BTG-based Machine Translation with Simple Reordering Model using Structured Perceptron

Hao Wang
Graduate School of Information, Production and Systems, Waseda University
oko_ips@ruri.waseda.jp

Yves Lepage
Graduate School of Information, Production and Systems, Waseda University
yves.lepage@waseda.jp

Abstract

In this paper, we present a novel statistical machine translation method which employs a BTG-based reordering model during decoding. BTG-based reordering models for *preordering* have been widely explored, aiming to improve the standard phrase-based statistical machine translation system. Less attention has been paid to incorporating such a reordering model into decoding directly. Our reordering model differs from previous models built using a syntactic parser or directly from annotated treebanks. Here, we train without using any syntactic information. The experiment results on an English–Japanese translation task show that our BTG-based decoder achieves comparable or better performance than the more complex state-of-the-art SMT decoders.

1 Introduction

The phrase-based method (Koehn et al., 2003) and the syntax-based method (Yamada and Knight, 2001) are two of the representative methods in statistical machine translation (SMT). On the one hand, in the phrase-based model, the lexical reordering model is a crucial component, but it is often be criticized, especially when translating a language pair with widely divergent syntax like English-Japanese, as the naïve distance-based lexical reordering model does not work so well when applied to longer reorderings. On the other hand, in syntax-based SMT method, word reordering is implicitly addressed by translation rules. The performance is thus directly subject to the parsing errors of the syntactic parser. Syntax-based translation models are usually built from annotated treebanks to extract grammar rules for reordering (Genzel, 2010). Such reordering models are thus more difficult to train. Between these two models, some loose hierarchical structure models have been proposed: the hierarchical phrase-based model (Chiang, 2007) and or the Bracketing Transduction Grammar (BTG) based model (Wu, 1997). Compared with the hierarchical phrase-based model, the BTG model has many advantages like its simplicity. Also, its well-formed rules avoid extracting a large number of rare or useless translation rules, as is the case of the hierarchical phrase-based model.

In recent proposals, phrase-based statistical machine translation has been shown to improve when BTG-based preordering is applied as a preprocessing (DeNero and Uszkoreit, 2011; Neubig et al., 2012; Nakagawa, 2015). The idea behind preordering is to reduce the structural complexity. It is preferable to apply the reordering operations in advance rather than during decoding as this benefits the word alignment step.

In this paper, following (Xiong et al., 2008), we propose to incorporate the BTG-based reordering model directly into the decoding step of a BTG-based SMT system using a simple Structured Perceptron (Rosenblatt, 1958; Collins and Roark, 2004). The rest of the paper is organized as follows. Section 2 briefly introduces previous BTG-based reordering methods both for preordering or determining the reorderings during decoding. Section 3 describes the principal model used in BTG-based machine translation. Section 4 gives the details of the

31st Pacific Asia Conference on Language, Information and Computation (PACLIC 31), pages 114–123
Cebu City, Philippines, November 16-18, 2017

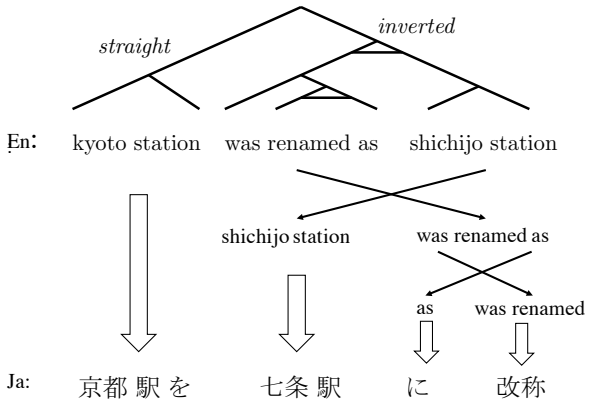

Figure 1: Example of translating a source sentence (English) into Japanese while reordering at the same time using a BTG tree.

proposed method and the model combination in the system construction. Section 5 reports the results of the experiment on an English-to-Japanese translation task. We conclude in Section 7.

2 Using Linguistic Contexts for BTG-based Reordering

A common problem in the distortion reordering models (Tillmann, 2004; Koehn et al., 2005; Galley and Manning, 2008) used in phrase-based SMT (PB-SMT) method is that they do not take contexts into account. Hence, we draw our attention on using linguistic-context information for reordering.

