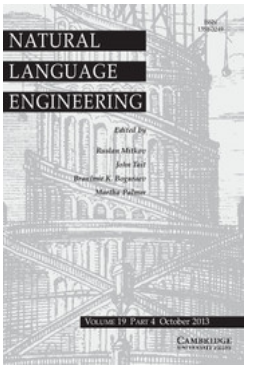
Natural Language Engineering

http://journals.cambridge.org/NLE

Additional services for ***Natural Language Engineering:***

Email alerts: Click here
Subscriptions: Click here
Commercial reprints: Click here
Terms of use : Click here



Improving shift-reduce constituency parsing with large-scale unlabeled data

MUHUA ZHU, JINGBO ZHU and HUIZHEN WANG

Natural Language Engineering / *FirstView* Article / October 2013, pp 1 - 26
DOI: 10.1017/S1351324913000119, Published online: 19 June 2013

Link to this article: http://journals.cambridge.org/abstract_S1351324913000119

How to cite this article:
MUHUA ZHU, JINGBO ZHU and HUIZHEN WANG Improving shift-reduce constituency parsing with large-scale unlabeled data. Natural Language Engineering, Available on CJO 2013 doi:10.1017/S1351324913000119

Request Permissions : Click here

CAMBRIDGE JOURNALS

Downloaded from http://journals.cambridge.org/NLE, IP address: 129.169.82.151 on 15 Oct 2013

Natural Language Engineering: page 1 of 26. © Cambridge University Press 2013
doi:10.1017/S1351324913000119

Improving shift-reduce constituency parsing with large-scale unlabeled data

MUHUA ZHU, JINGBO ZHU and HUIZHEN WANG†
Natural Language Processing Lab, Northeastern University, Shenyang 110819, China
e-mails: zhumuhua@gmail.com, {zhujingbo,wanghuizhen}@mail.neu.edu.cn

(*Received 15 July 2012; revised 14 May 2013; accepted 17 May 2013*)

Abstract

Shift-reduce parsing has been studied extensively for diverse grammars due to the simplicity and running efficiency. However, in the field of constituency parsing, shift-reduce parsers lag behind state-of-the-art parsers. In this paper we propose a semi-supervised approach for advancing shift-reduce constituency parsing. First, we apply the uptraining approach (Petrov, S. *et al.* 2010. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Cambridge, MA, USA, pp. 705–713) to improve part-of-speech taggers to provide better part-of-speech tags to subsequent shift-reduce parsers. Second, we enhance shift-reduce parsing models with novel features that are defined on lexical dependency information. Both stages depend on the use of large-scale unlabeled data. Experimental results show that the approach achieves overall improvements of 1.5 percent and 2.1 percent on English and Chinese data respectively. Moreover, the final parsing accuracies reach 90.9 percent and 82.2 percent respectively, which are comparable with the accuracy of state-of-the-art parsers.

1 Introduction

Due to the simplicity and running efficiency, shift-reduce parsing has been studied extensively for a variety of grammars, ranging from constituency parsing (Sagae and Lavie 2005; Sagae and Lavie 2006; Zhang and Clark 2009) through dependency parsing (Yamada and Matsumoto 2003; Nivre 2004; Huang, Jiang and Liu 2009b) to Combinatory Categorial Grammar parsing (Zhang and Clark 2011). In dependency and Combinatory Categorial Grammar parsing, shift-reduce parsing is among the best-performing algorithms (Huang and Sagae 2010; Zhang and Clark 2011). However, compared with commonly used statistical parsers, such as the Charniak–Johnson re-ranking parser (Charniak and Johnson 2005) and the Berkeley parser (Petrov and Klein 2007), there still exists room to improve the performance of shift-reduce constituency parsers.

Prior efforts on improving shift-reduce constituency parsing focus on the following two directions. One direction is to design better training and decoding algorithms. For example, in the respect of decoding, Sagae and Lavie (2006) proposed the best-first search strategy to expand the search space. In the respect of training, Zhang and

†Corresponding author.

Clark (2009) replaced local classifiers with a global learning algorithm. The other direction is to enrich feature representations for better shift-reduce constituency parsing (Sagae and Lavie 2005; Sagae and Lavie 2006; Wang, Sagae and Mitamura 2006; Zhang and Clark 2009). All the previous works listed above discuss shift-reduce parsing in the setting of supervised learning. In this paper we instead focus on semi-supervised shift-reduce constituency parsing, which, to the best of our knowledge, has not been studied before. Specifically, considering that shift-reduce parsers typically take part-of-speech (POS) tagging as a preliminary step and the performance of shift-reduce parsers is heavily dependent on the quality of input POS tags,[1] we propose a two-stage approach: We first improve POS taggers to supply syntactic parsers with better POS tags and then enrich feature representation of a baseline shift-reduce parser. Both stages improve performance by utilizing large-scale unlabeled data.

In the respect of POS tagging preprocessing, most previous work on shift-reduce parsing simply utilizes off-the-shelf POS taggers. In terms of the performance metric of per-token accuracy, those POS taggers seem difficult to improve further, especially for languages that have been extensively studied, for example English (Manning 2011). However, if we go beyond stand-alone POS taggers, we find that parsers such as the Berkeley parser can also be used to assign POS tags (together with parse trees) to sentences. Moreover, POS tags supplied by the parsers give rise to substantially higher shift-reduce parsing accuracy than POS tags coming from state-of-the-art POS taggers (see Section 6.1.1). Although it is practically insignificant to utilize a parser, say the Berkeley parser, to feed POS tags to a shift-reduce parser, we can actually use such parsers to help construct better POS taggers for shift-reduce parsing. Hereafter, parsers that integrate POS tagging and syntactic parsing in a single step are referred to as *integrated parser*, examples of which include the Berkeley and Bikel parsers (Bikel 2004).[2] Briefly, our approach to improving POS taggers is to train a stand-alone POS tagger on the output of integrated parsers on large-scale unlabeled data.

To improve shift-reduce parsers, one natural idea is to follow the direction of our approach to POS tagging, that is, to train a shift-reduce parser on the combination of human-labeled training data and auto-parsed data produced by integrated parsers. However, such an approach is extremely time-consuming for shift-reduce parsers that require to parse training sentences repeatedly (the baseline parser used in this paper is one such instance). Thus, the approach proposed in this paper is to mine information from auto-parsed data in an off-line manner and then enrich feature representation based on the obtained information. Specifically, we use the information of lexical head–modifier[3] relations (also known as lexical dependencies) (Collins

[1] Hatori, Matsuzaki and Tsujii (2011) studied joint POS tagging and dependency parsing in a shift-reduce parsing framework. However, the pipeline approach to POS tagging and shift-reduce parsing is still the mainstream.

[2] The Bikel parser has two input formats: with/without POS tags respectively. We use the latter one in this paper.

[3] By 'head–modifier' we mean the linguistic notion that a word (*modifier*) modifies another word (*head*).

1996). Previous work on other constituency parsers has shown the effectiveness of lexical dependency information on disambiguating syntactic structures (Collins 1996, 1997; Eisner and Satta 1999). But in shift-reduce constituency parsing such information is not fully used. For instance, Zhang and Clark (2009) encoded lexical information as features but ignored dependency information between words. Sagae and Lavie (2005) and Wang *et al.* (2006) only incorporated as features the most recently recognized (left and right) modifiers of some designated words. By contrast, this paper defines features that *explicitly* encode the information of whether words in an input sentence tend to have head–modifier relations. Moreover, we propose to obtain lexical dependency information from large-scale auto-parsed data instead of human-labeled treebank data because lexical dependencies obtained from the latter source suffer from data sparseness (Section 5.2.1). We focus on bigram and trigram lexical dependencies from automatically parsed trees. Based on the extracted lexical dependencies, we design a set of features to enhance the baseline parser.

