Shift-Reduce Word Reordering for Machine Translation

Katsuhiko Hayashi†, Katsuhito Sudoh, Hajime Tsukada, Jun Suzuki, Masaaki Nagata
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
†hayashi.katsuhiko@lab.ntt.co.jp

Abstract

This paper presents a novel word reordering model that employs a shift-reduce parser for inversion transduction grammars. Our model uses rich syntax parsing features for word reordering and runs in linear time. We apply it to postordering of phrase-based machine translation (PBMT) for Japanese-to-English patent tasks. Our experimental results show that our method achieves a significant improvement of +3.1 BLEU scores against 30.15 BLEU scores of the baseline PBMT system.

1 Introduction

Even though phrase-based machine translation (PBMT) (Koehn et al., 2007) and tree-based MT (Graehl and Knight, 2004; Chiang, 2005; Galley et al., 2006) systems have achieved great success, many problems remain for distinct language pairs, including long-distant word reordering.

To improve such word reordering, one promising way is to separate it from the translation process as preordering (Collins et al., 2005; DeNero and Uszkoreit, 2011) or postordering (Sudoh et al., 2011; Goto et al., 2012). Many studies utilize a rule-based or a probabilistic model to perform a reordering decision at each node of a syntactic parse tree.

This paper presents a parser-based word reordering model that employs a shift-reduce parser for inversion transduction grammars (ITG) (Wu, 1997). To the best of our knowledge, this is the first study on a shift-reduce parser for word reordering.

The parser-based reordering approach uses rich syntax parsing features for reordering decisions. Our propoesd method can also easily define such non-local features as the N-gram words of reordered strings. Even when using these non-local features, the complexity of the shift-reduce parser does not increase at all due to give up achieving an optimal solution. Therefore, it works much more efficient.

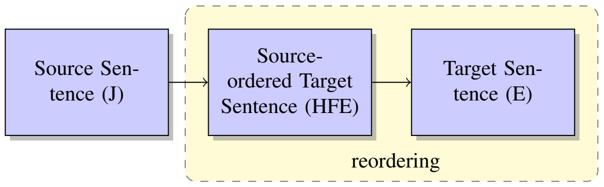

Figure 1: A description of the postordering MT system.

In our experiments, we apply our proposed method to postordering for J-to-E patent tasks because their training data for reordering have little noise and they are ideal for evaluating reordering methods. Although our used J-to-E setups need a language-dependent scheme and we describe our proposed method as a J-to-E postordering method, the key algorithm is language-independent and it can be applicable to preordering as well as postordering if the training data for reordering are available.

2 Postordering by Parsing

As shown in Fig.1, postordering (Sudoh et al., 2011) has two steps; the first is a translation step that translates an input sentence into source-ordered translations. The second is a reordering step in which the translations are reordered in the target language order. The key to postordering is the second step.

Goto et al. (2012) modeled the second step by parsing and created training data for a postordering parser using a language-dependent rule called **head-finalization**. The rule moves syntactic heads of a lexicalized parse tree of an English sentence to the end of the corresponding syntactic constituents. As a result, the terminal symbols of the English tree are sorted in a Japanese-like order. In Fig.2, we show an

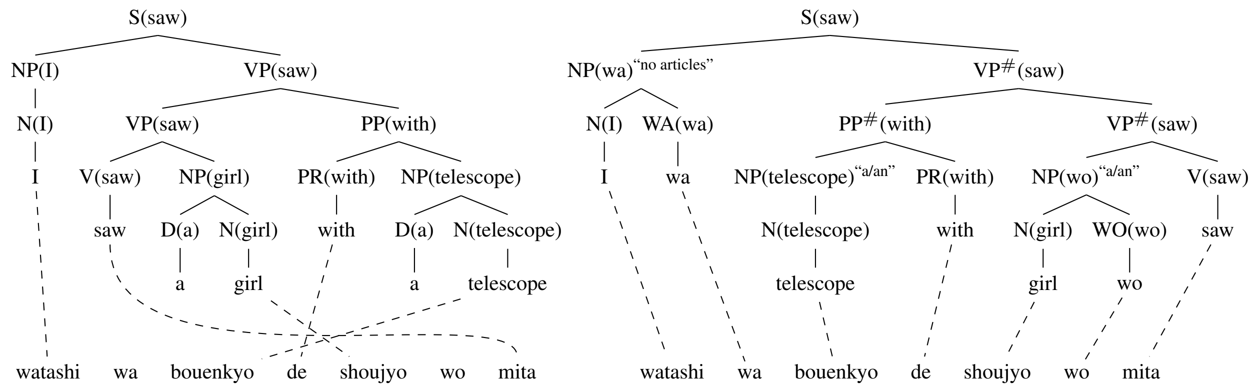

Figure 2: An example of the head-finzalizaton process for an English-Japanese sentence pair: the left-hand side tree is the original English tree, and the right-hand side tree is its head-final English tree.

