Dependency-Based Bracketing Transduction Grammar for Statistical Machine Translation

Jinsong Su, Yang Liu, Haitao Mi, Hongmei Zhao, Yajuan Lü, Qun Liu
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
{sujinsong,yliu,htmi,zhaohongmei,lvyajuan,liuqun}@ict.ac.cn

Abstract

In this paper, we propose a novel *dependency-based bracketing transduction grammar* for statistical machine translation, which converts a source sentence into a target dependency tree. Different from conventional bracketing transduction grammar models, we encode target dependency information into our lexical rules directly, and then we employ two different maximum entropy models to determine the reordering and combination of partial dependency structures, when we merge two neighboring blocks. By incorporating dependency language model further, large-scale experiments on Chinese-English task show that our system achieves significant improvements over the baseline system on various test sets even with fewer phrases.

1 Introduction

Bracketing transduction grammar (BTG) (Wu, 1995) is an important subclass of synchronous context free grammar, which employs a special synchronous rewriting mechanism to parse parallel sentence of both languages.

Due to the prominent advantages such as the simplicity of grammar and the good coverage of syntactic diversities in different language pairs, BTG has attracted increasing attention in statistical machine translation (SMT). In flat reordering model (Wu, 1996; Zens et al., 2004), which assigns constant reordering probabilities depending on the language pairs, BTG constraint proves to be very effective for reducing the search space of phrase reordering. To pursue a better method to predict the order between two neighboring blocks[1], Xiong et al. (2006) present an enhanced BTG with a maximum entropy (ME) based reordering model. Along this line, source-side syntactic knowledge is introduced into the reordering model to improve BTG-based translation (Setiawan et al., 2007; Zhang et al., 2007; Xiong et al., 2008; Zhang and Li, 2009). However, these methods mainly focus on the utilization of source syntactic knowledge, while ignoring the modeling of the target-side syntax that directly influences the translation quality. As a result, how to obtain better translation by exploiting target syntactic knowledge is somehow neglected. Thus, we argue that it is important to model the target-side syntax in BTG-based translation.

Recently, modeling syntactic information on the target side has progressed significantly. Depending on the type of output, these models can be divided into two categories: the *constituent-output* systems (Galley et al., 2006; Zhang et al., 2008; Liu et al., 2009) and *dependency-output* systems (Eisner, 2003; Lin, 2004; Ding and Palmer, 2005; Quirk et al., 2005; Shen et al., 2008). Compared with the *constituent-output* systems, the *dependency-output* systems provide a simpler platform to capture the target-side syntactic information, while also having the best interlingual phrasal cohesion properties (Fox, 2002). Typically, Shen et al. (2008) propose a string-to-dependency model, which integrates the target-side well-formed dependency structure into translation rules. With the dependency structure, this system employs a dependency language model (LM) to exploit long distance word relations, and achieves a significant improvement over the hierarchical phrase-based system (Chiang, 2007). So

[1] A block is a bilingual phrase without maximum length limitation.

Coling 2010: Poster Volume, pages 1185–1193,
Beijing, August 2010

we think it will be a promising way to integrate the target-side dependency structure into BTG-based translation.

In this paper, we propose a novel dependency-based BTG (DepBTG) for SMT, which represents translation in the form of dependency tree. Extended from BTG, our grammars operate on two neighboring blocks with target dependency structure. We integrate target syntax into bilingual phrases and restrict target phrases to the well-formed structures inspired by (Shen et al., 2008). Then, we adopt two ME models to predict how to reorder and combine partial structures into a target dependency tree, which gives us access to capturing the target-side syntactic information. To the best of our knowledge, this is the first effort to combine the translation generation with the modeling of target syntactic structure in BTG-based translation.

The remainder of this paper is structured as follows: In Section 2, we give brief introductions to the bases of our research: BTG and dependency tree. In Section 3, we introduce DepBTG in detail. In Section 4, we further illustrate how to create two ME models to predict the reordering and dependency combination between two neighboring blocks. Section 5 describes the implementation of our decoder. Section 6 shows our experiments on Chinese-English task. Finally, we end with a summary and future research in Section 7.