The two-stage approach described above can be regarded as a variant of the uptraining approach (Petrov *et al.* 2010) where auto-parsed data produced by the Berkeley parser is used as additional training data to improve POS taggers and shift-reduce parsers for dependency parsing. The most significant difference between our work and that of Petrov *et al.* (2010) lies in the second stage: We do not simply combine auto-parsed data to improve shift-reduce parsing; we instead extract partial information from auto-parsed data. In addition, it is notable that the approach to improving shift-reduce parsing is actually independent of the parsers that are used to generate auto-parsed data. In this paper we choose to utilize the same auto-parsed data as with POS tagging for the purpose of consistency. Experimental results show that our approach can improve shift-reduce constituency parsing to 90.9 percent on English and 82.2 percent on Chinese. In addition, our parser outperforms previously reported shift-reduce constituency parsers while maintaining the efficiency.

The remainder of this paper is organized as follows. Section 2 discusses in detail the related work. In Section 3 we describe the baseline parser and baseline features that are used in this paper. Section 4 presents the approach to improving POS taggers and the shift-reduce baseline parser. Section 5 introduces experimental setup and Section 6 presents the detailed experimental results. Finally, we conclude our work in Section 7.

2 Related work

Semi-supervised approaches to POS tagging have been widely studied. Merialdo (1994) attempted to improve a Hidden Markov Model-based tagger with expectation maximization. However, the results that Merialdo (1994) have achieved are negative. Wang *et al.* (2007) studied co-training with two distinct POS taggers but their approach seems to achieve positive results only when human-labeled training set is small in size. More recently, alternative methods based on system combination were proposed. Suzuki and Isozaki (2008) proposed an extension of semi-supervised conditional random fields by combining supervised and unsupervised probability models. Sϕgaard (2010) studied system combination in a tri-training framework.

Clark, Curran and Osborne (2003) adopted the self-training approach and achieved positive results only when human-labeled data are limited. Huang, Eidelman, and Harper (2009a) reported positive results using self-training of a hidden Markov model POS tagger. Unlike the previous work, this paper adopts the uptraining approach which improves stand-alone POS taggers with the aid of integrated POS taggers. The uptraining approach was first proposed in Petrov *et al.* (2010), where the Berkeley parser was used to improve dependency parsers in a target domain. In this paper we adapt the approach to in-domain POS tagging.

Shift-reduce parsing has been widely studied due to its simplicity and efficiency. For constituency parsing, Sagae and Lavie (2005) proposed a classifier-based shift-reduce parser which was extended with the best-first search strategy in Sagae and Lavie (2006). Wang *et al.* (2006) adapted the parser in Sagae and Lavie (2005) to Chinese parsing and compared some representative classifiers. Zhang and Clark (2009) proposed a global learning algorithm to replace local classifiers. Shift-reduce parsing was also widely applied to parsing with other grammars (Nivre 2004; Huang *et al.* 2009b; Huang and Sagae 2010; Zhang and Clark 2011). In this paper we focus on constituency parsing, especially on improving Zhang and Clark (2009) with a semi-supervised approach. To the best of our knowledge, semi-supervised shift-reduce constituency parsing has not been studied before. In this respect, self-training (and its variant, the uptraining approach) is the most successful approach (McClosky, Charniak and Johnson 2006; Huang and Harper 2009; Huang, Harper and Petrov 2010; Petrov *et al.* 2010). The difference between self-training and our approach is that we use partial information derived from auto-parsed data instead of entire automatically parsed trees. Chen *et al.* (2012a) and Noord (2007) exploited lexical dependencies from unlabeled data for dependency and Head-driven Phrase Structure Grammar parsing respectively. In this paper we for the first time use lexical dependency information for advancing state-of-the-art shift-reduce constituency parsers. It is worth emphasizing that although we follow the approach of Chen *et al.* (2012a) for lexical dependency extraction, new features proposed in this paper are different from the features proposed in Chen *et al.* (2012a) because lexical dependencies are used for different parsers. Specifically, Chen *et al.* (2012a) aims to improve a graph-based dependency parser whereas this paper focuses on a transition-based constituency parser.

3 Baseline parser

The baseline parser used in this paper is the beam-search shift-reduce parser (Zhang and Clark 2009). To our knowledge, this parser achieves the best performance among all the shift-reduce constituency parsers that have been reported before.

3.1 The shift-reduce parsing process

The shift-reduce process in the baseline parser assumes binary-branching trees, so binarization and debinarization are required for transforming training data and

parsing output respectively (Zhang and Clark 2009). Given an input sentence (words and POS tags), any possible parse tree yielding the sentence corresponds *exactly* to one sequence of states. Formally, each state in the sequence is denoted by a tuple $\langle S, Q \rangle$, where S is a stack containing partial parses that have been recognized and Q is a queue of word-POS pairs that remain unprocessed. In particular, the initial state is $\langle \phi, w_1 \dots w_n \rangle$, 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 pre-designated root label and Q is empty. Thus, 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:

- shift, which moves a pair of word and POS tag from the head of the queue to the stack. Here the queue is required to be non-empty.
- reduce-unary-X, which extends the top item on the stack by applying a unary rule and then replaces 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.
- reduce-binary-{L/R}-X, which moves top two items out of the stack and pushes 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 becoming the right child and the second popped item becoming the left child. The switch L/R indicates whether the left (L) or the right (R) child becomes the head child.
- terminate, which pops the root node off the stack and ends parsing. This action is applicable only when the stack contains a single parse and the queue is empty.

3.2 Beam-search extension

The shift-reduce parsing process described above can be extended with beam search, as presented in Algorithm 1. The algorithm starts by initializing a beam of size K with the initial state. In each iteration after the initialization, states are popped in turn out of the beam. For each popped state, all applicable actions are then evaluated with respect to the state. Scored action-state pairs are sorted in a temporary priority queue. When the beam gets empty, the top K highest scored action-state pairs are fetched from the priority queue and next states corresponding to the action-state pairs are inserted back into the beam. If the highest scored state in the beam is a final state, it will be returned as the parsing result; else the iteration continues. The algorithm has a time complexity of $O(nK)$, where n is the sentence length and K is the beam width.

3.3 Model and learning algorithm

To score an action A with respect to a state $Y = \langle S, Q \rangle$, we use a linear model as defined by

$$Score(\langle A, Y \rangle) = \vec{w} \cdot \Phi(\langle A, Y \rangle) = \sum_i \lambda_i f_i(\langle A, Y \rangle) \quad (1)$$

Algorithm 1 Beam-search shift-reduce parsing

Input: a POS-tagged word sequence $w_1 \dots w_n$
beam size K and *action set*

```
1: B ← {⟨φ, w1 ... wn⟩}          // initialize beam
2: loop
3:    priority queue P = []
4:    while B not empty do
5:       state ← pop(B)
6:       for all act ∈ action set do
7:          score ← evaluate act for state
8:          P·insert (⟨score, act, state⟩)
9:    for i = 0 to K do
10:      ⟨score, act, state⟩ ← Pop-Top (P)
11:      next ← apply act to state
12:      insert next to B
13:   best ← highest-scored state in B
14:   if best is complete then
15:      return best
```

where $f_i(\langle A, Y \rangle)$ are features extracted jointly from the action A and state Y. To learn parameters λ_i, we use the generalized perceptron algorithm proposed in Collins (2002).

Generalized perceptron is an online learning algorithm that learns one instance at a time. The basic procedure is to use the beam-search parsing algorithm (Algorithm 1) to parse the yield of a gold parse tree. Whenever the gold partial parse is pruned from the beam, parameters will be updated immediately and the learner moves to the next training instance. Such a strategy is known as 'early-update' (Collins and Roark 2004). Finally, model parameters are set to be an average of the weight vectors obtained during online learning.

3.4 Baseline features

Features used in the baseline parser are similar to those used in Zhang and Clark (2009). For convenience of reference, we repeat the features in Table 1, where the symbol S_i represents the ith item from the top of the stack S and the symbol Q_i denotes the ith item from the front end of the queue Q. The symbol w represents the lexical head for an item; c represents the label for an item; and t denotes POS of a lexical head. Note that Zhang and Clak (2009) also used bracket-related and separator features for Chinese parsing, which have been removed in the latest release of their parser. So in this paper we choose to ignore such language-specific features.

4 Our approach to better POS tagging and syntactic parsing

4.1 Uptraining of POS taggers

The uptraining approach to improving a POS tagger proceeds in the following two steps. The corresponding pipeline is depicted in Figure 1.

