Bilingually-Constrained (Monolingual) Shift-Reduce Parsing

Liang Huang
Google Research
1350 Charleston Rd.
Mountain View, CA 94043, USA
lianghuang@google.com
liang.huang.sh@gmail.com

Wenbin Jiang and **Qun Liu**
Key Lab. of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
jiangwenbin@ict.ac.cn

Abstract

Jointly parsing two languages has been shown to improve accuracies on either or both sides. However, its search space is much bigger than the monolingual case, forcing existing approaches to employ complicated modeling and crude approximations. Here we propose a much simpler alternative, *bilingually-constrained monolingual parsing*, where a source-language parser learns to exploit reorderings as additional observation, but *not* bothering to build the target-side tree as well. We show specifically how to enhance a shift-reduce dependency parser with alignment features to resolve shift-reduce conflicts. Experiments on the bilingual portion of Chinese Treebank show that, with just 3 bilingual features, we can improve parsing accuracies by 0.6% (absolute) for both English and Chinese over a state-of-the-art baseline, with negligible (∼6%) efficiency overhead, thus much faster than biparsing.

1 Introduction

Ambiguity resolution is a central task in Natural Language Processing. Interestingly, not all languages are ambiguous in the same way. For example, prepositional phrase (PP) attachment is (notoriously) ambiguous in English (and related European languages), but is strictly unambiguous in Chinese and largely unambiguous Japanese; see

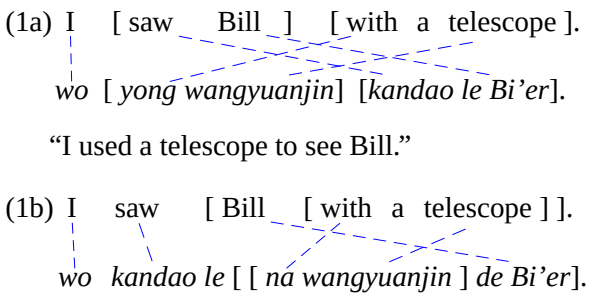

Figure 1: PP-attachment is unambiguous in Chinese, which can help English parsing.

Figure 1 for an example.[1] It is thus intuitive to use two languages for better disambiguation, which has been applied not only to this PP-attachment problem (Fossum and Knight, 2008; Schwartz et al., 2003), but also to the more fundamental problem of syntactic parsing which subsumes the former as a subproblem. For example, Smith and Smith (2004) and Burkett and Klein (2008) show that joint parsing (or reranking) on a bitext improves accuracies on either or both sides by leveraging bilingual constraints, which is very promising for syntax-based machine translation which requires (good-quality) parse trees for rule extraction (Galley et al., 2004; Mi and Huang, 2008).

However, the search space of joint parsing is inevitably much bigger than the monolingual case,

[1] Chinese uses word-order to disambiguate the attachment (see below). By contrast, Japanese resorts to case-markers and the unambiguity is limited: it works for the "V or N" attachment ambiguities like in Figure 1 (see (Schwartz et al., 2003)) but not for the "N_1 or N_2" case (Mitch Marcus, p.c.).

forcing existing approaches to employ complicated modeling and crude approximations. Joint parsing with a simplest synchronous context-free grammar (Wu, 1997) is $O(n^6)$ as opposed to the monolingual $O(n^3)$ time. To make things worse, languages are *non-isomorphic*, i.e., there is no 1-to-1 mapping between tree nodes, thus in practice one has to use more expressive formalisms such as synchronous tree-substitution grammars (Eisner, 2003; Galley et al., 2004). In fact, rather than joint parsing per se, Burkett and Klein (2008) resort to separate monolingual parsing and *bilingual reranking* over k^2 tree pairs, which covers a tiny fraction of the whole space (Huang, 2008).

We instead propose a much simpler alternative, *bilingually-constrained monolingual parsing*, where a source-language parser is extended to exploit the reorderings between languages as additional observation, but *not* bothering to build a tree for the target side simultaneously. To illustrate the idea, suppose we are parsing the sentence

(1) I saw Bill [$_{\text{PP}}$ with a telescope].

which has 2 parses based on the attachment of PP:

(1a) I [saw Bill] [$_{\text{PP}}$ with a telescope].

(1b) I saw [Bill [$_{\text{PP}}$ with a telescope]].

Both are possible, but with a Chinese translation the choice becomes clear (see Figure 1), because a Chinese PP always immediately precedes the phrase it is modifying, thus making PP-attachment strictly unambiguous.[2] We can thus use Chinese to help parse English, i.e., whenever we have a PP-attachment ambiguity, we will consult the Chinese translation (from a bitext), and based on the alignment information, decide where to attach the English PP. On the other hand, English can help Chinese parsing as well, for example in deciding the scope of relative clauses which is unambiguous in English but ambiguous in Chinese.

