Journal of Artificial Intelligence Research 58 (2017) 703-729 Submitted 07/16; published 03/17

A Neural Probabilistic Structured-Prediction Method for Transition-Based Natural Language Processing

Hao Zhou ZHOUH@NLP.NJU.EDU.CN
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, Jiangsu, China

Yue Zhang YUE_ZHANG@SUTD.EDU.SG
Information Systems Technology and Design
Singapore University of Technology and Design
Singapore

Chuan Cheng CHENGC@NLP.NJU.EDU.CN
Shujian Huang HUANGSJ@NLP.NJU.EDU.CN
Xinyu Dai DAIXY@NLP.NJU.EDU.CN
Jiajun Chen CHENJJ@NLP.NJU.EDU.CN
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, Jiangsu, China

Abstract

We propose a neural probabilistic structured-prediction method for transition-based natural language processing, which integrates beam search and contrastive learning. The method uses a global optimization model, which can leverage arbitrary features over non-local context. Beam search is used for efficient heuristic decoding, and contrastive learning is performed for adjusting the model according to search errors. When evaluated on both chunking and dependency parsing tasks, the proposed method achieves significant accuracy improvements over the locally normalized greedy baseline on the two tasks, respectively.

1. Introduction

Transition-based methods with global optimization and beam search (Zhang & Clark, 2011b) have been used to solve many structured-prediction problems in natural language processing. Such method construct outputs by using a sequence of transition actions over input sentences, adopting a structured model for an action *sequence* directly instead of modeling each action *individually*. As shown in Figure 1a, a greedy POS-tagging model selects the highest scoring action for each word in a sentence, which may suffer from label bias (Lafferty, McCallum, & Pereira, 2001). In contrast, a structured POS-tagging model in Figure 1b directly searches for the best scored action sequence globally, which alleviates the problems of greedy models. As a structured prediction model, global optimization with beam search yields fast speed and high accuracies, enabled by integrating search and learning, and the freedom to use arbitrary features. For natural language processing tasks such as Chinese word segmentation (Zhang & Clark, 2007), dependency parsing (Yamada & Matsumoto, 2003; Nivre & Scholz, 2004; Zhang & Clark, 2008; Huang & Sagae, 2010;

Zhang & Nivre, 2011; Goldberg & Nivre, 2013) and constituent parsing (Zhang & Clark, 2011b; Zhu, Zhang, Chen, Zhang, & Zhu, 2013; Zhang, Zhang, Che, & Liu, 2013), for which features play a important role, the transition-based method gives highly competitive accuracies and efficiencies.

Traditional transition-based systems represent context with large amounts of sparse features. More complex features can be obtained by combining atomic sparse features such as word or a POS tag. Chen and Manning (2014) propose a greedy neural dependency parsing model, which represents atomic features as dense vectors, obtaining feature combination via non-linear neural networks automatically rather than devising high-order features manually. The greedy neural parser achieves significant accuracy improvements upon the greedy discrete Maltparser model (Nivre, 2004), and runs in a very fast speed. The idea of the transition-based neural parser can be used for other NLP tasks such as part-of-speech tagging, name entity recognition, chunking and CCG parsing. However, the accuracy of the greedy transition-based neural parser of Chen and Manning lags behind state-of-the-art discrete systems with sparse features (Zhang & Nivre, 2011), which adopt global learning and beam search decoding (Zhang & Nivre, 2012; Huang, Fayong, & Guo, 2012).

The respective advantages of structured learning with beam search and neural network modeling give rise to the research question whether the two techniques can be combined for reaping the benefits from each. In this article, we propose a novel structured-prediction framework for transition-based neural models, which adopts global learning upon action sequences instead of local learning on individual actions. Following Zhang and Clark (2011), beam search is applied for decoding, and global structured learning is integrated with search (Daumé III, Langford, & Marcu, 2009; Doppa, Fern, & Tadepalli, 2014; Chang, Krishnamurthy, Agarwal, Daume, & Langford, 2015; Chang, He, Ross, Daume III, & Langford, 2016; Zhou, Zhang, Huang, Zhou, Dai, & Chen, 2016) using early-update (Collins & Roark, 2004). Different from Zhang and Clark (2011b), a neural network is utilized to score transition actions, rather than a linear model with discrete features. Designing such a framework is challenging for two main reasons:

First, applying global structured-prediction models for transition-based structured task such as parsing is non-trivial. A direct adaptation of the framework of Zhang and Clark (2011b) or Huang et al. (2012) under the neural probabilistic model setting does not yield good results. In the neural case, our systems are probabilistic models, which give each transition action a probability. Because the actions in a sequence are highly dependent, the log probability of an action sequence cannot be modeled just as the sum of log probabilities of each action in the sequence, which is the case of discrete structured prediction models. This issue can be understood as label bias, since a model with each action being individually normalized does not necessarily give optimal action sequence scores. We address the challenge by using a *softmax* function to directly normalize the distribution of action sequences.

Second, for the structured model above, maximum-likelihood training is computationally intractable, requiring summing over all possible action sequences, which is difficult for transition-based systems. To address this challenge, we take a contrastive learning approach (Hinton, 2002; LeCun & Huang, 2005; Liang & Jordan, 2008; Vickrey, Lin, & Koller, 2010; Liu & Sun, 2014), using the sum of log probabilities over the action sequences in the beam to approximate that over all possible action sequences. The larger the beam size is, the more exact the estimation of the probability mass is. The approximate estimation of

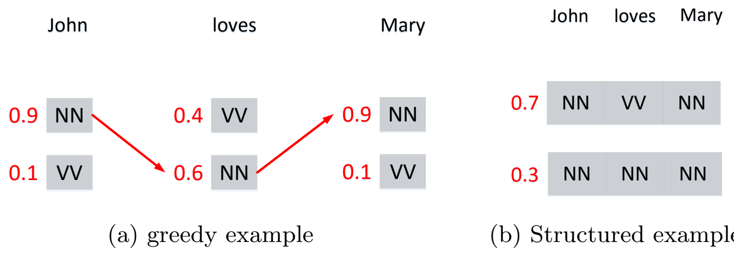

Figure 1: Example of greedy and structured POS-tagging models for the sentence "John loves Mary".

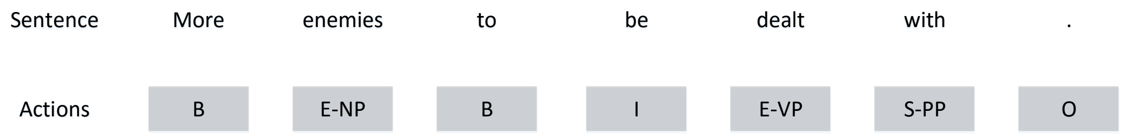

Figure 2: Example of chunking outputs: "[NP More enemies] [VP to be dealt] [PP with] [O .]" and its transition actions.

the beam contrastive learning is based on the assumption that the probability mass over the whole search space is concentrated on the limited best search candidates.

We evaluate the beam contrastive learning method on two fundamental NLP tasks: chunking and dependency parsing. For each task, both English and Chinese corpora are used. In the standard CoNLL 2000 data set for chunking (Tjong Kim Sang & Buchholz, 2000), our method obtains 0.39% and 0.99% accuracy improvements over a strong neural chunking baseline for English and Chinese, respectively. In standard dependency parsing evaluation (Marcus, Marcinkiewicz, & Santorini, 1993; Xue, Xia, Chiou, & Palmer, 2005), our method achieves accuracies of 93.31% and 85.01% for English and Chinese, respectively, which are significantly higher than the greedy neural parser of Chen and Manning (2014). The accuracy improvements achieved by the structured-prediction models show the effectiveness of the proposed beam contrastive learning framework, which could also be used for other structured-prediction tasks. This article is a much extended version of a conference paper (Zhou, Zhang, Huang, & Chen, 2015), which describes the neural structured-prediction model for dependency parsing.