Table 1. *A summary of baseline feature templates, where S_i represents the ith item in stack S and Q_i denotes the ith item in 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	$S_0cS_1cS_2c$, $S_0wS_1cS_2c$, $S_0cS_1wQ_0t$, $S_0cS_1cS_2w$, $S_0cS_1cQ_0t$, $S_0wS_1cQ_0t$, $S_0cS_1wQ_0t$, $S_0cS_1cQ_0w$

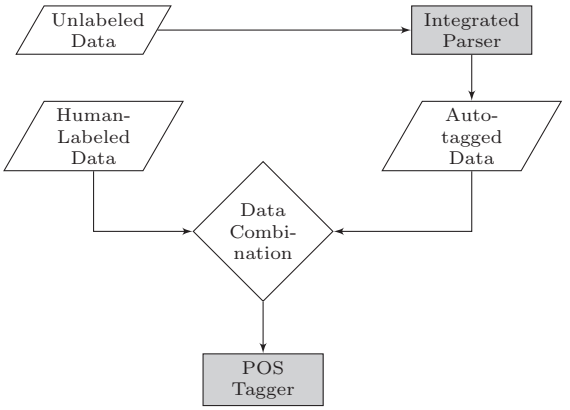


Fig. 1. The pipeline for uptraining of stand-alone POS taggers.

- Build an integrated parser on human-labeled training data and then use the parser to automatically process unlabeled data. This way we get a set of auto-tagged data.
- Merge human-labeled training data and auto-tagged data to get a data set on which we train a POS tagger.

It is noteworthy that although the focus of this paper is on shift-reduce parsing, the uptraining approach to improving POS taggers can benefit other parsers that take POS tagging as a pre-processing step, e.g. the chunking-based parser (Tsuruoka, Tsujii and Ananiadou 2009). Moreover, our approach requires labeled parse trees be available for training integrated parsers. Such a requirement seems to be a severe limitation at first sight. But it is not a real problem since we restrict our discussion of improving POS taggers in the context of syntactic parsing.

The uptraining approach described above is very similar to the ideas of standard self-training (Clark *et al.* 2003), with the difference that we use an integrated parser rather than a POS tagger to assign POS tags to unlabeled data. In addition, our approach aims to improve POS taggers for better syntactic parsing rather than

Table 2. *Types and ratios of first mistakes made by the baseline parser on the English development set*

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

higher POS tagging accuracy. In this sense, the approach has a philosophy similar to targeted self-training (Katz-Brown *et al.* 2011).

4.2 Enriching feature representation of shift-reduce parsing

4.2.1 Motivation

We first empirically analyze major sources of shift-reduce parsing errors with the parsing results of the baseline parser on the English development set. The baseline parser and a POS tagger are trained on the same human-labeled training data. Regarding the parsing results, we are especially concerned with *first mistakes* that the baseline parser makes because future mistakes are often caused by previous ones. There are 944 first mistakes in total in the parsing results. Table 2 shows the types and ratios of the top five most frequent first mistakes. Cases 1–2 consist of conflicts between shift and reduce actions. Case 3 comprises conflicts between reduce-binary and reduce-unary actions. Mistakes in cases 4–5 are caused by wrong choices of labels, where the symbols X, X_1, and X_2 refer to treebank phrase labels, and the symbol X^* denotes a temporary label introduced when a constituent with label X is binarized. We note in Table 2 that action conflicts between shift and reduce-binary are the largest source of parsing errors, which cover nearly half of first mistakes.

Intuitively, lexical dependency information is beneficial to resolving shift and reduce-binary conflicts. In the following, we use a real example to make clear the intuition. Figure 2 illustrates the shift-reduce parsing process of the baseline parser on a sentence with auto-assigned POS tags. The baseline parser proceeds correctly until it reaches the state in Figure 2(a). At that point, there is a conflict between reduce-binary (Figure 2(b)) and shift (Figure 2(c)) actions. We find that the baseline parser wrongly chooses the reduce-binary action because the word *bore* is (incorrectly) tagged as a verb.[4] The baseline parser tends to group the words preceding a verb as a constituent. However, if the parser is informed that the words *a* and *bore* have a lexical dependency relationship, the parser may correct its choice and switch to the shift action. In addition, we find that the human-labeled data used to train the baseline parser do not contain lexical dependencies between *a* and *bore*. We can see that extracting lexical dependencies solely from human-labeled training

[4] All the occurrences of *bore* in the training data of the POS tagger have the POS tag *VBD*.

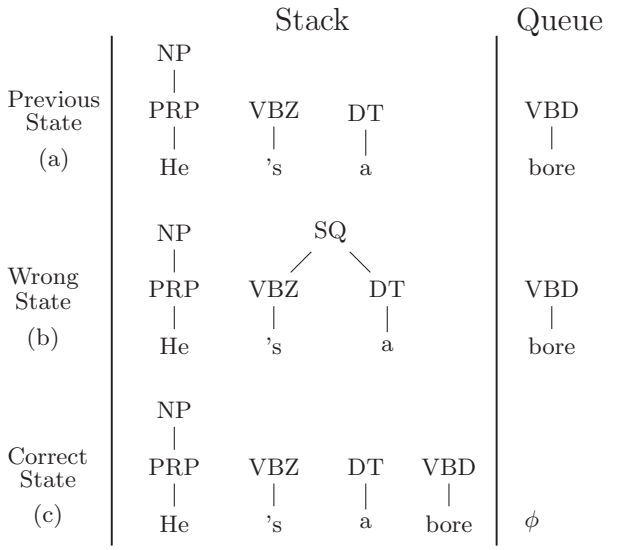


Fig. 2. An example of shift-reduce conflicts illustrating how lexical dependency information helps to disambiguate the conflicts.

data has a data sparseness problem. This motivates us to utilize unlabeled data as an additional source for lexical dependency extraction.

4.2.2 Data preprocessing

We propose to extract lexical dependencies from auto-parsed data. In principle, any parser, including the baseline parser of this paper, can be used to process unlabeled data to obtain automatically parsed trees. However, since we have produced large-scale auto-parsed in the POS tagging step by using integrated parsers, we will continue to use the same set of auto-parsed data. To simplify the extraction process, we convert automatically parsed constituency trees into dependency trees with Penn2Malt (or other conversion tools).[5] Given an input sentence x, we denote by y a dependency tree for x, and by $H(y)$ a set that includes the words that have at least one dependent. For each $x_h \in H(y)$, we have a dependency structure $D_h = (x_{Lk} \ldots, x_{L1}, x_h, x_{R1} \ldots x_{Rm})$, where $x_{Lk} \ldots x_{L1}$ and $x_{R1} \ldots x_{Rm}$ are dependents of the head x_h. Either k and m may be zero.

4.2.3 Extraction of lexical dependencies

After the conversion from constituency trees to dependency trees, the following lexical dependencies are read off from resulting dependency trees.

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

Bigram lexical dependencies: If two words are connected by an arc in a dependency tree, we claim these two words maintain a bigram lexical dependency. For pairs of words that have dependencies, we make a record of the words as well as their head–modifier relations. Formally, bigram lexical dependencies are denoted as $\langle w_1, w_2, L/R\rangle$, where L/R indicates the direction of the dependency arc that connects w_1 and w_2. Moreover, lexical dependencies are word-order sensitive, that is, $\langle w_1, w_2, L\rangle$ is regarded to be different from $\langle w_2, w_1, R\rangle$. Bigram lexical dependencies extracted from a dependency structure D_h can be represented as $\langle x_{Li}, x_h, R\rangle$ and $\langle x_h, x_{Ri}, L\rangle$.

Trigram lexical dependencies: Trigram lexical dependencies encode a head–modifier relationship among three words. As with bigram lexical dependencies, trigram lexical dependencies are also word-order sensitive. Although there exist distinct types of trigram lexical dependencies (Koo and Collins 2010), in this paper we only consider the type of trigram lexical dependency that has the first or the last word be the head and that requires the other two words to be siblings among all the modifiers of the head. Note that the two sibling modifiers are not necessary to be continuous in sentences. Among the types of trigram lexical dependencies we have experimented, this type is the only one that achieves positive effect on parsing accuracy. Such lexical dependencies can be represented formally as $\langle w_1, w_2, w_3, L/R\rangle$. Here the switch L/R indicates the head among the three words. Specifically, the symbol L specifies w_1 to be the head and the symbol R designates w_3 to be the head. In addition, we also consider the special case that w_2 is *NONE*, which indicates that w_1 (w_3) is the rightmost (leftmost) modifier of w_3 (w_1). Trigram lexical dependencies extracted from a dependency structure D_h can be represented as $\langle x_{Li}, x_{Li-1}, x_h, R\rangle$ and $\langle x_h, x_{Ri-1}, x_{Ri}, L\rangle$.

