Computational Intelligence — An International Journal

Computational Intelligence, Volume 33, Number 3, 2017

IMPROVING SHIFT-REDUCE PHRASE-STRUCTURE PARSING WITH CONSTITUENT BOUNDARY INFORMATION

WENLIANG CHEN,[1,2] MUHUA ZHU,[3] MIN ZHANG,[1,2]
YUE ZHANG,[4] AND JINGBO ZHU[3]

[1]*School of Computer Science and Technology, Soochow University, Suzhou, China*
[2]*Collaborative Innovation Center of Novel Software Technology and Industrialization, Jiansu, China*
[3]*Natural Language Processing Lab, Northeastern University, Shenyang, China*
[4]*Singapore University of Technology and Design, Singapore*

Shift-reduce parsing enjoys the property of efficiency because of the use of efficient parsing algorithms like greedy/deterministic search and beam search. In addition, shift-reduce parsing is much simpler and easy to implement compared with other parsing algorithms. In this article, we explore constituent boundary information to improve the performance of shift-reduce phrase-structure parsing. In previous work, constituent boundary information has been used to speed up chart parsers successfully. However, whether it is useful for improving parsing accuracy has not been investigated. We propose two different models to capture constituent boundary information, based on which two sets of novel features are designed for a shift-reduce parser. The first model is a boundary prediction model that uses a classifier to predict the boundaries of constituents. We use automatically parsed data to train the classifier. The second one is a Tree Likelihood Model that measures the validity of a constituent by its likelihood which is calculated on automatically parsed data. Experimental results show that our proposed method outperforms a strong baseline by 0.8% and 1.6% in F-score on English and Chinese data, respectively, achieving the competitive parsing accuracies on Chinese (84.8%) and English (90.8%). To our knowledge, this is the first time for shift-reduce phrase-structure parsing to advance the state-of-the-art with constituent boundary information.

Received 3 April 2015; Revised 25 February 2016; Accepted 15 March 2016

Key words: natural language processing, phrase-structure parsing, semi-supervised learning, shift-reduce parsing..

1. INTRODUCTION

In the field of phrase-structure parsing (aka constituency parsing), transition-based shift-reduce parsing is attractive because of its efficiency and simplicity. Specifically, shift-reduce parsing generally uses decoding algorithms such as greedy/determinisic search and beam search which have time complexity $O(n)$ and $O(nk)$, respectively. In addition, compared with complex parsing algorithms, shift-reduce parsing is easy to implement. Pioneering models of shift-reduce parsing rely on local classifiers and greedy search (Sagae and Lavie 2005; Wang et al. 2006). To advance shift-reduce phrase-structure parsing, Sagae and Lavie (2006) employed a best-first search strategy to expand search space, and Zhang and Clark (2009) proposed a global learning algorithm to replace local classifiers. More recently, Zhu et al. (2012) enriched feature representations of shift-reduce parsing by exploiting lexical dependencies extracted from large-scale automatically parsed data.

Zhu et al. (2012) summarized the types and ratios of action conflicts that were made by a shift-reduce parser (Table 2)[1] and found that the major type of action conflicts was whether a partial parse tree should be a complete constituent (i.e., the shift/reduce conflicts, accounting for 65.9% of all errors). In this article, we propose to resolve this type of action conflicts

Address correspondence to W. Chen, Soochow University; e-mail: chenwenliang@gmail.com

[1] By *action conflict* we refer to the phenomenon that plausible actions compete in the same state of shift-reduce parsing.

by introducing constituent boundary information into shift-reduce parsing. In addition, constituent boundary information is also useful in resolving other types of action conflicts (see details in Section 2).

The idea of modeling constituent boundaries has been previously explored in (Roark and Hollingshead 2008; Bodenstab et al. 2011), where boundary information was used to help improve the efficiency of chart parsers (Charniak 2000; Collins 2003; Petrov and Klein 2007). But so far, the boundary information has not been optimized toward improving accuracy of a constituent parser. This article focuses on the problem of integrating boundary information into shift-reduce parsers and the goal is to improve parsing accuracy. To this end, we further propose different ways to model constituent boundary information.

We propose two models that are designed from distinct perspectives. The first one is a constituent boundary prediction model, which directly predicts whether a text span should be any constituent in the parse tree of a sentence. Intuitively, such a model can help to judge whether a single word can be a constituent or not (Table 10 for the error type of *single word phrase).* The second one, a tree likelihood model, models the constituent boundary with tree structure information, which has effect on disambiguating attachment problems (refer to the error types of *PP, NP, VP* attachment in Table 10. The tree likelihood model is similar to syntactic language model (Charniak et al. 2003). The novelty of the proposed model is that it incorporates boundary information. It models the likelihood of a constituent by its context. In addition, to address the problem of data sparseness, we utilize in both models additional large-scale automatically parsed data, which are generated by a baseline shift-reduce parser on unlabeled data.

Based on the two models that capture constituent boundary information, we design two sets of novel features specifically for shift-reduce parsing. We compare the effects of two sets of features, and also the effect of using them jointly. Experimental results on benchmark data show that using the two sets of new features together improves the baseline parser significantly, by 0.8% and 1.6% on English and Chinese data, respectively. The final parsing accuracy of our new parser is 90.8% on English and 84.8% on Chinese, respectively.

In summary, we make the following contributions in this article:

(1) We utilize constituent boundary information to advance shift-reduce parsing. To the best of our knowledge, there is no previous work of using constituent boundary information to improve parsing accuracy, especially for shift-reduce parsing.
(2) With the help of our proposed methods, our parser achieves the state-of-the-art performance on English and Chinese.

The rest of this article is organized as follows. Section 2 introduces the shift-reduce constituency parsing model and the action conflicts. Section 3 describes how to capture the constituent boundary information and defines a set of new features for the parsing model. Sections 4 and 5 describe the experiment settings and report the experimental results. Section 6 discusses related work. Finally, in Section 7, we summarize the main points and suggest several possible extensions of this research.

2. BACKGROUND

2.1. Shift-Reduce Parsing

The basic idea of shift-reduce parsing is to scan an input sentence (word-POS pairs) from left to right, and construct binary-branching parse trees by using a stack and a sequence of shift-reduce actions (Aho and Ullman 1972). Given an input sentence, a possible parse tree of the sentence can be translated into one transition sequence of states. Formally speaking, each state in the sequence is denoted by $\langle S, Q \rangle$, where S is a stack containing partial parses that have been recognized so far and Q is a queue of word-POS pairs that remain unprocessed. The initial state is $\langle \phi, w_1 \ldots w_n \rangle$, where S is empty and Q contains the entire input sentence. The final state is $\langle S, \phi \rangle$, where S contains a single parse tree with a predesignated root label and Q is empty. The shift-reduce parsing process is a transition process from the initial state to the final state by performing a sequence of the following actions.

(1) **shift**: move a pair of word and POS tag from the front end of the (non-empty) queue Q onto the stack S.

(2) **reduce-unary-X**: extend the top item on the stack S by applying a unary rule and then replacing the top item with the newly generated constituent. Here X represents a treebank phrase label, such as NP, which is to be used as the root label of the new constituent.