2. Transition-Based Systems

Most structured prediction problems can be described using a transition-based framework, which constructs outputs by using sequences of incremental actions. In this section, we show how transition systems are defined for chunking and dependency parsing, respectively.

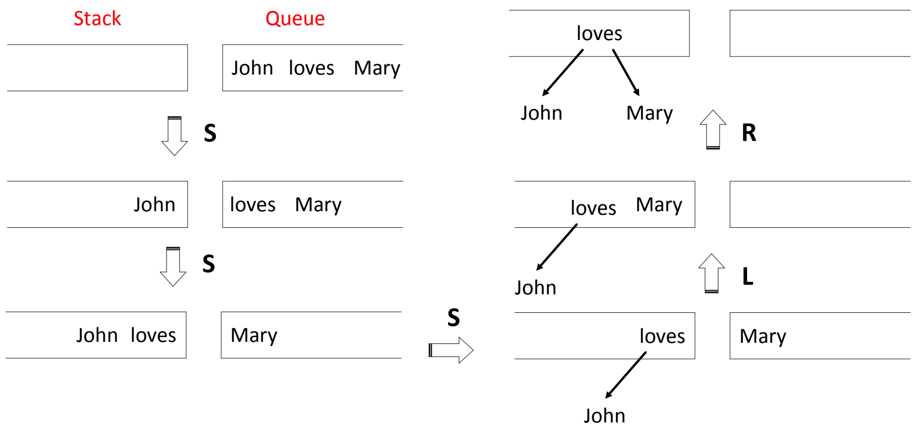

Figure 3: Example for processing the sentence "John loves Mary".

2.1 Transition System for Chunking

Given an input sentence, transition-based chunking employs a sequence of actions to generate a chunking output. The chunker maintains a *state* $\{q_c, q_w\}$, where q_c is the *chunk queue* for predicted chunks and q_w is the *word queue* for the incoming words to be processed. At each step of chunking, the chunker pops a word off q_w and chooses one action, determining the relationship between the word and the processing partial chunk. Denoting the front word of q_w as w_j, the chunker chooses one of the following transition actions at each step:

- *Begin* (B): mark w_j as the beginning of a chunk, which contains multiple words.
- *End* (E-l): mark w_j as the last word of the current partial chunk in q_c, with the syntactic chunking label l.
- *Outside* (O): mark w_j as an out-of-chunk word.
- *Inside* (I): mark w_j as an internal word of the current partial chunk in q_c.
- *Single* (S-l): mark w_j as the only word of a chunk with the syntactic chunking label l.

Given the sentence "More enemies to be dealt with .", the gold action sequence is *<B, E-NP, B, I, E-VP, S-PP, O>* as shown in Figure 2.

2.2 Transition System for Dependency Parsing

Transition-based dependency parsers scan an input sentence from left to right, performing a sequence of transition actions to predict its parse tree (Nivre, 2008). In this article, we employ the **arc-standard** system (Nivre, Hall, Nilsson, Chanev, Eryigit, Kübler, Marinov, & Marsi, 2007), which maintains partially-constructed outputs using a *stack*, and orders the incoming words in the input sentence in a *queue*. Parsing starts with an empty stack and a queue consisting of the whole input sentence. At each step, a *transition action* is taken to

$$\begin{array}{ll}
\text{input :} & w_0 \ldots w_{n-1} \\
\text{axiom :} & 0 : \langle 0, \phi, \phi, 0 \rangle \\
\text{goal :} & 2n-1 : \langle n, s_0, L \rangle \\
\textit{Shift} & \dfrac{k : \langle j, S, L \rangle}{k+1 : \langle j+1, S|w_j, L \rangle} \\
\textit{Left-Arc(l)} & \dfrac{k : \langle j, S|s_1|s_0, L \rangle}{k+1 : \langle j, S|s_0, L \cup \{ s_1 \xleftarrow{l} s_0 \} \rangle} \\
\textit{Right-Arc(l)} & \dfrac{k : \langle j, S|s_1|s_0, L \rangle}{k+1 : \langle j, S|s_1, L \cup \{ s_1 \xrightarrow{l} s_0 \} \rangle}
\end{array}$$

Figure 4: Deductive system for arc-standard dependency parsing.

consume the input and construct the output. The process repeats until the input queue is empty and the stack contains only one dependency tree.

Formally, a parsing state is denoted as $\langle j, S, L \rangle$, where S is a stack of subtrees $[\ldots s_2, s_1, s_0]$, j is the head of the queue (i.e. $[\ q_0 = w_j,\ q_1 = w_{j+1} \cdots]$), and L is a set of dependency arcs. At each step, the parser chooses one of the following actions:

- *Shift* (S): move the front word w_j from the queue onto the stacks.
- *Left-Arc* (L-l): add an arc with label l between the top two trees on the stack ($s_1 \leftarrow s_0$), and remove s_1 from the stack.
- *Right-Arc* (R-l): add an arc with label l between the top two trees on the stack ($s_1 \rightarrow s_0$), and remove s_0 from the stack.

The arc-standard parser can be summarized as the deductive system in Figure 4, where k denotes the current parsing step. For a sentence with size n, parsing stops after performing exactly $2n-1$ actions. Figure 3 shows an example of transition-based parsing. Given the sentence “John loves Mary”, the parsing system employs 5 transition actions to generate the output parsing tree.

3. Greedy Models

A greedy transition-based model generates structured outputs by employing a sequence of independent transition actions. At each step, the greedy model selects one optimal action with the locally highest score. The score of actions could be computed by using discrete and neural models.

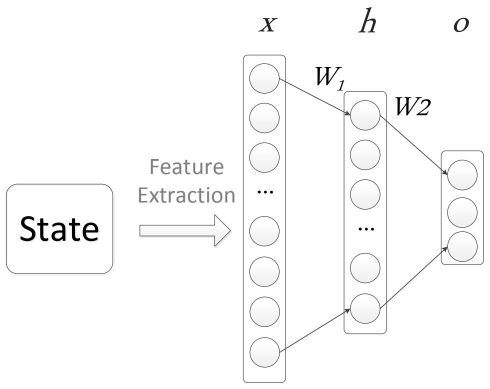

Figure 5: Neural network for greedy transition-based models .

3.1 Discrete Model

Many transition-based chunking and parsing systems use a discrete model for scoring transition actions at each step. In general, given an input x, the discrete model generates an output y by a multi-step action classification process. At each step, the action is:

$$a_i = \underset{a \in \mathrm{GEN}(x,i)}{\arg\max}\ score(a) \tag{1}$$

Here a_i is the action at the i_{th} step, and $\mathrm{GEN}(x, i)$ returns all possible actions at the i_{th} step. $score(y)$ is computed by a local classifier such as the support vector machine (SVM).

$$score(y) = \phi(y) \cdot \vec{\theta} \tag{2}$$

Here $\vec{\theta} \in R^d$ is the parameter vector of the classifier. $\phi(a_i)$ is the local feature vector, extracted using manually defined feature templates (Zhang & Clark, 2011b).

As a greedy transition-based example, Kudo and Matsumoto (2001) first propose to use SVM (Cortes & Vapnik, 1995) for syntactic chunking, which obtains the highest reported result in the CoNLL 2000 shared task for chunking. SVM is also adopted in greedy dependency parsing (Yamada & Matsumoto, 2003; Nivre, 2004). Their greedy discrete model runs in linear time, and obtains competitive performance by employing large numbers of manual sparse features.

