A Shift-Reduce Parsing Algorithm for Phrase-based String-to-Dependency Translation

Yang Liu
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology
Tsinghua University, Beijing 100084, China
liuyang2011@tsinghua.edu.cn

Abstract

We introduce a shift-reduce parsing algorithm for phrase-based string-to-dependency translation. As the algorithm generates dependency trees for partial translations left-to-right in decoding, it allows for efficient integration of both n-gram and dependency language models. To resolve conflicts in shift-reduce parsing, we propose a maximum entropy model trained on the derivation graph of training data. As our approach combines the merits of phrase-based and string-to-dependency models, it achieves significant improvements over the two baselines on the NIST Chinese-English datasets.

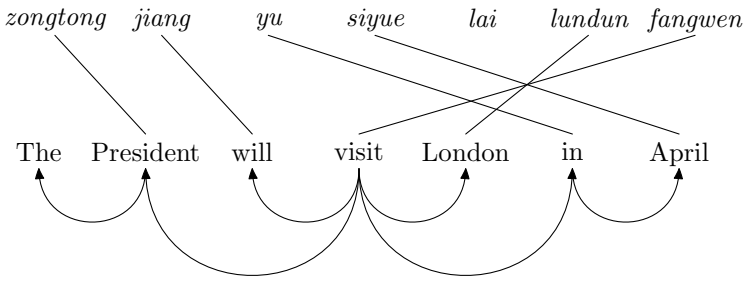
1 Introduction

Modern statistical machine translation approaches can be roughly divided into two broad categories: *phrase-based* and *syntax-based*. Phrase-based approaches treat phrase, which is usually a sequence of consecutive words, as the basic unit of translation (Koehn et al., 2003; Och and Ney, 2004). As phrases are capable of memorizing local context, phrase-based approaches excel at handling local word selection and reordering. In addition, it is straightforward to integrate n-gram language models into phrase-based decoders in which translation always grows left-to-right. As a result, phrase-based decoders only need to maintain the boundary words on one end to calculate language model probabilities. However, as phrase-based decoding usually casts translation as a string concatenation problem and permits arbitrary permutation, it proves to be NP-complete (Knight, 1999).

Syntax-based approaches, on the other hand, model the hierarchical structure of natural languages (Wu, 1997; Yamada and Knight, 2001; Chiang, 2005; Quirk et al., 2005; Galley et al., 2006; Liu et al., 2006; Huang et al., 2006; Shen et al., 2008; Mi and Huang, 2008; Zhang et al., 2008). As syntactic information can be exploited to provide linguistically-motivated reordering rules, predicting non-local permutation is computationally tractable in syntax-based approaches. Unfortunately, as syntax-based decoders often generate target-language words in a bottom-up way using the CKY algorithm, integrating n-gram language models becomes more expensive because they have to maintain target boundary words at both ends of a partial translation (Chiang, 2007; Huang and Chiang, 2007). Moreover, syntax-based approaches often suffer from the rule coverage problem since syntactic constraints rule out a large portion of non-syntactic phrase pairs, which might help decoders generalize well to unseen data (Marcu et al., 2006). Furthermore, the introduction of non-terminals makes the grammar size significantly bigger than phrase tables and leads to higher memory requirement (Chiang, 2007).

As a result, incremental decoding with hierarchical structures has attracted increasing attention in recent years. While some authors try to integrate syntax into phrase-based decoding (Galley and Manning, 2008; Galley and Manning, 2009; Feng et al., 2010), others develop incremental algorithms for syntax-based models (Watanabe et al., 2006; Huang and Mi, 2010; Dyer and Resnik, 2010; Feng et al., 2012). Despite these successful efforts, challenges still remain for both directions. While parsing algorithms can be used to parse partial translations in phrase-based decoding, the search space is significantly enlarged since there are exponentially many parse trees for exponentially many translations. On the other hand, although target words can be generated left-to-right by altering the way of tree transversal in syntax-based models, it is still difficult to reach full rule coverage as compared with phrase table.