example of head-finalization and a tree on the right-hand side is a head-finalized English (HFE) tree of an English tree on the left-hand side. We annotate each parent node of the swapped edge with $\#$ symbol. For example, a nonterminal symbol PP$^{\#}$(with) shows that a noun phrase "a/an telescope" and a word "with" are inverted.

For better word alignments, Isozaki et al. (2012) also deleted articles "the" "a" "an" from English because Japanese has no articles, and inserted Japanese particles "ga" "wo" "wa" into English sentences. We privilege the nonterminals of a phrase modified by a deleted article to determine which "the" "a/an" or "no articles" should be inserted at the front of the phrase. Note that an original English sentence can be recovered from its HFE tree by using $\#$ symbols and annotated articles.

As well as Goto et al. (2012), we solve postordering by a parser whose model is trained with a set of HFE trees. The main difference between Goto et al. (2012)'s model and ours is that while the former simply used the Berkeley parser (Petrov and Klein, 2007), our shift-reduce parsing model can use such non-local task specific features as the N-gram words of reordered strings without sacrificing efficiency.

Our method integrates postediting (Knight and Chander, 1994) with reordering and inserts articles into English translations by learning an additional "insert" action of the parser. Goto et al. (2012) solved the article generation problem by using an N-gram language model, but this somewhat complicates their approach. Compared with other parsers, one advantage of the shift-reduce parser is to easily define such additional operations as "insert".

HFE trees can be defined as monolingual ITG trees (DeNero and Uszkoreit, 2011). Our monolingual ITG G is a tuple $G = (V, T, P, I, S)$ where V is a set of nonterminals, T is a set of terminals, P is a set of production rules, I is a set of nonterminals on which "the" "a/an" or "no articles" must be determined, and S is the start symbol.

Set P consists of terminal production rules that are responsible for generating word $w (\in T)$:

$$\mathrm{X} \rightarrow w$$

and binary production rules in two forms:

$$\begin{aligned} \mathrm{X} &\rightarrow \mathrm{Y\,Z} \\ \mathrm{X}^{\#} &\rightarrow \mathrm{Y\,Z} \end{aligned}$$

where X, X$^{\#}$, Y and Z are nonterminals. On the right-hand side, the second rule generates two phrases Y and Z in the reverse order. In our experiments, we removed all unary production rules.

3 Shift-Reduce Parsing

Given an input sentence $w_1 \ldots w_n$, the shift-reduce parser uses a stack of partial derivations, a buffer of input words, and a set of actions to build a parse tree.

The following is the parser's configuration:

$$\ell : \langle i, j, S \rangle : \pi$$

where ℓ is the step size, S is a stack of elements $s_0, s_1, \ldots$, i is the leftmost span index of the stack top element s_0, j is an index of the next input word of the buffer, and π is a set of **predictor states**[1].

[1]Since our notion of predictor states is identical to that in (Huang and Sagae, 2010), we omit the details here.

Each stack element has at least the following components of its partial derivation tree:

$$s = \{\mathrm{H}, h, w_{left}, w_{right}, a\}$$

where H is a root nonterminal or a part-of-speech tag of the subtree, h is a head index of H, a is a variable to which "the" "a/an" "no articles" or null are assigned, and w_{left}, w_{right} are the leftmost and rightmost words of phrase H. When referring to component $*$, we use a $s.*$ notation.

Our proposed system has 4 actions shift-X, insert-x, reduce-MR-X and reduce-SR-X.

The shift-X action pushes the next input word onto the stack and assigns a part-of-speech tag X to the word. The deduction step is as follows:

$$\frac{\begin{array}{c}\mathrm{X} \to w_j \in P \\ \overbrace{\ell : \langle i, j, S|s'_0\rangle : \pi}^{p}\end{array}}{\ell + 1 : \langle j, j+1, S|s'_0|s_0)\rangle : \{p\}}$$

where s_0 is $\{\mathrm{X}, j, w_j, w_j, \mathrm{null}\}$.