This method is much simpler than joint parsing because it remains *monolingual* in the backbone, with alignment information merely as soft evidence, rather than hard constraints since automatic word alignment is far from perfect. It is thus straightforward to implement within a monolingual parsing algorithm. In this work we choose shift-reduce dependency parsing for its simplicity and efficiency. Specifically, we make the following contributions:

- we develop a baseline shift-reduce dependency parser using the less popular, but classical, "arc-standard" style (Section 2), and achieve similar state-of-the-art performance with the the dominant but complicated "arc-eager" style of Nivre and Scholz (2004);
- we propose bilingual features based on word-alignment information to prefer "target-side contiguity" in resolving shift-reduce conflicts (Section 3);
- we verify empirically that shift-reduce conflicts are the major source of errors, and correct shift-reduce decisions strongly correlate with the above bilingual contiguity conditions even with automatic alignments (Section 5.3);
- finally, with just three bilingual features, we improve dependency parsing accuracy by 0.6% for both English and Chinese over the state-of-the-art baseline (Section 5.4).

2 Simpler Shift-Reduce Dependency Parsing with Three Actions

The basic idea of classical shift-reduce parsing from compiler theory (Aho and Ullman, 1972) is to perform a left-to-right scan of the input sentence, and at each step, choose one of the two actions: either *shift* the current word onto the stack, or *reduce* the top two (or more) items on the stack, replacing them with their combination. This idea has been applied to constituency parsing, for example in Sagae and Lavie (2006), and we describe below a simple variant for dependency parsing similar to Yamada and Matsumoto (2003) and the "arc-standard" version of Nivre (2004).

2.1 The Three Actions

Basically, we just need to split the reduce action into two symmetric (sub-)actions, reduce$_{\text{L}}$ and reduce$_{\text{R}}$, depending on which one of the two

[2] to be precise, in Fig. 1(b), the English PP is translated into a Chinese relative clause, but nevertheless all phrasal modifiers attach to the immediate right in Mandarin Chinese.

	stack	queue	arcs
previous	S	$w_i \vert Q$	A
shift	$S \vert w_i$	Q	A
previous	$S \vert s_{t-1} \vert s_t$	Q	A
reduce_L	$S \vert s_t$	Q	$A \cup \{(s_t, s_{t-1})\}$
reduce_R	$S \vert s_{t-1}$	Q	$A \cup \{(s_{t-1}, s_t)\}$

Table 1: Formal description of the three actions. Note that shift requires non-empty queue while reduce requires at least two elements on the stack.

items becomes the head after reduction. More formally, we describe a parser configuration by a tuple $\langle S, Q, A \rangle$ where S is the stack, Q is the queue of remaining words of the input, and A is the set of dependency arcs accumulated so far.[3] At each step, we can choose one of the three actions:

1. shift: move the head of (a non-empty) queue Q onto stack S;

2. reduce_L: combine the top two items on the stack, s_t and s_{t-1} ($t \geq 2$), and replace them with s_t (as the head), and add a left arc (s_t, s_{t-1}) to A;

3. reduce_R: combine the top two items on the stack, s_t and s_{t-1} ($t \geq 2$), and replace them with s_{t-1} (as the head), and add a right arc (s_{t-1}, s_t) to A.

These actions are summarized in Table 1. The initial configuration is always $\langle \emptyset, w_1 \ldots w_n, \emptyset \rangle$ with empty stack and no arcs, and the final configuration is $\langle w_j, \emptyset, A \rangle$ where w_j is recognized as the root of the whole sentence, and A encodes a spanning tree rooted at w_j. For a sentence of n words, there are exactly $2n - 1$ actions: n shifts and $n - 1$ reductions, since every word must be pushed onto stack once, and every word except the root will eventually be popped in a reduction. The time complexity, as other shift-reduce instances, is clearly $O(n)$.

2.2 Example of Shift-Reduce Conflict

Figure 2 shows the trace of this paradigm on the example sentence. For the first two configurations

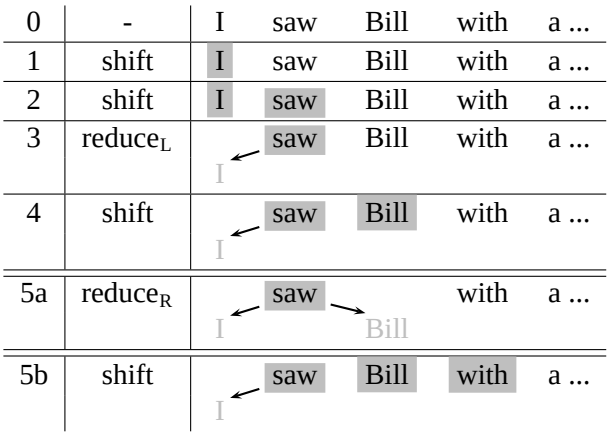

Figure 2: A trace of 3-action shift-reduce on the example sentence. Shaded words are on stack, while gray words have been popped from stack. After step (4), the process can take either (5a) or (5b), which correspond to the two attachments (1a) and (1b) in Figure 1, respectively.

