t) = \overrightarrow{w} \cdot \Phi(s,t)$, where $\Phi(s,t)$ is a vector of hand-crafted features for each state-action pair and $\overrightarrow{w}$ is the weight vector for these features. The hand-crafted features are usually constructed by compounding primitive units according to some feature templates. For example, almost all the previous work employed the list of primitive units in Table 1(a), and constructed hand-crafted features by concatenating these primitive units according to the feature templates in Table 1(b). Obviously, these feature templates are only a small subset of the cross products of all the primitive units. This feature set is the result of a large number of experiments through trial and error from previous work. Still we cannot say for sure that this is the optimal subset of features for the parsing task.

To cope with this problem, we propose to simultaneously optimize feature representation and parsing accuracy via a neural network model. Figure 2 illustrates the architecture of our model. Our model consists of input, projection, hidden and output layers. First, in the input layer, all primitive units (shown in Table 1(a)) are imported to the network. We also import the suffixes and prefixes of the first word in the queue, because these units have been shown to be very effective for predicting POS tags (Ratnaparkhi, 1996). Then, in the projection layer, each primitive unit is projected into a vector. Specifically, word-type units are represented as word embeddings, and other units are transformed into one-hot representations. The

(1)	p_0w, p_0t,p_0c, p_1w, p_1t,p_1c, p_2w, p_2t,p_2c, p_3w, p_3t,p_3c
(2)	$p_{0l}w$, $p_{0l}c$, $p_{0r}w$, $p_{0r}c$,$p_{0u}w$, $p_{0u}c$, $p_{1l}w$, $p_{1l}c$, $p_{1r}w$, $p_{1r}c$,$p_{1u}w$, $p_{1u}c$
(3)	q_0w, q_1w, q_2w, q_3w

(a) Primitive Units

unigrams	p_0tc, p_0wc, p_1tc, p_1wc, p_2tc p_2wc, p_3tc, p_3wc, q_0wt, q_1wt q_2wt, q_3wt, $p_{0l}wc$, $p_{0r}wc$ $p_{0u}wc$, $p_{1l}wc$, $p_{1r}wc$, $p_{1u}wc$
bigrams	p_0wp_1w, p_0wp_1c, p_0cp_1w, p_0cp_1c p_0wq_0w, p_0wq_0t, p_0cq_0w, p_0cq_0t q_0wq_1w, q_0wq_1t, q_0tq_1w, q_0tq_1t p_1wq_0w, p_1wq_0t, p_1cq_0w, p_1cq_0t
trigrams	$p_0cp_1cp_2c$, $p_0wp_1cp_2c$, $p_0cp_1wq_0t$ $p_0cp_1cp_2w$, $p_0cp_1cq_0t$, $p_0wp_1cq_0t$ $p_0cp_1wq_0t$, $p_0cp_1cq_0w$

(b) Feature Templates

Table 1: Primitive units (a) and feature templates (b) for shift-reduce constituent parsing, where p_i represents the i_{th} subtree in the stack and q_i denotes the i_{th} word in the queue. w refers to the head word, t refers to the head POS, and c refers to the constituent label. p_{il} and p_{ir} refer to the left and right child for a binary subtree p_i, and p_{iu} refers to the child of a unary subtree p_i.

vectors of all primitive units are concatenated to form a holistic vector for the projection layer. The hidden layer corresponds to the feature representation we want to learn. Each dimension in the hidden layer can be seen as an abstract factor of all primitive units, and it calculates a weighted sum of all nodes from the projection layer and applies a non-linear *activation function* to yield its activation. We choose the *logistic sigmoid* function for the hidden layer. The output layer is used for making parsing predictions. Each node in the output layer corresponds to a shift-reduce action. We want to interpret the activation of the output layer as a probability distribution over all possible shift-reduce actions, therefore we normalize the output activations (weighted summations of all nodes from the hidden layer) with the *softmax* function.