2 Background

2.1 BTG

BTG is a special case of synchronous context free grammar. There are three rules utilized in BTG:

$$A \rightarrow [A^1, A^2] \quad (1)$$

$$A \rightarrow \langle A^1, A^2 \rangle \quad (2)$$

$$A \rightarrow x/y \quad (3)$$

where the reordering rules (1) and (2) are used to merge two neighboring blocks A^1 and A^2 in a straight or inverted order, respectively. The lexical rule (3) is used to translate the source phrase x into the target phrase y.

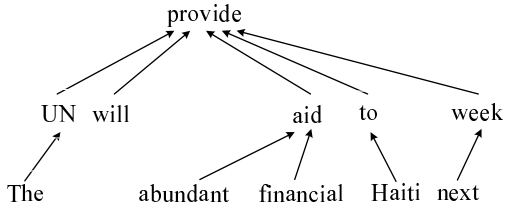

Figure 1: The dependency tree for sentence *The UN will provide abundant financial aid to Haiti next week.*

2.2 Dependency Tree

In a given sentence, each word depends on a parent word, except for the root word. The dependency tree for a given sentence reflects the long distance dependency and grammar relations between words. Figure 1 shows an example of a dependency tree, where a black arrow points from a child word to its parent word.

Compared with constituent tree, dependency tree directly models semantic structure of a sentence in a simpler form. Thus, it provides a desirable platform for us to utilize the target-side syntactic knowledge.

3 Dependency-based BTG

3.1 Grammars

In this section, we extend the original BTG into DepBTG. The rules of DepBTG, which derive from that of BTG, merge blocks with target dependency structure into a larger one. These rules take the following forms:

$$A_d \rightarrow [A_d^1, A_d^2]_{CC} \quad (4)$$

$$A_d \rightarrow [A_d^1, A_d^2]_{LA} \quad (5)$$

$$A_d \rightarrow [A_d^1, A_d^2]_{RA} \quad (6)$$

$$A_d \rightarrow \langle A_d^1, A_d^2 \rangle_{CC} \quad (7)$$

$$A_d \rightarrow \langle A_d^1, A_d^2 \rangle_{LA} \quad (8)$$

$$A_d \rightarrow \langle A_d^1, A_d^2 \rangle_{RA} \quad (9)$$

$$A_d \rightarrow x/y \quad (10)$$

where A_d^1 and A_d^2 represent two neighboring blocks with target dependency structure. Rules (4)$\sim$(9) are used to determine the reordering and combination of two dependency structures, when

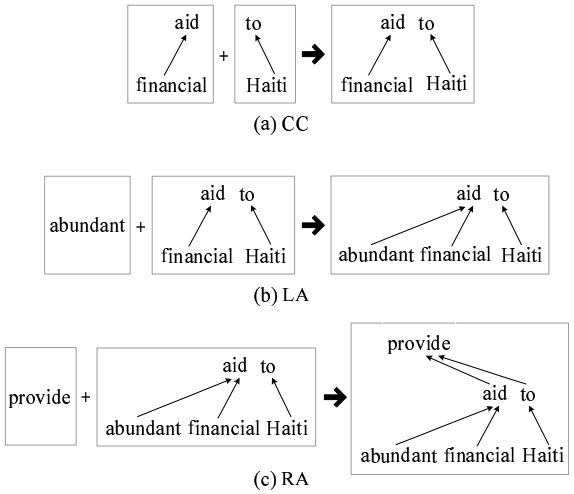

Figure 2: Dependency operations on the neighboring dependency structures. CC = coordinate concatenate, LA = left adjoining, and RA = right adjoining.

we merge two neighboring blocks. Rule (10) is applied to generate bilingual phrase (x, y) with target dependency structure learned from training corpus. To distinguish the rules with different functions, the rules (4)∼(9) and rule (10) are named as **merging rules** and **lexical rule**, respectively.