(0) and (1), only shift is possible since there are not enough items on the stack for reduction. At step (3), we perform a reduce_L, making word "I" a modifier of "saw"; after that the stack contains a single word and we have to shift the next word "Bill" (step 4). Now we face a *shift-reduce conflict*: we can either combine "saw" and "Bill" in a reduce_R action (5a), or shift "Bill" (5b). We will use features extracted from the configuration to resolve the conflict. For example, one such feature could be a bigram $s_t \circ s_{t-1}$, capturing how likely these two words are combined; see Table 2 for the complete list of feature templates we use in this baseline parser.

We argue that this kind of shift-reduce conflicts are the major source of parsing errors, since the other type of conflict, reduce-reduce conflict (i.e., whether left or right) is relatively easier to resolve given the part-of-speech information. For example, between a noun and an adjective, the former is much more likely to be the head (and so is a verb vs. a preposition or an adverb). Shift-reduce resolution, however, is more non-local, and often involves a triple, for example, (saw, Bill, with) for a typical PP-attachment. On the other hand, if we indeed make a wrong decision, a reduce-reduce mistake just flips the head and the modifier, and often has a more local effect on the shape of the tree, whereas a shift-reduce mistake always leads

[3] a "configuration" is sometimes called a "state" (Zhang and Clark, 2008), but that term is confusing with the states in shift-reduce LR/LL parsing, which are quite different.

Type	Features		
Unigram	s_t	$T(s_t)$	$s_t \circ T(s_t)$
	s_{t-1}	$T(s_{t-1})$	$s_{t-1} \circ T(s_{t-1})$
	w_i	$T(w_i)$	$w_i \circ T(w_i)$
Bigram	$s_t \circ s_{t-1}$	$T(s_t) \circ T(s_{t-1})$	$T(s_t) \circ T(w_i)$
	$T(s_t) \circ s_{t-1} \circ T(s_{t-1})$	$s_t \circ s_{t-1} \circ T(s_{t-1})$	$s_t \circ T(s_t) \circ T(s_{t-1})$
	$s_t \circ T(s_t) \circ s_{t-1}$	$s_t \circ T(s_t) \circ s_{t-1} \circ T(s_{t-1})$	
Trigram	$T(s_t) \circ T(w_i) \circ T(w_{i+1})$	$T(s_{t-1}) \circ T(s_t) \circ T(w_i)$	$T(s_{t-2}) \circ T(s_{t-1}) \circ T(s_t)$
	$s_t \circ T(w_i) \circ T(w_{i+1})$	$T(s_{t-1}) \circ s_t \circ T(w_i)$	
Modifier	$T(s_{t-1}) \circ T(lc(s_{t-1})) \circ T(s_t)$	$T(s_{t-1}) \circ T(rc(s_{t-1})) \circ T(s_t)$	$T(s_{t-1}) \circ T(s_t) \circ T(lc(s_t))$
	$T(s_{t-1}) \circ T(s_t) \circ T(rc(s_t))$	$T(s_{t-1}) \circ T(lc(s_{t-1})) \circ s_t$	$T(s_{t-1}) \circ T(rc(s_{t-1})) \circ s_t$
	$T(s_{t-1}) \circ s_t \circ T(lc(s_t))$		

Table 2: Feature templates of the baseline parser. s_t, s_{t-1} denote the top and next to top words on the stack; w_i and w_{i+1} denote the current and next words on the queue. $T(\cdot)$ denotes the POS tag of a given word, and $lc(\cdot)$ and $rc(\cdot)$ represent the leftmost and rightmost child. Symbol $\circ$ denotes feature conjunction. Each of these templates is further conjoined with the 3 actions shift, reduce$_{\text{L}}$, and reduce$_{\text{R}}$.

to vastly incompatible tree shapes with crossing brackets (for example, [saw Bill] vs. [Bill with a telescope]). We will see in Section 5.3 that this is indeed the case in practice, thus suggesting us to focus on shift-reduce resolution, which we will return to with the help of bilingual constraints in Section 3.