3.2 Parameter Estimation

Our model consists of three groups of parameters: (1) the word embedding for each word type unit,

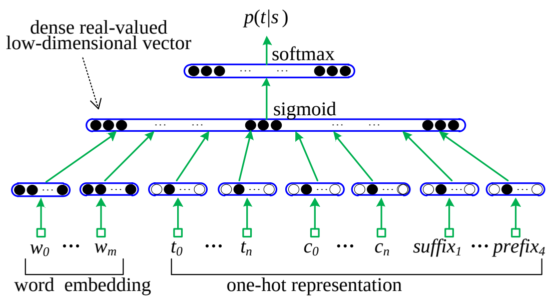

Figure 2: Neural network architecture for constituent parsing, where w_i denotes word type unit, t_i denotes POS tag unit, c_i denotes constituent label unit, $suffix_i$ and $prefix_i$ $(1 \leq i \leq 4)$ denotes i-character word suffix or prefix for the first word in the queue.

(2) the connections between the projection layer and the hidden layer which are used for learning an optimal feature representation and (3) the connections between the hidden layer and the output layer which are used for making accurate parsing predictions. We decided to learn word embeddings separately, so that we can take advantage of a large amount of unlabeled data. The remaining two groups of parameters can be trained simultaneously by the back propagation algorithm (Rumelhart et al., 1988) to maximize the likelihood over the training data.

We also employ three crucial techniques to seek more effective parameters. First, we utilize minibatched AdaGrad (Duchi et al., 2011), in which the learning rate is adapted differently for different parameters at different training steps. With this technique, we can start with a very large learning rate which decreases during training, and can thus perform a far more thorough search within the parameter space. In our experiments, we got a much faster convergence rate with slightly better accuracy by using the learning rate $\alpha = 1$ instead of the commonly-used $\alpha = 0.01$. Second, we initialize the model parameters by pre-training. Unsupervised pre-training has demonstrated its effectiveness as a way of initializing neural network models (Erhan et al., 2010). Since our model requires many run-time primitive units (POS tags and constituent labels), we employ an in-house shift-reduce parser to parse a large amount of unlabeled sentences, and pre-train the model with the automatically parsed data. Third, we utilize the Dropout strategy to address the overfitting problem. However, different from Hinton et al. (2012), we only use Dropout during testing, because we found that using Dropout during training did not improve the parsing performance (on the dev set) while greatly slowing down the training process.

4 Experiment

4.1 Experimental Setting

We conducted experiments on the Penn Chinese Treebank (CTB) version 5.1 (Xue et al., 2005) and the Wall Street Journal (WSJ) portion of Penn English Treebank (Marcus et al., 1993). To fairly compare with other work, we follow the standard data division. For Chinese, we allocated Articles 001-270 and 400-1151 as the training set, Articles 301-325 as the development set, and Articles 271-300 as the testing set. For English, we use sections 2-21 for training, section 22 for developing and section 23 for testing.

We also utilized some unlabeled corpora and used the word2vec[2] toolkit to train word embeddings. For Chinese, we used the unlabeled Chinese Gigaword (LDC2003T09) and performed Chinese word segmentation using our in-house segmenter. For English, we randomly selected 9 million sentences from our in-house newswire corpus, which has no overlap with our training, testing and development sets. We use Evalb[3] toolkit to evaluate parsing performance.

4.2 Characteristics of Our Model

There are several hyper-parameters in our model, e.g., the word embedding dimension ($wordDim$), the hidden layer node size ($hiddenSize$), the Dropout ratio ($dropRatio$) and the beam size for inference ($beamSize$). The choice of these hyper-parameters may affect the final performance. In this subsection, we present some experiments to demonstrate the characteristics of our model, and select a group of proper hyper-parameters that we use to evaluate our final model. All the experiments in this subsection were performed on Chinese data and the evaluation is performed on Chinese development set.

First, we evaluated the effectiveness of various primitive units. We set $wordDim = 300$, $hiddenSize = 300$, $beamSize = 8$, and did not apply Dropout ($dropRatio = 0$). Table 2 presents the results. By comparing numbers in other rows

[2]https://code.google.com/p/word2vec/
[3]http://nlp.cs.nyu.edu/evalb/

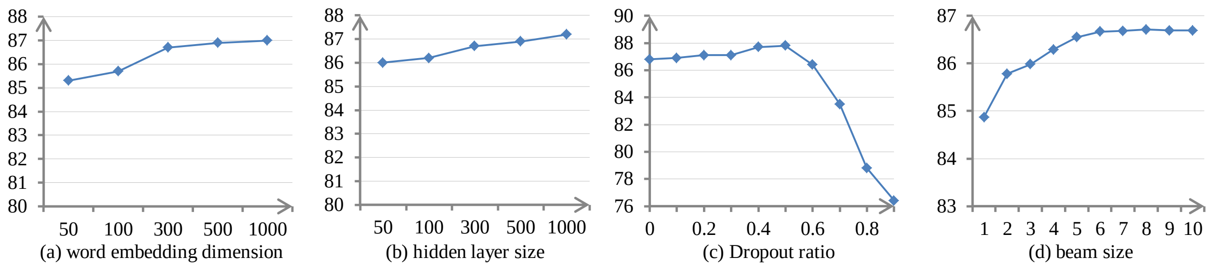

Figure 3: Influence of hyper-parameters.