3.2 Neural Model

Different from discrete models, a greedy neural model extracts n features from a state item, and represents the extracted features as *embeddings*, each of which is represented as a d-dimensional vector $e_i \in \mathbb{R}$. Chen and Manning (2014) propose to use a greedy neural model for transition-based parsing, which exploits the strong representation ability of neural networks, and achieves higher accuracy than discrete counterparts. Therefore, the full embedding matrix is $E \in \mathbb{R}^{d \times V}$, where V is the number of distinct features (Figure 5). A *projection layer* is used to concatenate the n input embeddings into a vector $x = [e_1; e_2 \ldots e_n]$, where $x \in \mathbb{R}^{d \cdot n}$. The purpose of this layer is to fine-tune the embedding

Algorithm 1: Beam Search Decoding Algorithm.

```
Input: $problem$; $agenda$; $candidates$; $B$
Output: $best$
$candidates \leftarrow$ StartItem($problem$)
$agenda \leftarrow$ Clear(agenda)
while $true$ do
    foreach $candidate \in candidates$ do
        $agenda \leftarrow$ Insert(Expand($candidate$, $problem$), $agenda$)
    $best \leftarrow$ Top($agenda$)
    if GoalTest($best$) then
        return $best$
    $candidates \leftarrow$ Top-B($agenda$, $B$) $agenda \leftarrow$ Clear($agenda$)
```

features. Then x is mapped to a d_h-dimensional *hidden layer* by a mapping matrix $W_1 \in \mathbb{R}^{d_h \times d \cdot n}$ and an activation function σ:

$$h = \sigma(W_1 x + b_1) \tag{3}$$

Finally, h is mapped into a *softmax output layer* for modeling the probabilistic distribution of candidate transition actions:

$$p = softmax(o) \tag{4}$$

where

$$o = W_2 h \tag{5}$$

Here $W_2 \in \mathbb{R}^{d_o \times d_h}$ and d_o is the number of actions. Each dimension of the output layer o corresponds to the probability of a next action. The greedy neural model selects the most probable action at each step and performs the transition process until chunking or parsing finishes.

Given a set of training examples, the training objective of the greedy neural model is to minimize the cross-entropy loss, plus a l_2-regularization term:

$$L(\theta) = -\sum_{i \in A} \log p_i + \frac{\lambda}{2} \parallel \theta \parallel^2 \tag{6}$$

θ is the set of all parameters (i.e. W_1, W_2, b, E), and A is the set of all gold actions in the training data. AdaGrad (Duchi, Hazan, & Singer, 2011) with mini-batch is adopted for optimization.

4. Structured-Prediction Using Global Discrete Model and Beam-Search

As mentioned in introduction, drawbacks of the greedy transition system include label bias and error propagation. An incorrect action will have a negative influence to its subsequent actions, leading to an incorrect output. To address this issue, Zhang and Clark (2011b) propose to use a globally optimized model for transition-based methods, learning a sequence of transition actions as a unit, rather than each action individually, which alleviates the label bias and error propagation problems of greedy discrete models by combining a global discrete model with a beam-search decoding algorithm, giving results competitive with the state-of-the-art on various tasks.

4.1 Beam-Search

The global discrete model uses beam search to obtain the highest scored output heuristically, where an agenda is employed to keep the K-best partial outputs at each incremental step, and the partially built outputs are represented as state items. For a shift-reduce parser, a state item includes the stack structure for the shift-reduce process and the queue of incoming unanalyzed words.

The agenda is initialized as empty, and the state item that corresponds to the initial structure is put onto it before decoding starts. At each step during decoding, each state item from the agenda is extended with one incremental step. When there are multiple choices to extend one state item, multiple new state items are generated. The new state items are ranked by their scores, and the B-best are put back onto the agenda. The process iterates until a stopping criterion is met, and the current best item from the agenda is taken as the output.

Pseudocode for the generic beam-search algorithm is given in Algorithm 1, where the variable *problem* represents a particular task, such as chunking or dependency parsing, and the variable *candidate* represents a state item, which has a different definition for each task. For example, for the dependency parsing task, a candidate is a state item, consisting of a stack of partially parsed tree structures and a buffer of remaining words to be parsed. The *agenda* is an ordered list, used to keep all the state items generated at each stage, ordered by score. The variable *candidates* is the set of state items that can be used to generate new state items, that is, the B-best state items from the previous stage. B is the number of state items retained at each stage.

StartItem initializes the start state item according to the problem; for example, for the dependency parsing task, the start state item is a pair consisting of an empty stack and the complete sequence of words and its corresponding POS-tags waiting to be parsed. Clear removes all items from the agenda. Insert puts one or more state items onto the agenda. Expand represents an incremental processing step, which takes a state item and generates new state items from it in all possible ways. For example, for the dependency parsing task, Expand takes the partially parsed tree structure in a state item, and extends it with all valid shift-reduce actions. Top returns the highest scoring state item on the *agenda*. GoalTest checks whether the incremental decoding process is completed; for example, for the dependency task, the process is complete if the stack contains only one tree and the queue is empty. Top-B returns the B-highest scoring state items on the *agenda*, which are used as candidates for the next incremental step. State items in the agenda are ranked by their scores.

Algorithm 2: Global Online Learning Algorithm.

```
Input: training examples $(x_i, y_i)$
Output: $\vec{\theta}$
$\vec{\theta} \leftarrow 0$;
for $r$ = 1..P, $i$ = 1..N do
    $y'_i = \text{Decode}(x_i)$;
    if $y'_i \neq y_i$ then
        $\vec{\theta} = \vec{\theta} + \Phi(y_i) - \Phi(y'_i)$;
```

4.2 Model

The key difference between structured-prediction models and greedy models is that, given an input, a structured-prediction model searches for the highest scored structural output (which corresponds to an action sequence), instead of obtaining the optimal action at each step as in the greedy case. Given an input x, the goal of a global discrete model is to find the highest-scored action sequence globally.

$$y' = \underset{y \in \text{GEN}(x)}{\arg\max}\, score(y), \tag{7}$$

where $\text{GEN}(x)$ denotes all possible action sequences on x. For the problem of dependency parsing, x is a sentence with POS-tags and y' is highest scored chunking or parsing output.

To compute $score(y)$, the output structure y is mapped into a global feature vector $\Phi(y) \in N^d$. Each dimension of $\Phi(y)$ represents a sparse feature, and discrete model always extracts billion of sparse indicator features from the output according to a set of feature templates (Zhang & Clark, 2011b). Note that $\Phi(y)$ is a global feature vector, which includes all feature vectors for the whole action sequence instead of one action in the sequence. Given the feature vector $\Phi(y)$, $score(y)$ is computed using a linear model:

$$score(y) = \Phi(y) \cdot \vec{\theta}, \tag{8}$$

where $\vec{\theta} \in R^d$ is the parameter vector of the model.

4.3 Training

Zhang and Clark (2011b) adopt the on-line perceptron algorithm (Collins, 2002) with early-update (Collins & Roark, 2004) for model training. Given a training instance $(x_i, y_i) \in (X, Y)$, a beam search decoder produces a highest scored output y'_i according to the parameter vector $\vec{\theta}$. Here, x_i is the input sentence and y_i is its corresponding output. For each instance, the output y'_i is compared with the gold structure y_i. If y'_i is correct, no update is performed. If y'_i is incorrect, the parameter vector is updated by adding the global feature vector of y_i and subtracting the global feature vector of y'_i. Intuitively, the training process effectively coerces the decoder to produce the correct output.

As shown in Algorithm 2, the training process is performed iteratively, guiding the search algorithm by correcting search errors. Here, P is the iteration numbers and N is the total size of training data (X, Y). The early-update strategy of Collins and Roark (2004)

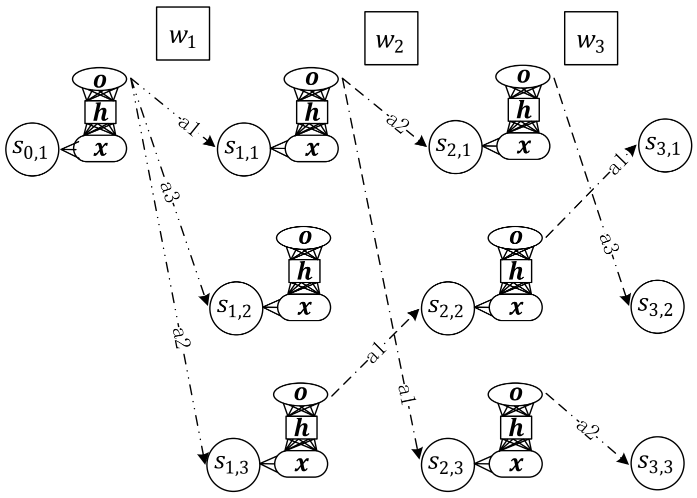

Figure 6: Example global neural model with beam size = 3. Here the action set is $\{a_1, a_2, a_3\}$ and 3 actions are performed for each state item in the example.

	local classifier	structured prediction
linear sparse	Section 3.1 (Nivre et al., 2007)	Section 4 (Zhang and Nivre, 2011)
neural dense	Section 3.2 (Chen and Manning, 2014)	this work

Table 1: Correlation between different parsing models.

is adopted in the on-line learning process. At any step during incremental decoding, if all the partial candidates in the agenda are incorrect, decoding is stopped and the parameter vector is updated immediately, according to the current best candidate in the agenda and the corresponding gold-standard partial output.

