A Ranking-based Approach to Word Reordering for Statistical Machine Translation*

Nan Yang[†], **Mu Li**[‡], **Dongdong Zhang**[‡], and **Nenghai Yu**[†]

[†]MOE-MS Key Lab of MCC
University of Science and Technology of China
v-nayang@microsoft.com, ynh@ustc.edu.cn
[‡]Microsoft Research Asia
{muli,dozhang}@microsoft.com

Abstract

Long distance word reordering is a major challenge in statistical machine translation research. Previous work has shown using source syntactic trees is an effective way to tackle this problem between two languages with substantial word order difference. In this work, we further extend this line of exploration and propose a novel but simple approach, which utilizes a ranking model based on word order precedence in the target language to reposition nodes in the syntactic parse tree of a source sentence. The ranking model is automatically derived from word aligned parallel data with a syntactic parser for source language based on both lexical and syntactical features. We evaluated our approach on large-scale Japanese-English and English-Japanese machine translation tasks, and show that it can significantly outperform the baseline phrase-based SMT system.

1 Introduction

Modeling word reordering between source and target sentences has been a research focus since the emerging of statistical machine translation. In phrase-based models (Och, 2002; Koehn et al., 2003), phrase is introduced to serve as the fundamental translation element and deal with local reordering, while a distance based distortion model is used to coarsely depict the exponentially decayed word movement probabilities in language translation. Further work in this direction employed lexicalized distortion models, including both generative (Koehn et al., 2005) and discriminative (Zens and Ney, 2006; Xiong et al., 2006) variants, to achieve finer-grained estimations, while other work took into account the hierarchical language structures in translation (Chiang, 2005; Galley and Manning, 2008).

Long-distance word reordering between language pairs with substantial word order difference, such as Japanese with Subject-Object-Verb (SOV) structure and English with Subject-Verb-Object (SVO) structure, is generally viewed beyond the scope of the phrase-based systems discussed above, because of either distortion limits or lack of discriminative features for modeling. The most notable solution to this problem is adopting syntax-based SMT models, especially methods making use of source side syntactic parse trees. There are two major categories in this line of research. One is tree-to-string model (Quirk et al., 2005; Liu et al., 2006) which directly uses source parse trees to derive a large set of translation rules and associated model parameters. The other is called *syntax pre-reordering* – an approach that re-positions source words to approximate target language word order as much as possible based on the features from source syntactic parse trees. This is usually done in a preprocessing step, and then followed by a standard phrase-based SMT system that takes the re-ordered source sentence as input to finish the translation.

In this paper, we continue this line of work and address the problem of word reordering based on source syntactic parse trees for SMT. Similar to most previous work, our approach tries to rearrange the source tree nodes sharing a common parent to mimic

*This work has been done while the first author was visiting Microsoft Research Asia.

the word order in target language. To this end, we propose a simple but effective ranking-based approach to word reordering. The ranking model is automatically derived from the word aligned parallel data, viewing the source tree nodes to be reordered as list items to be ranked. The ranks of tree nodes are determined by their relative positions in the target language – the node in the most front gets the highest rank, while the ending word in the target sentence gets the lowest rank. The ranking model is trained to directly minimize the mis-ordering of tree nodes, which differs from the prior work based on maximum likelihood estimations of reordering patterns (Li et al., 2007; Genzel, 2010), and does not require any special tweaking in model training. The ranking model can not only be used in a pre-reordering based SMT system, but also be integrated into a phrase-based decoder serving as additional distortion features.

We evaluated our approach on large-scale Japanese-English and English-Japanese machine translation tasks, and experimental results show that our approach can bring significant improvements to the baseline phrase-based SMT system in both pre-ordering and integrated decoding settings.

In the rest of the paper, we will first formally present our ranking-based word reordering model, then followed by detailed steps of modeling training and integration into a phrase-based SMT system. Experimental results are shown in Section 5. Section 6 consists of more discussions on related work, and Section 7 concludes the paper.

