Fast and Accurate Shift-Reduce Constituent Parsing

Muhua Zhu[†], Yue Zhang[‡], Wenliang Chen[*], Min Zhang[*] and Jingbo Zhu[†]
[†]Natural Language Processing Lab., Northeastern University, China
[‡]Singapore University of Technology and Design, Singapore
[*] Soochow University, China and Institute for Infocomm Research, Singapore
zhumuhua@gmail.com yue_zhang@sutd.edu.sg
chenwenliang@gmail.com mzhang@i2r.a-star.edu.sg
zhujingbo@mail.neu.edu.cn

Abstract

Shift-reduce dependency parsers give comparable accuracies to their chart-based counterparts, yet the best shift-reduce constituent parsers still lag behind the state-of-the-art. One important reason is the existence of unary nodes in phrase structure trees, which leads to different numbers of shift-reduce actions between different outputs for the same input. This turns out to have a large empirical impact on the framework of global training and beam search. We propose a simple yet effective extension to the shift-reduce process, which eliminates size differences between action sequences in beam-search. Our parser gives comparable accuracies to the state-of-the-art chart parsers. With linear run-time complexity, our parser is over an order of magnitude faster than the fastest chart parser.

1 Introduction

Transition-based parsers employ a set of shift-reduce actions and perform parsing using a sequence of state transitions. The pioneering models rely on a classifier to make local decisions, and search greedily for a transition sequence to build a parse tree. Greedy, classifier-based parsers have been developed for both dependency grammars (Yamada and Matsumoto, 2003; Nivre et al., 2006) and phrase-structure grammars (Sagae and Lavie, 2005). With linear run-time complexity, they were commonly regarded as a faster but less accurate alternative to graph-based chart parsers (Collins, 1997; Charniak, 2000; McDonald et al., 2005).

Various methods have been proposed to address the disadvantages of greedy local parsing, among which a framework of beam-search and global discriminative training have been shown effective for dependency parsing (Zhang and Clark, 2008; Huang and Sagae, 2010). While beam-search reduces error propagation compared with greedy search, a discriminative model that is globally optimized for whole sequences of transition actions can avoid local score biases (Lafferty et al., 2001). This framework preserves the most important advantage of greedy local parsers, including linear run-time complexity and the freedom to define arbitrary features. With the use of rich non-local features, transition-based dependency parsers achieve state-of-the-art accuracies that are comparable to the best-graph-based parsers (Zhang and Nivre, 2011; Bohnet and Nivre, 2012). In addition, processing tens of sentences per second (Zhang and Nivre, 2011), these transition-based parsers can be a favorable choice for dependency parsing.

The above global-learning and beam-search framework can be applied to transition-based phrase-structure (constituent) parsing also (Zhang and Clark, 2009), maintaining all the aforementioned benefits. However, the effects were not as significant as for transition-based dependency parsing. The best reported accuracies of transition-based constituent parsers still lag behind the state-of-the-art (Sagae and Lavie, 2006; Zhang and Clark, 2009). One difference between phrase-structure parsing and dependency parsing is that for the former, parse trees with different numbers of unary rules require different numbers of actions to build. Hence the scoring model needs to disambiguate between transitions sequences with different sizes. For the same sentence, the largest output can take twice as many as actions to build as the

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 434–443,
Sofia, Bulgaria, August 4-9 2013.

smallest one. This turns out to have a significant empirical impact on parsing with beam-search.

We propose an extension to the shift-reduce process to address this problem, which gives significant improvements to the parsing accuracies. Our method is conceptually simple, requiring only one additional transition action to eliminate size differences between different candidate outputs. On standard evaluations using both the Penn Treebank and the Penn Chinese Treebank, our parser gave higher accuracies than the Berkeley parser (Petrov and Klein, 2007), a state-of-the-art chart parser. In addition, our parser runs with over 89 sentences per second, which is 14 times faster than the Berkeley parser, and is the fastest that we are aware of for phrase-structure parsing. An open source release of our parser (version 0.6) is freely available on the Web. [1]

In addition to the above contributions, we apply a variety of semi-supervised learning techniques to our transition-based parser. These techniques have been shown useful to improve chart-based parsing (Koo et al., 2008; Chen et al., 2012), but little work has been done for transition-based parsers. We therefore fill a gap in the literature by reporting empirical results using these methods. Experimental results show that semi-supervised methods give a further improvement of 0.9% in F-score on the English data and 2.4% on the Chinese data. Our Chinese results are the best that we are aware of on the standard CTB data.

2 Baseline parser

We adopt the parser of Zhang and Clark (2009) for our baseline, which is based on the shift-reduce process of Sagae and Lavie (2005), and employs global perceptron training and beam search.

2.1 Vanilla Shift-Reduce

Shift-reduce parsing is based on a left-to-right scan of the input sentence. At each step, a transition action is applied to consume an input word or construct a new phrase-structure. A stack is used to maintain partially constructed phrase-structures, while the input words are stored in a buffer. The set of transition actions are

- *SHIFT*: pop the front word from the buffer, and push it onto the stack.