Specifically, we first merge the neighboring blocks in the straight order using rules (4)∼(6) or in the inverted order using rules (7)∼(9). Then, according to different merging rules, we conduct some operations to combine the corresponding dependency structures in the target order: coordinate concatenate (**CC**), left adjoining (**LA**) and right adjoining (**RA**).

To clearly illustrate our operations, we show the process of applying three dependency operations to build larger structures in Figure 2. Adopting rule (4), the dependency structures "*((financial) aid)*"[1] and "*(to (Haiti))*" can be combined into a larger one consisting of two sibling subtrees (see Figure 2(a)). Adopting rule (5), we can adjoin the left dependency structure "*(abundant)*" to the leftmost sub-root of the right dependency structure "*((financial) aid) (to (Haiti))*" (see Figure 2(b)). Adopting rule (6), we can include the right dependency structure "*((abundant) (financial) aid) (to (Haiti))*" as a child of the rightmost sub-root of the left dependency structure "*(provide)*" (see Figure 2(c)). In a similar way, rules (7)∼(9) are applied to deal with two partial structures in the inverted order.

[1]We use the lexicon dependency grammar (Hellwig, 2006) to express the projective dependency tree. Using this grammar, the words in the brackets are defined as the child words depending on the parent word outside the brackets.

3.2 Well-Formed Dependency Structures

As illustrated in the previous sub section, the rules of DepBTG operate on the blocks with target dependency structure. Following (Shen et al., 2008), we restrict the target phrases to the well-formed dependency structures. The main difference is that we use more relaxed constraints to extract more bilingual phrases with rational structure. Take a sentence $S = w_1 w_2 ... w_n$ for example, we denote the parent word ID of word w_i with d_i, and show the definitions of structures as follows.

Defination 1 *A dependency structure $d_{i...j}$ is* **fixed on head** *h, where $h \in [i, j]$, if and only if it meets the following conditions*

- $d_h \notin [i, j]$
- $\forall k \in [i, j]\ and\ k \neq h, d_k \in [i, j]$
- $\forall k \in [i, j],\ d_k = h\ or\ d_k \in [i, j]$

Defination 2 *A dependency structure $d_i ... d_j$ is* **floating with children** C, *for a non-empty set $C \subseteq \{i...j\}$, if and only if it meets the following conditions*

- $\exists h \notin [i, j], s.t. \forall k \in C, d_k = h$
- $\forall k \in [i, j]\ and\ k \notin C, d_k \in [i, j]$
- $\forall k \notin [i, j],\ d_k \notin [i, j]$ $\underline{or\ d_k = c_l}$ $\underline{or\ d_k = c_r}$

where c_l and c_r represent the IDs of the leftmost and rightmost words in the set *C*, respectively. Note that the underline indicates the difference between our definition and that of (Shen et al., 2008). In our model, we regard the floating structure, which is not complete on its boundary sub-roots, as an useful structure, since it will become a complete constituent by combining it with other partial structures. For example, the dependency

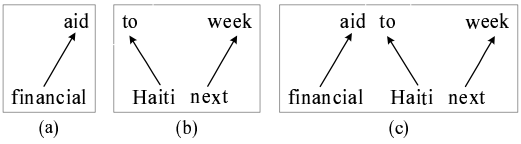

Figure 3: (a) A fixed structure and (b) (c) two floating structures. Note that (c) is ill-formed in (Shen et al., 2008).

structures shown in Figure 3 are all well-formed structures. However, according to the definitions of (Shen et al., 2008), 3(c) is ill-formed because *aid* does not include its leftmost child word *abundant* in the structure.

4 ME Models for Merging Rules

4.1 The Models

A simple way to estimate the probabilities of the merging rules is to adopt maximum likelihood estimation to obtain the conditional probabilities. However, this method is not applicable to merging rules because the dependency structures become larger and larger during decoding, which are very sparse in the corpus.

Inspired by MEBTG translation (Xiong et al., 2006), which considers phrase reordering as a classification problem, we model the reordering and combination of two neighboring dependency structures based on the ME principle. Owing to data sparseness and the complexity of multi-class classification, we establish two ME models rather than an unified ME model: one for the reordering between blocks, called **reordering model**; the other for the dependency operations on the corresponding dependency structures, called **operation model**.