2 Word Reordering as Syntax Tree Node Ranking

Given a source side parse tree T_e, the task of word reordering is to transform T_e to T'_e, so that e' can match the word order in target language as much as possible. In this work, we only focus on reordering that can be obtained by permuting *children* of every tree nodes in T_e. We use *children* to denote direct descendants of tree nodes for constituent trees; while for dependency trees, *children* of a node include not only all direct dependents, but also the head word itself. Figure 1 gives a simple example showing the word reordering between English and Japanese. By rearranging the position of tree nodes in the English

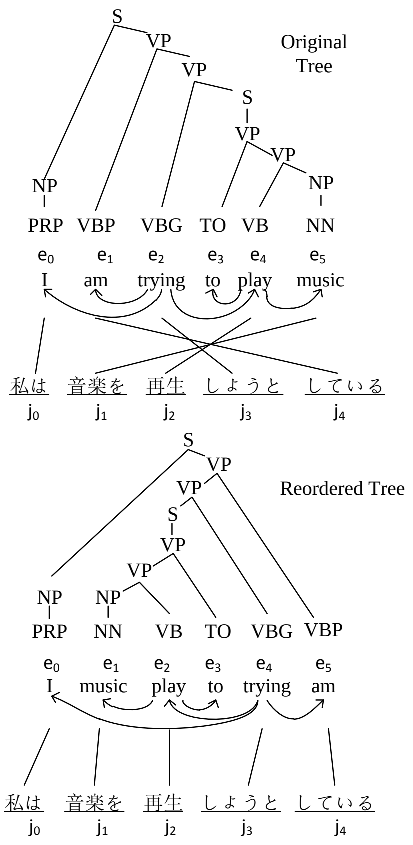


Figure 1: An English-to-Japanese sentence pair. By permuting tree nodes in the parse tree, the source sentence is reordered into the target language order. Constituent tree is shown above the source sentence; arrows below the source sentences show head-dependent arcs for dependency tree; word alignment links are lines without arrow between the source and target sentences.

parse tree, we can obtain the same word order of Japanese translation. It is true that tree-based reordering cannot cover all word movement operations in language translation, previous work showed that this method is still very effective in practice (Xu et al., 2009, Visweswariah et al., 2010).

Following this principle, the word reordering task can be broken into sub-tasks, in which we only need to determine the order of children nodes for all non-leaf nodes in the source parse tree. For a tree node t with children $\{c_1, c_2, \ldots, c_n\}$, we rearrange the children to target-language-like order $\{c_{\pi(i_1)}, c_{\pi(i_2)}, \ldots, c_{\pi(i_n)}\}$. If we treat the reordered position $\pi(i)$ of child c_i as its "*rank*", the reordering problem is naturally translated into a ranking problem: to reorder, we determine a "rank" for each child, then the children are sorted according to their "ranks". As it is often impractical to directly assign a score for each permutation due to huge number of possible permutations, a widely used method is to use a real valued function f to assign a value to each node, which is called a ranking function (Herbrich et al., 2000). If we can guarantee $(f(i) - f(j))$ and $(\pi(i) - \pi(j))$ always has the same sign, we can get the same permutation as π because values of f are only used to sort the children. For example, consider the node rooted at *trying* in the dependency tree in Figure 1. Four children form a list $\{$*I*, *am*, *trying*, *play*$\}$ to be ranked. Assuming ranking function f can assign values $\{0.94, -1.83, -1.50, -1.20\}$ for $\{$*I*, *am*, *trying*, *play*$\}$ respectively, we can get a sorted list $\{$*I*, *play*, *trying*, *am*$\}$, which is the desired permutation according to the target.

More formally, for a tree node t with children $\{c_1, c_2, \ldots, c_n\}$, our ranking model assigns a rank $f(c_i, t)$ for each child c_i, then the children are sorted according to the rank in a descending order. The ranking function f has the following form:

$$f(c_i, t) = \sum_j \theta_j(c_i, t) \cdot w_j \tag{1}$$

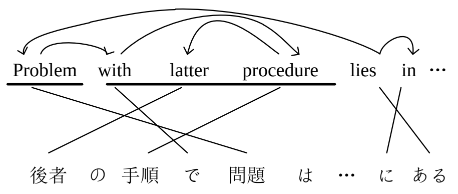
where the θ_j is a feature representing the tree node t and its child c_i, and w_j is the corresponding feature weight.