(3) **reduce-binary-{L/R}-X**: move the top two items off the stack S and push a new item onto the stack. The new item has X as its root label and consists of two children with the first popped item being the right child and the second popped item being the left child. The switch L/R indicates whether the left (L) or the right (R) child is the head child.

Given the input sentence "布朗(Brown) 访问(visits) 上海(Shanghai)", its syntactic structure can be constructed using the sequence of shift-reduce action shown in Figure 1. One implementation of the aforementioned shift-reduce parsing process is to employ a classifier to choose a shift-reduce action at each state transition (Sagae and Lavie 2005, 2006). In the decoding phase, either greedy or best-first search can be applied.

An alternative implementation is to replace local classifiers with a global online learning algorithm (Zhang and Clark 2011), where beam search is used for both training and decoding. In the training phase, the "early-update" strategy (Collins and Roark 2004) is used: whenever the gold partial parse is pruned from the beam, parameters are updated immediately. We adopt this method because of its relatively high accuracy. More details are given in Section 2.2. Binarization of training data and debinarization of parsing output are required if the training sentences are not binary-branching.

2.2. Baseline Parser

Our baseline parser is an extension of the beam-search shift-reduce parser proposed in Zhang and Clark (2009).[2] To score an action A with respect to a state $Y = \langle S, Q \rangle$, we choose to use a linear model, defined as follows:

$$\begin{aligned} Score(\langle A, Y \rangle) &= \overrightarrow{w} \cdot \Phi(\langle A, Y \rangle) \\ &= \sum_i \lambda_i f_i(\langle A, Y \rangle) \end{aligned}$$

where $f_i(\langle A, Y \rangle)$ are features extracted jointly from the action A and the state Y and λ_i are the parameters.

[2] http://faculty.sutd.edu.sg/~yue_zhang/doc/.

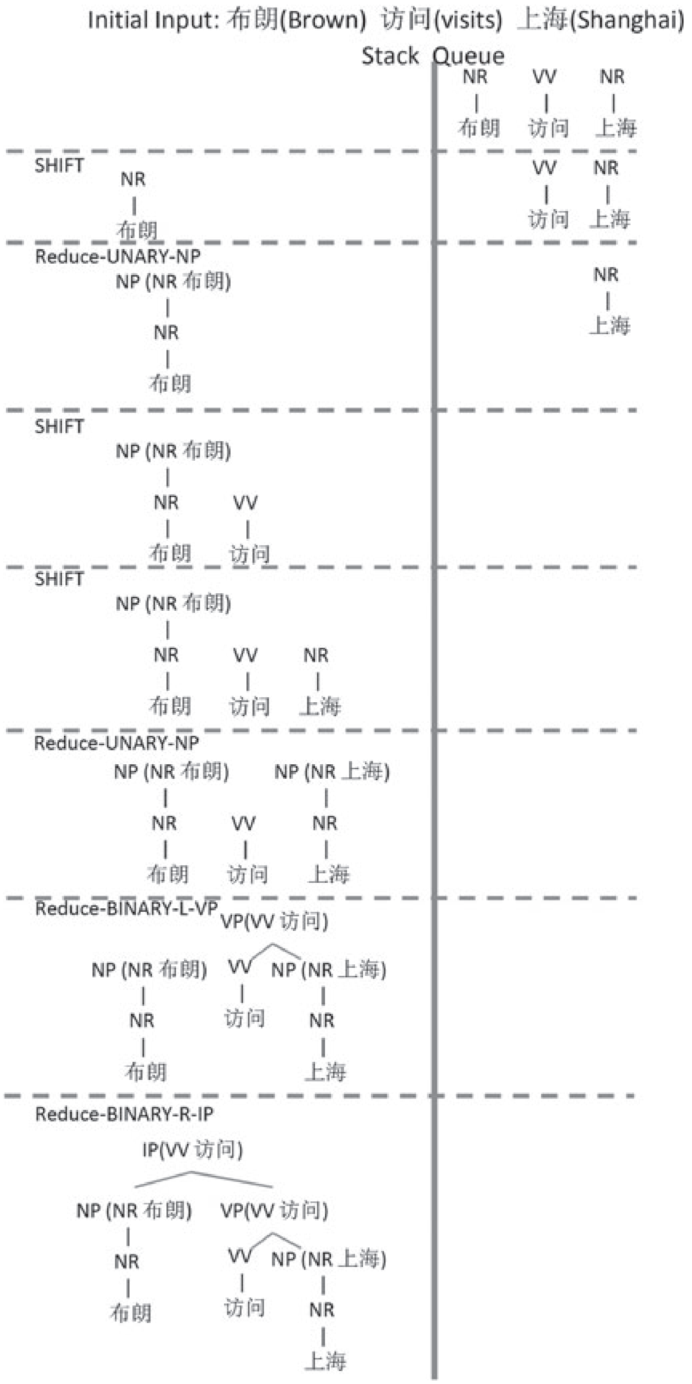

FIGURE 1. An example of shift-reduce parsing process.

Beam-search decoding is applied to find the globally highest-scored sequence of actions. A beam with size K is used to maintain the highest scored K states at each step. At initialization, the beam contains only the initial state. At each step, each state in the beam is extended by applying all possible actions, resulting in a set of new states. Each new state is rescored by considering the latest action, and the K highest scored states are used to replace the content of the beam, before the next step starts. The same process repeats until the highest-scored state in the beam is a final state, which is then used as the output structure.

Online learning based on the decoding process is used to train the parameter vector $\overrightarrow{w} =< \lambda_1, ..., \lambda_i, ..., \lambda_{N_w} >$, where N_w is the number of parameters. In particular, $\overrightarrow{w}$ is initialized to all zeroes and used to decode the training sentences. For each sentence, if the decoder gives the correct answer, training continues to the next sentence without changing the model. Otherwise the model is updated when the first search error is made. Here a search error happens when the gold-standard sequence of actions is pruned off the beam. A search error can happen at an intermediate step. In this case, the gold-standard output cannot be constructed, because there is a one-to-one correspondence between the action sequence and the output structure. As a result, search is stopped with the model parameters updated by using the gold-standard sequence of actions till this step as the positive example, and the highest-scored sequence of actions in the beam at this step as the negative example.

To learn the parameters λ_i, we employ the passive-aggressive algorithm (Crammer et al. 2006) rather than the averaged perceptron (Collins 2002), which is used in Zhang and Clark (2011). In our preliminary experiments, the passive-aggressive algorithm outperforms the averaged perceptron algorithm. Pseudo code of the search and learning algorithm is shown in Algorithm 1, where INITIALSTATE() sets the initial state, POSSIBLEACTIONS() returns all possible actions for a state, EXPAND() applies an action to a state and derives a new state, TOP() returns the highest-scored state in the agenda (agenda is generally implemented as a priority queue which sorts states according to their scores.), and CORRECT() checks whether a state is the gold-standard state.

We adopt the feature set of Zhang and Clark (2009) but remove the Chinese specific features. The baseline features used in this article are summarized in Table 1. The symbol w means the lexical head of an item; the symbol c denotes the constituent label; the symbol t is the POS of a lexical head.