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 1–10,
Sofia, Bulgaria, August 4-9 2013.



	source phrase	target phrase	dependency	category
r_1	*fangwen*	visit	$\{\}$	fixed
r_2	*yu siyue*	in April	$\{1 \rightarrow 2\}$	fixed
r_3	*zongtong jiang*	The President will	$\{2 \rightarrow 1\}$	floating left
r_4	*yu siyue lai lundun*	London in April	$\{2 \rightarrow 3\}$	floating right
r_5	*zongtong jiang*	President will	$\{\}$	ill-formed

Figure 1: A training example consisting of a (romanized) Chinese sentence, an English dependency tree, and the word alignment between them. Each translation rule is composed of a source phrase, a target phrase with a set of dependency arcs. Following Shen et al. (2008), we distinguish between *fixed, floating, and ill-formed* structures.

In this paper, we propose a shift-reduce parsing algorithm for phrase-based string-to-dependency translation. The basic unit of translation in our model is *string-to-dependency phrase pair*, which consists of a phrase on the source side and a dependency structure on the target side. The algorithm generates well-formed dependency structures for partial translations left-to-right using string-to-dependency phrase pairs. Therefore, our approach is capable of combining the advantages of both phrase-based and syntax-based approaches:

1. **compact rule table**: our rule table is a subset of the original string-to-dependency grammar (Shen et al., 2008; Shen et al., 2010) by excluding rules with non-terminals.

2. **full rule coverage**: all phrase pairs, both syntactic and non-syntactic, can be used in our algorithm. This is the same with Moses (Koehn et al., 2007).

3. **efficient integration of n-gram language model**: as translation grows left-to-right in our algorithm, integrating n-gram language models is straightforward.

4. **exploiting syntactic information**: as the shift-reduce parsing algorithm generates target language dependency trees in decoding, dependency language models (Shen et al., 2008; Shen et al., 2010) can be used to encourage linguistically-motivated reordering.

5. **resolving local parsing ambiguity**: as dependency trees for phrases are memorized in rules, our approach avoids resolving local parsing ambiguity and explores in a smaller search space than parsing word-by-word on the fly in decoding (Galley and Manning, 2009).

We evaluate our method on the NIST Chinese-English translation datasets. Experiments show that our approach significantly outperforms both phrase-based (Koehn et al., 2007) and string-to-dependency approaches (Shen et al., 2008) in terms of BLEU and TER.

2 Shift-Reduce Parsing for Phrase-based String-to-Dependency Translation

Figure 1 shows a training example consisting of a (romanized) Chinese sentence, an English dependency tree, and the word alignment between them. Following Shen et al. (2008), string-to-dependency rules without non-terminals can be extracted from the training example. As shown in Figure 1, each rule is composed of a source phrase and a target dependency structure. Shen et al. (2008) divide dependency structures into two broad categories:

1. **well-formed**

   (a) **fixed**: the head is known or fixed;