3 Ranking Model Training

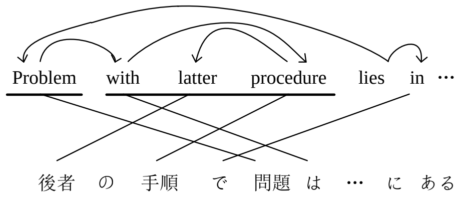
To learn ranking function in Equation (1), we need to determine the feature set $\boldsymbol{\theta}$ and learn weight vector $\mathbf{w}$ from reorder examples. In this section, we first describe how to extract reordering examples from parallel corpus; then we show our features for ranking function; finally, we discuss how to train the model from the extracted examples.

3.1 Reorder Example Acquisition

For a sentence pair (e, f, a) with syntax tree T_e on the source side, we need to determine which reordered tree $T'_{e'}$ best represents the word order in target sentence f. For a tree node t in T_e, if its children align to disjoint target spans, we can simply arrange them in the order of their corresponding target



(a) gold alignment



(b) auto alignment

Figure 2: Fragment of a sentence pair. (a) shows gold alignment; (b) shows automatically generated alignment which contains errors.

spans. Figure 2 shows a fragment of one sentence pair in our training data. Consider the subtree rooted at word "*Problem*". With the gold alignment, "*Problem*" is aligned to the 5th target word, and "*with latter procedure*" are aligned to target span [1, 3], thus we can simply put "*Problem*" after "*with latter procedure*". Recursively applying this process down the subtree, we get "*latter procedure with Problem*" which perfectly matches the target language.

As pointed out by (Li et al., 2007), in practice, nodes often have overlapping target spans due to erroneous word alignment or different syntactic structures between source and target sentences. (b) in Figure 2 shows the automatically generated alignment for the sentence pair fragment. The word "*with*" is incorrectly aligned to the 6th Japanese word "*ha*"; as a result, "*with latter procedure*" now has target span [1, 6], while "*Problem*" aligns to [5, 5]. Due to this overlapping, it becomes unclear which permutation of "*Problem*" and "*with latter procedure*" is a better match of the target phrase; we need a better metric to measure word order similarity between reordered source and target sentences. We choose to find the tree $T'_{e'}$ with minimal *alignment crossing-link number* (*CLN*) (Genzel, 2010) to f as our golden reordered tree.[1] Each crossing-

[1]A simple solution is to exclude all trees with overlapping target spans from training. But in our experiment, this method

link (i_1j_1, i_2j_2) is a pair of alignment links crossing each other. *CLN* reaches zero if f is monotonically aligned to e', and increases as there are more word reordering between e' and f. For example, in Figure 1, there are 6 crossing-links in the original tree: (e_1j_4, e_2j_3), (e_1j_4, e_4j_2), (e_1j_4, e_5j_1), (e_2j_3, e_4j_2), (e_2j_3, e_5j_1) and (e_4j_2, e_5j_1); thus *CLN* for the original tree is 6. *CLN* for the reordered tree is 0 as there are no crossing-links. This metric is easy to compute, and is not affected by unaligned words (Genzel, 2010).

We need to find the reordered tree with minimal *CLN* among all reorder candidates. As the number of candidates is in the magnitude exponential with respect to the degree of tree T_e [2], it is not always computationally feasible to enumerate through all candidates. Our solution is as follows.

First, we give two definitions.

- $CLN(t)$: the number of crossing-links (i_1j_1, i_2j_2) whose source words e'_{i_1} and e'_{i_2} both fall under sub span of the tree node t.
- $CCLN(t)$: the number of crossing-links (i_1j_1, i_2j_2) whose source words e'_{i_1} and e'_{i_2} fall under sub span of t's two different children nodes c_1 and c_2 respectively.

Apparently CLN of a tree T' equals to $CLN(root\ of\ T')$, and $CLN(t)$ can be recursively expressed as:

$$CLN(t) = CCLN(t) + \sum_{child\ c\ of\ t} CLN(c)$$

Take the original tree in Figure 1 for example. At the root node *trying*, *CLN*(*trying*) is 6 because there are six crossing-links under its sub-span: (e_1j_4, e_2j_3), (e_1j_4, e_4j_2), (e_1j_4, e_5j_1), (e_2j_3, e_4j_2), (e_2j_3, e_5j_1) and (e_4j_2, e_5j_1). On the other hand, *CCLN*(*trying*) is 5 because (e_4j_2, e_5j_1) falls under its child node *play*, thus does not count towards *CCLN* of *trying*.