4.2.4 Proposed features

After extracting all lexical dependencies, we group bigram and trigram lexical dependencies *separately* into three categories according to their frequencies. Specifically, if a dependency relation is among top-10 percent most frequent records, then it receives the group tag *high frequency* (HF); if it is in top-20 percent, then we use the tag *Middle Frequency* (MF); else we use the tag *Low Frequency* (LF). Formally, the group strategy can be described as follows:

$$GR(r) = \begin{cases} HF & \text{if } r \in \textit{top}\text{-}10 \text{ percent} \\ MF & \text{else if } r \in \textit{top}\text{-}20 \text{ percent} \\ LF & \text{otherwise} \end{cases}$$

Although such a grouping strategy is heuristic in some sense, it has been proven effective in Chen *et al.* (2012a). After grouping, we finally get two lists, containing bigram and trigram lexical dependencies respectively.

Based on the bigram and trigram lexical dependency lists, we propose a set of dependency features which is described in detail in the following. Here s_i denotes the ith item from the top of the stack S, and q_i is the ith item from the front end of

Table 3. *New features designed on the basis of lexical dependencies. Here the symbol w represents a word and the symbol t represents a POS tag*

Bigram dependency features		
$f_L(s_1w, s_0w)$	$f_L(s_1w, s_0w) \circ s_1t \circ s_0t$	$f_R(s_1w, s_0w)$
$f_R(s_1w, s_0w) \circ s_1t \circ s_0t$	$f_L(s_1w, q_0w) \circ s_1t \circ q_0t$	$f_L(s_1w, q_0w)$
$f_R(s_1w, q_0w)$	$f_R(s_1w, q_0w) \circ s_1t \circ q_0t$	$f_L(s_0w, q_0w)$
$f_L(s_0w, q_0w) \circ s_0t \circ q_0t$	$f_R(s_0w, q_0w) \circ s_0t \circ q_0t$	$f_R(s_0w, q_0w)$
Trigram dependency features		
$f_L(s_1w, s_1rdw, s_0w)$	$f_L(s_1w, s_1rdw, s_0w) \circ s_1t \circ s_0t$	$f_R(s_1w, s_0ldw, s_0w)$
$f_R(s_1w, s_0ldw, s_0w) \circ s_1t \circ s_0t$	$f_L(s_0w, s_0rdw, q_0w) \circ s_0t \circ q_0t$	$f_L(s_0w, s_0rdw, q_0w)$
$f_R(s_0w, NONE, q_0w)$	$f_R(s_0w, NONE, q_0w) \circ s_0t \circ q_0t$	

the queue Q. In addition, s_iw (s_it) refers to the head word (POS) of s_i and q_iw (q_it) refers to the word (POS) of q_i.

Bigram dependency features: Bigram dependency features have a generic form of $f_{L/R}(w_1, w_2)$, which returns a group tag (HF, MF, or LF) if the lexical dependency $\langle w_1, w_2, L/R\rangle$ is found in the bigram lexical dependency list; else it returns *NULL*. The above feature template is instantiated into three pairs of features: $\{f_L(s_1w, s_0w), f_R(s_1w, s_0w)\}$, $\{f_L(s_0w, q_0w), f_R(s_0w, q_0w)\}$, and $\{f_L(s_1w, q_0w), f_R(s_1w, q_0w)\}$.

We also combine the above features with POS tags of w_1 and w_2. Thus, we have three more pairs of features in the generic form of $f_{L/R}(w_1, w_2) \circ t(w_1) \circ t(w_2)$, where $t(w_i)$ represents the POS tag of the word w_i. All the bigram dependency features are listed in Table 3.

Trigram dependency features: Trigram dependency features have the generic form of $f_{L/R}(w_1, w_2, w_3)$. In this paper this feature template is instantiated into two pairs of features. The feature function $f_L(s_1w, s_1rdw, s_0w)$ returns a group tag if $\langle s_1w, s_1rdw, s_0w, L\rangle$ is found in the trigram lexical dependency list, where s_1rdw denotes the rightmost modifier of s_1w that has been recognized so far during the shift-reduce parsing process. Note that s_1rdw might be *NONE* if no right modifiers have been recognized for s_1w. The other trigram dependency features, $f_R(s_1w, s_0ldw, s_0w)$, $f_L(s_0w, s_0rdw, q_0w)$, and $f_R(s_0w, NONE, q_0w)$, can be explained in a similar way. As with bigram dependency features, POS tags are combined with the above features to obtain richer feature representations. Trigram dependency features used in the paper are summarized in Table 3.

4.2.5 Parsing with proposed features

To use the proposed dependency features, we only need to update the scoring function defined in (1) (Section 2.3). The new scoring function is shown in the

following:

$$Score'(\langle A, Y\rangle) = \sum_i \lambda_i f_i(\langle A, Y\rangle) + \sum_j \lambda_j^d f_j^d \quad (2)$$

where $f_i(\langle A, Y\rangle)$ are the baseline features listed in Table 1 and f_j^d refer to the dependency features defined in Table 3.

5 Experimental setup

5.1 Data preparation

For English experiments, our labeled data came from the *Wall Street Journal* (WSJ) corpus of the Penn Treebank (Marcus, Santorini and Marcinkiewiz 1993). Following the divisions adopted in Charniak and Johnson (2005), we used sections 2–21 as training data, section 24 for system development, and section 23 for performance evaluation. In terms of English unlabeled data, we used the TIPSTER corpus (LDC93T3A), which contains news articles from various sources, although we only used *Wall Street Journal* articles.

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

We conducted necessary preprocessing on English and Chinese unlabeled data before they were used. Specifically, we applied OpenNLP for English sentence boundary detection and tokenization.[6] For the Chinese unlabeled data, we retained sentences that come from the body of documents, and then conducted sentence boundary detection simply according to sentence ending punctuations, including Chinese full stops, exclamation marks, and question marks. In addition, raw sentences were automatically segmented with a Conditional Random Field-based word segmenter, which achieves a segmentation accuracy of 97.2 percent on the testing data of CTB 5.1.

Table 4 contains detailed statistics of all the data used in the experiments.

5.2 POS taggers and integrated parsers

In principle, a shift-reduce parser can be pipelined with any POS taggers. In this paper we adopted representative POS taggers for English and Chinese experiments respectively. Specifically, for English experiments, LAPOS (Tsuruoka, Miyao and Kazama 2011) was applied which has the trait of looking ahead to POS tags to the right of the prediction position. Hereafter, the LAPOS POS tagger is referred to as **POS-LAPOS**. For experiments on Chinese data, we applied the Stanford POS tagger (Toutanova *et al.* 2003). Hereafter, we refer to this POS tagger simply as **POS-Stanford**.

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

Table 4. *Data statistics, including the number of words and sentences, together with average sentence length.* The numbers are approximate due to the use of automatic preprocessing techniques*

Language	Statistics	Train	Dev	Test	Unlabeled
English	#sentences	39.8k	1.7k	2.4k	3,139.1k
	#words	950.0k	40.1k	56.7k	76,041.4k*
	#ave. length	28.9	25.1	25.1	25.22*
Chinese	#sentences	18.1k	350	348	9,871.3k
	#words	493.8k	8.0k	6.8k	282,450.9k*
	#ave. length	27.3	19.5	23.0	28.6*

With respect to integrated parsers, we used the Berkeley parser as well as the emulation of Collins parsing model 2 (Collins 1999) in the Bikel parser. The parsers represent two important research directions in the field of constituency paring: unlexicalized PCFG with latent annotations, and lexicalized PCFG respectively. Moreover, these two parsers have distinct performance levels in POS tagging and syntactic parsing. Thus, we can empirically study the effect of performance of integrated parsers on the results of our approach. In the experiments of POS tagging, the POS tagging components in the Berkeley and Bikel parsers are regarded as a kind of POS taggers (called integrated POS tagger) and are referred to as **POS-Berkeley** and **POS-Bikel** respectively. In contrast, tagging systems, such as POS-LAPOS and POS-Stanford, are named as stand-alone POS taggers or simply as POS taggers.