Bracketing Transduction Grammar (BTG) (Wu, 1997) is a binary and simplified synchronous context-free grammar with only one non-terminal symbol. It has three types for the right hand side of the rules γ: *S–straight* keeps the order of child nodes, *I–inverted* reverses the order, and *T–terminal* generates a terminal symbol.

$$X \rightarrow \gamma = \begin{cases} [X_1 X_2] & \textit{straight} \\ < X_1 X_2 > & \textit{inverted} \\ f/e & \textit{terminal} \end{cases} \quad (1)$$

where X, X_1, X_2 are non-terminal symbols and f/e is a source/target phrase pair. BTG provides an easy and simple mechanism for modeling word permutation across languages. Figure 1 illustrates this mechanism.

There exists some solutions for BTG grammar induction, which typically focus on unsupervised approaches, like inside-outside algorithm (Pereira and Schabes, 1992) for probabilistic context-free grammar (PCFG), monolingual bracketing representation (Klein and Manning, 2002) or bilingual bracketing grammar induction (Wu, 1995). The common problem is that these models suffer from a higher computational complexity.

Some supervised versions focus on supervised approach, ranging from simple flat reordering model (Wu, 1997), maximum-entropy based model (Zens and Ney, 2006; Xiong et al., 2008) and Tree Kernel-based SVM (Zhang and Li, 2009). Other approaches, use pre-annotated treebanks to train a monolingual/synchronous parser (Collins and Roark, 2004; Genzel, 2010). In this case, the rules are learned directly from the treebank. The majority of works (Zhang and Gildea, 2005; Xiong et al., 2008) rely on syntactic parsers available in one of a source or target language.

However, bilingual parallel treebanks are not always available. As to building a bilingual synchronous parser using the BTG formalism, there exist rare works without the use of such a constituency/dependency parser, and sometimes bilingual parallel treebanks are not always available. Zens and Ney (2006) and DeNero and Uszkoreit (2011) proposed semi-supervised approaches for synchronous grammar induction based on source-side information only when bilingual word alignments are given in advance, instead of training the parser in a supervised way on pre-annotated treebanks. This strategy does not require syntactic annotations in the training data, making training easier.

Rather than developing a novel BTG-decoder incorporated with a BTG-based reordering model, using reordering models for *preordering* have been widely explored to improve the standard phrase-based statistical machine translation system. Neubig et al. (2012) present a bottom-up method for inducing a preorder for SMT by training a discriminative model to minimize the loss function on the hand-aligned corpus. Their method makes use of the general framework of large margin online structured prediction (Crammer et al., 2006). Lerner and Petrov (2013) present a simple classifier-based preordering approach using the source-side dependency tree. Nakagawa (2015) further develop a more efficient top-down incremental parser for preordering

via online training using simple structured Perceptron algorithm. Differing from the mentioned methods to pre-reorder the sentence before the phase of decoding, in this paper; we propose to build a reordering model directly for building a BTG-based decoder.

3 BTG-based Machine Translation

Given the three types of rules in Equation 1, we define a BTG derivation $\mathcal{D}$ as a sequence of independent operations $d_1, \ldots, d_K$ that apply bracketing rules $X \rightarrow \gamma$ as each stage when parsing a source-target sentence pair $< \mathbf{f}, \mathbf{e} >$. We write $\mathcal{D} = [d_1, \ldots, d_k, \ldots, d_K]$. We can produce one single BTG tree accordingly for one given $\mathcal{D}$. The probability of a synchronous derivation (parse tree) under the framework of Probabilistic Synchronous Context Free Grammar (PSCFG) is computed as:

$$P(\mathcal{D}) = \prod_{d \in \mathcal{D}} P(d : X \rightarrow \gamma) \tag{2}$$

where $d : X \rightarrow \gamma$ stands for the derivation with the grammar rule $X \rightarrow \gamma$. Given an input sentence pair $< \mathbf{f}, \mathbf{e} >$ and the word alignment $\mathbf{a}$, the problem of finding the best derivation $\tilde{\mathcal{D}}$ can be defined as:

$$\tilde{\mathcal{D}} = \arg\max_{\mathcal{D}} P(\mathcal{D}|\mathbf{e}, \mathbf{f}, \mathbf{a}) \tag{3}$$