[1] http://sourceforge.net/projects/zpar/

Axioms	$[\phi, 0, \textit{false}, 0]$
Goal	$[S, n, \textit{true}, C]$

Inference Rules:

SHIFT	$\dfrac{[S, i, \textit{false}, c]}{[S\|w, i+1, \textit{false}, c+c_s]}$
REDUCE-L/R-X	$\dfrac{[S\|s_1 s_0, i, \textit{false}, c]}{[S\|X, i, \textit{false}, c+c_r]}$
UNARY-X	$\dfrac{[S\|s_0, i, \textit{false}, c]}{[S\|X, i, \textit{false}, c+c_u]}$
FINISH	$\dfrac{[S, n, \textit{false}, c]}{[S, n, \textit{true}, c+c_f]}$

Figure 1: Deduction system of the baseline shift-reduce parsing process.

- *REDUCE-L/R-X*: pop the top two constituents off the stack, combine them into a new constituent with label X, and push the new constituent onto the stack.

- *UNARY-X*: pop the top constituent off the stack, raise it to a new constituent with label X, and push the new constituent onto the stack.

- *FINISH*: pop the root node off the stack and ends parsing.

The deduction system for the process is shown in Figure 1, where the item is formed as $\langle$*stack, buffer front index, completion mark, score*$\rangle$, and c_s, c_r, and c_u represent the incremental score of the *SHIFT*, *REDUCE*, and *UNARY* parsing steps, respectively; these scores are calculated according to the context features of the parser state item. n is the number of words in the input.

2.2 Global Discriminative Training and Beam-Search

For a given input sentence, the initial state has an empty stack and a buffer that contains all the input words. An agenda is used to keep the k best state items at each step. At initialization, the agenda contains only the initial state. At each step, every state item in the agenda is popped and expanded by applying a valid transition action, and the top k from the newly constructed state items are put back onto the agenda. The process repeats until the agenda is empty, and the best completed state item (recorded as *candidate output*) is taken for

Description	Templates
unigrams	s_0tc, s_0wc, s_1tc, s_1wc, s_2tc
	s_2wc, s_3tc, s_3wc, q_0wt, q_1wt
	q_2wt, q_3wt, s_0lwc, s_0rwc
	s_0uwc, s_1lwc, s_1rwc, s_1uwc
bigrams	s_0ws_1w, s_0ws_1c, s_0cs_1w, s_0cs_1c,
	s_0wq_0w, s_0wq_0t, s_0cq_0w, s_0cq_0t,
	q_0wq_1w, q_0wq_1t, q_0tq_1w, q_0tq_1t,
	s_1wq_0w, s_1wq_0t, s_1cq_0w, s_1cq_0t
trigrams	$s_0cs_1cs_2c$, $s_0ws_1cs_2c$, $s_0cs_1wq_0t$
	$s_0cs_1cs_2w$, $s_0cs_1cq_0t$, $s_0ws_1cq_0t$
	$s_0cs_1wq_0t$, $s_0cs_1cq_0w$

Table 1: A summary of baseline feature templates, where s_i represents the i_{th} item on the stack S and q_i denotes the i_{th} item in the queue Q. w refers to the head lexicon, t refers to the head POS, and c refers to the constituent label.

the output.

The score of a state item is the total score of the transition actions that have been applied to build the item:

$$C(\alpha) = \sum_{i=1}^{N} \Phi(a_i) \cdot \vec{\theta}$$

Here $\Phi(a_i)$ represents the feature vector for the i_{th} action a_i in state item α. It is computed by applying the feature templates in Table 1 to the context of α. N is the total number of actions in α.

The model parameter $\vec{\theta}$ is trained with the averaged perceptron algorithm, applied to state items (sequence of actions) globally. We apply the early update strategy (Collins and Roark, 2004), stopping parsing for parameter updates when the gold-standard state item falls off the agenda.

2.3 Baseline Features

Our baseline features are adopted from Zhang and Clark (2009), and are shown in Table 1 Here s_i represents the i_{th} item on the top of the stack S and q_i denotes the i_{th} item in the front end of the queue Q. The symbol w denotes the lexical head of an item; the symbol c denotes the constituent label of an item; the symbol t is the POS of a lexical head. These features are adapted from Zhang and Clark (2009). We remove Chinese specific features and make the baseline parser language-independent.