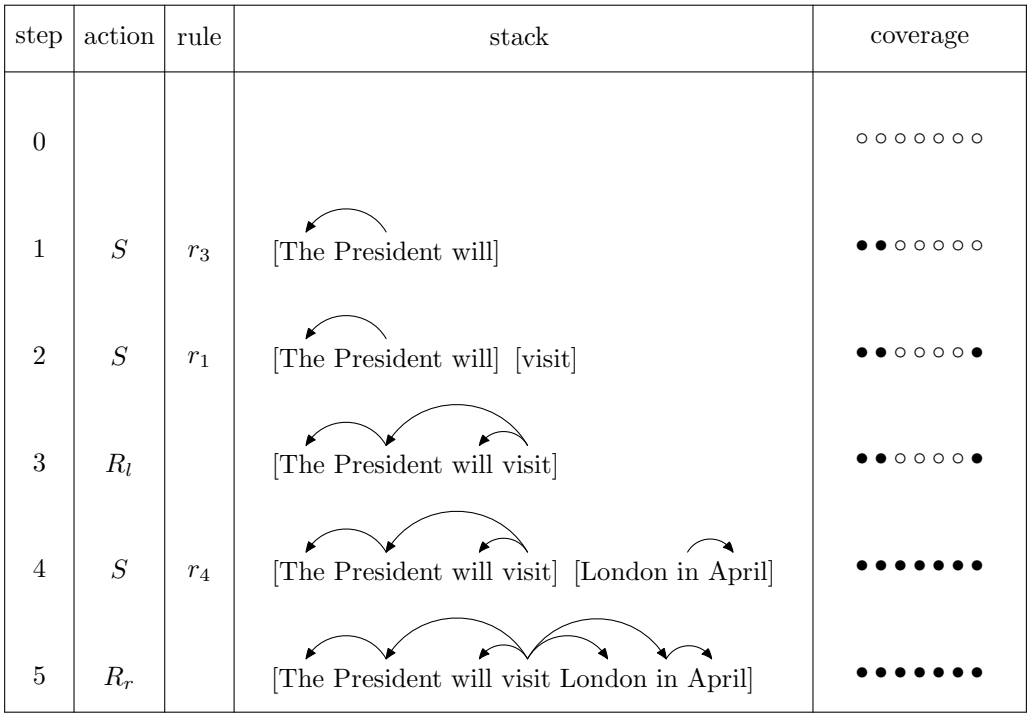


Figure 2: Shift-reduce parsing with string-to-dependency phrase pairs. For each state, the algorithm maintains a stack to store items (i.e., well-formed dependency structures). At each step, it chooses one action to extend a state: shift (S), reduce left (R_l), or reduce right (R_r). The decoding process terminates when all source words are covered and there is a complete dependency tree in the stack.

(b) **floating**: sibling nodes of a common head, but the head itself is unspecified or floating. Each of the siblings must be a complete constituent.

2. **ill-formed**: neither fixed nor floating.

We further distinguish between *left* and *right* floating structures according to the position of head. For example, as "The President will" is the left dependant of its head "visit", it is a left floating structure.

To integrate the advantages of phrase-based and string-to-dependency models, we propose a shift-reduce algorithm for phrase-based string-to-dependency translation.

Figure 2 shows an example. We describe a *state* (i.e., parser configuration) as a tuple $\langle \mathcal{S}, \mathcal{C} \rangle$ where $\mathcal{S}$ is a *stack* that stores *items* and $\mathcal{C}$ is a *coverage vector* that indicates which source words have been translated. Each item $s \in \mathcal{S}$ is a well-formed dependency structure. The algorithm starts with an empty state. At each step, it chooses one of the three actions (Huang et al., 2009) to extend a state:

1. **shift** (S): move a target dependency structure onto the stack;

2. **reduce left** (R_l): combine the two items on the stack, s_t and s_{t-1} ($t \geq 2$), with the root of s_t as the head and replace them with a combined item;

3. **reduce right** (R_r): combine the two items on the stack, s_t and s_{t-1} ($t \geq 2$), with the root of s_{t-1} as the head and replace them with a combined item.

The decoding process terminates when all source words are covered and there is a complete dependency tree in the stack.

Note that unlike monolingual shift-reduce parsers (Nivre, 2004; Zhang and Clark, 2008; Huang et al., 2009), our algorithm does not maintain a queue for remaining words of the input because the future dependency structure to be shifted is unknown in advance in the translation scenario. Instead, we use a coverage vector on the source side to determine when to terminate the algorithm.

For an input sentence of J words, the number of actions is $2K - 1$, where K is the number of rules used in decoding. [1] There are always K shifts and

[1] Empirically, we find that the average number of stacks for J words is about $1.5 \times J$ on the Chinese-English data.

