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 proposed 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 giving up achieving an optimal solution. Therefore, it works much more efficient.

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
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 $w(\in T)$:

$$X \rightarrow w$$

and binary production rules in two forms:

$$X \rightarrow YZ$$

$$X\# \rightarrow YZ$$

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:

$$\ell : (i, j, S) : \pi$$

where $\ell$ 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 $\pi$ 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:

\[ s = \{ H, h, w_{\text{left}}, w_{\text{right}}, a \} \]

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 \( w_{\text{left}}, w_{\text{right}} \) 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:

\[
X \rightarrow w_j \in P \\
\ell : (i, j, S|s'_0) : \pi \\
\ell + 1 : (i, j + 1, S|s'_0|s_0) : \{p\}
\]

where \( s_0 = \{ X, j, w_j, w_j, \text{null} \} \).

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

\[
s_0', X \in I \land (s'_0, a = \text{null or } x \neq \text{“no article”}) \\
\ell : (i, j, S|s'_0) : \pi \\
\ell + 1 : (i, j, S|s'_0|s_0) : \pi
\]

where \( s'_0 \) is \( \{ X, h, w_{\text{left}}, w_{\text{right}}, a \} \) and \( s_0 \) is \( \{ X, h, w_{\text{left}}, w_{\text{right}}, a \} \) \( (i \leq h, \text{left}, \text{right} < j) \).

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 \( w_{\text{lefft1}} \) of \( Y \), and the rightmost word of phrase \( X \) is set to rightmost word \( w_{\text{wright0}} \) 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 \( X^\# \rightarrow Y Z \):

\[ s_0 = \{ X^\#, h_0, w_{\text{lefft0}}, w_{\text{wright0}}, a_0 \} \]

This action expands \( Y \) and \( Z \) in a reverse order, and the leftmost word of \( X^\# \) is set to \( w_{\text{lefft0}} \) of \( Z \), and the rightmost word of \( X^\# \) is set to \( w_{\text{wright1}} \) 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 NTDCIR-9 and 10 patent data using a Japanese-English language pair.
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 interpolation 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 outperformed 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.

### 4.3 Analysis

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

---

\(^2\)https://code.google.com/p/mecab/  

\(^3\)All the data and the MT toolkits used in our experiments are the same as theirs.
results 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 precisions 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