Thus, according to the ME scheme, we decompose the probability Ω of each merging rule into

$$\begin{aligned}\Omega &= p_{\theta_1}(o|A_d^1, A_d^2) \cdot p_{\theta_2}(d|A_d^1, A_d^2) \\ &= \frac{\exp(\Sigma_i \theta_{1i} h_{1i}(o, A_d^1, A_d^2))}{\Sigma_o \exp(\Sigma_i \theta_{1i} h_{1i}(o, A_d^1, A_d^2))} \cdot \\ &\quad \frac{\exp(\Sigma_j \theta_{2j} h_{2j}(d, A_d^1, A_d^2))}{\Sigma_d \exp(\Sigma_j \theta_{2j} h_{2j}(d, A_d^1, A_d^2))}\end{aligned}$$

where the functions $h_{1i} \in \{0, 1\}$ are the features of the ME-based reordering model, θ_{1i} are the corresponding weights, and $o \in \{straight, inverted\}$. Similarly, the functions $h_{2j} \in \{0, 1\}$ and the weights θ_{2j} are trained for the ME-based operation model, and $d \in \{CC, LA, RA\}$.

4.2 Example Extraction

To train the ME models, we extract examples from a string-to-dependency word-aligned corpus during the process of bilingual phrases extraction (Koehn et al., 2005), and then collect various features for the models.

For the reordering model, we adopt the method of (Xiong et al., 2006) to extract reordering examples. Due to the limit of space, we skip the details of this method.

For the operation model, given an operation training example consisting of two neighboring dependency structures: the left structure d_l and the right structure d_r, we firstly classify it into different categories by the dependency relation between d_l and d_r:

- if d_l and d_r have the same parent, the category of the example is CC;
- if d_l depends on the leftmost sub-root of d_r, the category of the example is LA;
- if d_r depends on the rightmost sub-root of d_l, the category of the example is RA.

For instance, Figure 4 shows an operation example with RA operation, where the sub-root word *week* of d_r depends on the rightmost sub-root word *provide* of d_l.

Then, we collect various features from the following nodes: the rightmost sub-root of d_l, and its rightmost child node; the leftmost sub-root of d_r, and its leftmost child node. Here, we speculate that these nodes may carry useful information for the dependency combination of the two structures, since they locate nicely at the boundary subtrees of d_l and d_r. For simplicity, we refer to these nodes as the *feature nodes* of the example. Let's revisit Figure 4, the feature nodes of the example are marked with dashed ellipses. The rightmost sub-root word of d_l is *provide*, and its rightmost child word is *to*; The leftmost sub-root word of d_r is *week*, and its leftmost child word is *next*.

Type	Name	Description
Lexical Features	$W_{lh}(d_r)$	The leftmost sub-root word of d_r
	$W_{rh}(d_l)$	The rightmost sub-root word of d_l
	$W_{llc}(d_r)$	The leftmost child word of $W_{lh}(d_r)$
	$W_{rrc}(d_l)$	The rightmost child word of $W_{rh}(d_l)$
POS Features	$P_{lh}(d_r)$	The POS of $W_{lh}(d_r)$
	$P_{rh}(d_l)$	The POS of $W_{rh}(d_l)$
	$P_{llc}(d_r)$	The POS of $W_{llc}(d_r)$
	$P_{rrc}(d_l)$	The POS of $W_{rrc}(d_l)$

Table 1: Feature categories in the ME-based operation model.