3 Improved hypotheses comparison

Unlike dependency parsing, constituent parse trees for the same sentence can have different numbers of nodes, mainly due to the existence of unary nodes. As a result, completed state

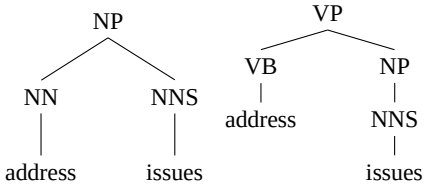

Figure 2: Example parse trees of the same sentence with different numbers of actions.

items for the same sentence can have different numbers of unary actions. Take the phrase "address issues" for example, two possible parses are shown in Figure 2 (a) and (b), respectively. The first parse corresponds to the action sequence [*SHIFT*, *SHIFT*, *REDUCE-R-NP*, *FINISH*], while the second parse corresponds to the action sequence [*SHIFT*, *SHIFT*, *UNARY-NP*, *REDUCE-L-VP*, *FINISH*], which consists of one more action than the first case. In practice, variances between state items can be much larger than the chosen example. In the extreme case where a state item does not contain any unary action, the number of actions is $2n$, where n is the number of words in the sentence. On the other hand, if the maximum number of consequent unary actions is 2 (Sagae and Lavie, 2005; Zhang and Clark, 2009), then the maximum number of actions a state item can have is $4n$.

The significant variance in the number of actions N can have an impact on the linear separability of state items, for which the feature vectors are $\sum_{i=1}^{N} \Phi(a_i)$. This turns out to have a significant empirical influence on perceptron training with early-update, where the training of the model interacts with search (Daume III, 2006).

One way of improving the comparability of state items is to reduce the differences in their sizes, and we use a *padding* method to achieve this. The idea is to extend the set of actions by adding an *IDLE* action, so that completed state items can be further expanded using the *IDLE* action. The action does not change the state itself, but simply adds to the number of actions in the sequence. A feature vector is extracted for the *IDLE* action according to the final state context, in the same way as other actions. Using the *IDLE* action, the transition sequence for the two parses in Figure 2 can be [*SHIFT*, *SHIFT*, *REDUCE-NP*, *FINISH*, *IDLE*] and [*SHIFT*, *SHIFT*, *UNARY-NP*, *REDUCE-L-VP*, *FINISH*], respectively. Their

Axioms $[\phi, 0, \textit{false}, 0, 0]$

Goal $[S, n, \textit{true}, m : 2n \leq m \leq 4n, C]$

Inference Rules:

SHIFT
$$\frac{[S, i, \textit{false}, k, \text{c}]}{[S|w, i+1, \textit{false}, k+1, c+c_s]}$$

REDUCE-L/R-X
$$\frac{[S|s_1 s_0, i, \textit{false}, k, c]}{[S|X, i, \textit{false}, k+1, c+c_r]}$$

UNARY-X
$$\frac{[S|s_0, i, \textit{false}, k, c]}{[S|X, i, \textit{false}, k+1, c+c_u]}$$

FINISH
$$\frac{[S, n, \textit{false}, k, c]}{[S, n, \textit{true}, k+1, c+c_f]}$$

IDLE
$$\frac{[S, n, \textit{true}, k, c]}{[S, n, \textit{true}, k+1, c+c_i]}$$

Figure 3: Deductive system of the extended transition system.

corresponding feature vectors have about the same sizes, and are more linearly separable. Note that there can be more than one action that are padded to a sequence of actions, and the number of IDLE actions depends on the size difference between the current action sequence and the largest action sequence without IDLE actions.

Given this extension, the deduction system is shown in Figure 3. We add the number of actions k to an item. The initial item (Axioms) has $k = 0$, while the goal item has $2n \leq k \leq 4n$. Given this process, beam-search decoding can be made simpler than that of Zhang and Clark (2009). While they used a *candidate output* to record the best completed state item, and finish decoding when the agenda contains no more items, we can simply finish decoding when all items in the agenda are completed, and output the best state item in the agenda. With this new transition process, we experimented with several extended features,and found that the templates in Table 2 are useful to improve the accuracies further. Here $s_i ll$ denotes the left child of s_i's left child. Other notations can be explained in a similar way.

4 Semi-supervised Parsing with Large Data

This section discusses how to extract information from unlabeled data or auto-parsed data to further improve shift-reduce parsing accuracies. We consider three types of information, including

s_0llwc, s_0lrwc, s_0luwc
s_0rlwc, s_0rrwc, s_0ruwc
s_0ulwc, s_0urwc, s_0uuwc
s_1llwc, s_1lrwc, s_1luwc
s_1rlwc, s_1rrwc, s_1ruwc

Table 2: New features for the extended parser.

paradigmatic relations, dependency relations, and structural relations. These relations are captured by word clustering, lexical dependencies, and a dependency language model, respectively. Based on the information, we propose a set of novel features specifically designed for shift-reduce constituent parsing.

4.1 Paradigmatic Relations: Word Clustering

Word clusters are regarded as lexical intermediaries for dependency parsing (Koo et al., 2008) and POS tagging (Sun and Uszkoreit, 2012). We employ the Brown clustering algorithm (Liang, 2005) on unannotated data (word segmentation is performed if necessary). In the initial state of clustering, each word in the input corpus is regarded as a cluster, then the algorithm repeatedly merges pairs of clusters that cause the least decrease in the likelihood of the input corpus. The clustering results are a binary tree with words appearing as leaves. Each cluster is represented as a bit-string from the root to the tree node that represents the cluster. We define a function *CLU(w)* to return the cluster ID (a bit string) of an input word *w*.