2.3. Shift-Reduce Action Conflicts

The research of this article is directly inspired by the error analysis conducted in Zhu et al. (2012), which investigates first mistakes made by a shift-reduce parser on the English

TABLE 1. A Summary of Baseline Features, where S_i Denotes the i^{th} Item in the Stack S and Q_i Denotes the i^{th} Item in the Queue Q from the Front end.

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	$S_0cS_1cS_2c$, $S_0wS_1cS_2c$, $S_0cS_1wQ_0t$
	$S_0cS_1cS_2w$, $S_0cS_1cQ_0t$, $S_0wS_1cQ_0t$
	$S_0cS_1wQ_0t$, $S_0cS_1cQ_0w$

Regarding tree nodes that have two children, l and r refer to the left and right child respectively. u refers to the child of tree nodes that have a single child.

Algorithm 1 The search and learning algorithm

```
function SearchAndTrain(Sentence, isTraining)
candidates ← {INITIALSTATE()}
agenda ← []
step ← 0
while true:
  for each candidate in candidates:
    for each action in POSSIBLEACTIONS(candidate):
      agenda ← Insert(EXPAND(candidate, action), agenda)
  if isTraining and not ANY CORRECT(candidates):
    positive ← GOLDATSTEP(step)
    negative ← TOP(agenda)
    PASSIVEAGGRESSIVE(positive, negative)
    return NULL
  best ← TOP(agenda)
  if IsFinalState(best); break
  candidates ← TOP_K(agenda, K)
  step ← step+1
if isTraining and not CORRECT(best)
  positive ← GOLDATSTEP(step)
  negative ← TOP(agenda)
  PASSIVEAGGRESSIVE(positive, negative)
  return NULL
return best
```

development set. By *first mistake*, we mean the position that a shift-reduce parser choose a wrong action for the first time during the parsing process. We are concerned with first mistakes because future mistakes are often caused by previous ones. Table 2 reports the analysis, which lists the types and ratios of first mistakes. From the analysis, we can see that action conflicts between shift and reduce-binary are the major cause of parsing errors, which cover nearly half (47.8%) of first mistakes. This type of first mistakes is attributed to the ambiguity about whether the combination of the top two stack items S_1 and S_0 is a constituent. Therefore, incorporating constituent boundary information is helpful to resolving such ambiguity. Specifically, if the concatenation of the spans of S_1 and S_0 is predicted to be a constituent, we will prefer reduce-binary actions to the shift action. Similarly, such information is also useful in resolving the conflicts between reduce-binary-L/R-X and reduce-binary-L/R-X*, that is, incorrect choices between a treebank label X and its corresponding temporary label X*, which is introduced during the tree binarization process. If the concatenation of S_1 and S_0 is a constituent, the reduce-binary-L/R-X action is preferred.

Table 2. The Types and Ratios of First Mistakes Made by a Shift-Reduce Parser on the English Development Set.

ID	Mistake type	Ratio % (count)
1	Shift versus reduce-binary	47.8 (451)
2	Shift versus reduce-unary	18.1 (171)
3	Reduce-binary versus reduce-unary	5.4 (92)
4	Reduce-binary-L/R-{X versus X*}	16.5 (156)
5	Reduce-unary-{X_1 versus X_2}	5.7 (54)

3. CAPTURING CONSTITUENT BOUNDARY INFORMATION

Here, we describe two models that capture constituent boundary information, based on which constituent boundary features are designed. In the following sections, we first introduce the approach to obtaining automatically parsed data, then describe in detail the two models. Finally, we list the set of new features that we propose.

3.1. Auto-Parsed Data Preparation

Both the boundary prediction model and tree likelihood model utilize information extracted from large-scale automatically parsed data. We run the baseline parser on the unlabeled data to obtain a set of automatically parsed trees. Because the shift-reduce parser used in this article requires POS tags as input, POS tagging should be applied to the unlabeled data before performing syntactic parsing.

3.2. Boundary Prediction Model

Given a sentence w_1^n, the constituent boundary prediction model seeks a function $f(w_i^j) \in \{+1, -1\}$ that tells whether a text span w_i^j is a valid constituent (omitting treebank grammar labels). We formalize such a boundary prediction task as a binary classification problem. Training data for building the model is collected from human-labeled parse trees that are used for training syntactic parsers. Formally, given a sentence w_1^n, we denote the set of all the spans of the sentence by $I(w_1^n)$. For each $w_i^j \in I(w_1^n)$, we use it as a positive instance if the w_i^j matches a constituent in the corresponding gold parse tree, and a negative instance otherwise. Based on the training data, a maximum entropy classifier is used to learn model parameters.[3] After building the classification model, we use it in both the training and decoding phases of shift-reduce parsing. Note that the integration of the classification model does not increase time complexity of the parsing model, because the classification model only processes candidates generated in the shift-reduce parsing, instead of $I(w_1^n)$.

Table 3 lists the feature templates used in the boundary prediction model, which can be separated into two subsets. Supervised features are extracted solely from human-labeled training data, while semi-supervised features are defined on automatically parsed data. To get semi-supervised features, we first parse unlabeled data with our baseline parser, and then extract two lists from auto-parsed data, based on which semi-supervised features are designed. To the best of our knowledge, previous work on boundary prediction (Roark and Hollingshead 2009; Bodenstab et al. 2011) does not consider semi-supervised features.

Supervised Features for Building BPM
Our supervised features for BPM are similar to the features used in (Bodenstab et al. 2011). The supervised features are defined on the surface of the input span w_i^j and its surrounding contexts. These features are generated from manually labeled data and used in the baseline model. The symbol w refers to word tokens and the symbol t represents POS tags. Subscripts of the symbols denote indices of word tokens in the whole input sentence w_1^n. Besides word and POS tags, we also have three feature functions that encode the length information of w_i^j. The first function (F1) returns the length $L(w_i^j)$ of the span if the length is ≤ 5. The other two functions (F2 and F3) are BIN functions to divide numeric values into a set of disjoint intervals called bins. The subscripts of the BIN functions (10 and 0.2) are bin sizes. Note that there is a BIN for each multiple of bin size.

[3] http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html.

TABLE 3. Feature templates of the constituent boundary prediction model.

Supervised Feature Templates	
Unigrams	$w_i, w_{i-1}, w_j, w_{j+1}, t_i, t_{i-1}$
	$t_{i-2}, t_{i+1}, t_{t_i+2}, t_j, t_{j+1}, t_{j+2}$
Bigrams	$t_{i-1}t_i, t_{j+1}t_{j+2}$
Length	F1: $L(w_i^j)$ if $L(w_i^j) \leq 5$
	F2: $BIN_{10}(L(w_i^j))$ if $L(w_i^j) \geq 10$
	F3: $BIN_{0.2}(L(w_i^j)/L(w_1^n))$
Semi-supervised Feature Templates	
$List_s(w_{i-1}w_i), List_s(w_j w_{j+1}), List_f(w_i w_j)$	

Here, w denotes word tokens, t denotes POS tags, $L(w_i^j)$ denotes the length of the span, and $L(w_1^n)$ denotes the length of the whole input sentence.