Type	Features and Instances
Unigram Features	$W_{rh}(d_l)$ = provide $W_{rrc}(d_l)$ = to $W_{lh}(d_r)$ = week $W_{llc}(d_r)$ = next $P_{rh}(d_l)$=VV $P_{rrc}(d_l)$ = TO $P_{lh}(d_r)$ = NN $P_{llc}(d_r)$ = ADJ
Bigram Features	$W_{rh}(d_l)\_W_{lh}(d_r)$ = provide_week $W_{rh}(d_l)\_P_{lh}(d_r)$ = provide_NN $P_{rh}(d_l)\_W_{lh}(d_r)$ = VV_week $P_{rh}(d_l)\_P_{lh}(d_r)$ = VV_NN
	$W_{rh}(d_l)\_W_{llc}(d_r)$ = provide_next $W_{rh}(d_l)\_P_{llc}(d_r)$ = provide_ADJ $P_{rh}(d_l)\_W_{llc}(d_r)$ = VV_next $P_{rh}(d_l)\_P_{llc}(d_r)$ = VV_ADJ
	$W_{rrc}(d_l)\_W_{lh}(d_r)$ = to_week $W_{rrc}(d_l)\_P_{lh}(d_r)$ = to_NN $P_{rrc}(d_l)\_W_{lh}(d_r)$ = TO_week $P_{rrc}(d_l)\_P_{lh}(d_r)$ = TO_NN

Table 2: ME operation features and instances of the example shown in Figure 4.

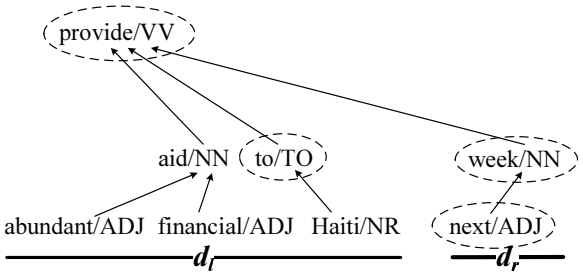

Figure 4: An example with RA category consisting of the neighboring dependency structures d_l and d_r. The dashed ellipses denote the *feature nodes* of the example, and each node consists of one word and its corresponding POS tag.

In addition, to keep the number of operation examples acceptable, we follow (Xiong et al., 2006) to only extract the smallest one from the examples with the same *feature nodes* in each sentence.

4.3 Features

To capture reordering information, we use the boundary words of bilingual blocks as features, which are proved to be very effective in (Xiong et al., 2006).

To capture dependency operation information, we design two kinds of features on the *feature nodes*: the Lexical features and Parts-of-speech (POS) features. With the POS features, the operation ME model will do exact predicating to the best of its ability, and then can back off to approximately predicating if exact predicating fails. Table 1 shows these feature categories in detail.

Furthermore, we also use some bigram features, since it is generally admitted that the combination of different features can lead to better performance than unigram features. To better understand our operation features, we continue with the example shown in Figure 4, listing features and instances in Table 2.

5 Implementation Details

5.1 Decoder

We develop a CKY-style decoder which uses the following features: (1) Phrase translation probabilities in two directions, (2) Lexical translation probabilities in two directions, (3) N-gram LM

score, (4) ME-based reordering model score, (5) Number of phrases, (6) Number of target words, (7) ME-based operation model score, (8) Dependency LM scores at word level and POS level separately, and (9) Discount on ill-formed dependency structures. Here, the former six features are also used in MEBTG translation.

5.2 Dependency Language Model

Following (Shen et al., 2008), we apply different tri-gram dependency LMs at word level and POS level separately to DepBTG translation.

Given a dependency structure, where w_h is the parent word, $w_L = w_{l_1}...w_{l_n}$ and $w_R = w_{r_1}...w_{r_m}$ are child word sequences on the left side and right side respectively, the probability of a tri-gram is computed as follows:

$$\begin{aligned} & P(w_L, w_R|w_h\text{-as-head}) \\ = \; & P(w_L|w_h\text{-as-head}) \cdot P(w_R|w_h\text{-as-head}) \end{aligned}$$

Here $P(w_L|w_h\text{-as-head})$ can be decomposed into:

$$\begin{aligned} & P(w_L|w_h\text{-as-head}) \\ = \; & P(w_{l_1}|w_h\text{-as-head}) \cdot P(w_{l_2}|w_{l_1}, w_h\text{-as-head}) \\ & ... \cdot P(w_{l_n}|w_{l_{n-1}}, w_{l_{n-2}}) \end{aligned}$$

where '-as-head' is used to distinguish the head word from child word in the language model. In like manner, $P(w_R|w_h\text{-as-head})$ has a similar calculation method.