5.3 Performance scoring

We measured tagging accuracy of POS taggers with per-token accuracy, which is defined to be the ratio of correctly tagged words over all the words in a test set. For performance evaluation of syntactic parsing, we used *EVALB* to provide bracket scoring as well as complete match scoring.[7] For significance tests, we adopted the comparator to compute p-value.[8] When we use significance testing to compare two results, we run *compare.pl* five times and then report the average of the obtained five p-values.

5.4 Running parameters

We set the beam size of the shift-reduce parser to 16 in both training and decoding phases which maintains a good trade-off between parsing efficiency and accuracy (Zhang and Clark 2009). With respect to the iteration number of perceptron learning, we tuned the parameter on the development sets and finally set the value to twenty-one for both English and Chinese experiments. In the respect of stand-alone POS taggers and integrated parsers, we used default parameter settings.

[7] http://nlp.cs.nyu.edu/evalb
[8] http://www.cis.upenn.edu/~dbikel/download/compare.pl

6 Experimental results

6.1 Improved POS tagging

We compared our approach to POS tagging with other representative approaches. POS taggers that were experimented are summarized in the following:

1. *Supervised training*: It trains POS taggers solely on human-labeled training data. Specifically, we trained POS-LAPOS (POS-Stanford) on human-labeled English (Chinese) training data and obtained a POS tagger named Base-LAPOS (Base-Stanford). The integrated POS taggers, POS-Berkeley and POS-Bikel, were applied to both English and Chinese POS tagging. We always refer to the resulting POS taggers as Base-Berkeley and Base-Bikel regardless of the specific language.
2. *Standard self-training*: It applies a base POS tagger to process unlabeled data and retrain the POS tagger on the combination of human-labeled training data and automatically tagged data. For English and Chinese POS tagging, Base-LAPOS and Base-Stanford are used as the base tagger respectively. The resulting self-trained taggers are referred to as Self-LAPOS and Self-Stanford.
3. *Uptraining*: It trains POS-LAPOS on the combination of labeled training data and auto-labeled POS data generated by Base-Berkeley (Base-Bikel) on unlabeled data. The resulting tagger is referred to as Berkeley-LAPOS (Bikel-LAPOS). For Chinese POS tagging, we replaced POS-LAPOS with POS-Stanford and the resulting Chinese POS taggers are referred to as Berkeley-Stanford and Bikel-Stanford.

When merging human-labeled data and auto-tagged data in the data combination, we simply gave our human-labeled training data a relative weight of one.

6.1.1 Stand-alone POS tagger versus integrated POS tagger

Table 5 shows the comparative results of stand-alone and integrated POS taggers in the setting of supervised learning. We evaluated POS taggers on the English and Chinese test sets by using the metrics of per-token accuracy as well as parsing accuracy of the baseline parser. As the results show, the integrated POS taggers achieve lower per-token accuracy than the stand-alone POS tagger does on the English test set, but they generate POS tags that result in higher parsing F1-scores ($\geq$0.7 percent). On Chinese data, the integrated POS taggers also provide better POS tags for syntactic parsing. In this sense, we can infer that integrated POS taggers are 'better' than stand-alone taggers. The reason is that integrated POS taggers use syntactic information as high-level constraints during assigning POS tags. So it is feasible to utilize an integrated POS tagger to improve a stand-alone POS tagger for better syntactic parsing.

6.1.2 Uptraining versus self-training

Table 6 shows the results of Berkeley-LAPOS (uptraining) and Self-LAPOS (self-training) on the English development set. This experiment serves the following two

Table 5. *Comparative results between stand-alone and integrated POS taggers on section 23 of the WSJ corpus and the CTB 5.1 test set*

Language	POS tagger	POS accuracy (%)	F1-score (%)
English	Base-LAPOS	97.46	89.4
	Base-Berkeley	97.35	90.2
	Base-Bikel	96.76	90.1
Chinese	Base-Stanford	95.38	80.1
	Base-Berkeley	95.58	82.4
	Base-Bikel	95.65	82.3

Table 6. *Comparative results between uptraining (Berkeley-LAPOS) and standard self-training (Self-LAPOS) on section 24 of the WSJ corpus*

POS tagger	Sent. added	POS accuracy (%)	F1-score (%)
	0 (Base-LAPOS)	97.21	88.4
Berkeley-LAPOS	100k	97.15	88.7
	250k	97.12	88.7
	500k	97.08	88.8
	1000k	97.07	88.8
	1,500k	**97.06**	**88.8**
	2,000k	97.08	88.8
Self-LAPOS	100k	97.23	88.4
	250k	97.19	88.3
	500k	97.18	88.3
	1,000k	97.21	88.4
	1,500k	97.22	88.4
	2,000k	97.21	88.4

purposes: (1) To demonstrate the advantage of uptraining over standard self-training in improving POS taggers for better syntactic parsing, and (2) to show the effect of different sizes of auto-tagged data that are used in uptraining. From the results we can see that the self-training approach has trivial effect on both POS tagging and parsing accuracy. Such results coincide with previous work on self-training for POS tagging (Clark *et al.* 2003). On the contrary, the uptraining approach achieves substantial improvements on parsing accuracy. Specifically, after 500k auto-tagged sentences are added, the approach improves parsing accuracy by 0.4 percent. We also decide the optimal size of auto-tagged sentences to be 1,500k because it achieves the highest parsing score when parsing scores are accurate to two decimal places. On the other hand, we note that uptraining with auto-tagged data causes POS tagging accuracy to regress. One major cause lies in tagging errors contained in auto-tagged data. In Section 6.1.4 we will analyze the reason why the uptraining approach improves parsing accuracy but degrades POS tagging accuracy.

A similar experiment was conducted on the Chinese development set by using the POS taggers Berkeley-Stanford and Self-Stanford. The results are shown in Table 7. We can see that the uptraining approach improves parsing accuracy substantially by

Table 7. *Comparative results between uptraining (Berkeley-Stanford) and standard self-training (Self-Stanford) on the CTB 5.1 development set*

POS tagger	Sent. Added	POS accuracy (%)	F1-score (%)
	0 (Base-Stanford)	95.76	84.1
Berkeley-Stan	100k	95.68	84.4
	250k	95.58	85.0
	500k	**95.53**	**85.0**
	750k	95.48	84.9
	1,000k	95.49	84.7
Self-Stanford	100k	95.76	83.9
	250k	95.95	84.4
	500k	95.99	84.4
	750k	95.94	84.4
	1,000k	95.87	84.3

0.9 percent, and it also outperforms the self-training approach. In addition, Berkeley-Stanford achieves the highest paring accuracy when 500k auto-tagged sentences are added.

6.1.3 POS tagging results on testing data

Table 8 presents the results of the uptraining approach on the English and Chinese test sets. We used 1,500k and 500k auto-tagged sentences for English and Chinese experiments respectively. Moreover, we compared two different integrated POS taggers: POS-Berkeley and POS-Bikel. From the English results we can see that uptraining with POS-Berkeley improves parsing accuracy by 0.7 percent in F1-score (from 89.4 to 90.1 percent; significant at the level of $p < 10^{-3}$). Moreover, although POS-Bikel has lower tagging and parsing accuracy than POS-Berkeley, uptraining with POS-Bikel also achieves an improvement of 0.6 percent (from 89.4 to 90.0 percent; significant at the level of $p < 10^{-3}$). Given that the improvement is close to that achieved by POS-Berkeley, the property that the parsers integrate POS tagging and syntactic parsing together seems to be more important than the accuracy of the parsers.

In contrast to experimental results on English, the uptraining approach achieves bigger improvements on Chinese ($\geq$1.0 percent; significant at the level of $p < 10^{-3}$). One possible reason is that the Chinese baseline POS tagger (Base-Stanford) has more room left for further performance improvements than the English baseline POS tagger (Base-LAPOS).