From the definition, we can easily see that *CCLN*(*t*) can be determined solely by the order of t's direct children, and $CLN(t)$ is only affected by the reorder in the subtree of t. This observation enables us to divide the task of finding the reordered tree $T'_{e'}$ with minimal *CLN* into independently finding the children permutation of each node with minimal *CCLN*. Unfortunately, the time cost for the subtask is still $O(n!)$ for a node with n children. Instead of enumerating through all permutations, we only search the *Inversion Transduction Grammar neighborhood* of the initial sequence (Tromble, 2009). As pointed out by (Tromble, 2009), the ITG neighborhood is large enough for reordering task, and can be searched through efficiently using a CKY decoder.

After finding the best reordered tree $T'_{e'}$, we can extract one reorder example from every node with more than one child.

3.2 Features

Features for the ranking model are extracted from source syntax trees. For English-to-Japanese task, we extract features from Stanford English Dependency Tree (Marneffe et al., 2006), including lexicons, Part-of-Speech tags, dependency labels, punctuations and tree distance between head and dependent. For Japanese-to-English task, we use a chunk-based Japanese dependency tree (Kudo and Matsumoto, 2002). Different from features for English, we do not use dependency labels because they are not available from the Japanese parser. Additionally, Japanese function words are also included as features because they are important grammatical clues. The detailed feature templates are shown in Table 1.

3.3 Learning Method

There are many well studied methods available to learn the ranking function from extracted examples., ListNet (**?**) etc. We choose to use RankingSVM (Herbrich et al., 2000), a pair-wised ranking method, for its simplicity and good performance.

For every reorder example t with children $\{c_1, c_2, \ldots, c_n\}$ and their desired permutation $\{c_{\pi(i_1)}, c_{\pi(i_2)}, \ldots, c_{\pi(i_n)}\}$, we decompose it into a set of pair-wised training instances. For any two children nodes c_i and c_j with $i < j$, we extract a positive instance if $\pi(i) < \pi(j)$, otherwise we extract a negative instance. The feature vector for both positive instance and negative instance is $(\theta_{c_i} - \theta_{c_j})$, where $\theta_{\mathbf{c_i}}$ and $\theta_{\mathbf{c_j}}$ are feature vectors for c_i and c_j

discarded too many training instances and led to degraded reordering performance.

[2]In our experiments, there are nodes with more than 10 children for English dependency trees.

E-J		
c_l	$c_l \cdot dst$	$c_l \cdot pct$
$c_l \cdot dst \cdot pct$	$c_l \cdot lc_l$	$c_l \cdot rc_l$
$c_l \cdot lc_l \cdot dst$	$c_l \cdot rc_l \cdot dst$	$c_l \cdot c_{lex}$
$c_l \cdot c_{lex}$	$c_l \cdot c_{lex} \cdot dst$	$c_l \cdot c_{lex} \cdot dst$
$c_l \cdot h_{lex}$	$c_l \cdot h_{lex}$	$c_l \cdot h_{lex} \cdot dst$
$c_l \cdot h_{lex} \cdot dst$	$c_l \cdot c_{lex} \cdot pct$	$c_l \cdot c_{lex} \cdot pct$
$c_l \cdot h_{lex} \cdot pct$	$c_l \cdot h_{lex} \cdot pct$	
J-E		
c_{tf}	$c_{tf} \cdot dst$	$c_{tf} \cdot lc_t$
$c_{tf} \cdot rc_t$	$c_{tf} \cdot lc_t \cdot dst$	$c_l \cdot rc_t \cdot dst$
$c_{tf} \cdot c_{lex}$	$c_{tf} \cdot c_{lex}$	$c_{tf} \cdot c_{lex} \cdot dst$
$c_{tf} \cdot c_{lex} \cdot dst$	$c_{tf} \cdot h_f$	$c_{tf} \cdot h_f$
$c_{tf} \cdot h_f \cdot dst$	$c_{tf} \cdot h_f \cdot dst$	$c_{tf} \cdot h_{lex}$
$c_{tf} \cdot h_{lex}$	$c_{tf} \cdot h_{lex} \cdot dst$	$c_{tf} \cdot h_{lex} \cdot dst$

Table 1: Feature templates for ranking function. All templates are implicitly conjuncted with the pos tag of head node.
c: child to be ranked; h: head node
lc: left sibling of c; rc: right sibling of c
l: dependency label; t: pos tag
lex: top frequency lexicons
f: Japanese function word
dst: tree distance between c and h
pct: punctuation node between c and h

respectively. In this way, ranking function learning is turned into a simple binary classification problem, which can be easily solved by a two-class linear support vector machine.