4.2 Dependency Relations: Lexical Dependencies

Lexical dependencies represent linguistic relations between words: whether a word modifies another word. The idea of exploiting lexical dependency information from auto-parsed data has been explored before for dependency parsing (Chen et al., 2009) and constituent parsing (Zhu et al., 2012).

To extract lexical dependencies, we first run the baseline parser on unlabeled data. To simplify the extraction process, we can convert auto-parsed constituency trees into dependency trees by using Penn2Malt. [2] From the dependency trees, we extract bigram lexical dependencies $\langle w_1, w_2, L/R \rangle$ where the symbol *L (R)* means that w_1 (w_2) is the head of w_2 (w_1). We also extract trigram lexical

[2] http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

dependencies $\langle w_1, w_2, w_3, L/R\rangle$, where L means that w_1 is the head of w_2 and w_3, meanwhile w_2 and w_3 are required to be siblings.

Following the strategy of Chen et al. (2009), we assign categories to bigram and trigram items **separately** according to their frequency counts. Specifically, top-10% most frequent items are assigned to the category of *High Frequency (HF)*; otherwise if an item is among top 20%, we assign it to the category of *Middle Frequency (MF)*; otherwise the category of *Low Frequency (LF)*. Hereafter, we refer to the bigram and trigram lexical dependency lists as *BLD* and *TLD*, respectively.

4.3 Structural Relations: Dependency Language Model

The dependency language model is proposed by Shen et al. (2008) and is used as additional information for graph-based dependency parsing in Chen et al. (2012). Formally, given a dependency tree y of an input sentence x, we can denote by $H(y)$ the set of words that have at least one dependent. For each $x_h \in H(y)$, we have a corresponding dependency structure $D_h = (x_{Lk}, \ldots x_{L1}, x_h, x_{R1}, \ldots, x_{Rm})$. The probability $P(D_h)$ is defined to be

$$P(D_h) = P_L(D_h) \times P_R(D_h)$$

where $P_L(D_h)$ can be in turn defined as:

$$\begin{aligned} P_L(D_h) \approx \quad & P(x_{L1}|x_h) \\ & \times P(x_{L2}|x_{L1}, x_h) \\ & \times \ldots \\ & \times P(x_{Lk}|x_{Lk-1}, \ldots, x_{Lk-N+1}, x_h) \end{aligned}$$

$P_R(D_h)$ can be defined in a similar way.

We build dependency language models on auto-parsed data. Again, we convert constituency trees into dependency trees for the purpose of simplicity. From the dependency trees, we build a bigram and a trigram language model, which are denoted by BLM and TLM, respectively. The following are the templates of the records of the dependency language models.

(1)	$\langle x_{Li}, x_h, P(x_{Li}\|x_h)\rangle$
(2)	$\langle x_{Ri}, x_h, P(x_{Ri}\|x_h)\rangle$
(3)	$\langle x_{Li}, x_{Li-1}, x_h, P(x_{Li}\|x_{Li-1}, x_h)\rangle$
(4)	$\langle x_{Ri}, x_{Ri-1}, x_h, P(x_{Ri}\|x_{Ri-1}, x_h)\rangle$

Here the templates (1) and (2) belong to BLM and the templates (3) and (4) belong to TLM. To use the dependency language models, we employ a map function $\Phi(r)$ to assign a category to each record *r* according to its probability, as in Chen et al. (2012). The following is the map function.

$$\Phi(r) = \begin{cases} HP & \text{if } P(r) \in top{-}10\% \\ MP & \text{else if } P(r) \in top{-}30\% \\ LP & \text{otherwise} \end{cases}$$

	Stat	Train	Dev	Test	Unlabeled
EN	# sent	39.8k	1.7k	2.4k	3,139.1k
	# word	950.0k	40.1k	56.7k	76,041.4k
CH	# sent	18.1k	350	348	11,810.7k
	# word	493.8k	8.0k	6.8k	269,057.2k

Table 4: Statistics on sentence and word numbers of the experimental data.

4.4 Semi-supervised Features

We design a set of features based on the information extracted from auto-parsed data or unannotated data. The features are summarized in Table 3. Here *CLU* returns a cluster ID for a word. The functions $BLD_{l/r}(\cdot)$, $TLD_{l/r}(\cdot)$, $BLM_{l/r}(\cdot)$, and $TLM_{l/r}(\cdot)$ check whether a given word combination can be found in the corresponding lists. For example, $BLD_l(s_1w, s_0w)$ returns a category tag (*HF*, *MF*, or *LF*) if $\langle s_1w, s_0w, L\rangle$ exits in the list BLD, else it returns *NONE*.

5 Experiments

5.1 Set-up

Labeled English data employed in this paper were derived from the Wall Street Journal (WSJ) corpus of the Penn Treebank (Marcus et al., 1993). We used sections 2-21 as labeled training data, section 24 for system development, and section 23 for final performance evaluation. For labeled Chinese data, we used the version 5.1 of the Penn Chinese Treebank (CTB) (Xue et al., 2005). Articles 001-270 and 440-1151 were used for training, articles 301-325 were used as development data, and articles 271-300 were used for evaluation.