5.3 Ill-Formed Dependency Structure

To preserve the good coverage of bilingual phrases, we keep some bilingual phrases with the special ill-formed dependency structure. Different from the well-formed structures, where all the children of the sub-roots are complete, these ill-formed structures are not complete on the children of the boundary sub-roots, lacking a well-formed sub structure on the boundary. We consider them as useful structures with gaps, each of which can be combined with some well-formed structures into a larger well-formed one. To reduce the search space, we constrain the number of gap to one on each boundary. During decoding, we directly substitute the gap in a structure with another well-formed structure which has the same direction.

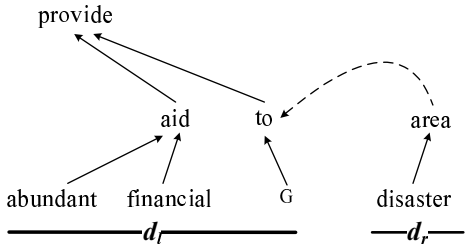

Figure 5: Dependency combination of the ill-formed dependency structure d_l with the right well-formed dependency structure d_r. G denotes gap and the dotted line denotes the substitution of the gap G with d_r.

For example, there are two dependency structures in Figure 5: d_l is an ill-formed structure with a right gap, and d_r is a well-formed one. Instead of investigating three operations to combine these structures, we fill the gap of d_l with d_r, and then compute the corresponding score of the RA operation on the sub structures "*(to)*" and "*((disaster) area)*" in the ME-based operation model.

6 Experiment

6.1 Setup

The training corpus[1] comes from LDC with 1.54M bilingual sentences (41M Chinese words and 48M English words). We run GIZA++ (Och and Ney, 2000) to obtain word alignments with the heuristic method "grow-diag-final-and". Then we parse the English sentences to generate a string-to-dependency word-aligned corpus using the parser (Huang et al., 2009). From this corpus, we extract bilingual phrases with dependency structure. Here, the maximum length of the source phrase is set to 7. For the n-gram LM, we use SRILM Toolkits (Stolcke, 2002) to train a 4-gram LM on the Xinhua portion of the Gigaword corpus. For the dependency LM, we train different 3-gram dependency LMs at word level and POS level separately on the English side of the training corpus.

During the process of bilingual phrase extraction, we collect the neighboring blocks without

[1]The training corpus consists of six LDC corpora: LDC2002E18, LDC2003E07, LDC2003E14, Hansards part of LDC2004T07, LDC2004T08 , LDC2005T06.

any length limitation to obtain examples for two ME models. For the reordering model, we obtain about $22.6M$ examples with monotone order and $4.8M$ examples with inverted order. For the operation model, we obtain about $5.9M$ examples with CC operation, $14.8M$ examples with LA operation, and $9.7M$ examples with RA operation. After collecting various features from the examples, we use the ME training toolkit developed by Zhang (2004) to train ME models with the following parameters: iteration number i=200 and Gaussian prior g=1.0.

The 2002 NIST MT Evaluation test set is used as the development set. The 2003 and 2005 NIST MT Evaluation test sets are our test sets. We perform the MERT training (Och, 2003) to tune the optimal feature weights on the development set. To run the decoder, we prune the phrase table with b = 100, prune the chart with n = 50, α = 0.1. See (Xiong et al., 2006) for the meanings of these parameters. The translation quality is evaluated by case-insensitive BLEU-4 metric (Papineni et al., 2002), as calculated by mteval-v11b.pl.

6.2 Results

Since (Xiong et al., 2006) has made a deep investigation on the ME-based reordering model, we mainly focus on the study of the ME-based operation model. To explore the utility of the various features in the operation model, we randomly select about $10K$ examples from all the operation examples as held-out data, and use the rest examples as training data. Then, we train the operation models on different feature sets and investigate the performance of models on the held-out data.