with row "All Units", we found that ablating the Prefix and Suffix units ("w/o Prefix & Suffix") significantly hurts both POS tagging and parsing performance. Ablating POS units ("w/o POS") or constituent label units ("w/o NT") has little effect on POS tagging accuracy, but hurts parsing performance. When only keeping the word type units ("Only Word"), both the POS tagging and parsing accuracy drops drastically. So the Prefix and Suffix units are crucial for POS tagging, and POS units and constituent label units are helpful for parsing performance. All these primitive units are indispensable to better performance.

Second, we uncovered the effect of the dimension of word embedding. We set $hiddenSize = 300$, $beamSize = 8$, $dropRatio = 0$ and varied $wordDim$ among {50, 100, 300, 500, 1000}. Figure 3(a) draws the parsing performance curve. When increasing $wordDim$ from 50 to 300, parsing performance improves more than 1.5 percentage points. After that, the curve flattens out, and parsing performance only gets marginal improvement. Therefore, in the following experiments, we fixed $wordDim = 300$.

Third, we tested the effect of hidden layer node size. We varied $hiddenSize$ among {50, 100, 300, 500, 1000}. Figure 3(b) draws the parsing performance curve. We found increasing $hiddenSize$ is helpful for parsing performance. However, higher $hiddenSize$ would greatly increase the amount of computation. To keep the efficiency of our model, we fixed $hiddenSize = 300$ in the following experiments.

Fourth, we applied Dropout and tuned the Dropout ratio through experiments. Figure 3(c) shows the results. We found that the peak performance occurred at $dropRatio = 0.5$, which brought about an improvement of more than 1 percentage point over the model without Dropout ($dropRatio = 0$). Therefore, we fixed $dropRatio = 0.5$.

Primitive Units	F_1	POS
All Units	86.7	96.7
w/o Prefix & Suffix	85.7	95.4
w/o POS	86.0	96.7
w/o NT	86.2	96.6
Only Word	82.7	95.2

Table 2: Influence of primitive units.

Finally, we investigated the effect of beam size. Figure 3(d) shows the curve. We found increasing $beamSize$ greatly improves the performance initially, but no further improvement is observed after $beamSize$ is greater than 8. Therefore, we fixed $beamSize = 8$ in the following experiments.

4.3 Semi-supervised Training

In this subsection, we investigated whether we can train more effective models using automatically parsed data. We randomly selected 200K sentences from our unlabeled data sets for both Chinese and English. Then, we used an in-house shift-reduce parser[4] to parse these selected sentences. The size of the automatically parsed data set may have an impact on the final model. So we trained many models with varying amounts of automatically parsed data. We also designed two strategies to exploit the automatically parsed data. The first strategy (Mix-Train) is to directly add the automatically parsed data to the hand-annotated training set and train models with the mixed data set. The second strategy (Pre-Train) is to first pretrain models with the automatically parsed data, and then fine-tune models with the hand-annotated training set.

Table 3 shows results of different experimental configurations for Chinese. For the Mix-Train

[4]Its performance is F_1 =83.9 on Chinese and F_1 =90.8% on English.

	Mix-Train		Pre-Train	
# Auto Sent	F_1	POS	F_1	POS
0	87.8	97.0	—	—
50K	87.2	96.8	88.4	97.1
100K	88.7	96.9	89.5	97.1
200K	89.2	97.2	89.5	97.4

Table 3: Semi-supervised training for Chinese.