2.3 Comparison with Arc-Eager

The three action system was originally described by Yamada and Matsumoto (2003) (although their methods require multiple passes over the input), and then appeared as "arc-standard" in Nivre (2004), but was argued against in comparison to the four-action "arc-eager" variant. Most subsequent works on shift-reduce or "transition-based" dependency parsing followed "arc-eager" (Nivre and Scholz, 2004; Zhang and Clark, 2008), which now becomes the dominant style. But we argue that "arc-standard" is preferable because:

1. in the three action "arc-standard" system, the stack always contains a list of *unrelated* subtrees recognized so far, with no arcs between any of them, e.g. (I← saw) and (Bill) in step 4 of Figure 2), whereas the four action "arc-eager" style can have left or right arrows between items on the stack;

2. the semantics of the three actions are atomic and disjoint, whereas the semantics of 4 actions are *not* completely disjoint. For example, their Left action assumes an implicit Reduce of the left item, and their Right action assumes an implicit Shift. Furthermore, these two actions have non-trivial preconditions which also causes the next problem (see below). We argue that this is rather complicated to implement.

3. the "arc-standard" scan always succeeds, since at the end we can always reduce with empty queue, whereas the "arc-eager" style sometimes goes into deadends where no action can perform (prevented by preconditions, otherwise the result will not be a well-formed tree). This becomes parsing failures in practice (Nivre and Scholz, 2004), leaving more than one fragments on stack.

As we will see in Section 5.1, this simpler arc-standard system performs equally well with a state-of-the-art arc-eager system (Zhang and Clark, 2008) on standard English Treebank parsing (which is never shown before). We argue that all things being equal, this simpler paradigm should be preferred in practice. [4]

2.4 Beam Search Extension

We also enhance deterministic shift-reduce parsing with beam search, similar to Zhang and Clark (2008), where k configurations develop in parallel. Pseudocode 1 illustrates the algorithm, where we keep an agenda $\mathbf{V}$ of the current active configurations, and at each step try to extend them by applying one of the three actions. We then dump the best k new configurations from the buffer back

[4] On the other hand, there are also arguments for "arc-eager", e.g., "incrementality"; see (Nivre, 2004; Nivre, 2008).

Pseudocode 1 beam-search shift-reduce parsing.

Input: POS-tagged word sequence $w_1 \dots w_n$
$start \leftarrow \langle \emptyset, w_1 \dots w_n, \emptyset \rangle$ ▷ initial config: empty stack, no arcs
$\mathbf{V} \leftarrow \{start\}$ ▷ initial agenda
for $step \leftarrow 1 \dots 2n-1$ **do**
  $\mathbf{BUF} \leftarrow \emptyset$ ▷ buffer for new configs
  for each *config* in agenda $\mathbf{V}$ **do**
    for $act \in \{\text{shift}, \text{reduce}_\text{L}, \text{reduce}_\text{R}\}$ **do**
      if act is applicable to *config* **then**
        $next \leftarrow$ apply act to *config*
        insert $next$ into buffer $\mathbf{BUF}$
  $\mathbf{V} \leftarrow$ top k configurations of $\mathbf{BUF}$
Output: the tree of the best config in $\mathbf{V}$

into the agenda for the next step. The complexity of this algorithm is $O(nk)$, which subsumes the determinstic mode as a special case ($k = 1$).

2.5 Online Training

To train the parser we need an "oracle" or gold-standard action sequence for gold-standard dependency trees. This oracle turns out to be *non-unique* for the three-action system (also non-unique for the four-action system), because left dependents of a head can be reduced either before or after all right dependents are reduced. For example, in Figure 2, "I" is a left dependent of "saw", and can in principle wait until "Bill" and "with" are reduced, and then finally combine with "saw". We choose to use the heuristic of "shortest stack" that always prefers reduce_L over shift, which has the effect that all left dependents are first recognized inside-out, followed by all right dependents, also inside-out, which coincides with the head-driven constituency parsing model of Collins (1999).

We use the popular online learning algorithm of structured perceptron with parameter averaging (Collins, 2002). Following Collins and Roark (2004) we also use the "early-update" strategy, where an update happens whenever the gold-standard action-sequence falls off the beam, with the rest of the sequence neglected. As a special case, for the deterministic mode, updates always co-occur with the first mistake made. The intuition behind this strategy is that future mistakes are often caused by previous ones, so with the parser on the wrong track, future actions become irrelevant for learning. See Section 5.3 for more discussions.