Semi-Supervised Features for Building BPM

Semi-supervised features rely on the use of two lists of records extracted from automatically parsed data and are used in the semi-supervised models. One list is called the **frontier-word-pair** list, which contains pairs that consist of the left-most and right-most frontier word of a constituent. For example, the words *The* and *market* are frontier words of the NP constituent in the parse tree of Figure 2. The other list is called the **separate-word-pair** list, which contains two consecutive words that belong to different constituents. For example, the words *market* and *does* in the example parse tree is a separate-word-pair. For items in the two lists, we assign tags according to their frequency counts, following the strategy used in Chen et al. (2009) and Chen et al. (2012a). Specifically, the 10% most frequent items are assigned to the category of *high frequency*; or if an item is among top 20%, we assign it to the category of *middle frequency*; others are assigned to the category of *low frequency*. With the two lists, we design semi-supervised features listed in Table 3, which return a category tag if a given word pair is found in the specified list (the subscript f = frontier-word-pair; s = separate-word-pair). To the best of our knowledge, these is no previous work that adopts such semi-supervised features for the task of boundary prediction.

3.3. Tree Likelihood Model

The constituent boundary prediction model relies on surface information (words and POS tags) to classify spans. The model has the advantage of simplicity, but it ignores underlying syntactic structures of spans. In contrast, the tree likelihood model measures the validity of a constituent (partial parse tree) by its likelihood, calculated over large automatically parsed data.

The model first identifies the head of each node in a parse tree by using head-finding rules. In the example of Figure 2, heads are marked by framed boxes. Sisters of a head are referred to as modifiers of the head. Second, each nonterminal label is extended with its head word and head POS which are obtained through propagation from the bottom up to the tree root. In the example, head words are presented in parentheses. This two-step process is known as lexicalization (Collins 2003). Hereafter, we write a nonterminal as $NT = X(x)$, where $x = \langle w_h, t_h \rangle$ and X is a grammar label. Thus a lexicalized context-free grammar rule can be written as

$$P(h) \rightarrow L_n \ldots L_1 H(h) R_1 \ldots R_m$$

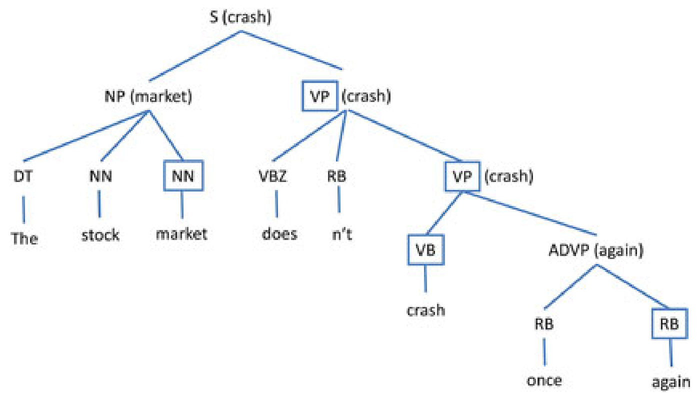

FIGURE 2. An example parse tree with heads being enclosed with boxes. [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 4. Templates of Records We Extract.

(1) $\langle L_i, H(h), Left\rangle$
(2) $\langle H(h), R_i, Right\rangle$
(3) $\langle L_i, L_{i-1}, H(h), Left\rangle$
(4) $\langle H(h), R_i, R_{i-1}, Right\rangle$

where H is the head; $L_1 \ldots L_n$ and $R_1 \ldots R_m$ are modifiers of the head. Either m or n may be zero.

After lexicalization, we can acquire a phrase structure $S = (L_n \ldots L_1 H(h) R_1 \ldots R_m)$ from each lexicalized grammar rule whose likelihood probability is defined as follows:

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

where $P_L(S)$ can be defined as

$$\begin{aligned} P_L(S) \approx\ & P(L_1|H(h)) \\ & \times P(L_2|L_1, H(h) \\ & \times \ldots \\ & \times P(L_n|L_{n-1}, \ldots, L_{n-k+1}, H(h)) \end{aligned}$$

Here, we make a k^{th} order Markov assumption. $P_R(S)$ can be defined similarly.

We incorporate tree likelihood information into a shift-reduce parser using a similar method as Chen et al. (2012b), which incorporates a dependency language model into a graph-based dependency parser. Specifically, we decompose phrase structures into components by setting k to different values. Here, we consider two cases: $k = 1$ (bigram) and $k = 2$ (trigram), but ignore higher-order dependency structures because of data sparseness. Table 4 shows the templates of records that we extract.

Here, *Left/Right* indicate the location of modifiers relative to the head. The first two templates denote bigram records, while the latter two templates denote trigram records. Moreover, we group records into one of the categories {Outer, Inner} depending on whether L_i (R_i) is the leftmost (rightmost) modifier; note that this is where we encode boundary information, and such boundary information is what make difference between Tree Likelihood Model and dependency language models (Chen et al. 2012b). We finally obtain four separate lists: Bigram-Outer (BO), Bigram-Inner (BI), Trigram-Outer (TO), and Trigram-Inner (TI). For example, $\langle L_n, H(h), \mathit{Left}\rangle$ and $\langle H(h), R_m, \mathit{Right}\rangle$ are contained in the BO list, while $\langle L_{i:i\neq n}, H(h), \mathit{Left}\rangle$ and $\langle H(h), R_{i:i\neq m}, \mathit{Right}\rangle$ are contained in the BI list. In each record list, we map records into discrete categories according to their probabilities $P(L_i|HIS)$ or $P(R_i|HIS)$, where *HIS* refers to the head and zero or one immediate previous sister. The strategy used here is similar to the one used previously in Section 3.2, with the difference that we use probabilities instead of frequencies: a record is assigned to *high probability (HP)* if it is among the 10% most probable records; or if it is among the top-20%, we use the category *middle probability*; otherwise we use the category of *low probability*.

3.4. Constituent Boundary Features

Based on the BPM and TLM model described in the aforementioned text, we design two sets of novel features for syntactic parsing, as listed in Table 5. These features are used together with the baseline features in the semi-supervised models.

Boundary Prediction Model Features

The boundary prediciton model (BPM) features are defined on the base of the output of BPM. The symbol $S_i b$ is an indicator function which returns 1 if S_i is predicted to be a constituent by the BPM, and 0 otherwise. Remember that S_i refers to the i^{th} stack item in the stack of beam-search process. $S_1 \circ S_0$ represents a concatenation of the spans of S_1 and S_0. We also consider the combination of $S_i b$ with the head tag t or constituent label c of S_i.

Tree Likelihood Model Features

The TLM features are defined on the four lists extracted by the tree likelihood model. Nonterminals in the extracted lists consist of grammar labels, head words, and head POS, but in this article, we use only head words. Thus, the extracted lists are simplified to encode relationship between words. The basic idea of TLM features is to check whether a word is at the boundary. Therefore, we check all the possible combinations. In Table 5, $B_{l/r}(\cdot,\cdot)$ means the use of a bigram list, either the Bigram-outer list or the Bigram-inner list, and the subscript (l or r) refers to the direction of a dependency relation. For example, $B_l(S_1 w, S_0 w)$ is instantiated into feature functions $BO_l(S_1 w, S_0 w)$ and $BI_l(S_1 w, S_0 w)$, which check whether $\langle S_1 w, S_0 w, \mathit{Left}\rangle$ can be found in BO and BI. $T_{l/r}(\cdot,\cdot,\cdot)$ can be explained in a similar way. $S_i ldw$ ($S_i rdw$) means the left (right) most dependent word of S_i.