6.1.4 Additional analysis

The results on the testing data show that the uptraining approach either decreases per-token accuracy or only achieves trivial improvements in per-token accuracy. However, the approach benefits syntactic parsing consistently. It is of interest to examine the reasons for such a phenomenon. To this end, we took Chinese POS

Table 8. *Final results of uptraining of POS tagging on section 23 of the WSJ corpus and the CTB 5.1 test set*

Language	POS tagger	POS accuracy (%)	F1-score (%)
English	Base-LAPOS	97.5	89.4
	Berkeley-LAPOS	97.6	90.1
	Bikel-LAPOS	97.2	90.0
Chinese	Base-Stanford	95.4	80.1
	Berkeley-Stanford	95.5	81.3
	Bikel-Stanford	95.2	81.1

Table 9. *Confusion matrix of POS tagging results on the CTB 5.1 test set*

	VV	NN	DEC	DEG	JJ	NR	VA	AD
VV	0/0	76/67	0/0	0/0	4/2	0/0	2/2	6/5
NN	57/36	0/0	0/0	0/0	9/8	8/28	2/1	5/6
DEC	0/0	0/0	0/0	41/33	0/0	0/0	0/0	0/0
DEG	0/0	0/0	17/18	0/0	0/0	0/0	0/0	0/0
JJ	6/2	26/34	0/0	0/0	0/0	0/0	1/1	9/6
NR	3/0	12/30	0/0	0/0	0/0	0/0	0/0	0/0
VA	9/10	5/6	0/0	0/0	6/4	0/0	0/0	4/1
AD	6/5	5/6	0/0	0/0	7/5	0/0	0/0	0/0

tagging as a case study and compared the results of uptraining with 500k auto-tagged sentences with the results of the baseline POS tagger. The comparative results are shown in Table 9, where the row labels are correct POS tags and the column labels represent automatically assigned POS tags. For example, the numbers [NN, VV] = 57/36 in the table refer to the frequency that words of POS tag NN were incorrectly assigned to VV: the number 57 preceding the backslash is the result of uptraining, while the number 36 behind the backslash is the result of the baseline POS tagger.

From the results we can see that Chinese POS tagging has severe errors between the following POS: NN–VV, DEC–DEG, NN–JJ, and NN–NR. The uptraining approach has different effect on the POS pairs. Specifically, the approach has positive effect on disambiguating NN–VV and DEG–DEC but has negative effect on disambiguating NN–NR and NN–JJ. These POS have distinct effect on subsequent parsing accuracy. To simplify the analysis, we chose from the test set the sentences that have only one POS modification caused by the uptraining approach. Among the seventy-eight sentences that have been picked up, twenty-five sentences contain alternations between NN and NR. Among the twenty-five sentences, we found that parsing accuracy of thirteen sentences remained unchanged; parsing accuracy on five sentences was improved; and parsing accuracy on seven sentences was decreased. In contrast, nineteen sentences of the seventy-eight sentences contain alternations between NN and VV, among which parsing accuracy on fourteen sentences is increased, while parsing accuracy on the other five sentences is decreased. We can

Table 10. *Comparative results on English and Chinese development sets with lexical dependencies extracted from diverse sources*

	English			Chinese		
Data Source	LR (%)	LP (%)	F1-score (%)	LR (%)	LP (%)	F1-score (%)
Baseline	88.2	88.2	88.2	83.7	84.4	84.0
Human-labeled	88.5	88.7	88.6	83.8	84.4	84.1
Auto-parsed	88.9	89.0	89.0	85.2	85.1	85.1
Combined	89.0	89.2	89.1	85.3	85.2	85.2

see that, for the subsequent syntactic parsing task, the errors between NN and VV have more serious effect than the errors between NN and NR. The uptraining approach is able to reduce POS tagging errors that have significant effect on subsequent syntactic parsing.

6.2 Improved shift-reduce parsing

This section is devoted to showing the effectiveness of our approach to improving the baseline shift-reduce parser. To simplify the discussion, the results presented in this section are based on the baseline POS taggers, i.e. Base-LAPOS and Base-Stanford.

6.2.1 Comparison of different sources

Table 10 shows comparative results on English and Chinese development sets with lexical dependencies obtained from different sources. We experimented with four different settings: no lexical dependency information was used (Baseline), lexical dependencies were solely from human-labeled training data (Human-Labeled), lexical dependencies were solely from auto-parsed data (Auto-Parsed), and lexical dependencies were from the combination of human-labeled and auto-parsed data (Combined). For the latter three settings, all the dependency features listed in Table 3 were incorporated. In the data combination, we simply gave our human-labeled training data a relative weight of one. Note that in this and the following experiments, auto-parsed data were produced on the whole sets of unlabeled data.

From the results we can see that although lexical dependency information from human-labeled training data can improve the performance on both English and Chinese, the improvement on Chinese is moderate (0.1 percent in F1-score). One reason is that lexical dependencies from human-labeled training data have a sparseness problem. In contrast, the use of large-scale auto-parsed data brings on much bigger improvements (0.8 percent F1-score on English and 1.1 percent F1-score on Chinese). In addition, we find that data combination has trivial effect on performance. One possible reason is that lexical dependencies from human-labeled training data are overwhelmed by those from auto-parsed data. We will leave further discussions on data combination to our future work and focus on the setting of obtaining lexical dependencies from auto-parsed data.

Table 11. *Main results on section 23 of the WSJ corpus using POS tags automatically assigned by Base-LAPOS and lexical dependencies extracted from auto-parsed data. Two types of dependency features are added* ***incrementally***

Features	LR (%)	LP (%)	F1-score (%)	EX (%)
Baseline	89.5	89.3	89.4	39.2
+Bigram features	90.1	90.0	90.0	39.9
+Trigram features	90.4	90.2	90.3	41.0

Table 12. *Main results on the CTB 5.1 test set using POS tags automatically assigned by Base-Stanford and lexical dependencies extracted from auto-parsed data. Two types of dependency features are added* ***incrementally***

Features	LR (%)	LP (%)	F1-score (%)	EX (%)
Baseline	79.5	80.7	80.1	28.2
+Bi-lexical features	80.3	81.6	80.9	28.2
+Tri-lexical features	80.6	81.9	81.2	29.3

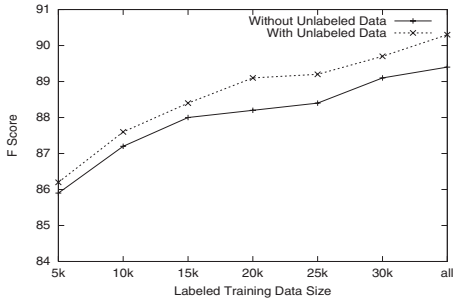
6.2.2 *Effect of different features*

Table 11 shows the results on the English test set, where the two types of dependency features were added incrementally to the baseline parser. As the results show, both bigram and trigram dependency features have positive effect on the parsing accuracy. Specifically, on the whole test set the overall improvement over the baseline parser is 0.9 percent in F1-score, where bigram dependency features contribute an absolute 0.6 percent improvement and trigram dependency features further improve the performance by 0.3 percent over the results of using bigram dependency features. Significance tests show that the overall improvement on the whole test set is statistically significant on the level of $p < 10^{-4}$.

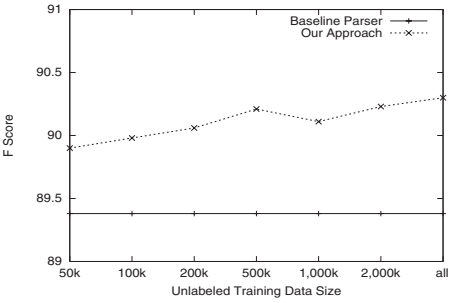
In parallel to the results on the English test set, Table 12 shows the results on the Chinese test set, using auto-assigned POS tags and lexical dependencies extracted from auto-parsed data. From the results on the whole test set we can see that dependency features contribute a bigger absolute improvement on Chinese than that on English (1.1 percent versus 0.9 percent). One possible reason is that the size of Chinese unlabeled data used in the paper is much bigger. Significance tests show that the overall improvement induced by bigram and trigram dependency features on the whole test is statistically significant on the level of $p < 10^{-3}$. These results indicate that the new features are very effective.