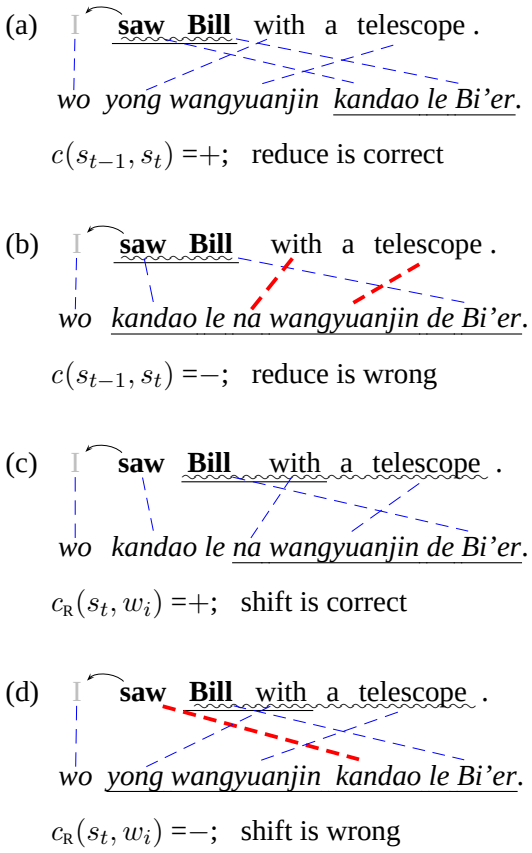

Figure 3: Bilingual contiguity features $c(s_{t-1}, s_t)$ and $c_\text{R}(s_t, w_i)$ at step (4) in Fig. 2 (facing a shift-reduce decision). Bold words are currently on stack while gray ones have been popped. Here the stack tops are s_t = **Bill**, s_{t-1} = **saw**, and the queue head is w_i = with; underlined texts mark the source and target spans being considered, and wavy underlines mark the *allowed spans* (Tab. 3). Red bold alignment links violate contiguity constraints.

3 Soft Bilingual Constraints as Features

As suggested in Section 2.2, shift-reduce conflicts are the central problem we need to address here. Our intuition is, whenever we face a decision whether to combine the stack tops s_{t-1} and s_t or to shift the current word w_i, we will consult the other language, where the word-alignment information would hopefully provide a preference, as in the running example of PP-attachment (see Figure 1). We now develop this idea into *bilingual contiguity features*.

3.1 A Pro-Reduce Feature $c(s_{t-1}, s_t)$

Informally, if the correct decision is a reduction, then it is likely that the corresponding words of s_{t-1} and s_t on the target-side should also form a contiguous span. For example, in Figure 3(a), the source span of a reduction is [saw .. Bill], which maps onto [*kandao* ... *Bi'er*] on the Chinese side. This target span is contiguous, because no word within this span is aligned to a source word outside of the source span. In this case we say feature $c(s_{t-1}, s_t)$ =+, which encourages "reduce".

However, in Figure 3(b), the source span is still [saw .. Bill], but this time maps onto a much longer span on the Chinese side. This target span is discontiguous, since the Chinese words *na* and *wangyuanjin* are alinged to English "with" and "telescope", both of which fall outside of the source span. In this case we say feature $c(s_{t-1}, s_t)$ =−, which discourages "reduce" .

3.2 A Pro-Shift Feature $c_{\text{R}}(s_t, w_i)$

Similarly, we can develop another feature $c_{\text{R}}(s_t, w_i)$ for the shift action. In Figure 3(c), when considering shifting "with", the source span becomes [Bill .. with] which maps to [*na .. Bi'er*] on the Chinese side. This target span looks like discontiguous in the above definition with *wangyuanjin* aligned to "telescope", but we tolerate this case for the following reasons. There is a crucial difference between shift and reduce: in a shift, we do not know yet the subtree spans (unlike in a reduce we are always combining two well-formed subtrees). The only thing we are sure of in a shift action is that s_t and w_i will be combined *before* s_{t-1} and s_t are combined (Aho and Ullman, 1972), so we can tolerate any target word aligned to source word still in the queue, but do not allow any target word aligned to an already recognized source word. This explains the notational difference between $c_{\text{R}}(s_t, w_i)$ and $c(s_{t-1}, s_t)$, where subscript "R" means "right contiguity".

As a final example, in Figure 3(d), Chinese word *kandao* aligns to "saw", which is already recognized, and this violates the right contiguity. So $c_{\text{R}}(s_t, w_i)$ =−, suggesting that shift is probably wrong. To be more precise, Table 3 shows the formal definitions of the two features. We basically map a source span sp to its target span $M(sp)$, and check whether its reverse image back onto the source language $M^{-1}(M(sp))$ falls inside the allowed span ap. For $c_{\text{R}}(s_t, w_i)$, the allowed span extends to the right end of the sentence.[5]

feature f	source span sp	target span tp	allowed span ap
$c(s_{t-1}, s_t)$	$[s_{t-1}..s_t]$	$M(sp)$	$[s_{t-1}..s_t]$
$c_{\text{R}}(s_t, w_i)$	$[s_t..w_i]$	$M(sp)$	$[s_t..w_n]$
$f = +$	iff. $M^{-1}(M(sp)) \subseteq ap$		

Table 3: Formal definition of bilingual features. $M(\cdot)$ is maps a source span to the target language, and $M^{-1}(\cdot)$ is the reverse operation mapping back to the source language.