4 Integration into SMT system

There are two ways to integrate the ranking reordering model into a phrase-based SMT system: the pre-reorder method, and the decoding time constraint method.

For pre-reorder method, ranking reorder model is applied to reorder source sentences during both training and decoding. Reordered sentences can go through the normal pipeline of a phrase-based decoder.

The ranking reorder model can also be integrated into a phrase based decoder. Integrated method takes the original source sentence e as input, and ranking model generates a reordered e' as a word order reference for the decoder. A simple penalty scheme is utilized to penalize decoder reordering violating ranking reorder model's prediction e'. In this paper, our underlying decoder is a CKY decoder following Bracketing Transduction Grammar (Wu, 1997; Xiong et al., 2006), thus we show how the penalty is implemented in the BTG decoder as an example. Similar penalty can be designed for other decoders without much effort.

Under BTG, three rules are used to derive translations: one unary terminal rule, one *straight* rule and one *inverse* rule:

$$\begin{aligned} A &\rightarrow e/f \\ A &\rightarrow [A_1, A_2] \\ A &\rightarrow \langle A_1, A_2 \rangle \end{aligned}$$

We have three penalty triggers when any rules are applied during decoding:

- Discontinuous penalty f_{dc}: it fires for all rules when source span of either A, A_1 or A_2 is mapped to discontinuous span in e'.
- Wrong straight rule penalty f_{st}: it fires for straight rule when source spans of A_1 and A_2 are not mapped to two adjacent spans in e' in straight order.
- Wrong inverse rule penalty f_{iv}: it fires for inverse rule when source spans of A_1 and A_2 are not mapped to two adjacent spans in e' in inverse order.

The above three penalties are added as additional features into the log-linear model of the phrase-based system. Essentially they are soft constraints to encourage the decoder to choose translations with word order similar to the prediction of ranking reorder model.

5 Experiments

To test our ranking reorder model, we carry out experiments on large scale English-To-Japanese, and Japanese-To-English translation tasks.

5.1 Data

5.1.1 Evaluation Data

We collect 3,500 Japanese sentences and 3,500 English sentences from the web. They come from a wide range of domains, such as technical documents, web forum data, travel logs etc. They are manually translated into the other language to produce 7,000 sentence pairs, which are split into two parts: 2,000 pairs as development set (***dev***) and the other 5,000 pairs as test set (***web*** test).

Beside that, we collect another 999 English sentences from newswire domain which are translated into Japanese to form an out-of-domain test data set (***news*** test).

5.1.2 Parallel Corpus

Our parallel corpus is crawled from the web, containing news articles, technical documents, blog entries etc. After removing duplicates, we have about 18 million sentence pairs, which contain about 270 millions of English tokens and 320 millions of Japanese tokens. We use Giza++ (Och and Ney, 2003) to generate the word alignment for the parallel corpus.

5.1.3 Monolingual Corpus

Our monolingual Corpus is also crawled from the web. After removing duplicate sentences, we have a corpus of over 10 billion tokens for both English and Japanese. This monolingual corpus is used to train a 4-gram language model for English and Japanese respectively.

5.2 Parsers

For English, we train a dependency parser as (Nivre and Scholz, 2004) on WSJ portion of Penn Treebank, which are converted to dependency trees using Stanford Parser (Marneffe et al., 2006). We convert the tokens in training data to lower case, and re-tokenize the sentences using the same tokenizer from our MT system.

For Japanese parser, we use CABOCHA, a chunk-based dependency parser (Kudo and Matsumoto, 2002). Some heuristics are used to adapt CABOCHA generated trees to our word segmentation.

5.3 Settings

5.3.1 Baseline System

We use a BTG phrase-based system with a Max-Ent based lexicalized reordering model (Wu, 1997; Xiong et al., 2006) as our baseline system for both English-to-Japanese and Japanese-to-English Experiment. The distortion model is trained on the same parallel corpus as the phrase table using a home implemented maximum entropy trainer.