6.2.3 *Additional analysis*

We performed several types of analysis, focusing on English, to investigate the effect of different sizes of human-labeled training data and auto-parsed data, as well as how lexical dependency information changes the distribution of first mistakes made



(a)Varying labeled training data.



(b)Varying unlabeled data.

Fig. 3. Results with varying sizes of human-labeled training data and unlabeled data.

by the baseline parser.

Effect of different sizes of labeled and unlabeled Data: We studied the effect of varying sizes of human-labeled training data by randomly sampling from sections 2–21. Meanwhile, we always used the whole set of auto-parsed data. The results are depicted in Figure 3(a). As the results show, auto-parsed data improve parsing accuracy even when the human-labeled training data are small in size. In addition, by using our approach, a fraction of sections 2–21 plus the whole set of auto-parsed data is sufficient to achieve the F1-scores obtained by the parser trained solely on the whole sections 2–21.

We also examined the effect of varying sizes of unlabeled data. In this experiment, we used sections 2–21 as human-labeled training data and changed the size of unlabeled data through random sampling. The results are depicted in Figure 3(b). From the results we can see that improvements achieved by using unlabeled data are enlarged with the increment of the size of unlabeled data until the performance finally levels off.

Reduction on first mistakes: The baseline parser made 1,522 first mistakes on the English test set. We analyzed how our approach changed the first mistakes. We grouped the changes into four cases. *No-Change* (1,090) refers to the case that the baseline and our parser make the same first mistakes at the same positions. In the case of *Correct* (249), first mistakes made by the baseline parser are corrected by our parser. *Wrong* (121) means that our parser makes first mistakes earlier than the baseline parser. Finally, *Others* (47) refers to the case that our parser and the baseline parser make first mistakes of different types at the same positions. In addition, we are especially interested in how our approach reduces first mistakes of the type *shift versus reduce-binary*. So we compared the numbers of the first mistakes of this type in the *Correct* and *Wrong* cases, which are 168 and 78 respectively.

6.3 Final results

We combined the efforts on improving POS tagging and shift-reduce parsing, and obtained the final results on the English and Chinese test sets. Table 13 shows the

Table 13. *Comparison with related work on section 23 of the WSJ corpus with POS tags automatically assigned by Berkeley-LAPOS.* The parsers are based on shift-reduce parsing.† The results are of self-training with a single latent annotation grammar*

Type	Parser	LR (%)	LP (%)	F1-score (%)
SINGLE	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 and Lavie (2005)*	86.1	86.0	86.0
	Sagae and Lavie (2006)*	87.8	88.1	87.9
	Petrov and Klein (2007)	90.1	90.2	90.1
RE	Carreras *et al.* (2008)	90.7	91.4	91.1
	Charniak and Johnson (2005)	91.2	91.8	91.5
	Huang (2008)	92.2	91.2	91.7
SELF	Huang and Harper (2009)	91.1	91.6	91.3
	McClosky *et al.* (2006)	92.1	92.5	92.3
	Huang *et al.* (2010)†	91.4	91.8	91.6
	This paper auto-parsed	90.9	90.8	90.9

comparison of our parser with a large body of representative related work. Here all the related work except Ratnaparkhi (1997), Sagae and Lavie (2005), Sagae and Lavie (2006), and Carreras, Collins and Koo (2008) utilized integrated POS taggers. With regard to the exceptions, Ratnaparkhi (1997) adopted a Maximum Entropy-based POS tagger (Ratnaparkhi 1996); Sagae and Laive (2005) and Sagae and Lavie (2006) used SVMTool whose tagging accuracy on section 23 of the WSJ corpus is 97.1 percent;[9] Carreras *et al.* (2008) took as input the POS tags assigned by the Collins parser (Collins 1997). For fair comparison, here we disregarded parsers that are based on combination methods such as Petrov (2010) and Zhang *et al.* (2009). Following the taxonomy adopted in Huang *et al.* (2010), we grouped the related work into single parsers (SINGLE), discriminative reranking approaches (RE), and self-training (SELF). Note that self-trained parsers in the table were built on distinct sets of unlabeled data. Specifically, McClosky *et al.* (2006) made use of 2,000k sentences from the North American News Text corpus, NANC (Graff 1995); Huang and Harper (2009) sampled 210k sentences from the BLLIP corpus; Huang *et al.* (2010) partitioned 1,769,055 BLLIP sentences into ten equally sized subsets and used a single subset to achieve accuracy in Table 13. From the results we can see that our parser outperforms all the single parsers listed in the table except Carreras *et al.* (2008). However, compared with Carreras *et al.* (2008), our parser has much smaller time-complexity: $O(nK)$ versus $O(n^3G)$, where K is the beam size used in our parser and G is the grammar constant in Carreras *et al.* (2008). Compared with reranking and self-training parsers, our parser has relatively low parsing accuracy. But the reranking and self-training techniques are actually complementary with our

[9] www.lsi.upc.edu/∼nlp/SVMTool

Table 14. *Comparison with related work on the test set of CTB 5.1 with POS tags automatically assigned by Berkeley-Stanford.* Huang (2009) adapted the parsers to Chinese parsing on CTB 5.1.† We run the parser on CTB 5.1 to get the results*

Type	Parser	LP (%)	LR (%)	F1-score (%)
RE SINGLE	Charniak (2000)*	79.6	82.1	80.8
	Bikel (2004)†	79.3	82.0	80.6
	Petrov and Klein (2007)	81.9	84.8	83.3
	Charniak and Johnson (2005)*	80.8	83.8	82.3
	This paper auto-parsed	81.2	83.1	82.2

Table 15. *Comparison of running time on section 23 of the WSJ corpus where the time for loading models is excluded.* The results of SVM-based shift-reduce parsing are with greedy search.† The results of MaxEnt-based shift-reduce parser are with best-first search.‡ Time is reported by authors by running on different hardware*

Parser		Time (min)
Ratnaparkhi (1997)		Unk
Collins (1999)		11.4
Charniak (2000)		7
Sagae and Lavie (2005)*		11‡
Sagae and Lavie (2006)†		17‡
Petrov and Klein (2007)		6.5
Carreras *et al.* (2008)		Unk
This paper	Baseline	0.41
	Auto-parsed	0.72

approach, so we might enhance our parser with these techniques in the future.

Comparing Chinese constituency parsers is difficult in the sense that previously reported results were achieved frequently on different versions of CTB and/or with different data split standards. Zhang and Clark (2009) presented a detailed comparison between the baseline parser of this paper and a large body of related work on CTB 2.0. Here we only compared our parser with the parsers available on the web for Chinese parsing, as shown in Table 14. From the results we can see that on Chinese parsing our parser outperforms Bikel (2004) and Charniak (2000) by 1.6 percent and 1.4percent respectively. However, our parser lags behind Petrov and Klein (2007) and the reranking parser (Charniak and Johnson 2005).

Table 15 shows the comparison of running time between our parser and the single parsers listed in Table 13. For completeness, we also measured the running time of the baseline parser. This experiment was conducted on an Intel processor of 2.93 GHz and 8 GB memory. From the comparison we can see that incorporating lexical dependency features incurs additional running overhead (by comparing the running time of our parser and the baseline parser), but our parser still has an obvious advantage over other parsers. It should be noted that the running time of

the classifier-based Sagae–Laive parsers (Sagae and Lavie 2005; Sagae and Lavie 2006) was evaluated on different hardware, which is much longer than the running time of our parsers. However, we note that time complexity of the parsers of Sagae and Lavie (2005) is linear, smaller than that of our parsers ($O(n)$ versus $O(kn)$). The long running time of Sagae and Lavie (2005) in Table 15 is largely attributed to the use of support vector machines. Sagae and Laive (2006) reported the running time of a Maximum Entropy-based Implementation of Sagae and Lavie (2005), which is less then 1 min, close to the running time of our parsers. Taking into consideration the tradeoff between parsing accuracy and running speed, our parser is a suitable choice for practical applications on massive data, for example parsing the whole web.