The insert-x action determines whether to generate "the" "a/an" or "no articles" ($= x$):

$$\frac{\begin{array}{l}s'_0.\mathrm{X} \in I \wedge (s'_0.a = \text{null or } x \neq \text{"no article"} \\ \wedge s'_0.a \neq \text{"the"} \wedge s'_0.a \neq \text{"a/an"}) \\ \ell : \langle i, j, S|s'_0)\rangle : \pi\end{array}}{\ell + 1 : \langle i, j, S|s_0\rangle : \pi}$$

where s'_0 is $\{\mathrm{X}, h, w_{left}, w_{right}, a\}$ and s_0 is $\{\mathrm{X}, h, w_{left}, w_{right}, x\}$ ($i \leq h, left, right < j$). The side condition prevents the parser from inserting articles into phrase X more than twice. During parsing, articles are not explicitly inserted into the input string: they are inserted into it when backtracking to generate a reordered string after parsing.

The reduce-MR-X action has a deduction rule:

$$\frac{\begin{array}{l}\mathrm{X} \to \mathrm{Y\ Z} \in P \wedge q \in \pi \\ \overbrace{\_ : \langle k, i, S|s'_2|s'_1\rangle : \pi'}^{q} \ \ \ell : \langle i, j, S|s'_1|s'_0\rangle : \pi\end{array}}{\ell + 1 : \langle k, j, S|s'_2|s_0\rangle : \pi'}$$

where s'_0 is $\{\mathrm{Z}, h_0, w_{left0}, w_{right0}, a_0\}$ and s'_1 is $\{\mathrm{Y}, h_1, w_{left1}, w_{right1}, a_1\}$. The action generates s_0 by combining s'_0 and s'_1 with binary rule X→Y Z:

$$s_0 = \{\mathrm{X}, h_0, w_{left1}, w_{right0}, a_1\}.$$

$s_0.w_h \circ s_0.t_h$ $s_0.\mathrm{H}$ $s_0.\mathrm{H} \circ s_0.t_h$ $s_0.w_h \circ s_0.\mathrm{H}$
$s_1.w_h \circ s_1.t_h$ $s_1.\mathrm{H}$ $s_1.\mathrm{H} \circ s_1.t_h$ $s_1.w_h \circ s_1.\mathrm{H}$
$s_2.t_h \circ s_2.\mathrm{H}$ $s_2.w_h \circ s_2.\mathrm{H}$ $q_0.w$ $q_1.w$ $q_2.w$
$s_0.t_l \circ s_0.\mathrm{L}$ $s_0.w_l \circ s_0.\mathrm{L}$ $s_1.t_l \circ s_1.\mathrm{L}$ $s_1.w_l \circ s_1.\mathrm{L}$
$s_0.w_h \circ s_0.\mathrm{H} \circ s_1.w_h \circ s_1.\mathrm{H}$ $s_0.\mathrm{H} \circ s_1.w_h$ $s_0.w_h \circ s_1.\mathrm{H}$
$s_0.\mathrm{H} \circ s_1.\mathrm{H}$ $s_0.w_h \circ s_0.\mathrm{H} \circ q_0.w$ $s_0.\mathrm{H} \circ q_0.w$
$s_1.w_h \circ s_1.\mathrm{H} \circ q_0.w$ $s_1.\mathrm{H} \circ q_0.w$ $s_1.t_h \circ q_0.w \circ q_1.w$
$s_0.w_h \circ s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ q_0.w$ $s_0.\mathrm{H} \circ s_1.w_h \circ s_1.\mathrm{H} \circ q_0.w$
$s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ q_0.w$ $s_0.t_h \circ s_1.t_h \circ q_0.w$
$s_0.w_h \circ s_1.\mathrm{H} \circ q_0.w \circ q_1.w$ $s_0.\mathrm{H} \circ q_0.w \circ q_1.w$
$s_0.t_h \circ q_0.w \circ q_1.w$ $s_0.w_h \circ s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ s_2.\mathrm{H}$
$s_0.\mathrm{H} \circ s_1.w_h \circ s_1.\mathrm{H} \circ s_2.\mathrm{H}$ $s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ s_2.w_h \circ s_2.\mathrm{H}$
$s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ s_2.\mathrm{H}$ $s_0.t_h \circ s_1.t_h \circ s_2.t_h$
$s_0.\mathrm{H} \circ s_0.\mathrm{R} \circ s_0.\mathrm{L}$ $s_1.\mathrm{H} \circ s_1.\mathrm{R} \circ s_1.\mathrm{L}$ $s_0.\mathrm{H} \circ s_0.\mathrm{R} \circ q_0.w$
$s_0.\mathrm{H} \circ s_0.\mathrm{L} \circ s_1.\mathrm{H}$ $s_0.\mathrm{H} \circ s_0.\mathrm{L} \circ s_1.w_h$ $s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ s_1.\mathrm{L}$
$s_0.w_h \circ s_1.\mathrm{H} \circ s_1.\mathrm{R}$
$s_0.w_{left} \circ s_1.w_{right}$ $s_0.t_{left} \circ s_1.t_{right}$
$s_0.w_{right} \circ s_1.w_{left}$ $s_0.t_{right} \circ s_1.t_{left}$
$s_0.a \circ s_0.w_{left}$ $s_0.a \circ s_0.t_{left}$ $s_0.a \circ s_0.w_{left} \circ s_1.w_{right}$
$s_0.a \circ s_0.t_{left} \circ s_1.t_{right}$ $s_0.a \circ s_0.w_h$ $s_0.a \circ s_0.t_h$