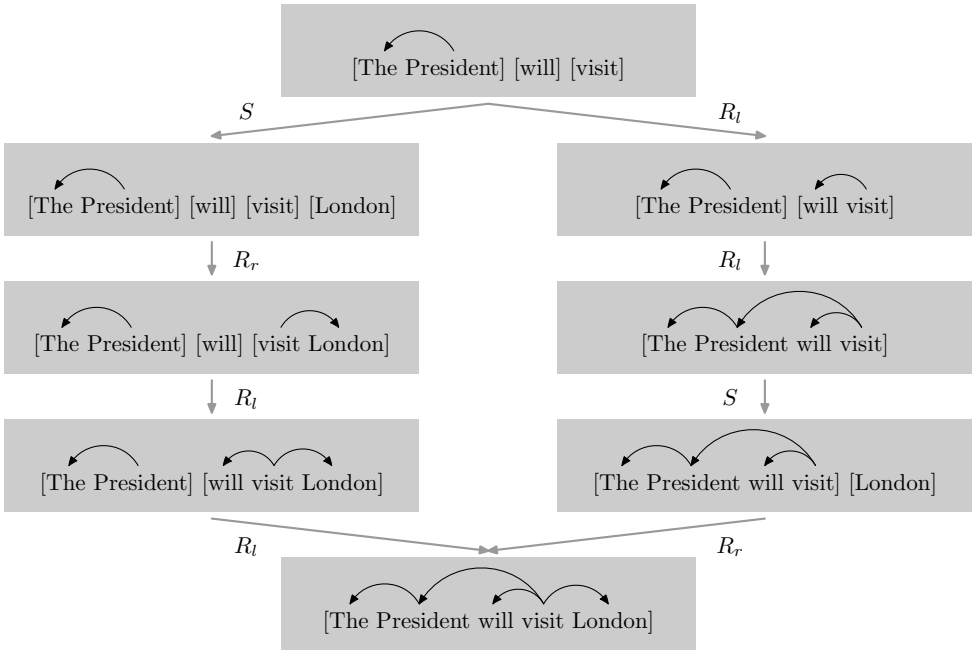


Figure 3: Ambiguity in shift-reduce parsing.

s_{t-1}	s_t	legal	action(s)
		yes	S
	h	yes	S
	l	yes	S
	r	no	
h	h	yes	S, R_l, R_r
h	l	yes	S
h	r	yes	R_r
l	h	yes	R_l
l	l	yes	S
l	r	no	
r	h	no	
r	l	no	
r	r	no	

Table 1: Conflicts in shift-reduce parsing. s_t and s_{t-1} are the top two items in the stack of a state. We use "h" to denote fixed structure, "l" to denote left floating structure, and "r" to denote right floating structure. It is clear that only "h+h" is ambiguous.

$K-1$ reductions.

It is easy to verify that the reduce left and reduce right actions are equivalent to the left adjoining and right adjoining operations defined by Shen et al. (2008). They suffice to operate on well-formed structures and produce projective dependency parse trees.

Therefore, with dependency structures present in the stacks, it is possible to use dependency language models to encourage linguistically plausible phrase reordering.

3 A Maximum Entropy Based Shift-Reduce Parsing Model

Shift-reduce parsing is efficient but suffers from parsing errors caused by syntactic ambiguity. Figure 3 shows two (partial) derivations for a dependency tree. Consider the item on the top, the algorithm can either apply a shift action to move a new item or apply a reduce left action to obtain a bigger structure. This is often referred to as **conflict** in the shift-reduce dependency parsing literature (Huang et al., 2009). In this work, the shift-reduce parser faces four types of conflicts:

1. shift vs. shift;
2. shift vs. reduce left;
3. shift vs. reduce right;
4. reduce left vs. reduce right.

Fortunately, if we distinguish between left and right floating structures, it is possible to rule out most conflicts. Table 1 shows the relationship between conflicts, dependency structures and actions. We use s_t and s_{t-1} to denote the top two

type	feature templates		
Unigram	c	$W_h(s_t)$	$W_h(s_{t-1})$
	$W_{lc}(s_t)$	$W_{rc}(s_{t-1})$	$T_h(s_t)$
	$T_h(s_{t-1})$	$T_{lc}(s_t)$	$T_{rc}(s_{t-1})$
Bigram	$W_h(s_t) \circ W_h(s_{t-1})$	$T_h(S_t) \circ T_h(s_{t-1})$	$W_h(s_t) \circ T_h(s_t)$
	$W_h(s_{t-1}) \circ T_h(s_{t-1})$	$W_h(s_t) \circ W_{rc}(s_{t-1})$	$W_h(s_{t-1}) \circ W_{lc}(s_t)$
Trigram	$c \circ W_h(s_t) \circ W(s_{t-1})$	$c \circ T_h(s_t) \circ T_h(s_{t-1})$	$W_h(s_t) \circ W_h(s_{t-1}) \circ T_{lc}(s_t)$
	$W_h(s_t) \circ W_h(s_{t-1}) \circ T_{rc}(s_{t-1})$	$T_h(s_t) \circ T_h(s_{t-1}) \circ T_{lc}(s_t)$	$T_h(s_t) \circ T_h(s_{t-1}) \circ T_{rc}(s_{t-1})$

Figure 4: Feature templates for maximum entropy based shift-reduce parsing model. c is a boolean value that indicate whether all source words are covered (shift is prohibited if true), $W_h(\cdot)$ and $T_h(\cdot)$ are functions that get the root word and tag of an item, $W_{lc}(\cdot)$ and $T_{lc}(\cdot)$ returns the word and tag of the left most child of the root, $W_{rc}(\cdot)$ amd $T_{rc}(\cdot)$ returns the word and tag of the right most child of the root. Symbol $\circ$ denotes feature conjunction. In this example, $c = true$, $W_h(s_t) =$ in, $T_h(s_t) =$ IN, $W_h(s_{t-1}) =$ visit, $W_{lc}(s_{t-1}) =$ London.