Table 5. New Features Based on the Boundary Prediction Model and the Tree Likelihood Model.

	Feature templates
BPM	$S_0 b$, $S_0 b S_0 t$, $S_0 b S_0 c$
	$S_1 b$, $S_1 b S_1 t$, $S_1 b S_1 c$, $(S_1 \circ S_0)b$
TLM	$B_l(S_1 w, S_0 w)$, $B_r(S_1 w, S_0 w)$, $B_l(S_1 w, Q_0 w)$
	$B_l(S_1 w, S_0 w)S_1 t S_0 t$, $B_r(S_1 w, S_0 w)S_1 t S_0 t$
	$B_l(S_1 w, Q_0 w)S_1 t Q_0 t$, $B_r(S_1 w, Q_0 w)S_1 t Q_0 t$
	$B_l(S_0 w, Q_0 w)S_0 t Q_0 t$, $B_r(S_0 w, Q_0 w)S_0 t Q_0 t$
	$B_l(S_0 w, Q_0 w)$, $B_r(S_0 w, Q_0 w)$, $B_r(S_1 w, Q_0 w)$
	$T_r(S_1 w, S_0 ldw, S_0 w)$, $T_r(S_1 w, S_0 ldw, S_0 w)S_0 t S_1 t$
	$T_l(S_1 w, S_1 rdw, S_0 w)$, $T_l(S_1 w, S_1 rdw, S_0 w)S_0 t S_1 t$

BPM, boundary prediction model; TLM, tree likelihood model.

3.5. Parsing with the Proposed Features

To use the proposed new features, we update the scoring function defined in Section 2. The new scoring function is shown as follows:

$$Score'(\langle A, Y\rangle) = \sum\nolimits_i \lambda_i f_i(\langle A, Y\rangle) + \sum\nolimits_j \lambda_j^P f_j^P + \sum\nolimits_k \lambda_k^L f_k^L$$

where $f_i(\langle A, Y\rangle)$ are the baseline features listed in Table 1, and f_j^P and f_k^L denote the BPM and TLM features in Table 5, respectively. Here, indices j and k go through all the features in Table 5, so the size of j is 7 and the size of k is 16. We can see that incorporating boundary information increases the cost of extracting parsing features, but does not change the time complexity of the parsing model.

4. EXPERIMENTAL SETUP

4.1. Data Preparation

For the English experiments, the *Wall Street Journal* (WSJ) corpus of the Penn Treebank (Marcus et al. 1993) was used as labeled data. We used the standard divisions as Charniak and Johnson (2005), that is, trained on sections 2–21 and evaluated on section 23. Section 24 was used for system development. With regard to English unlabeled data, we used WSJ articles from the TIPSTER corpus (LDC93T3A).

For the Chinese experiments, we used the Chinese Treebank (CTB) version 5.1 (Xue et al. 2005) as labeled data. Specifically, articles 001–270 and 440–1151 were used for training, articles 271–300 for evaluation, and articles 301–325 were development data. For Chinese unlabeled data, we utilized the Chinese Gigaword corpus (LDC2003T09) with some basic cleanups.

We conducted necessary preprocessing on English and Chinese unlabeled data before feeding the data to the baseline parser. Specifically, we removed from the English unlabeled data all the sentences of the WSJ corpus and then applied OpenNLP for English sentence boundary detection and tokenization.[4] For English POS tagging, we adopted SVMTool, which has a per-token accuracy of 97.1% on section 23 of the WSJ corpus.[5] For the Chinese unlabeled data, we conducted sentence boundary detection simply according to sentence ending punctuations (i.e., 。, ！, and ？). Raw sentences were automatically segmented with a CRF-based word segmenter, which achieves an accuracy of 97.2% on the CTB5.1 testing data. For automatic Chinese POS tagging, we utilized the Stanford POS tagger.[6] We trained the tagger on the CTB5.1 training data which has the accuracy of 95.4% on the CTB5.1 test data. Table 6 contains detailed data statistics of all the data sets.

4.2. Parameter Configurations

In all the experiments, we follow the previous work of Zhang and Clark (2009) to set the beam size of our parser to 16 in both training and decoding phases. The optimal iteration number of passive-aggressive training was obtained on the development sets. In addition,

[4] http://incubator.apache.org/opennlp/.

[5] http://www.lsi.upc.edu/~nlp/SVMTool/.

[6] http://nlp.stanford.edu/software/tagger.shtml.

TABLE 6. Data Statistics Including the Numbers of Sentences and Words (in Parentheses).

Lang.	Train	Dev	Test	Unlabeled
English	39.8k (950.0k)	1.7k (40.1k)	2.4k (56.7k)	3,139.1k (76,041.4k)
Chinese	18.1k (493.8k)	350 (8.0k)	348 (6.8k)	9,870.2k (282,361k)

we used ten-way jack-knifing (Collins and Koo 2005) to assign POS tags to labeled (English and Chinese) training data. Maximum Entropy (Berger et al. 1996) was used as the classifier in the boundary prediction model.

4.3. Evaluation Metrics

For performance evaluation, we employed *EVALB* to provide bracket scoring.[7] For significance tests, we adopted the comparator developed by Daniel Bikel to compute p-values.[8]

5. EXPERIMENTAL RESULTS

5.1. Effect of Constituent Boundary Features

We examined the effect of our proposed new features on the English and Chinese test sets, respectively. The results are reported in Table 7, where we compared three different settings: (1) adding the BPM features alone to the baseline; (2) incorporating the TLM features only into the baseline; (3) including all the new features (BPM and TLM features).

As the results show, the BPM features and the TLM features improve parsing accuracy on both English and Chinese data when used separately. Specifically, on the English data, the BPM features achieve an absolute improvement of 0.4% F-score, while the improvement achieved by adding the TLM features is 0.6% F-score. On the Chinese data, the BPM features and the TLM features achieve improvements of 0.4% F-score and 1.3% F-score, respectively. We can see that the tree likelihood model performs better than the boundary prediction model, especially on the Chinese data. One reason is that the tree likelihood model provides dependency information in addition to boundary information. Note that the BPM classification models used in the aforementioned experiments were built with both supervised and semi-supervised features (all the features in Table 3). It is informative to show the necessity of using semi-supervised features. To this end, we built a BPM classification model with supervised features only. Parsing features designed on the base of such a BPM model achieved a limited improvement of 0.15% over the baseline on the English test set, whereas Base+PBM in Table 7 achieved an improvement of 0.4%.

Using all the new features (BPM and TLM features) together further improves performance: an overall improvement of 0.8% on English and 1.6% on Chinese over the baseline. This suggests that the BPM and TLM features provide complementary information. Significance tests show that the improvements on English and Chinese are significant on the level of $p < 10^{-5}$ and $p < 10^{-4}$, respectively.

[7] http://nlp.cs.nyu.edu/evalb.

[8] http://www.cis.upenn.edu/∼dbikel/download/compare.pl.

TABLE 7. Main Results on the English and Chinese Test Sets, with the BPM and/or TLM Features Added to the Baseline Parser.