5. Structured-Prediction Using Neural Networks and Beam Search

The global discrete model introduced in Section 4 obtains better performance compared to the greedy discrete model (Zhang & Clark, 2011b), which shows the strength of structured-prediction. We propose a globally normalized neural model, which combines the advantages of structured learning and neural network modeling over greedy discrete models.

A direct adaptation of the framework of Zhang and Clark (2011b) under the neural probabilistic model setting does not yield good results. We propose a neural probabilistic structured-prediction model, which finds the distribution of action sequences directly, instead of computing probability of action sequences by multiplying the probabilities of each individual action in the sequence. Following Zhang and Clark (2011), beam search is applied to decoding, and global structured learning is integrated with beam search using early-update (Collins & Roark, 2004). As shown in Table 1, for dependency parsing, the model can be seen as a neural probabilistic alternative of Zhang and Clark (2011b), or a structured-prediction alternative of Chen and Manning (2014).

5.1 Model

Given a sentence x and a set of neural network parameter θ, the probability of the action sequence y_i is given by the *softmax* function:

$$p(y_i \mid x, \theta) = \frac{e^{f(x,\ \theta)_i}}{\sum\limits_{y_j \in \mathrm{GEN}(x)} e^{f(x,\ \theta)_j}} \quad , \tag{9}$$

where

$$f(x,\ \theta)_i = \sum_{a_k \in y_i} o(x,\ y_i,\ k,\ a_k) \tag{10}$$

Here $\mathrm{GEN}(s)$ is the set of all possible valid action sequences for a sentence x and $o(x,\ y_i,\ k,\ a_k)$ denotes the neural network score for the action a_k given x and y_i. We use the same neural network as Chen and Manning (2014) to calculate $o(x,\ y_i,\ k,\ a_k)$ (Equation 5). The beam search algorithm described in Algorithm 1 is used to obtain the most probable action sequence, according to Equation 9.

5.2 Tranining

Given the training data as $(X,\ Y)$, our training objective is to minimize the negative log-likelihood:

$$L(\theta) = -\sum_{(x_i,\, y_i) \in (X, Y)} \log p(y_i \mid x_i, \theta) \tag{11}$$

$$= -\sum_{(x_i,\, y_i) \in (X, Y)} \log \frac{e^{f(x_i, \theta)_i}}{Z(x_i, \theta)} \tag{12}$$

$$= \sum_{(x_i,\, y_i) \in (X, Y)} \log Z(x_i, \theta) - f(x_i, \theta)_i \quad , \tag{13}$$

where

$$Z(x, \theta) = \sum_{y_j \in \mathrm{GEN}(x)} e^{f(x,\ \theta)_j} \tag{14}$$

Here, $Z(x, \theta)$ is called the *partition function*. Following Chen and Manning (2014), we apply l_2-regularization for training. AdaGrad (Duchi et al., 2011) with mini-batch is adopted for optimization.

Algorithm 3: Training Algorithm for Global Neural Model with Neural Networks.

```
Input: training examples ($\mathbf{X}, \mathbf{Y}$)
Output: $\theta$
$\theta \leftarrow$ pretrained embedding
for $i \leftarrow 1$ to $N$ do
    $\mathbf{x}, \mathbf{y}$ = RandomSample($\mathbf{X}, \mathbf{Y}$)
    $\delta = \mathbf{0}$
    foreach $x_j, y_j \in \mathbf{x}, \mathbf{y}$ do
        $beam = \phi$
        $goldState =$ null
        $notEnd =$ true
        $beamGold =$ true
        while $beamGold$ and $notEnd$ do
            $beam =$ decode($beam$, $x_j$, $y_j$)
            $goldState =$ goldMove($goldState$, $x_j$, $y_j$)
            if not isGold($beam$) then
                $beamGold =$ false
            if isEnd($beam$) then
                $notEnd =$ false;
        $\delta = \delta +$ update($goldState$, $beam$)
    $\theta = \theta + \delta$;
```

For optimization, we need to compute gradients for $L(\theta)$, which includes gradients of exponential numbers of negative examples in the partition function $Z(x, \theta)$. However, beam search is used for transition-based parsing with non-local features, and no efficient optimal dynamic program is available to estimate $Z(x, \theta)$ accurately. We adopt a novel contrastive learning approach to approximately compute $Z(x, \theta)$. As an alternative to maximize the likelihood on some observed data, contrastive learning (Hinton, 2002; LeCun & Huang, 2005; Liang & Jordan, 2008; Vickrey et al., 2010; Liu & Sun, 2014) is an approach that assigns higher probabilities to observed data and lower probabilities to noisy data.

We adopt contrastive learning by assigning higher probabilities to the gold action sequence compared to incorrect action sequences in the beam. Our new training objective is approximated as:

$$L'(\theta) = -\sum_{(x_i, y_i) \in (X,Y)} \log p'(y_i \mid x_i, \theta) \tag{15}$$

$$= -\sum_{(x_i, y_i) \in (X,Y)} \log \frac{e^{f(x_i, \theta)_i}}{Z'(x_i, \theta)} \tag{16}$$

$$= \sum_{(x_i, y_i) \in (X,Y)} \log Z'(x_i, \theta) - f(x_i, \theta)_i \tag{17}$$

where

$$Z'(x,\theta) = \sum_{y_j \in \mathrm{BEAM}(x)} e^{f(x,\ \theta)_j} \tag{18}$$

$p'(y_i \mid x, \theta)$ is the relative probability of the action sequence y_i, computed over only the action sequences in the beam. $Z'(x,\theta)$ is the contrastive approximation of $Z(x,\theta)$. $\mathrm{BEAM}(x)$ returns the predicted action sequences in the beam. Intuitively, this method only penalizes incorrect action sequences with high probabilities.

We assume that the probability mass concentrates on a relatively small number of action sequences, which allows the use of a limited number of probable sequences to approximate the full set of action sequences. The concentration may be enlarged dramatically with an exponential activation function of the neural network (i.e. $a > b \Rightarrow e^a \gg e^b$).

5.3 Integrating Search and Learning

As shown in Algorithm 3, the proposed model also adopts the beam-search (Section 4.1) algorithm for decoding, integrating search and learning (Figure 6). At each training iteration i, we randomly sample a set of training instances, and perform on-line learning with early update (Collins & Roark, 2004). In particular, given a training example, we use beam-search to decode the sentence. At any step. if the gold action sequence falls out of the beam, we take all the incorrect action sequences in the beam as negative examples, and the current gold sequence as a positive example for parameter update, using the training algorithm of Section 5.2. In this way, the distribution of not only full action sequences (i.e. complete parse trees), but also partial action sequences (i.e. partial outputs) are modeled. Due to the model integrating search and learning, early update is triggered frequently when the model is not well learned, and our model pays more attention to correct the partial outputs in search and learning. With the increasingly in-depth training, the global model is trained well on the partial outputs, and will begin to focus on the complete outputs. This process shows the effectiveness of integrating search and learning.