7 Conclusion and future work

We have presented a semi-supervised approach to improving shift-reduce constituency parsing from two aspects. In respect of POS tagging, we enhanced POS taggers targeted at syntactic parsing by using auto-tagged POS data produced by integrated parsers; in respect of shift-reduce parsing, lexical dependency information extracted from auto-parsed data was utilized, based on which a set of new features was proposed and integrated into the shift-reduce parser. Our approach succeeded to construct POS taggers that can provide better POS tags for syntactic parsing and well addressed the action conflict problem. We evaluated the approach on English and Chinese data. The results show that our new parsers provide comparable accuracies with state-of-the-art parsers while maintaining the advantage in parsing speed.

There are many ways in which this research can be continued. First, we are concerned with the generalization of lexical dependencies. To this end, we propose to combine POS with words to obtain generalized dependencies. We can also use semantic classes of words to generalize dependencies. Second, we will experiment the shift-reduce parser proposed in this paper for bitext parsing (Zhao *et al.* 2009; Chen *et al.* 2012b). In bitext parsing, bilingual information can be used as additional information to further improve shift-reduce parsing. Intuitively, bilingual information and dependency information are complementary in disambiguating shift-reduce actions. All these will be conducted in our future work.

Acknowledgments

This work was supported in part by the National Science Foundation of China (61073140; 61272376; 61100089), Specialized Research Fund for the Doctoral Program of Higher Education (20100042110031), and the Fundamental Research Funds for the Central Universities (N110404012; N100204002).

References

Bikel, D. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models.* PhD thesis, University of Pennsylvania.

Carreras, X., Collins, M., and Koo, T. 2008. TAG, dynamic programming and the perceptron for efficient, feature-rich parsing. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL)*, Manchester, UK, pp. 9–16.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Washington, USA, pp. 132–9.

Charniak, E., and Johnson, M. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, University of Michigan, Ann Arbor, MI, USA, pp. 173–80.

Chen, W., Kazama, J., Uchimoto, K., and Torisawa, K. 2012. Exploiting subtrees in auto-parsed data to improve dependency parsing. *Computational Intelligence Journal* **28**(3): 426–51 (John Wiley).

Chen, W., Kazama, J., Zhang, M., Tsuruoka, Y., Zhang, Y., Wang, Y., Torisawa, K., and Li, H. 2012. Bitext dependency parsing with auto-generated bilingual treebanks. *IEEE Transactions on Audio, Speech and Language Processing* **20**(5): 1461–72.

Clark, S., Curran, J., and Osborne, M. 2003. Bootstrapping POS taggers using unlabeled data. In *Proceedings of the 7th Conference on Computational Natural Language Learning (CoNLL)*, Edmonton, Canada.

Collins, M. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, California, USA.

Collins, M. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, Madrid, Spain.

Collins, M. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Collins, M. 2002. Discriminative training methods for hidden Markov models: theory and experiemnts with perceptron algorithm. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, PA, USA, pp. 1–8.

Collins, M., and Roark, B. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain.

Eisner, J., and Satta, G. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, Maryland, USA.

Graff, D. 1995. *North American News Text Corpus*. Linguistic Data Consortium, Philadelphia, PA. LDC Catalog No. LDC95T21.

Hatori, J., Matsuzaki, T., and Tsujii, J. 2011. Incremental joint POS tagging and dependency parsing in Chinese. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, Chiang Mai, Thailand, pp. 8–13.

Huang, L. 2008. Forest reranking: discriminative parsing with non-local features. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, Ohio, USA, pp. 586–94.

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

Huang, Z., Eidelman, V., and Harper, M. 2009a. Improving a simple bigram HMM part-of-speech tagger by latent annotation and self-training. In *Proceedings of Huamn Language Technology: Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, 2009, Colorado, USA, pp. 213–6.

Huang, Z., and Harper, M. 2009. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Singapore, pp. 832–41.

Huang, Z., Harper, M., and Petrov, S. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Cambridge, MA, USA, pp. 12–22.

Huang, L., Jiang W., and Liu, Q. 2009b. Billingually constrained (monolingual) shift-reduce parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Singapore, pp. 1222–31.

Huang, L., and Sagae, K. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden, pp. 1077–86.

Katz-Brown, J., Petrov, S., McDonald, R., Och, F., Talbot, D., Ichikawa, H., Seno, M., and Kazawa, H. 2011. Training a parser for machine translation reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Edinburgh, UK, pp. 183–92.

Koo, T., and Collins, M. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden, pp. 1–11.

McClosky, D, Charniak, E., and Johnson, M. 2006. Effective self-training for parsing. In *Proceedings of Human Language Technology Conference – North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, New York, USA, pp. 152–9.

Manning, C. D. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Proceedings of Computational Linguistics and Intelligence Text Processing – 12th International Conference (CICLing)*, Tokyo, Japan, pp. 171–89.

Marcus, P., Santorini, B., and Marcinkiewiz, A. 1993. Building a large annotated corpus of English. *Computational Linguistics* **19**(2): 313–30 (MIT Press).

Merialdo, B. 1994. Tagging English text with a probabilistic model. *Computational Linguistics* **20**(2): 155–71, MIT Press.

Nivre, J. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the ACL Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. Workshop at ACL, Barcelona, Spain.

Noord, G. 2007. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, Prague, Czech Republic, pp. 1–10.

Petrov, S. 2010. Products of random latent variable grammars. In *Proceedings of Human Language Technologies Conference – North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, California, USA, pp. 19–27.

Petrov, S., Chang, P., Ringgaard, M., and Alshawi H. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Cambridge, MA, USA, pp. 705–13.

Petrov, S., and Klein, D. 2007. Improved inference for unlexicalized parsing. In *Proceedings of Human Language Technology Conference – North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, New York, USA, pp. 404–11.

Ratnaparkhi, A. 1996. A maximum entropy part of speech tagger. In *Proceedings of the 1996 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, University of Pennsylvania.

Ratnaparkhi, A. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the 1997 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Rhode Island, USA.

Sagae, K., and Lavie, A. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, Vancouver, BC, Canada, pp. 125–32.

Sagae, K., and Lavie, A. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, Sydney, Australia, pp. 691–8.

Sϕgaard, A. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden, pp. 205–8.

Suzuki, J., and Isozaki, H. 2008. Semi-supervised labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, Ohio, USA, pp. 665–73.

Toutanova, K., Klein, D., Manning, C., and Singer, Y. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of Human Language Technology Conference – North Amrican Chapter of the Association for Computational Linguistics (HLT-NAACL)*, Edmonton, Canada, pp. 252–9.

Tsuruoka, Y., Miyao, Y., and Kazama, J. 2011. Learning with lookahead: can history-based models rival globally optimized models? In *Proceedings of the 15th Conference on Computational Natural Language Learning (CoNLL)*, Oregon, USA, pp. 238–46.

Tsuruoka, Y., Tsujii, J., and Ananiadou, S. 2009. Fast full parsing by linear-chain conditional random fields. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Athens, Greece, pp. 790–8.

Wang, W., Huang, Z., and Harper, M. 2007. Semi-supervised learning for part-of-speech tagging of Mandarin transcribed speech. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Hawaii, USA.

Wang, M., Sagae, K., and Mitamura, T. 2006. A fast, accurate deterministic parser for Chinese. In *Proceedings of 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, Sydney, Australia, pp. 425–32.

Xue, N., Xia, F., Chiou, F., and Palmer, M. 2006. The Penn Chinese Treebank: phrase structure annotation of a large corpus. *Natural Language Engineering* **11**(2): 207–38 (Cambridge University Press).

Yamada, H., and Matsumoto, Y. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, Nancy, France, pp. 195–206.

Zhang, Y., and Clark, S. 2011. Shift-reduce CCG parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, Oregon, USA, pp. 683–92.

Zhang, Y., and Clark, S. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proceedings of 11th International Conference on Parsing Technologies (IWPT)*, Paris, France, pp. 162–71.

Zhang, H., Zhang, M., Tan, C., and Li, H. 2009. K-best combination of syntactic parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Singapore, pp. 1552–60.

Zhao, H., Song, Y., Kit C., and Zhou, G. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, Singapore, pp. 55–63.