Lang.	**System**	**LR**	**LP**	**F1**
English	Baseline	89.7	90.3	90.0
	Base+BPM	90.0	90.7	90.4
	Base+TLM	90.3	90.8	90.6
	Base+BPM+TLM	90.6	91.0	90.8
Chinese	Baseline	81.9	84.4	83.2
	Base+BPM	82.2	84.9	83.6
	Base+TLM	83.3	85.6	84.5
	Base+BPM+TLM	83.7	86.1	84.8

5.2. Performance Comparison

The aforementioned section reports the performance of our methods against the baseline, showing the effectiveness of our methods. In this section, we further compared our work with a large body of related work empirically. We grouped all the parsers into thee categories: single parsers (SINGLE)[9], reranking parsers (RE), and semi-supervised parsers (SEMI). Table 8 and Table 9 report the comparison on the English and Chinese test sets, respectively.

From the results, we see that our baseline parser is at the state-of-the-art performance level on both English and Chinese: very close to Petrov and Klein (2007), which is generally regarded as a representative state-of-the-art parser. On the English test set, the accuracy of our parser is 90.8%, which is significantly higher than the Berkeley parser (Petrov and Klein 2007). In addition, the accuracy of our parser is comparable with the performance of self-trained parsers. In the future, we will combine techniques such as reranking to further improve accuracies. On the Chinese test set, the accuracy of our parser reaches 84.8%. This is one of the best results on the Chinese data (CTB5.1). Zhu et al. (2013) give higher accuracy. However, their semi-supervised information is different from ours, which can be incorporated into our parser. Wang et al. (2015) is another parser that outperforms our one, but note that their parser builds on a joint model of POS tagging and syntactic parsing.

5.3. Analysis

We analyzed the parsing results on the English test set, with two perspectives: (1) we examine how the distribution of first mistakes made by the baseline parser is changed with our parser (first mistakes are critical due to error propagation) and (2) we classify parsing errors into several specific categories and compare the occurrences of errors in the baseline and our parser.

5.3.1. First-Mistake Reduction. The baseline parser made 1,534 first mistakes on the English test set. Changes of first mistakes made by our parser fall into four categories. In the following, the numbers in parentheses are instance counts in the corresponding categories. *No-change* (1,081) refers to the category that the baseline and our parser made the

[9] Our baseline is an extension of the parser from Zhang and Clark (2009) by replacing perceptron with passive-aggressive algorithm. Zhu et al. (2013) (SINGLE) is also an extension from Zhang and Clark (2009) by introducing the IDLE action and padding method.

TABLE 8. Comparative results of our parser and related work on the English test set.

Type	Parser	LR	LP	F1
SINGLE	Ratnaparkhi (1997)	86.3	87.5	86.9
	Collins (2003)	88.1	88.3	88.2
	Charniak (2000)	89.5	89.9	89.5
	Sagae and Lavie (2005)*	86.1	86.0	86.0
	Sagae and Lavie (2006)*	87.8	88.1	87.9
	Wang et al. (2015)*	–	–	89.4
	Our Baseline*	89.7	90.3	90.0
	Petrov and Klein (2007)	90.1	90.2	90.1
	Roark and Hollingshead (2009)	89.9	90.4	90.2
	Zhu et al. (2013)*	90.2	90.7	90.4
	Watanabe and Sumita (2015)	–	–	90.7
	Carreras et al. (2008)	90.7	91.4	91.1
	Shindo et al. (2012)	–	–	91.1
RE	Charniak and Johnson (2005)	91.2	91.8	91.5
	Huang (2008)	92.2	91.2	91.7
SEMI	Zhu et al. (2012)*	90.4	90.5	90.4
	Wang et al. (2015)*	–	–	90.7
	Our Improved Parser*	90.6	91.0	90.8
	Huang and Harper (2009)	91.1	91.6	91.3
	Zhu et al. (2013)*	91.1	91.5	91.3
	Huang et al. (2010)†	91.4	91.8	91.6
	McClosky et al. (2006)	92.1	92.5	92.3

* Shift-reduce parsers. † The results of self-training with a single latent annotation grammar.

same first mistake at the same position of action sequences. *Positive* (282) means our parser either postponed the positions of first mistakes or parses correctly on the sentences that were parsed incorrectly by the baseline parser. *Negative* means that our parser made first mistakes earlier than the baseline (119) or made first mistakes on the sentences that were correctly parsed by the baseline (83). Finally, *Others* (52) refers to the case that our parser and the baseline parser make different first mistakes at the same position. We are especially interested in the details of the *Positive* category. We find that 77.3% (218/282) are related to reduce-binary actions (shift vs. red-binary or red-binary vs. red-binary). In addition, 53.1% reduce-binary-related first mistakes made by the baseline parser were postponed or resolved by incorporating boundary information. This suggests that boundary information is useful in differentiating these actions.

5.3.2. Error Breakdown. We are also interested in breaking down parsing errors into specific types. To this end, we utilized the Berkeley error analyzer (Kummerfeld et al. 2012) to classify parsing errors into 12 types (see the error types in Table 10).[10] Refer to Kummerfeld et al. (2012) for detailed descriptions of the error types. Table 10 shows the occurrence

[10] http://code.google.com/p/berkeley-parser-analyser/.

TABLE 9. Comparison with Related Work on the CTB5.1 Test Set.

Type	Parser	LR	LP	F1
SINGLE	Charniak (2000)[‡]	79.6	82.1	80.8
	Bikel (2004)[†]	79.3	82.0	80.6
	Zhu et al. (2013)*	81.9	84.3	83.2
	Wang et al. (2015)*	–	–	83.2
	Our Baseline*	81.9	84.4	83.2
	Petrov and Klein (2007)	81.9	84.8	83.3
	Watanabe and Sumita (2015)	–	–	84.3
	Roark and Hollingshead (2009)	83.4	86.1	84.7
RE	Charniak and Johnson (2005)[‡]	80.8	83.8	82.3
SEMI	Zhu et al. (2012)*	80.6	81.9	81.2
	Our Improved Parser*	83.7	86.1	84.8
	Zhu et al. (2013)*	84.4	86.8	85.6
	Wang et al. (2015)*	–	–	86.6

* Shift-reduce parsers. [‡] Huang (2009) adapted the parsers to CTB5.1.
[†] We run the parser on CTB5.1 to get the results.

counts of parsing errors in the baseline and the improved parsers: Base+BPM, Base+TLM, and Base+BPM+TLM. The table also reports absolute and relative error reductions of the improved parsers over the baseline. By comparing the baseline and Base+BPM+TLM, we can see that boundary information is beneficial to resolving the following attachment errors: PP attachment, clause attachment, and VP attachment. These are the three types that have the biggest relative error reduction. In addition, boundary information is also helpful in recognizing structures of noun phrase (NP Internal Structure) and single word phrases. We notice that the attachment problem and single word phrase problem are closely related with recognition of phrases. This is constituent with our intuition that boundary information can help. Moreover, we can see BPM-based features and TLM-based features have different bias at reducing different types of parsing errors. For example, BPM-based features perform better at reducing *single word phrase* and *clause attach* errors, whereas TLM-based features perform better regarding the types of *PP attach* and *NP attachment*.