6. Features

One advantage of Chen and Manning (2014) is that neural networks has strong power in feature representation and can achieve feature combination automatically. In this article, we use the same feature templates in both greedy neural baseline model and the global neural model.

6.1 Chunking

Following previous work (Zhou, Qu, & Zhang, 2012; Huang, Xu, & Yu, 2015; Lyu, Zhang, & Ji, 2016), we divide chunking features into 5 types. As shown in Table 2, F_w, F_t, F_l, F_c, F_a represent word, POS-tag, action, word capital and affix features, respectively. $q_c^0.w$ represents the first word in q_c^0; $q_w^1.t$ represents the POS-tag of the second word in q_w; $q_c^0.l$ represents the transition action for q_c^0; $q_c^0.c$ represents capitalization information for $q_c^0.w$, namely whether the word is all in lower case, all in upper case, with the first letter in upper case or with at least one letter in upper case. $prefix_i(w)$, $suffix_i(w)$ return the i-th prefix or

Type	Templates	Size
F_w	$q_c^1.w$; $q_c^0.w$; $q_w^0.w$; $q_w^1.w$; $q_w^2.w$	50
F_t	$q_c^1.t$; $q_c^0.t$; $q_w^0.t$; $q_w^1.t$; $q_w^2.t$; $q_c^1.t \circ q_c^0.t$; $q_c^0.t \circ q_w^0.t$; $q_w^0.t \circ q_w^1.t$; $q_w^1.t \circ q_w^2.t$; $start(q_c.C^2).t$; $end(q_c.C^2).t$; $head(q_c.C^2).t$; $start(q_c.C^1).t$; $end(q_c.C^1).t$; $head(q_c.C^1).t$ $start(q_c.C^0).t$	50
F_l	$q_c^1.l$; $q_c^0.l$; $start(q_c.C^0).l$; $end(q_c.C^2).l$; $end(q_c.C^1).l$	50
F_c	$q_c^1.c$; $q_c^0.c$; $q_w^0.c$; $q_w^1.c$; $q_w^2.c$	5
F_a	$prefix_1(q_w^0.w)$; $prefix_2(q_w^0.w)$; $suffix_2(q_w^0.w)$; $prefix_3(q_w^0.w)$; $suffix_1(q_w^0.w)$; $suffix_3(q_w^0.w)$	20

Table 2: Feature templates for chunking, where *Size* denotes the dimension size of the corresponding feature embedding.

suffix letters of given a word w. $q_c.C_0$ is the first chunk in a chunk queue. $start(C)$, $end(C)$ and $head(C)$ return the start word, the end word and the head word[1] for a chunk C.

6.2 Dependency Parsing

The atomic features are defined by following Zhang and Nivre (2011). As shown in Table 3, the features are categorized into three types: F^w, F^t, F^l, which represent word features, POS-tag features and dependency label features, respectively. For example, s_0w and q_0w represent the first word on the stack and queue, respectively; $lc_1(s_0)w$ and $rc_1(s_0)w$ represent the leftmost and rightmost child of s_0, respectively. Similarly, $lc_1(s_0)t$ and $lc_1(s_0)l$ represent the POS-tag and dependency label of the leftmost child of s_0, respectively. In detail, F_w contains $n_w = 18$ elements: The top 3 words on the stack and buffer: s_1w, s_2w, s_3w, b_1w, b_2w, b_3w; the first and second leftmost / rightmost children of the top two words on the stack: $lc_1(s_i)$, $rc_1(s_i)$, $lc_2(s_i)$, $rc_2(s_i)$, $i = 1, 2$. (3) The leftmost of the leftmost / rightmost of the rightmost children of the top two words on the stack: $lc_1(lc_1(s_i))$, $rc_1(rc_1(s_i))$, i = 1, 2. For the POS-tagging feature F_p, we use the corresponding POS tags for F_w ($n_t = 18$), and the corresponding arc labels of words excluding those words on the stack/buffer ($n_l = 12$).

7. Experiments

We conduct chunking and parsing experiments on both English and Chinese corporaa.

7.1 Data and Evaluation

7.1.1 Chunking

For English chunking, we evaluate the systems using the CoNLL 2000 dataset (Tjong Kim Sang & Buchholz, 2000), which consists of the Wall Street Journal (WSJ) corpus

1. We use manual head rules to extract the head word of chunks (Collins, 1999; Zhang & Clark, 2009).

Type	Templates	Size
F^w	$s_0w, s_2w, q_0w, q_1w, q_2w, lc_1(s_0)w, lc_2(s_0)w$ $s_1w, rc_2(s_0)w, lc_1(s_1)w, lc_2(s_1)w, rc_2(s_1)w$ $lc_1(lc_1(s_0))w, lc_1(lc_1(s_2))w$ $rc_1(s_1)w, rc_1(rc_1(s_0))w, rc_1(rc_1(s_1))w$	50
F^t	$s_0t, s_2t, q_0t, q_1t, q_2t, lc_1(s_0)t, lc_2(s_0)t$ $s_1t, rc_2(s_0)t, lc_1(s_1)t, lc_2(s_1)t, rc_2(s_1)t$ $lc_1(lc_1(s_0))t, lc_1(lc_1(s_2))t$ $rc_1(s_1)t, rc_1(rc_1(s_0))t, rc_1(rc_1(s_1))t$	50
F^l	$rc1(s_0)t, lc_1(s_1)t, lc_2(s_1)t, lc1(lc_1(s_0))t$ $lc_1(s_0)t, lc_1(lc_1(s_1))t, rc2(s_0)t, lc_2(s_0)t$ $rc2(s_1)t, rc_1(rc_1(s_0))t, rc_1(rc_1(s_1))t, rc1(s_1)t$	50

Table 3: Feature templates for dependency parsing. Size denotes the dimension size of the corresponding feature embedding.

and is widely used for chunking evaluation. Following previous work on CoNLL, we use sections 15–18 of WSJ for training, section 21 for development (Sha & Pereira, 2003) and section 20 for testing. The training set consists of 8936 sentences, and the test set consists of 2012 sentences. The part-of-speech tags of the chunking data are derived using the Brill tagger.

Chinese chunking experiments were performed on the CTB4 dataset (Chen, Zhang, & Isahara, 2006), which consists of 838 files. We used the first 728 files (FID from chtb 001.fid to chtb 899.fid) as the training data, and the other 110 files (FID from chtb 900.fid to chtb 1078.fid) as the test data. The training set consists of 9878 sentences, and the test set consists of 5920 sentences.

We extract head words using the head rules of Collins (1999) for English, and those of Zhang and Clark (2011b) for Chinese. The standard evaluation metrics for chunking are used, which includes the precision p (the fraction of output chunks matching the reference chunks), the recall r (the fraction of reference chunks returned), and the F-measure given by $F = 2pr/(p + r)$.

7.1.2 Dependency Parsing

Our English experiments are performed using the English Penn Treebank (PTB Marcus et al., 1993). We follow the standard splits of PTB3, using sections 2-21 for training, section 22 for development testing and section 23 for final testing. For comparison with previous work, we use Penn2Malt to convert constituent trees to dependency trees. We use the POS-tagger of Collins (2002) to assign POS automatically. 10-fold jackknifing is performed for tagging the training data. We use the labeled attachment score (LAS) and the unlabeled attachment score (UAS) to evaluate the parsing accuracy.

7.1.3 Pre-training Word Embeddings

For English, we follow Chen and Manning (2014) and use the set of pre-trained word embeddings from Collobert et al. (2011) with a dictionary size of 13,000. The word embeddings

Type	Chunking	Parsing
embedding size d	50	50
hidden layer size d_h	300	300
regularization λ	10^{-8}	10^{-8}
initial learning rate α	0.05	0.01
batch size b	6000	2000

Table 4: Hyper-parameters.