In the real case of machine translation, we do not know the word alignment $\mathbf{a}$ when training set is the parallel corpus. In order to find the best translation $\tilde{\mathbf{e}}$ from all translation candidates, we assume two latent variables $\mathbf{a}, \mathcal{D}$ were required as following:

$$\tilde{\mathbf{e}} = \arg\max_{\mathbf{e}} P(\mathbf{e}|\mathbf{f}) \tag{4}$$

$$\propto \arg\max_{\mathbf{e}} P(\mathbf{e}, \mathcal{D}, \mathbf{a}|\mathbf{f}) \tag{5}$$

$$\propto \arg\max_{\mathbf{e}} P(\mathcal{D}|\mathbf{a}, \mathbf{f}, \mathbf{e}) \times P(\mathbf{a}|\mathbf{f}, \mathbf{e}) \times P(\mathbf{e}) \tag{6}$$

In Equation 6, $P(\mathbf{e})$ is the language model and $\mathbf{a}, \mathcal{D}$ are latent variables that should be learnt from the training data. The generative story of Equation 6 is understood as follows: Once we found the hidden word alignment $\mathbf{a}$ with an alignment model $P(\mathbf{a}|\mathbf{f}, \mathbf{e})$ and the hidden derivation $\mathcal{D}$ using BTG-based reordering model $P(\mathcal{D}|\mathbf{a}, \mathbf{f}, \mathbf{e})$, we can translate the input source sentence $\mathbf{f}$ with the target translation $\tilde{\mathbf{e}}$.

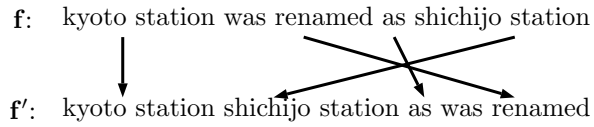

Figure 2: Example of preordering a source sentence given the target word order.

3.1 Training Alignment Model

There are two sub-models in Equation 6, one is the alignment model $P(\mathbf{a}|\mathbf{f}, \mathbf{e})$ and the other one is the reordering model $P(\mathcal{D}|\mathbf{a}, \mathbf{f}, \mathbf{e})$. Since state-of-the-art alignment methods yield high-quality word-to-word alignments, it is not necessary to design a new alignment model to obtain the intermediate variable $\mathbf{a}$. We use the standard method to get word-to-word alignments.

3.2 Training Reordering Model

Recently, some research also showed that treating the parse tree as latent variables (Loehlin, 1998) can benefit the BTG tree inference but for preordering (see Figure 2). The reordering model is trained to maximize the conditional likelihood of trees that license the reorderings implied by observed word alignments in a given parallel corpus. For example, Neubig et al. (2012) proposed a BTG-based reordering model trained from word-aligned parallel text directly. With assuming that there is an underlying derivation $\mathcal{D}$ that produced $\mathbf{f}'$, where $\mathbf{f}'$ is the reordered source sentence given the corresponding target word orders under the constraints of BTGs.

$$\mathbf{f} \xrightarrow[\mathbf{a}]{\text{preordering with } \mathcal{D}} \mathbf{f}' \tag{7}$$

To learn such a reordering model, they handled the derivations $\mathcal{D}$ as a latent variable directly from the source side linguistic contexts. The objective function in their work can be represented as:

$$\tilde{\mathbf{f}}' = \arg\max_{\mathbf{f}'} Score(\mathbf{f}', \mathcal{D}|\mathbf{f}) \tag{8}$$

Since their model is based on reorderings $\mathbf{f}'$ licensed by BTG derivations $\mathcal{D}$, notes $\mathcal{D} \rightarrow \mathbf{f}'$, the objective function also can be written as:

$$\tilde{\mathcal{D}} = \arg\max_{\mathcal{D} \rightarrow \mathbf{f}'} Score(\mathcal{D}|\mathbf{f}) \tag{9}$$

The learning problem defined here is fairly simple. With treating the derivation $\mathcal{D}$ as the latent variable, they want to find the derivation with maximal score of $Score(\mathcal{D}|\mathbf{f})$. Furthermore, following (Collins, 2002; Collins and Roark, 2004), they assume that $Score(\mathcal{D}|\mathbf{f})$ is the linear combination of feature functions defined over $\mathcal{D}$ and $\mathbf{f}$.