5.3.3. Effect of Unlabeled Data Size. Our method is semi-supervised parsing in essence, so it is interesting to show the effect of varying sizes of unlabeled data on parsing accuracy. For this purpose, we randomly sampled four different subsets from Chinese unlabeled data by adopting the percentage of 0.1%, 1%, 10%, and 100%, respectively. Then four parsers were built on the base of the sampled unlabeled data and evaluated on the Chinese test set. The results are shown in Figure 3. From the results, we can see that the improvement levels out when more than 1% unlabeled data is used.

6. RELATED WORK

Shift-reduce parsing has been widely studied because of its efficiency. Sagae and Lavie (2005) proposed a classifier-based approach to shift-reduce parsing, which was extended by Sagae and Lavie (2006) to expand the search space. Zhang and Clark (2009) proposed a global learning algorithm to replace local classifiers. All these previous works belong to the spectrum of supervised parsing. Zhu et al. (2012) and Zhu et al. (2013) for the first

TABLE 10. Breakdown of Parsing Errors of the Baseline and Our Parser on the English Test Set.

Error Type	Baseline	Base+ +BPM	Error Reduction(%)	Base+ +TLM	Error Reduction (%)	Final Parser (%)	Error Reduction (%)
PP Attachment	778	767	12 (1.5)	721	57 (7.3)	710	68 (8.7)
Single word phrase	615	584	31 (5.0)	602	13 (2.1)	575	40 (6.5)
Unary	497	509	−12 (−2.4)	471	16 (3.2)	497	0 (0)
Different label	366	366	0 (0)	364	2 (0.5)	355	11 (3.0)
Clause attach	349	309	40 (11.4)	335	14 (4.0)	319	30 (8.6)
NP Internal structure	341	325	16 (4.7)	324	17 (5.0)	318	23 (6.7)
Coordination	279	285	−6 (−2.1)	286	−7 (−2.5)	287	−8 (−2.9)
Modifier attachment	297	280	17 (5.7)	285	124.0	282	15 (5.1)
NP Attachment	167	169	−2 (−1.2)	151	16 (9.5)	172	−5 (−3.0)
VP Attachment	71	66	5 (7.0)	57	14 (19.7)	60	11 (15.5)
Unary clause label	85	84	1 (1.2)	85	0 (0)	83	2 (2.4)
Unclassified	553	556	−3 (−0.5)	520	33 (6.0)	498	55 (9.9)

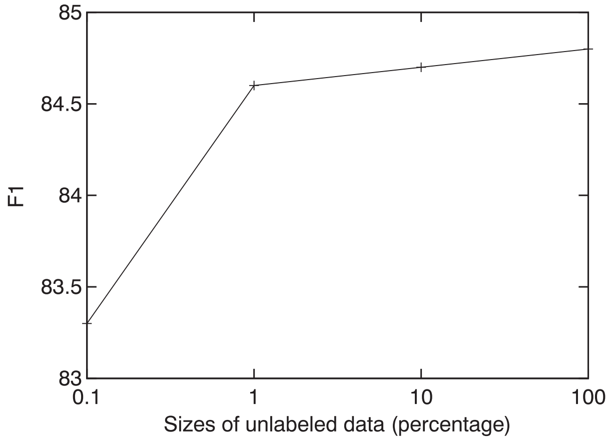

FIGURE 3. The parsing accuracy with varying sizes of unlabeled data on the Chinese test set.

time studied semi-supervised shift-reduce constituency parsing. They enriched feature representations of a shift-reduce parser with a set of features that encodes lexical dependency information extracted from automatically parsed data. Our work is also a case of semi-supervised shift-reduce parsing. One major difference between the work of Zhu et al. (2012) and Zhu et al. (2013) and our work is that we integrate constituent boundary information into shift-reduce parsing. To our knowledge, such information has not been used to improve parsing accuracy before, especially targeting at shift-reduce parsing.

The idea of integrating boundary information into syntactic parsers has been studied before. Roark and Hollingshead (2008) classified each word in a sentence into two categories: starting a multiword constituent and ending a multiword constituent. Such boundary information was used to prune candidate parses in a chart-based parser. In Roark and Hollingshead (2009), the idea was extended further to handle single-word spans,and experimental results were reported on both English and Chinese. A similar idea was used in Zhang et al. (2010) to increase the efficiency of a CCG parser. Zhang et al. (2010) extended the binary classifier of Roark and Hollingshead (2008) to predicting the maximum size of the constituents a word can start and end, and found increased pruning effects. In the same direction, Bodenstab et al. (2011) proposed to classify a text span instead of individual words to capture boundary information. Our work also focuses on integrating boundary information. However, compared with the previous work using boundary information, our work has three major differences. First, we utilize boundary information for shift-reduce parsing instead of chart parsing. Second, the purpose of integrating boundary information in this article is to achieve higher parsing accuracy instead of efficiency. Third, the tree likelihood model is our newly proposed approach to capturing boundary information.

In the direction of semi-supervised parsing, there are closely-related works on dependency parsing. Koo et al. (2008) proposed a simple yet effective word clustering-based approach to semi-supervised dependency parsing. Intuitively, word clustering information can be incorporated together with boundary information to achieve further improvements. Chen et al. (2009) extracted dependency subtrees from automatically parsed data, which were applied to advance a graph-based dependency parser. Chen et al. (2012b) built a dependency language model on automatically parsed data, which was integrated to improve a graph-based dependency parser. The tree likelihood model proposed in this article is similar to the dependency language model in Chen et al. (2012b), with the notable difference that tree likelihood model contains constituency boundary information.

Recent years have seen application of deep learning techniques in this direction of shift-reduce constituent parsing. Watanabe and Sumita (2015) propose a new neural network structure that models the unbounded history of actions performed on the stack and queue in transition-based parsing. Instead of modeling of the transition process, Wang et al. (2015) focus on automatic learning of features representations that was conducted by human in most of previous work. Both of these recent works achieve state-of-the-art parsing accuracy.

7. CONCLUSION

In this article, constituent boundary information is integrated into shift-reduce constituency parsing for the improvement of accuracy. We studied two different models to capture boundary information, and designed a set of novel features that encode boundary information collected from large-scale automatically parsed data. To our knowledge, this is the first work that aims to improve shift-reduce parsing performance with constituent boundary information. Experimental results show that boundary features improve parsing accuracies significantly by 0.8% on English and by 1.6% on Chinese on standard benchmarks.

For future work, there are several ways in which this research could be extended. First, we plan to use larger data to build the BPM and TLM models. Second, we could apply the proposed approach to other languages. Third, it might be possible to generate semi-supervised features based on the combination of word surfaces and POS tags. Finally, we could combine our approach with other methods on semi-supervised shift-reduce parsers.

ACKNOWLEDGMENTS

Wenliang Chen, Min Zhang, and Jingbo Zhu were supported by the National Natural Science Foundation of China (Grant No. 61572338, 61525205, 61272376) and Yue Zhang was supported by Singapore MOE grant 2012-T2-2-163. This work was partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization. Corresponding author is Muhua Zhu. We would also thank the anonymous reviewers for their detailed comments, which have helped us to improve the quality of this work.

REFERENCES

AHO, A. V., and J. D. ULLMAN. 1972. The Theory of Parsing, Translation, and Compiling. Prentice-Hall: Upper Saddle River, NJ.