Description	Parsing UAS		Chunking F1 Score	
Baseline	91.78		93.17	
	structured	greedy	structured	greedy
beam = 1	78.51	91.78	51.89	93.17
beam = 4	84.04	92.01	93.16	93.25
beam = 8	92.81	92.17	93.72	93.25
beam = 16	93.18	92.29	93.90	93.25
beam = 32	93.40	92.27	93.76	93.25
beam = 64	93.48	92.24	94.84	93.25

Table 5: Accuracies of the structured neural model and the greedy neural model with different beam sizes.

were trained on the entire English Wikipedia, which contains about 631 million words. For Chinese, we train 50-dimensional embeddings using word2vec on Wikipedia and Gigaword corpus for Chinese.

7.2 Hyper-parameters

We tune the hyper-parameters of neural networks on the development set for both chunking and parsing, setting them as shown in Table 4. In addition to the hyper-parameters, beam size is also important to the system performance of structured prediction. We tune the beam sizes for chunking and parsing on the development sets.

7.3 Beam Size

Beam search enlarges the number of search candidates. Typically, the larger the beam is, the more accurate search algorithm is. Contrastive learning approximates the exact probabilities of exponential many action sequences by computing the relative probabilities over action sequences in the beam (Equation 18). Therefore, the larger the beam is, the more accurate the approximation is. In this section, we evaluate the results of chunking and parsing with different beam sizes on the English data set.

As shown in the *structured* columns in Table 5, the performances of the beam contrastive models improve as the beam size increases. To verify that the accuracy improvements are not due to the relieved error propagation by the beam search decoding, we also report the results of the greedy neural baseline with beam search decoding, in the *greedy* columns in

System	F1 Score
Kudo and Matsumoto (2001)	93.91
Shen and Sarkar (2005)	94.01
Sun, Morency, Okanohara, and Tsujii (2008)	94.34
Collobert et al. (2011)	94.32
Huang et al. (2015)	94.46
baseline greedy model	94.19
beam contrastive learning	94.58

Table 6: English results on CoNLL for chunking.

Table 5. The score of a whole action sequence in the greedy model is computed by the sum of log action probabilities.

From Table 5, we can find that beam search can improve greedy parsing. With increasing beam sizes, decoding using a greedy model gives slightly improved chunking (+0.1) and parsing (+0.5) accuracies. In contrast, by integrating beam search and global learning, performances of the structured model benefit from larger beam sizes much more significantly. With smaller beam size (i.e. 4), accuracies of the structured model lag behind of those of the greedy model. Structured parsing and chunking with beam size 1 give really bad performances, which is likely because the partition function in the structured model is estimated poorly. With increasing beam sizes, accuracies of chunking and parsing improve significantly. When the beam size is set as 64, the structured models outperform baselines by 0.4% and 1.48% on chunking and parsing, respectively.

Zhang and Nivre (2012) find that global learning and beam search should be used jointly for improving parsing using a *discrete* transition-based model. In particular, by increasing the beam size, the accuracy of ZPar (Zhang & Nivre, 2011) increases significantly, but that of MaltParser does not. For structured *neural* models, our finding is similar: integrating search and learning is much more effective than using beam search only in decoding. For neural parsing, we also obtain slight accuracy improvements with beam search decoding. We speculate that the neural parser benefits from a softmax layer, which serves as a form of normalization. If we remove the softmax layer in testing with beam search, the accuracy of the structured system drops to 60%.

7.4 Results on English

In this section, we compare chunking and parsing performances using our neural structured-prediction framework with a line of previous work on English data.

7.4.1 Chunking

We compare a set of related chunking methods in Table 6. Our model achieves an accuracy of 94.58. To our best knowledge, this is the best reported result achieved by a single chunking model on this data set. The method of Kudo and Matsumoto (2001) gives the highest result in CoNLL 2000, which adopts the SVM training and dynamic programming decoding. The most related work includes Collobert et al. (2011) and Huang et al. (2015). Collobert et al. use neural networks to model the chunking context. With a CRF based training objective,

	System	UAS	LAS
SUP	Huang and Sagae (2010)	92.10	-
	Zhang and Nivre (2011)	92.90	91.80
	Choi and McCallum (2013)	92.96	91.93
	Ma, Zhang, and Zhu (2014a)	93.06	-
SEMI	Suzuki, Isozaki, Carreras, and Collins (2009)	93.79	-
	Koo, Carreras, and Collins (2008)	93.16	-
	Chen, Zhang, and Zhang (2014)	93.77	-
NEURAL	Andor et al. (2016)†	94.61	92.79
	Weiss, Alberti, Collins, and Petrov (2015)†	93.99	92.05
	Dyer, Ballesteros, Ling, Matthews, and Smith (2015)†	93.20	90.90
	baseline greedy model	91.67	90.62
	beam contrastive model	93.31	92.37

Table 7: Results on WSJ. SUP denotes the parsing methods with supervised models, SEMI denotes methods with semi-supervised models and NEURAL denotes neural parsing models. Results with † are reported on Stanford Dependencies.

it obtains an accuracy of 94.43%. Huang et al. extend Collebert's work with a bi-direction LSTM, and achieved further accuracy improvements. However, accuracies of their neural models are lower than ours, and the main reason is that their neural CRF method makes a Markov assumption on the label sequence, while our beam contrastive learning method is a global learning model, extracting arbitrary features from the context.

7.4.2 Parsing

Table 7 shows the results of our parser and previous work on the test set. Our structured parser achieves an accuracy of 93.31%, with s a 0.41% improvement over Zhang and Nivre (2011), which employ millions of high-order binary indicator features. The model size of ZPar (Zhang & Nivre, 2011) is over 250 MB on disk. In contrast, the model size of our structured neural parser is only 25 MB.

Bohnet and Nivre (2012) obtain an accuracy of 93.67%, which is higher than our parser. However, their parser is a joint model of parsing and POS-tagging, and they use external data in parsing. We list the result of Chen et al. (2014), Koo et al. (2008) and Suzuki et al. (2009) in Table 7, which make use of large-scale unannotated text to improve parsing accuracies. The input embeddings of our parser are also trained over large raw text, and in this perspective our model is correlated with the semi-supervised models.

We also compare our parser with parsing models using neural networks. Dyer et al. (2015) propose a locally normalized transition-based dependency parser, which gives higher parsing accuracies than Chen and Manning (2014) by using three stack LSTMs to represent the parsing features more effectively. Weiss et al. (2015) propose a structured neural transition-based parser, which uses a greedy parsing process for pre-training, and fine-tunes an additional perceptron layer consisting of the pre-trained hidden and output layers using structured perceptron updates. The main difference is that they adopt deeper neural networks and perform much carefully hype-parameter tuning. Compared to Dyer et al.,

System	F1 Score
Chen et al. (2006) *Memory-based*	87.88
Chen et al. (2006) *Transform-based*	89.95
Chen et al. (2006) *CRF*	90.74
Chen et al. (2006) *SVM*	91.46
Zhou et al. (2012)	92.11
baseline greedy model	91.0
beam contrastive model	91.99

Table 8: Chinese chunking results. Result with † is given by a semi-supervised model.

our parser adopts a structured-prediction model, which models the whole action sequence, reducing the error of label bias. Our proposed model can be used to further enhance the performance of Dyer et al., which is orthogonal to our method. Different to our model, Weiss et al. build a structured-prediction parsing model by only updating the parameters of the perceptron layer and employing a perceptron-based objective. Andor et al. (2016) use the same globally normalized networks as ours, except adopting deeper networks and careful hyper-parameters tuning. The theory behind our approach was further developed by Andor et al., who verify the effectiveness for the beam contrastive learning model in a variety of NLP tasks, giving a detailed proof that the structured-prediction neural model is strictly more expressive than the greedy neural model.

The improvements indicate that chunking and dependency parsing using neural networks can achieve state-of-the-art performances without devising high-order features manually. The results show that our parser combines the benefits of structured models and neural probabilistic models, offering high accuracies and slim model size.