For both English and Chinese data, we used ten-fold jackknifing (Collins, 2000) to automatically assign POS tags to the training data. We found that this simple technique could achieve an improvement of 0.4% on English and an improvement of 2.0% on Chinese. For English POS tagging, we adopted SVMTool, [3] and for Chinese POS tagging

[3] http://www.lsi.upc.edu/∼nlp/SVMTool/

Word Cluster Features		
$CLU(s_1w)$	$CLU(s_0w)$	$CLU(q_0w)$
$CLU(s_1w)s_1t$	$CLU(s_0w)s_0t$	$CLU(q_0w)q_0w$
Lexical Dependency Features		
$BLD_l(s_1w, s_0w)$	$BLD_l(s_1w, s_0w) \circ s_1t \circ s_0t$	$BLD_r(s_1w, s_0w)$
$BLD_r(s_1w, s_0w) \circ s_1t \circ s_0t$	$BLD_l(s_1w, q_0w) \circ s_1t \circ q_0t$	$BLD_l(s_1w, q_0w)$
$BLD_r(s_1w, q_0w)$	$BLD_r(s_1w, q_0w) \circ s_1t \circ q_0t$	$BLD_l(s_0w, q_0w)$
$BLD_l(s_0w, q_0w) \circ s_0t \circ q_0t$	$BLD_r(s_0w, q_0w) \circ s_0t \circ q_0t$	$BLD_r(s_0w, q_0w)$
$TLD_l(s_1w, s_1rdw, s_0w)$	$TLD_l(s_1w, s_1rdw, s_0w) \circ s_1t \circ s_0t$	$TLD_r(s_1w, s_0ldw, s_0w)$
$TLD_r(s_1w, s_0ldw, s_0w) \circ s_1t \circ s_0t$	$TLD_l(s_0w, s_0rdw, q_0w) \circ s_0t \circ q_0t$	$TLD_l(s_0w, s_0rdw, q_0w)$
$TLD_r(s_0w, NONE, q_0w)$	$TLD_r(s_0w, NONE, q_0w) \circ s_0t \circ q_0t$	
Dependency Language Model Features		
$BLM_l(s_1w, s_0w)$	$BLM_l(s_1w, s_0w) \circ s_1t \circ s_0t$	$BLM_r(s_1w, s_0w)$
$BLM_r(s_1w, s_0w) \circ s_1t \circ s_0t$	$BLM_l(s_0w, q_0w)$	$BLM_l(s_0w, q_0w) \circ s_0t \circ q_0t$
$BLM_r(s_0w, q_0w) \circ s_0t \circ q_0t$	$BLM_r(s_0w, q_0w)$	$TLM_l(s_1w, s_1rdw, s_0w)$
$TLM_l(s_1w, s_1rdw, s_0w) \circ s_1t \circ s_0t$	$TLM_r(s_1w, s_0ldw, s_0w)$	$TLM_r(s_1w, s_0ldw, s_0w) \circ s_1t \circ s_0t$

Table 3: Semi-supervised features designed on the base of word clusters, lexical dependencies, and dependency language models. Here the symbol s_i denotes a stack item, q_i denotes a queue item, w represents a word, and t represents a POS tag.

Lan.	System	LR	LP	F1
ENG	Baseline	88.4	88.7	88.6
	+padding	88.8	89.5	89.1
	+features	89.0	89.7	89.3
CHN	Baseline	85.6	86.3	86.0
	+padding	85.5	87.2	86.4
	+features	85.5	87.6	86.5

Table 5: Experimental results on the English and Chinese development sets with the padding technique and new supervised features added **incrementally**.

Lan.	Features	LR	LP	F1
ENG	+word cluster	89.3	90.0	89.7
	+lexical dependencies	89.7	90.3	90.0
	+dependency LM	90.0	90.6	90.3
CHN	+word cluster	85.7	87.5	86.6
	+lexical dependencies	87.2	88.6	87.9
	+dependency LM	87.2	88.7	88.0

Table 6: Experimental results on the English and Chinese development sets with different types of semi-supervised features added **incrementally** to the extended parser.

we employed the Stanford POS tagger. [4]

We took the WSJ articles from the TIPSTER corpus (LDC93T3A) as unlabeled English data. In addition, we removed from the unlabeled English data the sentences that appear in the WSJ corpus of the Penn Treebank. For unlabeled Chinese data, we used Chinese Gigaword (LDC2003T09), on which we conducted Chinese word segmentation by using a CRF-based segmenter. Table 4 summarizes data statistics on sentence and word numbers of the data sets listed above.