3.3 Variations and Implementation

To conclude so far, we have got two alignment-based features, $c(s_{t-1}, s_t)$ correlating with reduce, and $c_{\text{R}}(s_t, w_i)$ correlating with shift. In fact, the conjunction of these two features,

$$c(s_{t-1}, s_t) \circ c_{\text{R}}(s_t, w_i)$$

is another feature with even stronger discrimination power. If

$$c(s_{t-1}, s_t) \circ c_{\text{R}}(s_t, w_i) = + \circ -$$

it is strongly recommending reduce, while

$$c(s_{t-1}, s_t) \circ c_{\text{R}}(s_t, w_i) = - \circ +$$

is a very strong signal for shift. So in total we got three bilingual feature (templates), which in practice amounts to 24 instances (after cross-product with $\{-, +\}$ and the three actions). We show in Section 5.3 that these features do correlate with the correct shift/reduce actions in practice.

The naive implemention of bilingual feature computation would be of $O(kn^2)$ complexity in the worse case because when combining the largest spans one has to scan over the whole sentence. We envision the use of a clever datastructure would reduce the complexity, but leave this to future work, as the experiments (Table 8) show that

[5] Our definition implies that we only consider *faithful* spans to be contiguous (Galley et al., 2004). Also note that source spans include all dependents of s_t and s_{t-1}.

the parser is only marginally (∼6%) slower with the new bilingual features. This is because the extra work, with just 3 bilingual features, is not the bottleneck in practice, since the extraction of the vast amount of other features in Table 2 dominates the computation.

4 Related Work in Grammar Induction

Besides those cited in Section 1, there are some other related work on using bilingual constraints for grammar induction (rather than parsing). For example, Hwa et al. (2005) use simple heuristics to project English trees to Spanish and Chinese, but get discouraging accuracy results learned from those projected trees. Following this idea, Ganchev et al. (2009) and Smith and Eisner (2009) use constrained EM and parser adaptation techniques, respectively, to perform more principled projection, and both achieve encouraging results.

Our work, by constrast, never uses bilingual tree pairs not tree projections, and only uses word alignment alone to enhance a monolingual grammar, which learns to prefer target-side contiguity.

5 Experiments

5.1 Baseline Parser

We implement our baseline monolingual parser (in C++) based on the shift-reduce algorithm in Section 2, with feature templates from Table 2. We evaluate its performance on the standard Penn English Treebank (PTB) dependency parsing task, i.e., train on sections 02-21 and test on section 23 with automatically assigned POS tags (at 97.2% accuracy) using a tagger similar to Collins (2002), and using the headrules of Yamada and Matsumoto (2003) for conversion into dependency trees. We use section 22 as dev set to determine the optimal number of iterations in perceptron training. Table 4 compares our baseline against the state-of-the-art graph-based (McDonald et al., 2005) and transition-based (Zhang and Clark, 2008) approaches, and confirms that our system performs at the same level with those state-of-the-art, and runs extremely fast in the deterministic mode (k=1), and still quite fast in the beam-search mode (k=16).

parser	accuracy	secs/sent
McDonald et al. (2005)	90.7	0.150
Zhang and Clark (2008)	91.4	0.195
our baseline at k=1	90.2	0.009
our baseline at k=16	91.3	0.125

Table 4: Baseline parser performance on standard Penn English Treebank dependency parsing task. The speed numbers are not exactly comparable since they are reported on different machines.

	Training	Dev	Test
CTB Articles	1-270	301-325	271-300
Bilingual Paris	2745	273	290

Table 5: Training, dev, and test sets from bilingual Chinese Treebank à la Burkett and Klein (2008).

5.2 Bilingual Data

The bilingual data we use is the translated portion of the Penn Chinese Treebank (CTB) (Xue et al., 2002), corresponding to articles 1-325 of PTB, which have English translations with gold-standard parse trees (Bies et al., 2007). Table 5 shows the split of this data into training, development, and test subsets according to Burkett and Klein (2008). Note that not all sentence pairs could be included, since many of them are not one-to-one aligned at the sentence level. Our word-alignments are generated from the HMM aligner of Liang et al. (2006) trained on approximately 1.7M sentence pairs (provided to us by David Burkett, p.c.). This aligner outputs "soft alignments", i.e., posterior probabilities for each source-target word pair. We use a pruning threshold of 0.535 to remove low-confidence alignment links,[6] and use the remaining links as hard alignments; we leave the use of alignment probabilities to future work.

For simplicity reasons, in the following experiments we always supply gold-standard POS tags as part of the input to the parser.