In addition, a pre-reorder system using manual rules as (Xu et al., 2009) is included for the English-to-Japanese experiment (ManR-PR). Manual rules are tuned by a bilingual speaker on the development set.

5.3.2 Ranking Reordering System

Ranking reordering model is learned from the same parallel corpus as phrase table. For efficiency reason, we only use 25% of the corpus to train our reordering model. LIBLINEAR (Fan et al., 2008) is used to do the SVM optimization for RankingSVM.

We test it on both pre-reorder setting (*Rank-PR*) and integrated setting (*Rank-IT*).

5.4 End-to-End Result

	system	dev	web test	news test
E-J	Baseline	21.45	21.12	14.18
	ManR-PR	23.00	22.42	15.61
	Rank-PR	22.92	22.51	**15.90**
	Rank-IT	**23.14**	**22.85**	15.72
J-E	Baseline	25.39	24.20	14.26
	Rank-PR	26.57	25.56	**15.42**
	Rank-IT	**26.72**	**25.87**	15.27

Table 2: BLEU(%) score on dev and test data for both E-J and J-E experiment. All settings significantly improve over the baseline at 95% confidence level. Baseline is the BTG phrase system system; ManR-PR is pre-reorder with manual rule; Rank-PR is pre-reorder with ranking reorder model; Rank-IT is system with integrated ranking reorder model.

From Table 2, we can see our ranking reordering model significantly improves the performance for both English-to-Japanese and Japanese-to-English experiments over the BTG baseline system. It also out-performs the manual rule set on English-to-Japanese result, but the difference is not significant.

5.5 Reordering Performance

In order to show whether the improved performance is really due to improved reordering, we would like to measure the reorder performance directly.

As we do not have access to a golden reordered sentence set, we decide to use the alignment crossing-link numbers between aligned sentence pairs as the measure for reorder performance.

We train the ranking model on 25% of our parallel corpus, and use the rest 75% as test data (***auto***). We sample a small corpus (575 sentence pairs) and do manual alignment (***man-small***). We denote the automatic alignment for these 575 sentences as (***auto-small***). From Table 3, we can see

	setting	auto	auto-small	man-small
	None	36.3	35.9	40.1
E-J	Oracle	4.3	4.1	7.4
	ManR	13.4	13.6	16.7
	Rank	12.1	12.8	17.2
J-E	Oracle	6.9	7.0	9.4
	Rank	15.7	15.3	20.5

Table 3: Reorder performance measured by crossing-link number per sentence. *None* means the original sentences without reordering; *Oracle* means the best permutation allowed by the source parse tree; *ManR* refers to manual reorder rules; *Rank* means ranking reordering model.

our ranking reordering model indeed significantly reduces the crossing-link numbers over the original sentence pairs. On the other hand, the performance of the ranking reorder model still fall far short of oracle, which is the lowest crossing-link number of all possible permutations allowed by the parse tree. By manual analysis, we find that the gap is due to both errors of the ranking reorder model and errors from word alignment and parser.

Another thing to note is that the crossing-link number of manual alignment is higher than automatic alignment. The reason is that our annotators tend to align function words which might be left unaligned by automatic word aligner.

5.6 Effect of Ranking Features

Here we examine the effect of features for ranking reorder model. We compare their influence on RankingSVM accuracy, alignment crossing-link number, end-to-end BLEU score, and the model size. As Table 4 shows, a major part of reduction of *CLN* comes from features such as Part-of-Speech tags,

	Features	Acc.	CLN	BLEU	Feat.#
E-J	tag+$label$	88.6	16.4	22.24	26k
	+dst	91.5	13.5	22.66	55k
	+pct	92.2	13.1	22.73	79k
	+lex_{100}	92.9	12.1	22.85	347k
	+lex_{1000}	94.0	11.5	22.79	2,410k
	+lex_{2000}	95.2	10.7	22.81	3,794k
J-E	tag+fw	85.0	18.6	25.43	31k
	+dst	90.3	16.9	25.62	65k
	+lex_{100}	91.6	15.7	25.87	293k
	+lex_{1000}	92.4	14.8	25.91	2,156k
	+lex_{2000}	93.0	14.3	25.84	3,297k