We used *EVALB* to evaluate parser performances, including labeled precision (LP), labeled recall (LR), and bracketing F1. [5] For significance tests, we employed the randomized permutation-based tool provided by Daniel Bikel. [6]

In both training and decoding, we set the beam size to 16, which achieves a good tradeoff between efficiency and accuracy. The optimal iteration number of perceptron learning is determined on the development sets. For word clustering, we set the cluster number to 50 for both the English and Chinese experiments.

5.2 Results on Development Sets

Table 5 reports the results of the extended parser (baseline + padding + supervised features) on the English and Chinese development sets. We integrated the padding method into the baseline parser, based on which we further incorporated the supervised features in Table 2. From the results we find that the padding method improves the parser accuracies by 0.5% and 0.4% on English and Chinese, respectively. Incorporating the supervised features in Table 2 gives further improvements of 0.2% on English and 0.1% on Chinese.

Based on the extended parser, we experimented different types of semi-supervised features by adding the features incrementally. The results are shown in Table 6. By comparing the results in Table 5 and the results in Table 6 we can see that the semi-supervised features achieve an overall improvement of 1.0% on the English data and an im-

[4] http://nlp.stanford.edu/software/tagger.shtml
[5] http://nlp.cs.nyu.edu/evalb
[6] http://www.cis.upenn.edu/~dbikel/software.html#comparator

Type	Parser	LR	LP	F1
SI	Ratnaparkhi (1997)	86.3	87.5	86.9
	Collins (1999)	88.1	88.3	88.2
	Charniak (2000)	89.5	89.9	89.5
	Sagae & Lavie (2005)*	86.1	86.0	86.0
	Sagae & Lavie (2006)*	87.8	88.1	87.9
	Baseline	**90.0**	**89.9**	**89.9**
	Petrov & Klein (2007)	90.1	90.2	90.1
	Baseline+Padding	**90.2**	**90.7**	**90.4**
	Carreras et al. (2008)	90.7	91.4	91.1
RE	Charniak & Johnson (2005)	91.2	91.8	91.5
	Huang (2008)	92.2	91.2	91.7
SE	Zhu et al. (2012)*	90.4	90.5	90.4
	Baseline+Padding+Semi	**91.1**	**91.5**	**91.3**
	Huang & Harper (2009)	91.1	91.6	91.3
	Huang et al. (2010)†	91.4	91.8	91.6
	McClosky et al. (2006)	92.1	92.5	92.3

Table 7: Comparison of our parsers and related work on the English test set. * Shift-reduce parsers. † The results of self-training with a single latent annotation grammar.

Type	Parser	LR	LP	F1
SI	Charniak (2000)*	79.6	82.1	80.8
	Bikel (2004)†	79.3	82.0	80.6
	Baseline	**82.1**	**83.1**	**82.6**
	Baseline+Padding	**82.1**	**84.3**	**83.2**
	Petrov & Klein (2007)	81.9	84.8	83.3
RE	Charniak & Johnson (2005)*	80.8	83.8	82.3
SE	Zhu et al. (2012)	80.6	81.9	81.2
	Baseline+Padding+Semi	**84.4**	**86.8**	**85.6**

Table 8: Comparison of our parsers and related work on the test set of CTB5.1.* Huang (2009) adapted the parsers to Chinese parsing on CTB5.1. † We run the parser on CTB5.1 to get the results.

provement of 1.5% on the Chinese data.

5.3 Final Results

Here we report the final results on the English and Chinese test sets. We compared the final results with a large body of related work. We grouped the parsers into three categories: single parsers (SI), discriminative reranking parsers (RE), and semi-supervised parsers (SE). Table 7 shows the comparative results on the English test set and Table 8 reports the comparison on the Chinese test set.

From the results we can see that our extended parser (baseline + padding + supervised features) outperforms the Berkeley parser by 0.3% on English, and is comparable with the Berkeley parser on Chinese (-0.1% less). Here *+padding* means the padding technique and the features in Table 2. After integrating semi-supervised features, the parsing accuracy on English is improved to 91.3%. We note that the performance is on the same level as the performance of self-trained parsers, except for McClosky et al. (2006), which is based on the combination of reranking and self-training. On Chinese, the final parsing accuracy is 85.6%. To our knowledge, this is by far the best reported performance on this data set.

The padding technique, supervised features, and semi-supervised features achieve an overall improvement of 1.4% over the baseline on English, which is significant on the level of $p < 10^{-5}$. The overall improvement on Chinese is 3.0%, which is also significant on the level of $p < 10^{-5}$.

Parser		**#Sent/Second**
Ratnaparkhi (1997)		Unk
Collins (1999)		3.5
Charniak (2000)		5.7
Sagae & Lavie (2005)*		3.7‡
Sagae & Lavie (2006)†		2.2‡
Petrov & Klein (2007)		6.2
Carreras et al. (2008)		Unk
This Paper	Baseline	100.7
	Baseline+Padding	89.5
	Baseline+Padding+Semi	46.8

Table 9: Comparison of running times on the English test set, where the time for loading models is excluded. * The results of SVM-based shift-reduce parsing with greedy search. † The results of MaxEnt-based shift-reduce parser with best-first search. ‡ Times reported by authors running on different hardware.