BERGER, A. L., V. J. D. PIETRA, and S. A. D. PIETRA. 1996. A maximum entropy approach to natural language processing. Computational Linguistics, **22**(1): 39–71.

BIKEL, D. M. 2004. On the parameter space of generative lexicalized statistical parsing models, Ph.D. dissertation, University of Pennsylvania, Philadelphia.

BODENSTAB, N., A. DUNLOP, K. HALL, and B. ROARK. 2011. Beam-width prediction for efficient context-free parsing. *In* Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, Portland, OR, pp. 440–449.

CARRERAS, X., M. COLLINS, and T. KOO. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. *In* Proceedings of the Twelfth Conference on Computational Natural Language Learning. Association for Computational Linguistics, Manchester, UK, pp. 9–16.

CHARNIAK, E. 2000. A maximum-entropy-inspired parser. *In* Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference, Seattle, WA, pp. 132–139.

CHARNIAK, E., and M. JOHNSON. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. *In* Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Ann Arbor, MI, pp. 173–180.

CHARNIAK, E., K. KNIGHT, and K. YAMADA. 2003. Syntax-based language models for statistical machine translation. *In* Proceedings of MT Summit IX. Citeseer, New Orleans, LA, pp. 40–46.

CHEN, W., J. KAZAMA, K. UCHIMOTO, and K. TORISAWA. 2009. Improving dependency parsing with subtrees from auto-parsed data. *In* Proceedings of EMNLP 2009, Singapore, pp. 570–579.

CHEN, W., J. KAZAMA, K. UCHIMOTO, and K. TORISAWA. 2012a. Exploiting subtrees in auto-parsed data to improve dependency parsing. Computational Intelligence, **28**(3): 426–451.

CHEN, W., M. ZHANG, and H. LI. 2012b. Utilizing dependency language models for graph-based dependency parsing models. *In* Proceedings of ACL 2012, Jeju Island, pp. 213–222.

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, Philadelphia, PA, pp. 1–8.

COLLINS, M. 2003. Head-driven statistical models for natural language parsing. Computational Linguistics, **29**(4): 589–637.

COLLINS, M., and T. KOO. 2005. Discriminative reranking for natural language parsing. Computational Linguistics, **31**(1): 25–70.

COLLINS, M., and B. ROARK. 2004. Incremental parsing with the perceptron algorithm. *In* Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, Barcelona, Spain, pp. 111–118.

CRAMMER, K., O. DEKEL, J. KESHET, S. SHALEV-SHWARTZ, and Y. SINGER. 2006. Online passive-aggressive algorithms. The Journal of Machine Learning Research, **7**: 551–585.

HUANG, L. 2008. Forest reranking: discriminative parsing with non-local features. *In* ACL, Columbus, OH, pp. 586–594.

HUANG, L.-Y. 2009. Improve Chinese parsing with Max-Ent reranking parser. Master Project Report, Brown University, Providence, RI.

HUANG, Z., and M. HARPER. 2009. Self-training pcfg grammars with latent annotations across languages. *In* Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2, Singapore, pp. 832–841.

HUANG, Z., M. HARPER, and S. PETROV. 2010. Self-training with products of latent variable grammars. *In* Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, Cambridge, MA, pp. 12–22.

KOO, T., X. CARRERAS, and M. COLLINS. 2008. Simple semi-supervised dependency parsing. *In* Proceedings of ACL-08: HLT, Columbus, OH, pp. 595–603.

KUMMERFELD, J. K., D. HALL, J. R. CURRAN, and D. KLEIN. 2012. Parser showdown at the Wall Street corral: an empirical investigation of error types in parser output. *In* Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Korea, pp. 1048–1059.

MARCUS, M. P., B. SANTORINI, and M. A. MARCINKIEWICZ. 1993. Building a large annotated corpus of English: the Penn Treebank. Computational Linguisticss, **19**(2): 313–330.

MCCLOSKY, D., E. CHARNIAK, and M. JOHNSON. 2006. Effective self-training for parsing. *In* Proceedings of NAACL, New York City, pp. 152–159.

PETROV, S., and D. KLEIN. 2007. Improved inference for unlexicalized parsing. *In* HLT-NAACL, Vol. 7, Rochester, NY, pp. 404–411.

RATNAPARKHI, A. 1997. A linear observed time statistical parser based on maximum entropy models. arXiv preprint cmp-lg/9706014.

ROARK, B., and K. HOLLINGSHEAD. 2008. Classifying chart cells for quadratic complexity context-free inference. *In* Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1, Manchester, UK, pp. 745–751.

ROARK, B., and K. HOLLINGSHEAD. 2009. Linear complexity context-free parsing pipelines via chart constraints. *In* Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Boulder, CO, pp. 647–655.

SAGAE, K., and A. LAVIE. 2005. A classifier-based parser with linear run-time complexity. *In* Proceedings of the Ninth International Workshop on Parsing Technology. Association for Computational Linguistics, Ann Arbor, MI, pp. 125–132.

SAGAE, K., and A. LAVIE. 2006. A best-first probabilistic shift-reduce parser. *In* Proceedings of the COLING/ACL on Main Conference Poster Sessions, Sydney, Australia, pp. 691–698.

SHINDO, H., Y. MIYAO, A. FUJINO, and M. NAGATA. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. *In* Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, Jeju Island, Korea, pp. 440–448.

WANG, M., K. SAGAE, and T. MITAMURA. 2006. A fast, accurate deterministic parser for Chinese. *In* Coling-ACL2006, Sydney, Australia, pp. 425–432.

WANG, Z., H. MI, and N. XUE. 2015. Feature optimization for constituent parsing via neural network. *In* ACL (1), Beijing, China, pp. 1138–1147.

WATANABE, T., and E. SUMITA. 2015. Transition-based neural constituent parsing. *In* ACL (1), Beijing, China, pp. 1169–1179.

XUE, N., F. XIA, F.-D. CHIOU, and M. PALMER. 2005. The Penn Chinese Treebank: phrase structure annotation of a large corpus. Natural Language Engineering, **11**(02): 207–238.

ZHANG, Y., B.-G. AHN, S. CLARK, C. VAN WYK, J. R. CURRAN, and L. RIMELL. 2010. Chart pruning for fast lexicalised-grammar parsing. *In* Proceedings of the 23rd International Conference on Computational Linguistics: Posters, Beijing, China, pp. 1471–1479.

ZHANG, Y., and S. CLARK. 2009. Transition-based parsing of the Chinese Treebank using a global discriminative model. *In* Proceedings of the 11th International Conference on Parsing Technologies, Paris, France, pp. 162–171.

ZHANG, Y., and S. CLARK. 2011. Syntactic processing using the generalized perceptron and beam search. Computational Linguistics, **37**(1): 105–151.

ZHU, M., Y. ZHANG, W. CHEN, M. ZHANG, and J. ZHU. 2013. Fast and accurate shift-reduce constituent parsing. *In* ACL (1), Sofia, Bulgaria, pp. 434–443.

ZHU, M., J. ZHU, and H. WANG. 2012. Exploiting lexical dependencies from large-scale data for better shift-reduce constituency parsing. *In* COLING, Mumbai, India, pp. 3171–3186.