Table 4: Effect of ranking features. Acc. is RankingSVM accuracy in percentage on the training data; *CLN* is the crossing-link number per sentence on parallel corpus with automatically generated word alignment; BLEU is the BLEU score in percentage on **web** test set on *Rank-IT* setting (system with integrated rank reordering model); lex_n means n most frequent lexicons in the training corpus.

dependency labels (for English), function words (for Japanese), and the distance and punctuations between child and head. These features also correspond to BLEU score improvement for End-to-End evaluations. Lexicon features generally continue to improve the RankingSVM accuracy and reduce *CLN* on training data, but they do not bring further improvement for SMT systems beyond the top 100 most frequent words. Our explanation is that less frequent lexicons tend to help local reordering only, which is already handled by the underlying phrase-based system.

5.7 Performance on different domains

From Table 2 we can see that pre-reorder method has higher BLEU score on **news** test, while integrated model performs better on **web** test set which contains informal texts. By error analysis, we find that the parser commits more errors on informal texts, and informal texts usually have more flexible translations. Pre-reorder method makes “hard” decision before decoding, thus is more sensitive to parser errors; on the other hand, integrated model is forced to use a longer distortion limit which leads to more search errors during decoding time. It is possible to use system combination method to get the best of both systems, but we leave this to future work.

6 Discussion on Related Work

There have been several studies focusing on compiling hand-crafted syntactic reorder rules. Collins et al. (2005), Wang et al. (2007), Ramanathan et al. (2008), Lee et al. (2010) have developed rules for German-English, Chinese-English, English-Hindi and English-Japanese respectively. Xu et al. (2009) designed a clever precedence reordering rule set for translation from English to several SOV languages. The drawback for hand-crafted rules is that they depend upon expert knowledge to produce and are limited to their targeted language pairs.

Automatically learning syntactic reordering rules have also been explored in several work. Li et al. (2007) and Visweswariah et al. (2010) learned probability of reordering patterns from constituent trees using either Maximum Entropy or maximum likelihood estimation. Since reordering patterns are matched against a tree node together with all its direct children, data sparseness problem will arise when tree nodes have many children (Li et al., 2007); Visweswariah et al. (2010) also mentioned their method yielded no improvement when applied to dependency trees in their initial experiments. Genzel (2010) dealt with the data sparseness problem by using window heuristic, and learned reordering pattern sequence from dependency trees. Even with the window heuristic, they were unable to evaluate all candidates due to the huge number of possible patterns. Different from the previous approaches, we treat syntax-based reordering as a ranking problem between different source tree nodes. Our method does not require the source nodes to match some specific patterns, but encodes reordering knowledge in the form of a ranking function, which naturally handles reordering between any number of tree nodes; the ranking function is trained by well-established rank learning method to minimize the number of mis-ordered tree nodes in the training data.

Tree-to-string systems (Quirk et al., 2005; Liu et al., 2006) model syntactic reordering using minimal or composed translation rules, which may contain reordering involving tree nodes from multiple tree levels. Our method can be naturally extended to deal with such multiple level reordering. For a tree-to-string rule with multiple tree levels, instead of ranking the direct children of the root node, we rank all leaf nodes (Most are frontier nodes (Galley et al., 2006)) in the translation rule. We need to redesign our ranking feature templates to encode the reordering information in the source part of the translation rules. We need to remember the source side context of the rules, the model size would still be much smaller than a full-fledged tree-to-string system because we do not need to explicitly store the target variants for each rule.

7 Conclusion and Future Work

In this paper we present a ranking based reordering method to reorder source language to match the word order of target language given the source side parse tree. Reordering is formulated as a task to rank different nodes in the source side syntax tree according to their relative position in the target language. The ranking model is automatically trained to minimize the mis-ordering of tree nodes in the training data. Large scale experiment shows improvement on both reordering metric and SMT performance, with up to 1.73 point BLEU gain in our evaluation test.

In future work, we plan to extend the ranking model to handle reordering between multiple levels of source trees. We also expect to explore better way to integrate ranking reorder model into SMT system instead of a simple penalty scheme. Along the research direction of preprocessing the source language to facilitate translation, we consider to not only change the order of the source language, but also inject syntactic structure of the target language into source language by adding pseudo words into source sentences.

Acknowledgements

Nan Yang and Nenghai Yu were partially supported by Fundamental Research Funds for the Central Universities (No. WK2100230002), National Natural Science Foundation of China (No. 60933013), and National Science and Technology Major Project (No. 2010ZX03004-003).