5.3 Testing our Hypotheses

Before evaluating our bilingual approach, we need to verify empirically the two assumptions we made about the parser in Sections 2 and 3:

[6] and also removing notoriously bad links in $\{\text{the}, \text{a}, \text{an}\} \times \{de, le\}$ following Fossum and Knight (2008).

	sh ▷ re	re ▷ sh	sh-re	re-re
#	92	98	190	7
%	46.7%	49.7%	**96.4%**	**3.6%**

Table 6: [Hypothesis 1] Error distribution in the baseline model ($k = 1$) on English dev set. "sh ▷ re" means "should shift, but reduced". Shift-reduce conflicts overwhelmingly dominate.

1. (monolingual) shift-reduce conflict is the major source of errors while reduce-reduce conflict is a minor issue;

2. (bilingual) the gold-standard decisions of shift or reduce should correlate with contiguities of $c(s_{t-1}, s_t)$, and of $c_{\text{R}}(s_t, w_i)$.

Hypothesis 1 is verified in Table 6, where we count all the *first mistakes* the baseline parser makes (in the deterministic mode) on the English dev set (273 sentences). In shift-reduce parsing, further mistakes are often caused by previous ones, so only the first mistake in each sentence (if there is one) is easily identifiable;[7] this is also the argument for "early update" in applying perceptron learning to these incremental parsing algorithms (Collins and Roark, 2004) (see also Section 2). Among the 197 first mistakes (other 76 sentences have perfect output), the vast majority, 190 of them (96.4%), are shift-reduce errors (equally distributed between shift-becomes-reduce and reduce-becomes-shift), and only 7 (3.6%) are due to reduce-reduce conflicts.[8] These statistics strongly confirm our intuition that shift-reduce decision is much harder to make during parsing, and contributes to the overwhelming majority of errors, which is the topic of the next hypothesis.

Hypothesis 2 is verified in Table 7. We take the gold-standard shift-reduce sequence on the English dev set, and classify them into the four categories based on bilingual contiguity features: (a) $c(s_{t-1}, s_t)$, i.e. whether the top 2 spans on stack

$c(s_{t-1}, s_t)$	$c_{\text{R}}(s_t, w_i)$	shift		reduce
$+$	$-$	172	$\ll$	1,209
$-$	$+$	1,432	$>$	805
$+$	$+$	4,430	$\sim$	3,696
$-$	$-$	525	$\sim$	576
total		6,559	$=$	6,286

Table 7: [Hyp. 2] Correlation of *gold-standard* shift/reduce decisions with bilingual contiguity conditions (on English dev set). Note there is always one more shift than reduce in each sentence.

is contiguous, and (b) $c_{\text{R}}(s_t, w_i)$, i.e. whether the stack top is contiguous with the current word w_i. According to discussions in Section 3, when (a) is contiguous and (b) is not, it is a clear signal for reduce (to combine the top two elements on the stack) rather than shift, and is strongly supported by the data (first line: 1209 reduces vs. 172 shifts); and while when (b) is contiguous and (a) is not, it should suggest shift (combining s_t and w_i before s_{t-1} and s_t are combined) rather than reduce, and is mildly supported by the data (second line: 1432 shifts vs. 805 reduces). When (a) and (b) are both contiguous or both discontiguous, it should be considered a neutral signal, and is also consistent with the data (next two lines). So to conclude, this bilingual hypothesis is empirically justified.

On the other hand, we would like to note that these correlations are done with *automatic* word alignments (in our case, from the Berkeley aligner) which can be quite noisy. We suspect (and will finish in the future work) that using *manual* alignments would result in a better correlation, though for the main parsing results (see below) we can only afford automatic alignments in order for our approach to be widely applicable to *any* bitext.

5.4 Results

We incorporate the three bilingual features (again, with automatic alignments) into the baseline parser, retrain it, and test its performance on the English dev set, with varying beam size. Table 8 shows that bilingual constraints help more with larger beams, from almost no improvement with the deterministic mode (k=1) to +0.5% better with the largest beam (k=16). This could be explained by the fact that beam-search is more robust than

[7] to be really precise one can define "*independent mistakes*" as those not affected by previous ones, i.e., errors made after the parser *recovers* from previous mistakes; but this is much more involved and we leave it to future work.

[8] Note that shift-reduce errors include those due to the non-uniqueness of oracle, i.e., between some reduce_{L} and shift. Currently we are unable to identify "genuine" errors that would result in an incorrect parse. See also Section 2.5.

	baseline		+bilingual	
k	accuracy	time (s)	accuracy	time (s)
1	84.58	0.011	84.67	0.012
2	85.30	0.025	85.62	0.028
4	85.42	0.040	85.81	0.044
8	85.50	0.081	85.95	0.085
16	85.57	0.158	86.07	0.168

Table 8: Effects of beam size k on efficiency and accuracy (on English dev set). Time is average per sentence (in secs). Bilingual constraints show more improvement with larger beams, with a fractional efficiency overhead over the baseline.