5.4 Comparison of Running Time

We also compared the running times of our parsers with the related single parsers. We ran timing tests on an Intel 2.3GHz processor with 8GB memory. The comparison is shown in Table 9. From the table, we can see that incorporating semi-supervised features decreases parsing speed, but the semi-supervised parser still has the advantage of efficiency over other parsers. Specifically, the semi-supervised parser is 7 times faster than the Berkeley parser. Note that Sagae & Lavie (2005) and Sagae & Lavie (2006) are also shift-reduce parsers, and their running times were evaluated on different hardwares. In practice, the running times of the shift-reduce parsers should be much shorter than the reported times in the table.

5.5 Error Analysis

We conducted error analysis for the three systems: the baseline parser, the extended parser with

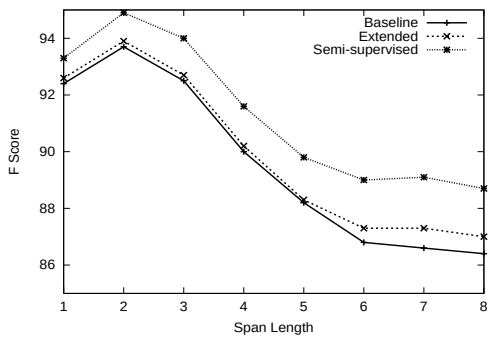

Figure 5: Comparison of parsing accuracies of the baseline, extended parser, and semi-supervised parsers on spans of different lengths.

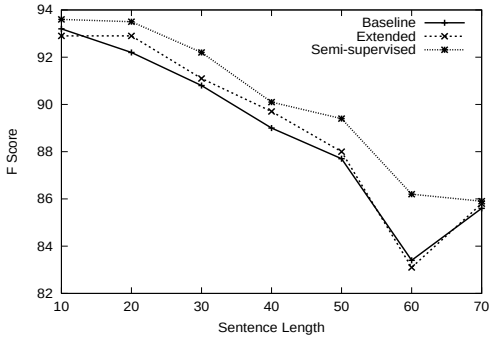

Figure 6: Comparison of parsing accuracies of the baseline, extended parser, and semi-supervised parser on sentences of different lengths.

the padding technique, and the semi-supervised parser, focusing on the English test set. The analysis was performed in four dimensions: parsing accuracies on different phrase types, on constituents of different span lengths, on different sentence lengths, and on sentences with different numbers of unknown words.

5.5.1 Different Phrase Types

Table 10 shows the parsing accuracies of the baseline, extended parser, and semi-supervised parser on different phrase types. Here we only consider the nine most frequent phrase types in the English test set. In the table, the phrase types are ordered from left to right in the descending order of their frequencies. We also show the improvements of the semi-supervised parser over the baseline parser (the last row in the table). As the results show, the extended parser achieves improvements on most of the phrase types with two exceptions: Preposition Prase (PP) and Quantifier Phrase (QP). Semi-supervised features further improve parsing accuracies over the extended parser (QP is an exception). From the last row, we can see that improvements of the semi-supervised parser over the baseline on VP, S, SBAR, ADVP, and ADJP are above the average improvement (1.4%).

5.5.2 Different Span Lengths

Figure 5 shows a comparison of the three parsers on spans of different lengths. Here we consider span lengths up to 8. As the results show, both the padding extension and semi-supervised features are more helpful on relatively large spans: the performance gaps between the three parsers are enlarged with increasing span lengths.

5.5.3 Different Sentence Lengths

Figure 6 shows a comparison of parsing accuracies of the three parsers on sentences of different lengths. Each number on the horizontal axis represents the sentences whose lengths are between the number and its previous number. For example, the number 30 refers to the sentences whose lengths are between 20 and 30. From the results we can see that semi-supervised features improve parsing accuracy on both short and long sentences. The points at 70 are exceptions. In fact, sentences with lengths between 60 and 70 have only 8 instances, and the statistics on such a small number of sentences are not reliable.

5.5.4 Different Numbers of Unknown Words

Figure 4 shows a comparison of parsing accuracies of the baseline, extended parser, and semi-supervised parser on sentences with different numbers of unknown words. As the results show, the padding method is not very helpful on sentences with large numbers of unknown words, while semi-supervised features help significantly on this aspect. This conforms to the intuition that semi-supervised methods reduce data sparseness and improve the performance on unknown words.

6 Conclusion

In this paper, we addressed the problem of different action-sequence lengths for shift-reduce phrase-structure parsing, and designed a set of novel non-local features to further improve parsing. The resulting supervised parser outperforms the Berkeley parser, a state-of-the-art chart parser, in both accuracies and speeds. In addition, we incorporated a set of semi-supervised features. The

System	NP	VP	S	PP	SBAR	ADVP	ADJP	WHNP	QP
Baseline	91.9	90.1	89.8	88.1	85.7	84.6	72.1	94.8	89.3
Extended	92.1	90.7	90.2	87.9	86.6	84.5	73.6	95.5	88.6
Semi-supervised	93.2	92.0	91.5	89.3	88.2	86.8	75.1	95.7	89.1
Improvements	+1.3	+1.9	+1.7	+1.2	+2.5	+2.2	+3.0	+0.9	-0.2

Table 10: Comparison of parsing accuracies of the baseline, extended parser, and semi-supervised parsers on different phrase types.