Table 1: Feature templates: s.L and s.R denote the left and right subnodes of s. l and r are head indices of L and R. q denotes a buffer element. t is a part-of-speech tag.

New nonterminal X is lexicalized with head word w_{h_0} of right nonterminal Z. This action expands Y and Z in a straight order. The leftmost word of phrase X is set to leftmost word w_{left1} of Y, and the rightmost word of phrase X is set to rightmost word w_{right0} of Z. Variable a is set to a_1 of Y.

The difference between reduce-MR-X and reduce-SR-X actions is new stack element s_0. The reduce-SR-X action generates s_0 by combining s'_0 and s'_1 with binary rule $\mathrm{X}^{\#}$ →Y Z:

$$s_0 = \{\mathrm{X}^{\#}, h_0, w_{left0}, w_{right1}, a_0\}.$$

This action expands Y and Z in a reverse order, and the leftmost word of $\mathrm{X}^{\#}$ is set to w_{left0} of Z, and the rightmost word of $\mathrm{X}^{\#}$ is set to w_{right1} of Y. Variable a is set to a_0 of Z.

We use a linear model that is discriminatively trained with the averaged perceptron (Collins and Roark, 2004). Table 1 shows the feature templates used in our experiments and we call the features in the bottom two rows "non-local" features.

4 Experiments

4.1 Experimental Setups

We conducted experiments for NTCIR-9 and 10 patent data using a Japanese-English language pair.

	test9			test10		
	BLEU	RIBES	time (sec.)	BLEU	RIBES	time (sec.)
PBMT (dist=6)	27.1	67.76	2.66	27.92	68.13	3.18
PBMT (dist=12)	29.55	69.84	4.15	30.03	69.88	4.93
PBMT (dist=20)	29.98	69.87	6.22	30.15	69.43	7.19
Tree-based MT** (Goto et al., 2012)	29.53	69.22	–	–	–	–
PBMT (dist=20)** (Goto et al., 2012)	30.13	68.86	–	–	–	–
Goto et al. (2012)**	31.75	72.57	–	–	–	–
PBMT (dist=0) + proposed w/o nf. (beam=12)	32.59	76.35	1.46 + 0.01	32.83	76.44	1.7 + 0.01
PBMT (dist=0) + proposed w/o nf. (beam=48)	32.61	76.58	1.46 + 0.06	32.86	76.6	1.7 + 0.06
PBMT (dist=0) + proposed w/ nf. (beam=12)	32.91	76.38	1.46 + 0.01	33.15	76.53	1.7 + 0.02
PBMT (dist=0) + proposed w/ nf. (beam=48)	**32.93**	**76.68**	1.46 + 0.07	**33.25**	**76.74**	1.7 + 0.07

Table 3: System comparison: time represents the average second per sentence. ** denotes "not our experiments".

	train	dev	test9	test10
# of sent.	3,191,228	2,000	2,000	2,300
ave. leng. (J)	36.4	36.6	37.0	43.1
ave. leng. (E)	33.3	33.3	33.7	39.6

Table 2: NTCIR-9 and 10 data statistics.