items in the stack. "h" stands for fixed structure, "l" for left floating structure, and "r" for right floating structure. If the stack is empty, the only applicable action is shift. If there is only one item in the stack and the item is either fixed or left floating, the only applicable action is shift. Note that it is illegal to shift a right floating structure onto an empty stack because it will never be reduced. If the stack contains at least two items, only "h+h" is ambiguous and the others are either unambiguous or illegal. Therefore, we only need to focus on how to resolve conflicts for the "h+h" case (i.e., the top two items in a stack are both fixed structures).

We propose a maximum entropy model to resolve the conflicts for "h+h": [2]

$$P_\theta(a|c, s_t, s_{t-1}) = \frac{\exp(\theta \cdot h(a, c, s_t, s_{t-1}))}{\sum_a \exp(\theta \cdot h(a, c, s_t, s_{t-1}))}$$

where $a \in \{S, R_l, R_r\}$ is an action, c is a boolean value that indicates whether all source words are covered (shift is prohibited if true), s_t and s_{t-1} are the top two items on the stack, $h(a, c, s_t, s_{t-1})$ is a vector of binary features and θ is a vector of feature weights.

Figure 4 shows the feature templates used in our experiments. $W_h(\cdot)$ and $T_h(\cdot)$ are functions that get the root word and tag of an item, $W_{lc}(\cdot)$ and $T_{lc}(\cdot)$ returns the word and tag of the left most child of the root, $W_{rc}(\cdot)$ and $T_{rc}(\cdot)$ returns the word and tag of the right most child of the root. In this example, $c = true$, $W_h(s_t) =$ in, $T_h(s_t) =$ IN, $W_h(s_{t-1}) =$ visit, $W_{lc}(s_{t-1}) =$ London.

To train the model, we need an "oracle" or gold-standard action sequence for each training example. Unfortunately, such oracle turns out to be non-unique even for monolingual shift-reduce dependency parsing (Huang et al., 2009). The situation for phrase-based shift-reduce parsing aggravates because there are usually multiple ways of segmenting sentence into phrases.

To alleviate this problem, we introduce a structure called **derivation graph** to compactly represent all derivations of a training example. Figure 3 shows a (partial) derivation graph, in which a node corresponds to a state and an edge corresponds to an action. The graph begins with an empty state and ends with the given training example.

More formally, a derivation graph is a directed acyclic graph $G = \langle V, E \rangle$ where V is a set of nodes and E is a set of edges. Each node v corresponds to a state in the shift-reduce parsing process. There are two distinguished nodes: v_0, the staring empty state, and $v_{|V|}$, the ending completed state. Each edge $e = (a, i, j)$ transits node v_i to node v_j via an action $a \in \{S, R_l, R_r\}$.

To build the derivation graph, our algorithm starts with an empty state and iteratively extends an unprocessed state until reaches the completed state. During the process, states that violate the training example are discarded. Even so, there are still exponentially many states for a training example, especially for long sentences. Fortunately, we

[2]The shift-shift conflicts always exist because there are usually multiple rules that can be shifted. This can be resolved using standard features in phrase-based models.

Algorithm 1 Beam-search shift-reduce parsing.

```
1:  procedure PARSE(f)
2:      V ← ∅
3:      ADD(v_0, V[0])
4:      k ← 0
5:      while V[k] ≠ ∅ do
6:          for all v ∈ V[k] do
7:              for all a ∈ {S, R_l, R_r} do
8:                  EXTEND(f, v, a, V)
9:              end for
10:         end for
11:         k ← k + 1
12:     end while
13: end procedure
```

only need to focus on "h+h" states. In addition, we follow Huang et al. (2009) to use the heuristic of "shortest stack" to always prefer R_l to S.

4 Decoding

Our decoder is based on a linear model (Och, 2003) with the following features:

1. relative frequencies in two directions;
2. lexical weights in two directions;
3. phrase penalty;
4. distance-based reordering model;
5. lexicaized reordering model;
6. n-gram language model model;
7. word penalty;
8. ill-formed structure penalty;
9. dependency language model;
10. maximum entropy parsing model.