Because it is also possible to apply the score function $Score(\mathcal{D}|\mathbf{f})$ as a reordering model during the BTG-based decoding, following (Neubig et al., 2012; Nakagawa, 2015), we propose to build such a reordering model with latent derivation for decoding instead of preordering. The natural difference between their works and our work is as follows: In (Neubig et al., 2012; Nakagawa, 2015), they train an incremental parser for preordering, following the order in the target language before decoding, but we do reordering while decoding. In other words, we adopt their model but make use of it as an online reordering heuristic during decoding.

4 Proposed Methods

In our method, we propose to train and use a BTG-based reordering model in three steps. Firstly, we train the BTG parser on the source side with shallow annotations (only POS-tags and word classes (Brown et al., 1992)) on word-aligned bilingual data. Then we select a large mount of features of unigrams, bigrams, and trigrams to represent the current parser state and we estimate feature weights using a Structured Perceptron (Nakagawa, 2015). Finally, the log-linear combination score for the current state is computed again during decoding. This works as an additional heuristic score and helps the decoder to select the best candidates in sub-hypothesis combination.

4.1 Reordering

We define a reordering model Φ_{RM} as a model composed of a *straight* reordering model Φ_{RMs} and an inverted reordering model Φ_{RMi}. $\mathcal{R}$ stands for the composition of Φ_{RMs} and Φ_{RMi}.

$$\mathcal{R} = \{\Phi_{RMs}, \Phi_{RMi}\} \tag{10}$$

Given a source sentence $\mathbf{f}$, we define the score for $\mathcal{R}$ the weighted sum of the score $\mathcal{P}(d)$ of the sub-derivation d at each parse state defined over D given a source sentence $\mathbf{f}$.

$$\mathcal{R}(\mathcal{D}|\mathbf{f}) = \sum_{d \in \mathcal{D}} \mathcal{P}(d : X \to \gamma) \tag{11}$$

Each atomic derivation d which belongs to $\mathcal{D}$ is weighted with various features in a log-linear form as (Xiong et al., 2008; Duan et al., 2009):

$$\mathcal{P}(d : X \to \gamma) = \sum_{\phi_i \in d} \pi_i \phi_i \tag{12}$$

where ϕ_i is the ith feature function and π_i is the ith weight can be trained on the training data.

Suppose that we know the word alignment $\mathbf{a}$. We want to train a parser which maximizes the number of times the source sentences in the training data are successfully parsed under the constraints of BTGs. Nakagawa (2015) propose an efficient top-down parser via online training for this problem. He uses a simple structured perceptron algorithm.

We assume that the parser has an independent state in each step. We define the parse state as a triple $\langle X, r, d \rangle$, where X is an unparsed span. For example, following the deductive proof system representations (Shieber et al., 1995; Goodman, 1999), $[X, p, q]$ covers $f_p, \ldots, f_q$. $d = \langle r, X \to \gamma \rangle$ is the derivation at the current state with r is the splitting position between f_{r-1} and f_r and $X \to \gamma$ is the applied BTG rule. To extract the features used to score the model, we assume that each word in a sentence has three types of features: lexical form, part-of-speech (POS) tag and word class (Brown et al., 1992) as (Nakagawa, 2015). We extract the unigrams, bigrams, and trigrams at each parse state and compute the model score defined in Equation 12[1].

$\mathbf{f}$: kyoto station was renamed as shichijo station
1 2 3 4 5 6 7

Derivations:

1. $f_1^7 \to [f_1^2\ f_3^7]$
2. $f_1^2 \to [f_1\ f_2]$
3. $f_3^7 \to \langle f_3^5\ f_6^7 \rangle$
4. $f_3^5 \to [f_3^4\ f_5]$
5. $f_3^4 \to \langle f_3\ f_4 \rangle$
6. $f_6^7 \to [f_6\ f_7]$

Figure 3: Example of step-by-step atomic derivations.