Mecab[2] was used for the Japanese morphological analysis. The data are summarized in Table 2.

We used Enju (Miyao and Tsujii, 2008) for parsing the English training data and converted parse trees into HFE trees by a head-finalization scheme. We extracted grammar rules from all the HFE trees and randomly selected 500,000 HFE trees to train the shift-reduce parser.

We used Moses (Koehn et al., 2007) with lexicalized reordering and a 6-gram language model (LM) trained using SRILM (Stolcke et al., 2011) to translate the Japanese sentences into HFE sentences.

To recover the English sentences, our shift-reduce parser reordered only the 1-best HFE sentence. Our strategy is much simpler than Goto et al. (2012)'s because they used a linear inteporation of MT cost, parser cost and N-gram LM cost to generate the best English sentence from the n-best HFE sentences.

4.2 Main Results

The main results in Table 3 indicate our method was significantly better and faster than the conventional PBMT system. Our method also ourperformed Goto et al. (2012)'s reported systems as well as a tree-based (moses-chart) system[3]. Our proposed model with "non-local" features (w/ nf.) achieved gains against that without the features (w/o nf.). Further feature engineering may improve the accuracy more.

[2]https://code.google.com/p/mecab/

	test9		test10	
	BLEU	RIBES	BLEU	RIBES
HFE w/ art.	28.86	73.45	29.9	73.52
proposed	**32.93**	**76.68**	**33.25**	**76.74**
w/o art.	19.86	75.62	20.17	75.63
N-gram	32.15	76.52	32.28	76.46

Table 4: The effects of article generation: "w/o art." denotes evaluation scores for translations of the best system ("proposed") in Table 3 from which articles are removed. "HFE w/ art." system used HFE data with articles and generated them by MT system and the shift-reduce parser performed only reordering. "N-gram" system inserted articles into the translations of "w/o art." by Goto et al. (2012)'s article generation method.

	(1–4)-gram precision
moses (dist=6)	67.1 / 36.9 / 20.7 / 11.5
moses (dist=20)	67.7 / 38.9 / 23.0 / 13.7
proposed	68.9 / 40.6 / 25.7 / 16.7

Table 5: N-gram precisions of moses (dist=6, dist=20) and proposed systems for test9 data.

4.3 Analysis

We show N-gram precisions of PBMT (dist=6, dist=20) and proposed systems in Table 5. The re-

[3]All the data and the MT toolkits used in our experiments are the same as theirs.

sults clearly show that improvements of 1-gram precisions are the main factors that contribute to better performance of our proposed system than PBMT systems. It seems that the gains of 1-gram presicions come from postediting (article generation).

In table 4, we show the effectiveness of our joint reordering and postediting approach ("proposed"). The "w/o art." results clearly show that generating articles has great effects on MT evaluations especially for BLEU metric. Comparing "proposed" and "HFE w/ art." systems, these results show that postediting is much more effective than generating articles by MT. Our joint approach also outperformed "N-gram" postediting system.

5 Conclusion

We proposed a shift-reduce word ordering model and applied it to J-to-E postordering. Our experimental results indicate our method can significantly improve the performance of a PBMT system.

Future work will investigate our method's usefulness on various language datasets. We plan to study more general methods that use word alignments to embed swap information in trees (Galley et al., 2006).

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 531–540.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

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 *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 961–968.

Isao Goto, Masao Utiyama, and Eiichiro Sumita. 2012. Post-ordering by parsing for japanese-english statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 311–316.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proc. HLT-NAACL*, pages 105–112.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2012. HPSG-based preprocessing for English-to-Japanese translation. *ACM Transactions on Asian Language Information Processing (TALIP)*, 11(3).

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. In *Proceedings of the National Conference on Artificial Intelligence*, pages 779–779.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic hpsg parsing. *Computational Linguistics*, 34(1):35–80.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human language technologies 2007: the conference of the North American chapter of the Association for Computational Linguistics*, pages 404–411.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. Srilm at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*.

Katsuhito Sudoh, Xianchao Wu, Kevin Duh, Hajime Tsukada, and Masaaki Nagata. 2011. Post-ordering in statistical machine translation. In *Proc. MT Summit*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.