Table 3 shows the accuracy rates of the ME operation models using different feature sets. We find that the bigram feature set provides the most persuasive evidences and achieves best performance than other feature sets.

To investigate the influences of various factors on the system performance, we carried out experiments on the NIST Chinese-English task with the following systems:

- MEBTG + all: an MEBTG translation system, which uses all bilingual phrases. It is our baseline system;
- MEBTG + filter1: a baseline system, which uses the bilingual phrases consistent to the well-formed dependency structures by (Shen et al., 2008);
- MEBTG + filter2: a baseline system, which uses the bilingual phrases consistent to our well-formed dependency structures;
- MEBTG + filter3: a baseline system, which uses the bilingual phrases consistent to our well-formed dependency structures and the special ill-formed dependency structures;
- DepBTG + unigram features: a DepBTG system which only uses the unigram features in the ME-based operation model;
- DepBTG + bigram features: a DepBTG system which only uses the bigram features in the ME-based operation model;
- DepBTG + all features: a DepBTG system which uses all features in the ME-based operation model;
- DepBTG + unigram features + dep LMs: a DepBTG system with dependency LMs, where only the unigram features are adopted in the ME-based operation model;
- DepBTG + bigram features + dep LMs: a DepBTG system with dependency LMs, where only the bigram features are adopted in the ME-based operation model;
- DepBTG + all features + dep LMs: a DepBTG system with dependency LMs, where all features are adopted in the ME-based operation model.

Model	Accuracy Rate
lexical features	87.614%
POS features	88.232%
unigram features	90.024%
bigram features	$\mathbf{93.907}\%$
all features	93.290%

Table 3: The accuracy rates of the ME-based operation models on the held-out data set using different feature sets. Unigram features include lexical features and POS features, and bigram features are the combinations of different unigram features.

System	Type	#Bp	MT03	MT05
MEBTG	all(baseline)	$81.4M$	33.41	32.65
	filter1	$27.8M$	32.17($\downarrow$ 1.24)	31.26($\downarrow$ 1.39)
	filter2	$33.7M$	32.77($\downarrow$ 0.64)	31.93($\downarrow$ 0.72)
	filter3	$58.5M$	33.29($\downarrow$ 0.12)	32.71($\uparrow$ 0.06)
DepBTG	unigram features	$59.9M$	33.46($\uparrow$ 0.05)	32.67($\uparrow$ 0.02)
	bigram features	$59.9M$	33.57($\uparrow$ 0.16)	32.89($\uparrow$ 0.24)
	all features	$59.9M$	33.59($\uparrow$ 0.18)	32.86($\uparrow$ 0.21)
	unigram features + dep LMs	$59.9M$	33.90($\uparrow$ 0.49)	33.29($\uparrow$ 0.64)
	bigram features + dep LMs	$59.9M$	**34.18**($\uparrow$ 0.77)	**33.58**($\uparrow$ 0.93)
	all features + dep LMs	$59.9M$	34.10($\uparrow$ 0.69)	33.55($\uparrow$ 0.90)

Table 4: Experimental results on Chinese-English NIST Task.

Experiment results are summarized in Table 4. Our baseline system extracts $81.4M$ bilingual phrases and achieves the BLEU scores of 33.41 and 32.65 on two test sets separately. Adopting the constraint of the well-formed structures by (Shen et al., 2008), we extract $27.8M$ bilingual phrases, which lead to great drops in BLEU score: 1.24 points and 1.39 points on two test sets separately(see Row 3). Using the constraint of our well-formed structures, the number of extracted bilingual phrases is $33.7M$. We observe the similar results that the performance drops 0.64 points and 0.72 points over the baseline system on two test sets, respectively (see Row 4). Furthermore, we add some bilingual phrases with the special ill-formed structure into our phrase table, and the number of the bilingual phrases in use is 58.5M accounting up 71.9% of the full phrases. For two test sets, our system achieves the BLEU scores of 33.29 and 32.71 (see Row 5), which are very close to the scores of baseline system. Those experimental results demonstrate that phrase coverage has a great effect on the system performance and our definitions of the allowed dependency structures are useful to retain rational bilingual phrases.