[1]We use the same set of features described in (Nakagawa, 2015)

Algorithm 1: Training the Reordering Model

Input : Training data $\{\langle \mathbf{e}, \mathbf{f}, \mathbf{a} \rangle\}_0^L$
Output: Feature weights $\boldsymbol{\pi}$ for $\mathcal{R}$

```
foreach iteration t do
    foreach example ⟨e, f, a⟩ do
        D̂ = arg max_D R(D|f);
        D* = arg max_{D ∧ Constraint(D,a,e,f)} R(D|f);
        if D̂ ≠ D* then
            π ← π + R(D*, f) − R(D̂, f);
        end
    end
    return π;
end
```

The training algorithm (see Algorithm 1) can be described briefly as following: The parser first produces a *system derivation* $\hat{\mathcal{D}}$ with the maximum model score given $\mathbf{f}$. If $\hat{\mathcal{D}}$ is not licensed by BTG constraints also given $(\mathbf{e}, \mathbf{a})$, we consider the parser entered a failure state and stop it. Another *oracle derivation* $\mathcal{D}^*$ was also selected, which satisfied the constraint of BTGs (notes $\textit{Constraint}(\mathcal{D}, \mathbf{a}, \mathbf{e}, \mathbf{f}) = true$). If the *system derivation* $\hat{\mathcal{D}}$ and the *oracle derivation* $\mathcal{D}^*$ are not equivalent, we update the model weights $\boldsymbol{\pi}$ towards $\mathcal{D}^*$.

Like all structured prediction learning frameworks, the online Structured Perceptron is costly to train as training complexity is proportional to inference, which is frequently non-linear in the length of example. To train the reordering model, we employ an in-house parser[2] which uses Batch Perceptron. It is a modified and boosted version of the original top-down parser (Nakagawa, 2015), which allows us to train on the whole training set[3].

4.2 Decoding

In decoding, we follow (Och and Ney, 2002; Chiang, 2007). That is, we remove the target side and use a more general linear model composition over derivations:

$$\tilde{\mathbf{e}} = \arg\max_{\mathbf{e}} P(\mathbf{e}, \mathcal{D}|\mathbf{f}) \tag{13}$$

$$\propto \arg\max_{\mathcal{D} \to \mathbf{e}} \textstyle\prod_i \Phi_i(\mathcal{D})^{\lambda_i} \tag{14}$$

where each Φ_i is a sub-model score function and λ is the corresponding weight. For each arbitrary score function Φ_i with a derivation D, we decompose it as a chain of independent derivations d with BTG rules $X \to \gamma$:

$$\Phi_i(\mathcal{D}) = \prod_{d \in \mathcal{D}} \Phi_i(d : X \to \gamma) \tag{15}$$

Therefore, given an input sentence $\mathbf{f} = f_1, \ldots, f_n$, notes f_1^n, the task to translate an input source sentence can be solved by finding the derivation with maximal score in Equation 14, which uniquely determines a target translation $\hat{\mathbf{e}}$ (e_1^m) with this latent derivation $\mathcal{D}$.

The decoder needs to generate all derivations for each segment spanning from f_i to f_j $(0 \leq i < j \leq n)$. Since our goal is to find the best derivation $\hat{D}$ that covers the whole input sentence $[f_1, \ldots, f_n]$, we employ a CKY-style decoder to generate the best derivation $\hat{\mathcal{D}}$ for each source sentence. This yields the best translation $\hat{\mathbf{e}}$ (e_1^m) at the same time.

4.2.1 The *-LM -RM* Decoder

The integration of a standard n-gram-based language model into a CKY-style decoder is not easy as in the standard phrase-based method (Koehn et al., 2003). Following (Chiang, 2007), we first introduce the *-LM -RM* model in which the reordering and language model are removed from the decoding model:

$$w(\mathcal{D}) = \prod_{i \notin \{RM, LM\}} \Phi_i(\mathcal{D})^{\lambda_i} \tag{16}$$

Using the deductive proof system (Shieber et al., 1995; Goodman, 1999) to describe our *-LM -RM* decoder, the inference rules are the following:

$$\frac{X \to f/e}{[X, p, q] : w} \tag{17}$$

$$\frac{X \to \langle X_1, X_2 \rangle : \quad [X_1, p, r] : w_1 \; [X_2, r+1, q] : w_2}{[X, p, q] : w_1 w_2} \tag{18}$$

$$\frac{X \to [X_1, X_2] : \quad [X_1, p, r] : w_1 \; [X_2, r+1, q] : w_2}{[X, p, q] : w_1 w_2} \tag{19}$$

[2] https://github.com/wang-h/HieraParser

[3] We skip the sentences which cannot be parsed under the constraints of BTGs.

where $X \rightarrow \gamma$ is the derivation rule, $[X, p, q]$ is the subtree rooted in a non-terminal X (see Section 2), w is the model score defined in Equation 16. When all terms on the top line are true, the item on the bottom line is derived. The final goal for the decoder is $[\mathbf{f}, 1, n]$, where $\mathbf{f}$ is the whole source sentence.