In practice, we extend deterministic shift-reduce parsing with beam search (Zhang and Clark, 2008; Huang et al., 2009). As shown in Algorithm 1, the algorithm maintains a list of stacks $\mathcal{V}$ and each stack groups states with the same number of accumulated actions (line 2). The stack list $\mathcal{V}$ initializes with an empty state v_0 (line 3). Then, the states in the stack are iteratively extended until there are no incomplete states (lines 4-12). The search space is constrained by discarding any state that has a score worse than:

1. β multiplied with the best score in the stack, or
2. the score of b-th best state in the stack.

As the stack of a state keeps changing during the decoding process, the context information needed to calculate dependency language model and maximum entropy model probabilities (e.g., root word, leftmost child, etc.) changes dynamically as well. As a result, the chance of risk-free *hypothesis recombination* (Koehn et al., 2003) significantly decreases because complicated contextual information is much less likely to be identical.

Therefore, we use *hypergraph reranking* (Huang and Chiang, 2007; Huang, 2008), which proves to be effective for integrating non-local features into dynamic programming, to alleviate this problem. The decoding process is divided into two passes. In the first pass, only **standard features** (i.e., features 1-7 in the list in the beginning of this section) are used to produce a hypergraph. [3] In the second pass, we use the hypergraph reranking algorithm (Huang, 2008) to find promising translations using additional **dependency features** (i.e., features 8-10 in the list). As hypergraph is capable of storing exponentially many derivations compactly, the negative effect of propagating mistakes made in the first pass to the second pass can be minimized.

To improve rule coverage, we follow Shen et al. (2008) to use ill-formed structures in decoding. If an ill-formed structure has a single root, it can treated as a (pseudo) fixed structure; otherwise it is transformed to one (pseudo) left floating structure and one (pseudo) right floating structure. We use a feature to count how many ill-formed structures are used in decoding.

5 Experiments

We evaluated our phrase-based string-to-dependency translation system on Chinese-English translation. The training data consists of 2.9M pairs of sentences with 76.0M Chinese words and 82.2M English words. We used the Stanford parser (Klein and Manning, 2003) to get dependency trees for English sentences. We used the SRILM toolkit (Stolcke, 2002) to train a

[3] Note that the first pass does not work like a phrase-based decoder because it yields dependency trees on the target side. A uniform model (i.e., each action has a fixed probability of 1/3) is used to resolve "h+h" conflicts.

system	MT02 (tune)		MT03		MT04		MT05	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
phrase	34.88	57.00	33.82	57.19	35.48	56.48	32.52	57.62
dependency	35.23	56.12	34.20	56.36	36.01	55.55	33.06	56.94
this work	35.71**	55.87**	34.81**+	55.94**+	36.37**	55.02**+	33.53**	56.58**

Table 2: Comparison with Moses (Koehn et al., 2007) and a re-implementation of the bottom-up string-to-dependency decoder (Shen et al., 2008) in terms of uncased BLEU and TER. We use randomization test (Riezler and Maxwell, 2005) to calculate statistical significance. *: significantly better than Moses ($p < 0.05$), **: significantly better than Moses ($p < 0.01$), +: significantly better than string-to-dependency ($p < 0.05$), ++: significantly better than string-to-dependency ($p < 0.01$).

features	BLEU	TER
standard	34.79	56.93
+ depLM	35.29*	56.17**
+ maxent	35.40**	56.09**
+ depLM & maxent	35.71**	55.87**

Table 3: Contribution of maximum entropy shift-reduce parsing model. "standard" denotes using standard features of phrase-based system. Adding dependency language model ("depLM") and the maximum entropy shift-reduce parsing model ("maxent") significantly improves BLEU and TER on the development set, both separately and jointly.

rules	coverage	BLEU	TER
well-formed	44.87	34.42	57.35
all	100.00	35.71**	55.87**

Table 4: Comparison of well-formed and ill-formed structures. Using all rules significantly outperforms using only well-formed structures. BLEU and TER scores are calculated on the development set.

4-gram language model on the Xinhua portion of the GIGAWORD coprus, which contians 238M English words. A 3-gram dependency language model was trained on the English dependency trees. We used the 2002 NIST MT Chinese-English dataset as the development set and the 2003-2005 NIST datasets as the testsets. We evaluated translation quality using *uncased* BLEU (Papineni et al., 2002) and TER (Snover et al., 2006). The features were optimized with respect to BLEU using the minimum error rate training algorithm (Och, 2003).