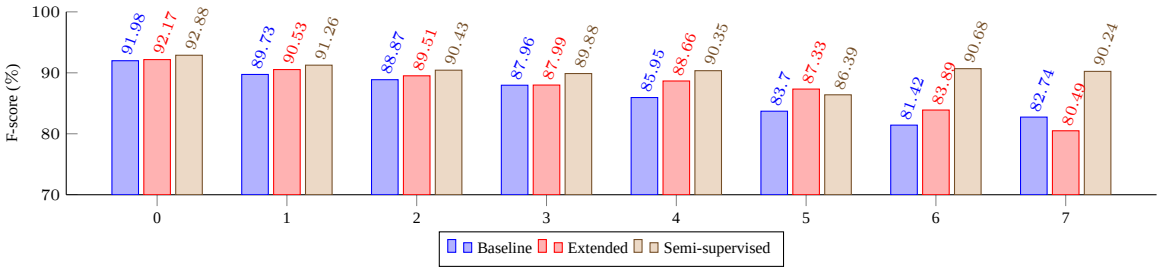

Figure 4: Comparison of parsing accuracies of the baseline, extended parser, and semi-supervised parser on sentences of different unknown words.

final parser reaches an accuracy of 91.3% on English and 85.6% on Chinese, by far the best reported accuracies on the CTB data.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. Yue Zhang and Muhua Zhu were supported partially by SRG-ISTD-2012-038 from Singapore University of Technology and Design. Muhua Zhu and Jingbo Zhu were funded in part by the National Science Foundation of China (61073140; 61272376), Specialized Research Fund for the Doctoral Program of Higher Education (20100042110031), and the Fundamental Research Funds for the Central Universities (N100204002). Wenliang Chen was funded partially by the National Science Foundation of China (61203314).

References

Daniel M. Bikel. 2004. *On the parameter space of generative lexicalized statistical parsing models.* Ph.D. thesis, University of Pennsylvania.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP*, pages 12–14, Jeju Island, Korea.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of CoNLL*, pages 9–16, Manchester, England.

Eugune Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, pages 173–180.

Eugune Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139, Seattle, Washington, USA.

Wenliang Chen, Junichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of EMNLP*, pages 570–579, Singapore.

Wenliang Chen, Min Zhang, and Haizhou Li. 2012. Utilizing dependency language models for graph-based dependency. In *Proceedings of ACL*, pages 213–222, Jeju, Republic of Korea.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, Stroudsburg, PA, USA.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL*, Madrid, Spain.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Michael Collins. 2000. Discriminative reranking for natural language processing. In *Proceedings of ICML*, pages 175–182, Stanford, CA, USA.

Hal Daume III. 2006. *Practical Structured Learning for Natural Language Processing*. Ph.D. thesis, USC.

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations

across languages. In *Proceedings of EMNLP*, pages 832–841, Singapore.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL*, pages 1077–1086, Uppsala, Sweden.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of EMNLP*, pages 12–22, Massachusetts, USA.

Liang Huang. 2008. Forest reranking: discriminative parsing with non-local features. In *Proceedings of ACL*, pages 586–594, Ohio, USA.

Liang-Ya Huang. 2009. Improve Chinese parsing with Max-Ent reranking parser. In *Master Project Report, Brown University*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, Massachusetts, USA, June.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewiz. 1993. Building a large annotated corpus of English. *Computational Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the HLT/NAACL, Main Conference*, pages 152–159, New York City, USA, June.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98, Ann Arbor, Michigan, June.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: a data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, pages 2216–2219.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT/NAACL*, pages 404–411, Rochester, New York, April.

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of EMNLP*, Rhode Island, USA.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of IWPT*, pages 125–132, Vancouver, Canada.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of HLT/NAACL, Companion Volume: Short Papers*, pages 129–132, New York, USA.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*, pages 577–585, Ohio, USA.

Weiwei Sun and Hans Uszkoreit. 2012. Capturing paradigmatic and syntagmatic lexical relations: towards accurate Chinese part-of-speech tagging. In *Proceedings of ACL*, Jeju, Republic of Korea.

Nianwen 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.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206, Nancy, France.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL/HLT*, pages 888–896, Columbus, Ohio.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese Treebank using a global discriminative model. In *Proceedings of IWPT*, Paris, France, October.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL*, pages 188–193, Portland, Oregon, USA.

Muhua Zhu, Jingbo Zhu, and Huizhen Wang. 2012. Exploiting lexical dependencies from large-scale data for better shift-reduce constituency parsing. In *Proceedings of COLING*, pages 3171–3186, Mumbai, India.