During decoding, the *-LM -RM* decoder flexibly explores the derivation without taking reordering into account. This strategy is a simple way to build a CYK-style decoder, but the decoder requires very large beam size to find the true best translation. Incorporating the *LM* and *RM* model directly into the translation construction will improve efficiency.

4.2.2 The *+LM +RM* Decoder

The computational complexity of online strategy is reduced by using dynamic programming and incorporating the language model and the reordering model into decoding. The similar method has been described in (Chiang, 2007). The decoder integrated with the n-gram language model is called: "*+LM* decoder". In our case, we also need to integrate the reordering model, so we call it "*+LM +RM* decoder". Given the inference rules described in Equations 17–19, we describe the *+LM +RM* decoding algorithm using Equations 20–23.

In our case, the reordering model affects computing the language model score if the derivation requires to swap the target sub-charts. We can calculate $\Phi_{RM}(X)$ by just taking the model score as the product of two sub-charts $\Phi_{RM}(X_1)$ and $\Phi_{RM}(X_2)$ with current reordering score $\Phi_{RM}(X \rightarrow \gamma)$. Since $\mathcal{R}$ is a log-linear expression, we compute the reordering score $\mathcal{R}(X)$ for a given span $X : [X, p, q]$ that consists of $X_1 : [X_1, p, r]$ and $X_2 : [X_2, r+1, q]$ with a grammar rule $X \rightarrow \gamma$ as:

$$\mathcal{R}(X) = \mathcal{R}(X_1) + \mathcal{R}(X_2) + \mathcal{P}(X \rightarrow \gamma) \quad (24)$$

When we merge the chart $X_1 : [X_1, p, r]$ with $X_2 : [X_2, r+1, q]$ using the rule $X \rightarrow \gamma$, we update the total score for the composition model after applying each rule dynamically, we call this the *+RM* strategy. The BTG *terminal* rule ($T : X \rightarrow f/e$) is used to translate the source phrase f into the target phrase e while the *straight* and *inverted* rules ($S : X \rightarrow [X_1 X_2]$ and $I : X \rightarrow < X_1 X_2 >$) are used to concatenate two neighbouring phrases with a *straight* or *inverted* order as following:

$$e_x^y = \begin{cases} e_1 \cdot e_2, & X \rightarrow [X_1 X_2] \\ e_2 \cdot e_1, & X \rightarrow \langle X_1 X_2 \rangle \end{cases} \quad (25)$$

where $\cdot$ stands for concatenation between strings. After having decided the word order on the target side, we compute the score in the language model, noted $\mathcal{L}(\cdot)$[4]. The language model score $P_{LM}(e_x^y)$ depends on the preceding $N-1$ words for any $e_x^y (|e_x^y| \geq N, 1 \leq x < y \leq m)$. It is computed as:

$$P_{LM}(e_x^y) = \prod_{x \leq z \leq y} p(\hat{e}_{z+N-1} | \hat{e}_z \dots \hat{e}_{z+N-2}) \quad (26)$$

The language model score function $\mathcal{L}(e_x^y)$ depends on the rule type γ as follows:

$$\mathcal{L}(e_x^y) = \begin{cases} P_{LM}(e_x^{y+1}), & |e_x^y| = |e_1^m| \\ 0, & |e_x^y| < N \\ P_{LM}(e_{x+N}^y), & \text{otherwise} \end{cases} \quad (27)$$

To determine whether we have the case $|e_x^y| = |e_1^m|$, we assume that, if the span of $X : [X, p, q]$ covers the entire source sentence f_1^n as $X : [X, 1, n]$, then the target translation e_x^y should also cover the entire target sentence. On the basis of *+RM* decoder, we add the *+LM* component into the decoder and build a *+LM+RM* decoder for CYK-style bottom-up decoding. cube pruning (Chiang, 2007) was also applied to speedup the decoder.