We chose the following two systems that are closest to our work as baselines:

1. The Moses phrase-based decoder (Koehn et al., 2007).

2. A re-implementation of bottom-up string-to-dependency decoder (Shen et al., 2008).

All the three systems share with the same target-side parsed, word-aligned training data. The histogram pruning parameter b is set to 100 and phrase table limit is set to 20 for all the three systems. Moses shares the same feature set with our system except for the dependency features. For the bottom-up string-to-dependency system, we included both well-formed and ill-formed structures in chart parsing. To control the grammar size, we only extracted "tight" initial phrase pairs (i.e., the boundary words of a phrase must be aligned) as suggested by (Chiang, 2007). For our system, we used the Le Zhang's maximum entropy modeling toolkit to train the shift-reduce parsing model after extracting 32.6M events from the training data. [4] We set the iteration limit to 100. The accuracy on the training data is 90.18%.

Table 2 gives the performance of Moses, the bottom-up string-to-dependency system, and our system in terms of uncased BLEU and TER scores. From the same training data, Moses extracted 103M bilingual phrases, the bottom-up string-to-dependency system extracted 587M string-to-dependency rules, and our system extracted 124M phrase-based dependency rules. We find that our approach outperforms both baselines systematically on all testsets. We use randomization test (Riezler and Maxwell, 2005) to calculate statistical significance. As our system can take full advantage of lexicalized reordering and depen-

[4] http://homepages.inf.ed.ac.uk/lzhang10/maxent.html

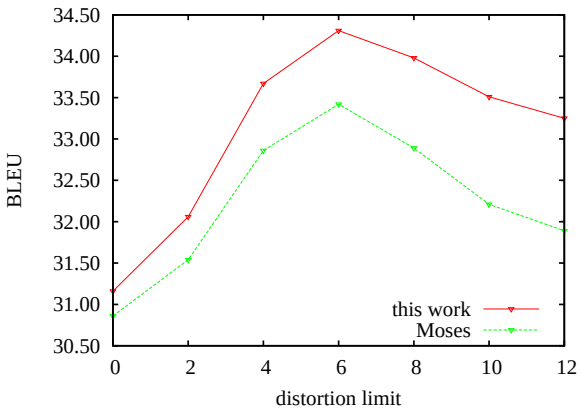


Figure 5: Performance of Moses and our system with various distortion limits.

dency language models without loss in rule coverage, it achieves significantly better results than Moses on all test sets. The gains in TER are much larger than BLEU because dependency language models do not model n-grams directly. Compared with the bottom-up string-to-dependency system, our system outperforms consistently but not significantly in all cases. The average decoding time for Moses is 3.67 seconds per sentence, bottom-up string-to-dependency is 13.89 seconds, and our system is 4.56 seconds.

Table 3 shows the effect of hypergraph reranking. In the first pass, our decoder uses standard phrase-based features to build a hypergraph. The BLEU score is slightly lower than Moses with the same configuration. One possible reason is that our decoder organizes stacks with respect to actions, whereas Moses groups partial translations with the same number of covered source words in stacks. In the second pass, our decoder reranks the hypergraph with additional dependency features. We find that adding dependency language and maximum entropy shift-reduce models consistently brings significant improvements, both separately and jointly.

We analyzed translation rules extracted from the training data. Among them, well-formed structures account for 43.58% (fixed 33.21%, floating left 9.01%, and floating right 1.36%) and ill-formed structures 56.42%. As shown in Table 4, using all rules clearly outperforms using only well-formed structures.

Figure 5 shows the performance of Moses and our system with various distortion limits on the development set. Our system consistently outperforms Moses in all cases, suggesting that adding dependency helps improve phrase reordering.

6 Related Work

The work of Galley and Manning (2009) is closest in spirit to ours. They introduce maximum spanning tree (MST) parsing (McDonald et al., 2005) into phrase-based translation. The system is phrase-based except that an MST parser runs to parse partial translations at the same time. One challenge is that MST parsing itself is not incremental, making it expensive to identify loops during hypothesis expansion. On the contrary, shift-reduce parsing is naturally incremental and can be seamlessly integrated into left-to-right phrase-based decoding. More importantly, in our work dependency trees are memorized for phrases rather than being generated word by word on the fly in decoding. This treatment might not only reduce decoding complexity but also potentially revolve local parsing ambiguity.