	English	Chinese
monolingual baseline	86.9	85.7
+bilingual features	**87.5**	**86.3**
improvement	+0.6	+0.6
signficance level	$p < 0.05$	$p < 0.08$
Berkeley parser	86.1	87.9

Table 9: Final results of dependency accuracy (%) on the test set (290 sentences, beam size k=16).

the deterministic mode, where in the latter, if our bilingual features misled the parser into a mistake, there is no chance of getting back, while in the former multiple configurations are being pursued in parallel. In terms of speed, both parsers run proportionally slower with larger beams, as the time complexity is linear to the beam-size. Computing the bilingual features further slows it down, but only fractionally so (just 1.06 times as slow as the baseline at k=16), which is appealing in practice. By contrast, Burkett and Klein (2008) reported their approach of "monolingual k-best parsing followed by bilingual k^2-best reranking" to be "3.8 times slower" than monolingual parsing.

Our final results on the test set (290 sentences) are summarized in Table 9. On both English and Chinese, the addition of bilingual features improves dependency arc accuracies by +0.6%, which is mildly significant using the Z-test of Collins et al. (2005). We also compare our results against the Berkeley parser (Petrov and Klein, 2007) as a reference system, with the exact same setting (i.e., trained on the bilingual data, and testing using gold-standard POS tags), and the resulting trees are converted into dependency via the same headrules. We use 5 iterations of split-merge grammar induction as the 6th iteration overfits the small training set. The result is worse than our baseline on English, but better than our bilingual parser on Chinese. The discrepancy between English and Chinese is probably due to the fact that our baseline feature templates (Table 2) are engineered on English not Chinese.

6 Conclusion and Future Work

We have presented a novel parsing paradigm, *bilingually-constrained monolingual parsing*, which is much simpler than joint (bi-)parsing, yet still yields mild improvements in parsing accuracy in our preliminary experiments. Specifically, we showed a simple method of incorporating alignment features as soft evidence on top of a state-of-the-art shift-reduce dependency parser, which helped better resolve shift-reduce conflicts.

The fact that we managed to do this with only three alignment feature templates is on one hand encouraging, but on the other hand leaving the bilingual feature space largely unexplored. So we are currently engineering more such features, especially by the use of lexicalization and soft alignments (Liang et al., 2006). The influence of alignment quality on parsing improvement is also worth studying. From a linguistics point of view, we would like to see how *linguistics distance* affects this approach, e.g., we suspect English-French would not help each other as much as English-Chinese do; and it would be very interesting to see what types of syntactic ambiguities can be resolved across different language pairs. Furthermore, we believe this bilingual-monolingual approach can easily transfer to shift-reduce constituency parsing (Sagae and Lavie, 2006).

Acknowledgment

We thank the anonymous reviewers for pointing to us references about "arc-standard". We also thank Aravind Joshi and Mitch Marcus for insights on PP attachment, Joakim Nivre for discussions on arc-standard vs. arc-eager, Yang Liu for suggestion to look at manual alignments, and David A. Smith and Jason Eisner for sending us their paper. The second and third authors were supported by Natiaonal Natural Science Foundation

of China, Contracts 60603095 and 60736014, and 863 State Key Project No. 2006AA010108.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume I: Parsing of *Series in Automatic Computation*. Prentice Hall, Englewood Cliffs, New Jersey.

Ann Bies, Martha Palmer, Justin Mott, and Colin Warner. 2007. English chinese translation treebank v1.0. LDC2007T02.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*.

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

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540, Ann Arbor, Michigan, June.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of ACL (poster)*, pages 205–208.

Victoria Fossum and Kevin Knight. 2008. Using bilingual chinese-english word alignments to resolve pp-attachment ambiguity in english. In *Proceedings of AMTA Student Workshop*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*, pages 273–280.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL-IJCNLP*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the ACL: HLT*, Columbus, OH, June.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of COLING-ACL*, Sydney, Australia, July.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd ACL*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP*, Honolulu, Haiwaii.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of COLING*, Geneva.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together. Workshop at ACL-2004*, Barcelona.

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

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of ACL (poster)*.

Lee Schwartz, Takako Aikawa, and Chris Quirk. 2003. Disambiguation of english pp attachment using multilingual aligned data. In *Proceedings of MT Summit IX*.

David A. Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous features. In *Proceedings of EMNLP*.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using english to parse korean. In *Proceedings of EMNLP*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proceedings of COLING*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of EMNLP*.