	Mix-Train		Pre-Train	
# Auto Sent	F_1	POS	F_1	POS
0	89.7	96.6	—	—
50K	89.4	96.1	90.2	96.4
100K	89.5	96.0	90.4	96.5
200K	89.2	95.8	90.8	96.7

Table 4: Semi-supervised training for English.

strategy, when we only use 50K automatically parsed sentences, the performance drops in comparison with the model trained without using any automatically parsed data. When we increase the automatically parsed data to 100K sentences, the parsing performance improves about 1 percent but the POS tagging accuracy drops slightly. When we further increase the automatically parsed data to 200K sentences, both the parsing performance and POS tagging accuracy improve. For the Pre-Train strategy, the performance of all three configurations improves performance against the model that does not use any automatically parsed data. The Pre-Train strategy consistently outperforms the Mix-Train strategy when the same amount of automatically parsed data is used. Therefore, for Chinese, the Pre-Train strategy is much more helpful, and the more automatically parsed data we use the better performance we get.

Table 4 presents results of different experimental configurations for English. The performance trend for the Mix-Train strategy is different from that of Chinese. Here, no matter how much automatically parsed data we use, there is a consistent degradation in performance against the model that does not use any automatically parsed data at all. And the more automatically parsed data we use, the larger the drop in accuracy. For the Pre-Train strategy, the trend is similar to Chinese. The parsing performance of the Pre-Train setting consistently improves as the size of automatically parsed data increases.

Type	System	F_1
Ours	Supervised*‡	83.2
	Pretrain-Finetune*‡	**86.6**
SI	Petrov and Klein (2007)	83.3
	Wang and Xue (2014)‡	83.6
SE	Zhu et al. (2013)‡	85.6
	Wang and Xue (2014)‡	86.3
RE	Charniak and Johnson (2005)	82.3
	Wang and Zong (2011)	85.7

Table 5: Comparison with the state-of-the-art systems on Chinese test set. * marks neural network based systems. ‡ marks shift-reduce parsing systems.

4.4 Comparing With State-of-the-art Systems

In this subsection, we present the performance of our models on the testing sets. We trained two systems. The first system ("Supervised") is trained only with the hand-annotated training set, and the second system ("Pretrain-Finetune") is trained with the Pre-Train strategy described in subsection 4.3 using additional automatically parsed data. The best parameters for the two systems are set based on their performance on the development set. To further illustrate the effectiveness of our systems, we also compare them with some state-of-the-art systems. We group parsing systems into three categories: supervised single systems (SI), semi-supervised single systems (SE) and reranking systems (RE). Both of our two models belong to semi-supervised single systems, because our "Supervised" system utilized word embeddings in its input layer.

Table 5 lists the performance of our systems as well as the state-of-the-art systems on Chinese test set. Comparing the performance of our two systems, we see that our "Pretrain-Finetune" system shows a fairly large gain over the "Supervised" system. One explanation is that our neural network model is a non-linear model, so the back propagation algorithm can only reach a local optimum. In our "Supervised" system the starting points are randomly initialized in the parameter space, so it only reaches local optimum. In comparison, our "Pretrain-Finetune" system gets to see large amount of automatically parsed data, and initializes the starting points with the pre-trained

Type	System	F_1
Ours	Supervised*‡	89.4
	Pretrain-Finetune*‡	90.7
SI	Collins (1999)	88.2
	Charniak (2000)	89.5
	Henderson (2003)*	88.8
	Petrov and Klein (2007)	90.1
	Carreras et al. (2008)	91.1
	Zhu et al. (2013)‡	90.4
SE	Huang et al. (2010)	91.6
	Collobert (2011)*	89.1
	Zhu et al. (2013)‡	91.3
RE	Henderson (2004)*	90.1
	Charniak and Johnson (2005)	91.5
	McClosky et al. (2006)	**92.3**
	Huang (2008)	91.7
	Socher et al. (2013)*	90.4

Table 6: Comparing with the state-of-the-art systems on English test set. * marks neural network based systems. ‡ marks shift-reduce parsing systems.

parameters. So it finds a much better local optimum than the "Supervised" system. Comparing our "Pretrain-Finetune" system with all the state-of-the-art systems, we see our system surpass all the other systems. Although our system only utilizes some basic primitive units (in Table 1(a)), it still outperforms Wang and Xue (2014)'s shift-reduce parsing system which uses more complex structural features and semi-supervised word cluster features. Therefore, our model can simultaneously learn an effective feature representation and make accurate parsing predictions for Chinese.