Our decoding algorithm is similar to Gimpel and Smith (2011)'s lattice parsing algorithm as we divide decoding into two steps: hypergraph generation and hypergraph rescoring. The major difference is that our hypergraph is not a phrasal lattice because each phrase pair is associated with a dependency structure on the target side. In other words, our second pass is to find the Viterbi derivation with addition features rather than parsing the phrasal lattice. In addition, their algorithm produces phrasal dependency parse trees while the leaves of our dependency trees are words, making dependency language models can be directly used.

Shift-reduce parsing has been successfully used in phrase-based decoding but limited to adding structural constraints. Galley and Manning (2008) propose a shift-reduce algorithm to integrate a hierarchical reordering model into phrase-based systems. Feng et al. (2010) use shift-reduce parsing to impose ITG (Wu, 1997) constraints on phrase permutation. Our work differs from theirs by going further to incorporate linguistic syntax into phrase-based decoding.

Along another line, a number of authors have developed incremental algorithms for syntax-based models (Watanabe et al., 2006; Huang and Mi, 2010; Dyer and Resnik, 2010; Feng et al., 2012). Watanabe et al. (2006) introduce an Early-style top-down parser based on binary-branching Greibach Normal Form. Huang et al. (2010), Dyer

and Resnik (2010), and Feng et al. (2012) use dotted rules to change the tree transversal to generate target words left-to-right, either top-down or bottom-up.

7 Conclusion

We have presented a shift-reduce parsing algorithm for phrase-based string-to-dependency translation. The algorithm generates dependency structures incrementally using string-to-dependency phrase pairs. Therefore, our approach is capable of combining the advantages of both phrase-based and string-to-dependency models, it outperforms the two baselines on Chinese-to-English translation.

In the future, we plan to include more contextual information (e.g., the uncovered source phrases) in the maximum entropy model to resolve conflicts. Another direction is to adapt the dynamic programming algorithm proposed by Huang and Sagae (2010) to improve our string-to-dependency decoder. It is also interesting to compare with applying word-based shift-reduce parsing to phrase-based decoding similar to (Galley and Manning, 2009).

Acknowledgments

This research is supported by the 863 Program under the grant No 2012AA011102 and No. 2011AA01A207, by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, and by a Research Fund No. 20123000007 from Tsinghua MOE-Microsoft Joint Laboratory.

References

David Chiang. 2005. A hiearchical phrase-based model for statistical machine translation. In *Proc. of ACL 2005*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chris Dyer and Philip Resnik. 2010. Context-free reordering, finite-state translation. In *Proc. of NAACL 2010*.

Yang Feng, Haitao Mi, Yang Liu, and Qun Liu. 2010. An efficient shift-reduce decoding algorithm for phrased-based machine translation. In *Proc. of COLING 2010*.

Yang Feng, Yang Liu, Qun Liu, and Trevor Cohn. 2012. Left-to-right tree-to-string decoding with prediction. In *Proc. of EMNLP 2012*.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proc. of EMNLP 2008*.

Michel Galley and Christopher D. Manning. 2009. Quadratic-time dependency parsing for machine translation. In *Proc. of ACL 2009*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of ACL 2006*.

Kevin Gimpel and Noah A. Smith. 2011. Quasi-synchronous phrase dependency grammars for machine translation. In *Proc. of EMNLP 2011*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL 2007*.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proc. of EMNLP 2010*.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proc. of ACL 2010*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA 2006*.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proc. of EMNLP 2009*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL 2008*.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL 2003*.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*.

Philipp Koehn, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL 2003*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL 2007*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL 2006*.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. Spmt: Statistical machine translation with syntactified target language phrases. In *Proc. of EMNLP 2006*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of EMNLP 2005*.

Haitao Mi and Liang Huang. 2008. Forest-based translation. In *Proc. of ACL 2008*.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proc. of ACL 2004 Workshop Incremental Parsing: Bringning Engineering and Cognition Together.*

Franz Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4).

Franz Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL 2003*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL 2002*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proc. of ACL 2005*.

S. Riezler and J. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for mt. In *Proc. of ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL 2008*.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-dependency statistical machine translation. *Computational Linguistics*, 36(4).

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA 2006*.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proc. of ICSLP 2002*.

Taro Watanabe, Hajime Tsukuda, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proc. of ACL 2006*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. of ACL 2001*.

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 *Proc. of EMNLP 2008*.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proc. of ACL 2008*.