4.2.3 Model Combination

HieraTrans is our newly-developed in-house BTG-based SMT translation platform. It adopts the constraints of BTG in both phrase translation and reordering. We combine the models in a log-liner manner as shown in Equation 14. The feature functions employed by HieraTrans are:

- Phrase-based translation models (TM): direct and inverse phrase translation probabilities, direct and inverse lexical translation probabilities.
- Language model (LM)

[4]For the case of start-of-the sentence and end of the sentence, we wrap the target sentence $\mathbf{e}$ (e_1^m) as $\hat{\mathbf{e}} = \hat{e}_1^{m+h} = \langle s \rangle^{N-1} e_1^m \langle \backslash s \rangle$.

$$\frac{X \to f/e}{[X,p,q] : w[\mathcal{L}(e)]^{\lambda_{\mathcal{L}}}} \tag{20}$$

$$\frac{X \to \langle X_1, X_2 \rangle : \quad [\exp \mathcal{P}(X \to \langle X_1, X_2 \rangle)]^{\lambda_{\mathcal{R}}} \quad [X_1,p,r] : w_1 \; [X_2, r+1, q] : w_2}{[X,p,q] : w_1 w_2 [\exp \mathcal{R}(X)]^{\lambda_{\mathcal{R}}} [\mathcal{L}(e_2 + e_1)]^{\lambda_{\mathcal{L}}}} \tag{21}$$

$$\frac{X \to [X_1, X_2] : \quad [\exp \mathcal{P}(X \to [X_1, X_2])]^{\lambda_{\mathcal{R}}} \quad [X_1,p,r] : w_1 \; [X_2, r+1, q] : w_2}{[X,p,q] : w_1 w_2 [\exp \mathcal{R}(X)]^{\lambda_{\mathcal{R}}} [\mathcal{L}(e_1 + e_2)]^{\lambda_{\mathcal{L}}}} \tag{22}$$

$$X_1 \to f_1/e_1, X_2 \to f_2/e_2 \tag{23}$$

- Reordering models (RM): *straight* and *inverted* scores combined within the log-linear framework.
- Penalties (PM): word penalty, phrase penalty, unknown word penalty.

The weights for each feature are tuned and estimated using the minimum error rate training (MERT) algorithm (Och, 2003).

5 Experiment

5.1 Experimental Setup

To evaluate our system, we conducted translation experiments on the KFTT Corpus (English–Japanese) and compared our system with baseline phrase-based (PB) and hierarchical phrase-based (HPB) SMT implementations in Moses[5] (Koehn et al., 2007). For each language, the training corpus is around 330,000 sentences. The development set contains nearly 1,235 sentences and nearly 1,160 sentences used for testing. We use the default training set for training translation model, and traditional lexical (Koehn et al., 2005) reordering model or our proposed BTG-based reordering model, and also target language model. We use the default tuning set for tuning the parameters and the default test set for evaluation.

For word alignment, we train word alignments in both directions with the default settings, i.e., the standard bootstrap for IBM model 4 alignment in `GIZA++` ($1^5H^5 3^3 4^3$). We then symmetrize the word alignments using *grow-diag-final-and* (+*gdfa*) and the standard phrase extraction heuristic (Koehn et al., 2003) for all systems. In our experiment, the maximum length of phrases entered into phrase table is limited to 7, and we input only the top 20 translation candidates. The language model storage of target language uses the implementation in KenLM (Heafield, 2011) which is trained and queried as a 5-gram model. For distortion model in phrase-based SMT baseline, we set the distortion limit to 6.

Word alignments used for training the reordering model are the intersection of both asymmetrical alignments in each mono-direction output by GIZA++[6] (Och and Ney, 2003). For pos-tagging, we make use of the Stanford Log-linear POS Tagger[7] (Toutanova and Manning, 2000). To produce word class tags for each source word, we use the implementation of (Liang, 2005) [8] of Brown's clustering algorithm (Brown et al., 1992). The size of the class tags is fixed to 256.

For tuning, the optimal weights for each feature are estimated using the minimum error rate training (MERT) algorithm (Och, 2003) and parameter optimization with ZMERT[9] (Zaidan, 2009).