Table 6 presents the performance of our systems as well as the state-of-the-art systems on the English test set. Our "Pretrain-Finetune" system still achieves much better performance than the "Supervised" system, although the gap is smaller than that of Chinese. Our "Pretrain-Finetune" system also outperforms all other neural network based systems (systems marked with *). Although our system does not outperform all the state-of-the-art systems, the performance is comparable to most of them. So our model is also effective for English parsing.

4.5 Cross Domain Evaluation

In this subsection, we examined the robustness of our model by evaluating it on data sets from various domains. We use the Berkeley Parser as our baseline parser, and trained it on our training set.

For Chinese, we performed our experiments on the cross domain data sets from Chinese Treebank 8.0 (Xue et al., 2013). It consists of six domains: newswire (nw), magazine articles (mz), broadcast news (bn), broadcast conversation (bc), weblogs (wb) and discussion forums (df). Since all of the mz domain data is already included in our training set, we only selected sample sentences from the other five domains as the test sets [5], and made sure these test sets had no overlap with our treebank training, development and test sets. Note that we did not use any data from these five domains for training or development. The models are still the ones described in the previous subsection. The results are presented in Table 7. Although our "Supervised" model got slightly worse performance than the Berkeley Parser (Petrov and Klein, 2007), as shown in Table 5, it outperformed the Berkeley Parser on the cross-domain data sets. This suggests that the learned features can better adapt to cross-domain situations. Compared with the Berkeley Parser, on average our "Pretrain-Finetune" model is 3.4 percentage points better in terms of parsing accuracy, and 3.2 percentage points better in terms of POS tagging accuracy. We also presented the performance of our pre-trained model ("Only-Pretrain"). We found the "Only-Pretrain" model performs poorly on this cross-domain data sets. But even pre-training based on this less than competitive model, our "Pretrain-Finetune" model achieves significant improvement over the "Supervised" model. So the Pre-Train strategy is crucial to our model.

For English, we performed our experiments on the cross-domain data sets from OntoNote 5.0 (Weischedel et al., 2013), which consists of nw, mz, bn, bc, wb, df and telephone conversations (tc). We also performed experiments on the SMS domain, using data annotated by the LDC for the DARPA BOLT Program. We randomly selected 300 sentences for each domain as the test sets [5]. Table 8 presents our experimental results. To save space, we only presented the results of our "Pretrain-Finetune" model and the Berkeley

[5]The selected sentences can be downloaded from http://www.cs.brandeis.edu/ xuen/publications.html

	Only-Pretrain		Supervised		Pretrain-Finetune		BerkeleyParser	
domain	F_1	POS	F_1	POS	F_1	POS	F_1	POS
bc	61.6	81.1	72.9	90.2	74.9	91.2	68.2	86.4
bn	—	—	78.2	93.2	80.8	94.2	78.3	91.2
df	65.6	84.5	76.2	91.7	78.5	92.6	75.9	90.3
nw	72.0	86.1	82.1	95.2	85.0	95.8	82.9	93.6
wb	65.4	81.5	74.6	89.5	76.9	90.2	73.8	86.7
average	66.2	83.3	76.8	92.0	79.2	92.8	75.8	89.6

Table 7: Cross-domain performance for Chinese. The "Only-Pretrain" model cannot successfully parse some sentences in bn domain, so we didn't give the numbers.

	Pretrain-Finetune		BerkeleyParser	
Domain	F_1	POS	F_1	POS
bc	77.7	92.2	76.0	91.1
bn	88.1	95.4	88.2	95.0
df	82.5	93.3	79.4	92.4
nw	89.6	95.3	86.2	94.6
wb	83.3	93.1	82.0	91.2
sms	79.2	85.8	74.6	85.3
tc	74.2	88.0	71.1	87.6
average	82.1	91.9	79.6	91.0

Table 8: Cross-domain performance for English.

Parser. Except for the slightly worse performance on the bn domain, our model outperformed the Berkeley Parser on all the other domains. While our model is only 0.6 percentage point better than the Berkeley Parser (Petrov and Klein, 2007) when evaluated on the standard Penn TreeBank test set (Table 6), our parser is 2.5 percentage points better on average on the cross domain data sets. So our parser is also very robust for English on cross-domain data sets.