Then, by employing the ME-based operation model and two 3-gram dependency LMs, the DepBTG system outperforms the MEBTG system in almost all cases. The experimental results indicate that the dependency LMs are more effective than the ME-based operation model for DepBTG system. Especially, using bigram features and dependency LMs, the DepBTG system obtains absolute improvements on two test sets: $\mathbf{0.77}$ BLEU points on NIST03 test set and $\mathbf{0.93}$ BLEU points on NIST05 test set (see Row 10), which are both statistically significant at $p < 0.05$ using the significance tester developed by Zhang et al. (2004).

7 Conclusion and Future Work

In this paper, we propose a novel dependency-based BTG to directly model the syntactic structure of the translation. Using the bilingual phrases with target dependency structure, our system employs two ME models to generate the translation in line with dependency structure. Based on the target dependency structure, our system filters 26.4% bilingual phrases (from $81.4M$ to $59.9M$), captures the target-side syntactic knowledge by dependency language models, and achieves significant improvements over the baseline system.

There is some work to be done in the future. To better utilize the syntactic information, we will put more effort on the study of the dependency LM with deeper syntactic knowledge. Moreover, we believe that modeling the syntax of both sides is a promising method to further improve BTG-based translation and this will become a study emphasis in our future research. Finally, inspired by (Tu et al., 2010), we will replace 1-best dependency trees with dependency forests to further increase the phrase coverage.

Acknowledgement

The authors were supported by National Natural Science Foundation of China, Contracts

60736014 and 60873167, and 863 State Key Project No.2006AA010108. We thank the anonymous reviewers for their insightful comments. We are also grateful to Zhaopeng Tu, Shu Cai and Xianhua Li for their helpful feedback.

References

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*.

Ding, Yuan and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of ACL*.

Eisner, Jason. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. of ACL*.

Fox, Heidi J. 2002. Phrasal cohesion and statistical machine translation. In *Proc. of EMNLP*.

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

Hellwig, Peter. 2006. Parsing with dependency grammars, volume ii. *An International Handbook of Contemporary Research*.

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

Koehn, Philipp, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Statistical phrase-based translation. In *Proceedings International Workshop on Spoken Language Translation*.

Lin, Dekang. 2004. A path-based transfer model for machine translation. In *Proc. of Coling*.

Liu, Yang, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proc. of ACL*.

Och, Franz Josef and Hermann Ney. 2000. Improved statistical alignment models. In *Proc. of ACL*.

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

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

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

Setiawan, Hendra, Min-Yen Kan, and Haizhou Li. 2007. Ordering phrases with function words. In *Proc. of ACL*.

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

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

Tu, Zhaopeng, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. 2010. Dependency forest for statistical machine translation. In *Proc. of COLING*.

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

Wu, Dekai. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. of ACL*.

Xiong, Deyi, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proc. of ACL*.

Xiong, Deyi, Min Zhang, Aiti Aw, and Haizhou Li. 2008. Linguistically annotated BTG for statistical machine translation. In *Proc. of Coling*.

Zens, Richard, Hermann Ney, Taro Watanabe, and Eiichiro Sumita. 2004. A polynomial-time algorithm for statistical machine translation. In *Proc. of Coling*.

Zhang, Min and Haizhou Li. 2009. Tree kernel-based svm with structured syntactic knowledgefor btg-based phrase reordering. In *Proc. of EMNLP*.

Zhang, Ying, Stephan Vogel, and Alex Waibel. 2004. Interpreting bleu/nist scores how much improvement do we need to have a better system? In *Proc. of LREC*.

Zhang, Dongdong, Mu Li, Chi-Ho Li, and Ming Zhou. 2007. Phrase reordering model integrating syntactic knowledge for smt. In *Proc. of EMNLP*.

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

Zhang, Le. 2004. Maximum entropy modeling tooklkit for python and c++.