5.2 Experiment Results

For evaluation of machine translation quality, standard automatic evaluation metrics are used, like BLEU (Papineni et al., 2002) and RIBES (Isozaki et al., 2010) in all experiments. BLEU is used as the default standard metric, RIBES takes more word order into consideration. Table 1 shows the performance of MT systems on the KFTT test data, which are (1) Moses, trained using the phrase-based model (PB-SMT). (2) Moses, trained using the hierarchical phrase-based model (HPB-SMT) and last one (3) HieraTrans, trained using the BTG-based model

[5] http://www.statmt.org/moses/

[6] http://www.statmt.org/moses/giza/GIZA++.html

[7] https://nlp.stanford.edu/software/tagger.shtml

[8] https://github.com/percyliang/brown-cluster

[9] http://www.cs.jhu.edu/ ozaidan/zmert/

	BLEU	RIBES
Moses (PB-SMT)	20.81	67.50
Moses (HPB-SMT)	**21.67**	66.58
HieraTrans (BTG-SMT)		
(beam=40)	**21.15**	65.80
(beam=100)	**21.24**	66.33

Table 1: Results on phrase-based baseline system, hierarchical phrase-based system and our BTG-based system. Bold scores indicate no statistically significant difference at $p < 0.05$ from the best system (Koehn, 2004).

(BTG-SMT).

5.3 Analysis

Compared with the PB-SMT, BTG-based SMT uses weak linguistic annotations on the source side which provides additional information for reordering. We found that this strategy does help tree structure construction and finding final translations. However, our BTG-based method underperformed the HPB-SMT method. Increasing the beam size will gain improvement slightly.

There are two explanations for the result: First, final machine translation performance is also related to the used tools, which is sensitive to parse errors, alignment errors or annotation errors. Inaccurate labeling hurts the performance. Second, strict constraints of BTGs makes the decoder difficult to find some discontinuous phrases (translations).

6 Conclusion

In this paper, we proposed a novel BTG-based translation approach using a BTG-based reordering model directly trained from the training data. Training such a reordering model does not require any syntactic annotations, hence no use of treebanks or parsers. This approach provides an alternative to building a BTG-based machine translation system using syntactic information. We also made several improvements over (Xiong et al., 2008): First, we developed a novel BTG-based parser using Batch Perceptron. It allows training the reordering model on the whole training set. Second, we made the reordering model serve as a model which can be queried during decoding. We compared and validated our method can achieve the comparable performance with state-of-the-art SMT approaches. For further improvements, we will work on towards higher-speed decoder and make the decoder open available.

Acknowledgments

This work is supported in part by China Scholarship Council (CSC) under the CSC Grant No.201406890026. We also thank the anonymous reviewers for their insightful comments.

References

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

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

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. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585.

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. Association for Computational Linguistics.

Xiangyu Duan, Deyi Xiong, Hui Zhang, Min Zhang, and Haizhou Li. 2009. I2r's machine translation system for iwslt 2009. In *IWSLT*, pages 50–54.

Michel Galley and Christopher D Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 848–856. Association for Computational Linguistics.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of the 23rd international conference on computational linguistics*, pages 376–384. Association for Computational Linguistics.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, David Talbot, and Michael White. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *IWSLT*, pages 68–75.

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. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395. Citeseer.

Uri Lerner and Slav Petrov. 2013. Source-side classifier preordering for machine translation. In *EMNLP*, pages 513–523.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

John C Loehlin. 1998. *Latent variable models: An introduction to factor, path, and structural analysis*. Lawrence Erlbaum Associates Publishers.

Tetsuji Nakagawa. 2015. Efficient top-down btg parsing for machine translation preordering. In *ACL (1)*, pages 208–218.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 843–853. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 160–167. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Stuart M Shieber, Yves Schabes, and Fernando CN Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of logic programming*, 24(1):3–36.

Christoph Tillmann. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 101–104. Association for Computational Linguistics.

Kristina Toutanova and Christopher D Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 63–70. Association for Computational Linguistics.

Dekai Wu. 1995. Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora. In *Proceedings*

of the 14th International Joint Conference on Artificial Intelligence, volume 95, pages 1328–1335.

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

Deyi Xiong, Min Zhang, Aiti Aw, and Haizhou Li. 2008. Linguistically annotated btg for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1009–1016. Association for Computational Linguistics.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

Omar Zaidan. 2009. Z-mert: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

Richard Zens and Hermann Ney. 2006. Discriminative reordering models for statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 55–63. Association for Computational Linguistics.

Hao Zhang and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 475–482. Association for Computational Linguistics.

Min Zhang and Haizhou Li. 2009. Tree kernel-based svm with structured syntactic knowledge for btg-based phrase reordering. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 698–707. Association for Computational Linguistics.