7.5 Results on Chinese

7.5.1 Chunking

As shown in Table 8, Chen et al. (2006) surveyed a line of chunking methods over the Chinese data. Zhou et al. (2012) proposed to utilize chunking level data, and achieved an accuracy of 92.11%, which is the best reported result on this data set. The structured chunker obtains 0.99% improvement over the greedy baseline, which shows the effectiveness of the beam contrastive learning.

7.5.2 Parsing

On Chinese Treebank, our parser achieves a 1.13% accuracy improvement over the greedy baseline parser of Chen and Manning (2014). Our parser obtains higher accuracies compared with the discrete structured-prediction transition-based parser of Zhang and Clark (2011b), which is consistent with the parsing results on English PTB. The parsing accuracy of our parser still lies slightly behind the parser of Huang and Sagae (2010), which adopts dynamic programming for decoding and searches much more tree candidates than ours.

System	UAS	LAS
Duan, Zhao, and Xu (2007)	83.88	-
Zhang and Clark (2011b)	84.33	-
Huang and Sagae (2010)	85.20	-
baseline greedy model	83.88	82.36
beam contrastive model	85.01	82.89

Table 9: Chinese Parsing results.

8. Related Work

Neural network methods have been widely applied in the natural language processing, obtaining state-of-the-art results for tasks such as part-of-speech tagging (Ma, Zhang, & Zhu, 2014b; Conti, Di Pietro, Mancini, & Mei, 2009; Ling, Luís, Marujo, Astudillo, Amir, Dyer, Black, & Trancoso, 2015), syntactic parsing (Chen & Manning, 2014; Dyer et al., 2015; Ballesteros, Dyer, & Smith, 2015) and machine translation (Devlin, Zbib, Huang, Lamar, Schwartz, & Makhoul, 2014; Bahdanau, Cho, & Bengio, 2014).

8.1 Parsing by Neural Networks

Neural models have been very useful for syntax parsing. Socher, Bauer, Manning, and Ng (2013) propose to use recursive neural networks in constituent parsing reranking, which achieves adequate accuracy improvements. Vinyals et al. (2015) adopt a sequence to sequence model (Sutskever, Vinyals, & Le, 2014) to generate linearized parsing trees, which achieves competitive parsing performances. Chen and Manning (2014) propose to use the neural networks for transition-based parsing, which represents atomic parsing features as dense vectors, obtaining feature combination automatically other than devising high-order features manually. A line of subsequent work proposes to extend the neural parser by dropping all their linguistic features, using LSTM (Hochreiter & Schmidhuber, 1997) to capture parsing features completely. Dyer et al. (2015) propose a greedy transition-based dependency parser, using three stack LSTMs to represent the input, the stack of partial syntactic trees and the history of parse actions, respectively. Based on Dyer et al. (2015). Ballesteros et al. (2015) further propose to use LSTM to model the relation among characters, leading to better parsing performances on morphology-rich languages. By modeling more history, the parser gives significant better accuracies compared to the greedy neural parser of Chen and Manning. Kiperwasser and Goldberg (2016a) propose to incorporate the hierarchical tree LSTM into an easy-first framework, and Kiperwasser and Goldberg (2016b) directly use a bi-direction LSTM for input feature modeling, both of which give state-of-the-art parsing accuracies.

8.2 Neural Structured-Prediction Parsing Models

Structured prediction approaches can be used to further enhanced the neural model performances. Most previous work uses locally optimized models, which suffer from the label bias problem and error propagation. Lafferty et al. (2001) propose the conditional random fields (CRF) model, utilizing a globally normalization to relieve the label bias problem, which is introduced into neural model by Do, Arti, et al. (2010). Durrett and Klein (2015) adopt the neural CRF model in constituent parsing, outperforming its linear counterparts.

The CRF objective (Lafferty et al., 2001) is used in many neural network models. models (Do et al., 2010; Collobert et al., 2011; Yao, Peng, Zweig, Yu, Li, & Gao, 2014; Zheng, Jayasumana, Romera-Paredes, Vineet, Su, Du, Huang, & Torr, 2015; Huang et al., 2015) for structured inference, which leads to significant accuracy improvements over locally optimized models. This shows the power of global optimized structured-prediction models, which are powerful for the strong ability of feature representation. However, the linear-chain neural CRF has limitations on feature extraction. In particular for dynamic programming decoding, the feature window is limited due to Markov assumptions. The strong representation learning ability of neural networks can be greatly limited from the local feature extraction.

8.3 Integrating Search and Learning

Weiss et al. (2015) and ourselves (Zhou et al., 2015) concurrently propose to enhance the parser of Chen and Manning with structured-prediction models. We (Zhou et al., 2015) use a globally normalized training objective for modeling the whole action sequence, while Weiss et al. only update the parameters of the perceptron layer by employing a perceptron-based objective. Watanabe and Sumita (2015), Xu, Auli, and Clark (2016) and Wiseman and Rush (2016) propose a different structured-prediction neural models with global optimization upon a sequence of actions. Watanabe and Sumita use the neural structured model and beam search on the constituent parsing task. Xu et al. use the same framework as ours except that their training objective is to maximize the expected F1 score instead of the probability of the predicted parsing tree. Additionally, Wiseman and Rush adopt a LaSO-like (Daumé III & Marcu, 2005) paradigm in training a search-based neural structured prediction model. Andor et al. (2016) adopt the same globally normalized networks as us (Zhou et al., 2015), except for using deeper networks and carefully tuning hyper-parameter, which lead to better performances on standard benchmarks. Andor et al. also verify the effectiveness of the beam contrastive learning in a variety of other NLP tasks, giving a detailed proof that the structured-prediction neural model is strictly more expressive than a greedy neural model.

The proposed beam contrastive learning integrates beam search and sentence-level log-likelihood in a neural transition-based framework, which allows rich global features and is fit for neural networks. The integrated search and learning approach was first proposed by Daumé III et al. (2009), the sprit of which is used in solving many other problems (Goldberg & Nivre, 2013; Doppa et al., 2014; Chang et al., 2015, 2016; Zhou et al., 2016). Combining heuristic search and approximated learning, the proposed model could obtain significant accuracy improvements empirically over a greedy baseline. Zhang and Clark (2011b) first propose to use a global optimized model for transition-based structured prediction task, using a perceptron algorithm with discrete manual features to solve a range of NLP problems, achieving high accuracies and efficiency. The correlation between our framework and neural CRF is the same as the one between the method of Zhang and Clark (2011b) and discrete CRF (Lafferty et al., 2001).

8.4 Applications

Our proposed model can also be used in many other tasks such as Chinese word segmentation (Zhang & Clark, 2007), CCG parsing (Zhang & Clark, 2011a; Xu, Clark, & Zhang, 2014), etc, which are structured-prediction tasks, aiming to predict a complex structure and could be solved by executing a sequence of transition actions. An action sequence corresponds to the resulting structure. For example, Chinese word segmentation can be solved by predicting a sequence of actions, which indicate whether the current character is the is an end of a word (Zhang, Zhang, & Fu, 2016).

9. Conclusion

We built a beam contrastive learning framework for transition-based NLP. Compared to greedy neural transition-based models, our proposed neural structured-prediction model integrates beam search and global contrastive learning for avoiding label bias. The framework outperforms the discrete baseline of Zhang and Clark (2011b) thanks to the strong context representation ability of neural networks, extracting richer context features compared the neural CRF. In standard evaluation, our proposed chunker and parser achieve average 0.69% and 1.39% accuracy improvements over a strong neural greedy baseline, respectively, which shows the effectiveness of combining search and learning. The structured neural probabilistic framework can be used for other incremental structured prediction tasks.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by the Natural Science Foundation of China (61672277, 61472183). Corresponding author: Shujian Huang and Yue Zhang.

References

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., & Collins, M. (2016). Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2442–2452. Association for Computational Linguistics.

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

Ballesteros, M., Dyer, C., & Smith, A. N. (2015). Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 349–359. Association for Computational Linguistics.

Chang, K.-W., He, H., Ross, S., Daume III, H., & Langford, J. (2016). A credit assignment compiler for joint prediction. In *Advances In Neural Information Processing Systems*, pp. 1705–1713.