5 Related Work

There has been some work on feature optimization in dependency parsing, but most prior work in this area is limited to selecting an optimal subset of features from a set of candidate features (Nilsson and Nugues, 2010; Ballesteros and Bohnet, 2014). Lei et al. (2014) proposed to learn features for dependency parsing automatically. They first represented all possible features with a multi-way tensor, and then transformed it into a low-rank tensor as the final features that are actually used by their system. However, to obtain competitive performance, they had to combine the learned features with traditional hand-crafted features. Chen and Manning (2014) proposed to learn a dense feature vector for transition-based dependency parsing via neural networks. Their model had to learn POS tag embeddings and dependency label embeddings first, and then induced the dense feature vector based on these embeddings. Comparing with their method, our model is much simpler. Our model learned features directly based on the original form of primitive units.

There have also been some attempts to use neural networks for constituent parsing. Henderson (2003) presented the first neural network for broad coverage parsing. Later, he also proposed to rerank k-best parse trees with a neural network model which achieved state-of-the-art performance (Henderson, 2004). Collobert (2011) designed a recurrent neural network model to construct parse tree by stacks of sequences labeling, but its final performance is significantly lower than the state-of-the-art performance. Socher et al. (2013) built a recursive neural network for constituent parsing. However, rather than performing full inference, their model can only score parse candidates generated from another parser. Our model also requires a parser to generate training samples for pre-training. However, our system is different in that, during testing, our model performs full inference with no need of other parsers. Vinyals et al. (2014) employed a Long Short-Term Memory (LSTM) neural network for parsing. By training on a much larger hand-annotated data set, their performance reached 91.6% for English.

6 Conclusion

In this paper, we proposed to learn features via a neural network model. By taking as input the primitive units, our neural network model learns feature representations in the hidden layer and made parsing predictions based on the learned features in the output layer. By employing the back-propagation algorithm, our model simultaneously induced features and learned prediction model parameters. We show that our model achieved significant improvement from pretraining on a substantial amount of pre-parsed data. Evaluated on standard data sets, our model outperformed all state-of-the-art parsers on Chinese and all neural network based models on English. We also show that our model is particularly effective on cross-domain tasks for both Chinese and English.

Acknowledgments

We thank the anonymous reviewers for comments. Haitao Mi is supported by DARPA HR0011-12-C-0015 (BOLT) and Nianwen Xue is supported by DAPRA HR0011-11-C-0145 (BOLT). The views and findings in this paper are those of the authors and are not endorsed by the DARPA.

References

Miguel Ballesteros and Bernd Bohnet. 2014. Automatic feature selection for agenda-based dependency parsing.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

Michael Collins. 1999. *HEAD-DRIVEN STATISTICAL MODELS FOR NATURAL LANGUAGE PARSING*. Ph.D. thesis, University of Pennsylvania.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*, number EPFL-CONF-192374.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660.

James Henderson. 2003. Neural network probability estimation for broad coverage parsing. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 131–138. Association for Computational Linguistics.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 95. Association for Computational Linguistics.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 12–22. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Daniel Jurafsky and James H Martin. 2008. Speech and language processing.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1381–1391.

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.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics.

Peter Nilsson and Pierre Nugues. 2010. Automatic discovery of feature sets for dependency parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 824–832. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*, pages 404–411.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, PA.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2014. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*.

Zhiguo Wang and Nianwen Xue. 2014. Joint pos tagging and transition-based constituent parsing in chinese with non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 733–742. Association for Computational Linguistics.

Zhiguo Wang and Chengqing Zong. 2011. Parse reranking based on higher-order lexical dependencies. In *IJCNLP*, pages 1251–1259.

Mengqiu Wang, Kenji Sagae, and Teruko Mitamura. 2006. A fast, accurate deterministic parser for chinese. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 425–432. Association for Computational Linguistics.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Nianwen Xue, Martha Palmer, Mitchell Marcus, Ann Taylor, et al. 2013. Ontonotes release 5.0. *Linguistic Data Consortium, Philadelphia*.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

Nianwen Xue, Xiuhong Zhang, Zixin Jiang, Martha Palmer, Fei Xia, Fu-Dong Chiou, and Meiyu Chang. 2013. Chinese treebank 8.0. *Philadelphia: Linguistic Data Consortium*, page https://catalog.ldc.upenn.edu/LDC2013T21.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 162–171. Association for Computational Linguistics.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, Sofia, Bulgaria, August. Association for Computational Linguistics.