References

David Chiang. 2005. *A Hierarchical Phrase-Based Model for Statistical Machine Translation*. In *Proc. ACL*, pages 263-270.

Michael Collins, Philipp Koehn and Ivona Kucerova. 2005. *Clause restructuring for statistical machine translation*. In *Proc. ACL*.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. 2008. *LIBLINEAR: A library for large linear classification*. In *Journal of Machine Learning Research*.

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. ACL-Coling*, pages 961-968.

Michel Galley and Christopher D. Manning. 2008. *A Simple and Effective Hierarchical Phrase Reordering Model*. In *Proc. EMNLP*, pages 263-270.

Dmitriy Genzel. 2010. *Automatically Learning Source-side Reordering Rules for Large Scale Machine Translation*. In *Proc. Coling*, pages 376-384.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer 2000. *Large Margin Rank Boundaries for Ordinal Regression*. In *Advances in Large Margin Classifiers*, pages 115-132.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne and David Talbot. 2005. *Edinborgh System Description for the 2005 IWSLT Speech Translation Evaluation*. In *International Workshop on Spoken Language Translation*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. *Statistical Phrase-Based Translation*. In *Proc. HLT-NAACL*, pages 127-133.

Taku Kudo, Yuji Matsumoto. 2002. *Japanese Dependency Analysis using Cascaded Chunking*. In *Proc. CoNLL*, pages 63-69.

Young-Suk Lee, Bing Zhao and Xiaoqiang Luo. 2010. *Constituent reordering and syntax models for English-to-Japanese statistical machine translation*. In *Proc. Coling*.

Chi-Ho Li, Minghui Li, Dongdong Zhang, Mu Li and Ming Zhou and Yi Guan 2007. *A Probabilistic Approach to Syntax-based Reordering for Statistical Machine Translation*. In *Proc. ACL*, pages 720-727.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. *Tree-to-String Alignment Template for Statistical Machine Translation*. In *Proc. ACL-Coling*, pages 609-616.

Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. *Generating Typed Dependency Parses from Phrase Structure Parses*. In *LREC 2006*

Joakim Nivre and Mario Scholz 2004. *Deterministic Dependency Parsing for English Text*. In *Proc. Coling*.

Franz J. Och. 2002. *Statistical Machine Translation: From Single Word Models to Alignment Template*. Ph.D.Thesis, RWTH Aachen, Germany

Franz J. Och and Hermann Ney. 2003. *A Systematic Comparison of Various Statistical Alignment Models*. *Computational Linguistics*, 29(1): pages 19-51.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. *Dependency Treelet Translation: Syntactically Informed Phrasal SMT*. In *Proc. ACL*, pages 271-279.

A. Ramanathan, Pushpak Bhattacharyya, Jayprasad Hegde, Ritesh M. Shah and Sasikumar M. 2008. *Simple syntactic and morphological processing can help English-Hindi Statistical Machine Translation*. In *Proc. IJCNLP*.

Roy Tromble. 2009. *Search and Learning for the Linear Ordering Problem with an Application to Machine Translation*. Ph.D. Thesis.

Karthik Visweswariah, Jiri Navratil, Jeffrey Sorensen, Vijil Chenthamarakshan and Nandakishore Kambhatla. 2010. *Syntax Based Reordering with Automatically Derived Rules for Improved Statistical Machine Translation*. In *Proc. Coling*, pages 1119-1127.

Chao Wang, Michael Collins, Philipp Koehn. 2007. *Chinese syntactic reordering for statistical machine translation*. In *Proc. EMNLP-CoNLL*.

Dekai Wu. 1997. *Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora*. *Computational Linguistics*, 23(3): pages 377-403.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. *Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation*. In *Proc. ACL-Coling*, pages 521-528.

Peng Xu, Jaeho Kang, Michael Ringgaard, Franz Och. 2009. *Using a Dependency Parser to Improve SMT for Subject-Object-Verb Languages*. In *Proc. HLT-NAACL*, pages 376-384.

Richard Zens and Hermann Ney. 2006. *Discriminative Reordering Models for Statistical Machine Translation*. In *Proc. Workshop on Statistical Machine Translation, HLT-NAACL*, pages 127-133.