Chang, K.-W., Krishnamurthy, A., Agarwal, A., Daume, H., & Langford, J. (2015). Learning to search better than your teacher. In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 2058–2066.

Chen, D., & Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Chen, W., Zhang, Y., & Zhang, M. (2014). Feature embedding for dependency parsing.. In *COLING*, pp. 816–826.

Chen, W., Zhang, Y., & Isahara, H. (2006). An empirical study of chinese chunking. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pp. 97–104. Association for Computational Linguistics.

Choi, J. D., & McCallum, A. (2013). Transition-based dependency parsing with selectional branching.. In *ACL (1)*, pp. 1052–1062.

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

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 1–8. Association for Computational Linguistics.

Collins, M., & Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, p. 111. Association for Computational Linguistics.

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

Conti, M., Di Pietro, R., Mancini, L. V., & Mei, A. (2009). (old) distributed data source verification in wireless sensor networks. *Inf. Fusion*, *10*(4), 342–353.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273––297.

Daumé III, H., Langford, J., & Marcu, D. (2009). Search-based structured prediction. *Machine learning*, *75*(3), 297–325.

Daumé III, H., & Marcu, D. (2005). Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, pp. 169–176. ACM.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R. M., & Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation.. In *ACL (1)*, pp. 1370–1380. Citeseer.

Do, T., Arti, T., et al. (2010). Neural conditional random fields. In *International Conference on Artificial Intelligence and Statistics*, pp. 177–184.

Doppa, J. R., Fern, A., & Tadepalli, P. (2014). Hc-search: A learning framework for search-based structured prediction.. *J. Artif. Intell. Res.(JAIR)*, *50*, 369–407.

Duan, X., Zhao, J., & Xu, B. (2007). Probabilistic parsing action models for multi-lingual dependency parsing..

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, *12*, 2121–2159.

Durrett, G., & Klein, D. (2015). Neural crf parsing. *arXiv preprint arXiv:1507.03641*.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Goldberg, Y., & Nivre, J. (2013). Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, *1*, 403–414.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, *14*(8), 1771–1800.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Huang, L., Fayong, S., & Guo, Y. (2012). Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–151. Association for Computational Linguistics.

Huang, L., & Sagae, K. (2010). Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1077–1086. Association for Computational Linguistics.

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Kiperwasser, E., & Goldberg, Y. (2016a). Easy-first dependency parsing with hierarchical tree lstms. *Transactions of the Association for Computational Linguistics*, *4*, 445–461.

Kiperwasser, E., & Goldberg, Y. (2016b). Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, *4*, 313–327.

Koo, T., Carreras, X., & Collins, M. (2008). Simple semi-supervised dependency parsing..

Kudo, T., & Matsumoto, Y. (2001). Chunking with support vector machines. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pp. 1–8. Association for Computational Linguistics.

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data..

LeCun, Y., & Huang, F. (2005). Loss functions for discriminative training of energybased models.. AIStats.

Liang, P., & Jordan, M. I. (2008). An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *Proceedings of the 25th international conference on Machine learning*, pp. 584–591. ACM.

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., & Trancoso, I. (2015). Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Liu, Y., & Sun, M. (2014). Contrastive unsupervised word alignment with non-local features. *arXiv preprint arXiv:1410.2082*.

Lyu, C., Zhang, Y., & Ji, D. (2016). Joint word segmentation, pos-tagging and syntactic chunking. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Ma, J., Zhang, Y., & Zhu, J. (2014a). Punctuation processing for projective dependency parsing. In *Proc. of ACL*.

Ma, J., Zhang, Y., & Zhu, J. (2014b). Tagging the web: Building a robust web tagger with neural network. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 144–154, Baltimore, Maryland. Association for Computational Linguistics.

Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, *19*(2), 313–330.

Nivre, J. (2004). Incrementality in deterministic dependency parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together (ACL Workshop)*, pp. 50–57.

Nivre, J. (2008). Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, *34*(4), 513–553.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., & Marsi, E. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, *13*(2), 95–135.

Nivre, J., & Scholz, M. (2004). Deterministic dependency parsing of English text. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pp. 64–70.

Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pp. 213–220.

Shen, H., & Sarkar, A. (2005). *Voting between multiple data representations for text chunking*. Springer.

Socher, R., Bauer, J., Manning, C. D., & Ng, A. Y. (2013). Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Sun, X., Morency, L.-P., Okanohara, D., & Tsujii, J. (2008). Modeling latent-dynamic in shallow parsing: a latent conditional model with improved inference. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pp. 841–848. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112.

Suzuki, J., Isozaki, H., Carreras, X., & Collins, M. (2009). An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of* *the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pp. 551–560. Association for Computational Linguistics.

Tjong Kim Sang, E. F., & Buchholz, S. (2000). Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pp. 127–132. Association for Computational Linguistics.

Vickrey, D., Lin, C. C., & Koller, D. (2010). Non-local contrastive objectives. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 1103–1110.

Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I., & Hinton, G. (2015). Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pp. 2773–2781.

Watanabe, T., & Sumita, E. (2015). Transition-based neural constituent parsing. *Proceedings of ACL-IJCNLP*.

Weiss, D., Alberti, C., Collins, M., & Petrov, S. (2015). Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158*.

Wiseman, S., & Rush, M. A. (2016). Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1296–1306. Association for Computational Linguistics.

Xu, W., Auli, M., & Clark, S. (2016). Expected f-measure training for shift-reduce parsing with recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 210–220, San Diego, California. Association for Computational Linguistics.

Xu, W., Clark, S., & Zhang, Y. (2014). Shift-reduce ccg parsing with a dependency model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 218–227, Baltimore, Maryland. Association for Computational Linguistics.

Xue, N., Xia, F., Chiou, F.-D., & Palmer, M. (2005). The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, *11*(2), 207–238.

Yamada, H., & Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, Vol. 3.

Yao, K., Peng, B., Zweig, G., Yu, D., Li, X., & Gao, F. (2014). Recurrent conditional random field for language understanding. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4077–4081. IEEE.

Zhang, M., Zhang, Y., Che, W., & Liu, T. (2013). Chinese parsing exploiting characters. In *51st Annual Meeting of the Association for Computational Linguistics*.

Zhang, M., Zhang, Y., & Fu, G. (2016). Transition-based neural word segmentation. *Proceedings of the 54nd ACL*.

Zhang, Y., & Clark, S. (2007). Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational*

Linguistics, pp. 840–847, Prague, Czech Republic. Association for Computational Linguistics.

Zhang, Y., & Clark, S. (2008). A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 562–571. Association for Computational Linguistics.

Zhang, Y., & Clark, S. (2009). Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies*, pp. 162–171. Association for Computational Linguistics.

Zhang, Y., & Clark, S. (2011a). Shift-reduce ccg parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 683–692. Association for Computational Linguistics.

Zhang, Y., & Clark, S. (2011b). Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, *37*(1), 105–151.

Zhang, Y., & Nivre, J. (2011). Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pp. 188–193. Association for Computational Linguistics.

Zhang, Y., & Nivre, J. (2012). Analyzing the effect of global learning and beam-search on transition-based dependency parsing.. In *COLING (Posters)*, pp. 1391–1400.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., & Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1529–1537.

Zhou, H., Zhang, Y., Huang, S., & Chen, J. (2015). A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1213–1222, Beijing, China. Association for Computational Linguistics.

Zhou, H., Zhang, Y., Huang, S., Zhou, J., Dai, X.-Y., & Chen, J. (2016). A search-based dynamic reranking model for dependency parsing. In *In Proceeding of ACL*.

Zhou, J., Qu, W., & Zhang, F. (2012). Exploiting chunk-level features to improve phrase chunking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 557–567. Association for Computational Linguistics.

Zhu, M., Zhang, Y., Chen, W., Zhang, M., & Zhu, J. (2013). Fast and accurate shift-reduce constituent parsing. In *51st Annual Meeting of the Association for Computational Linguistics*.