Joint Parsing and Translation

Yang Liu and **Qun Liu**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
{yliu,liuqun} @ict.ac.cn

Abstract

Tree-based translation models, which exploit the linguistic syntax of source language, usually separate decoding into two steps: parsing and translation. Although this separation makes tree-based decoding simple and efficient, its translation performance is usually limited by the number of parse trees offered by parser. Alternatively, we propose to parse and translate jointly by casting tree-based translation as parsing. Given a source-language sentence, our joint decoder produces a parse tree on the source side and a translation on the target side simultaneously. By combining translation and parsing models in a discriminative framework, our approach significantly outperforms a forest-based tree-to-string system by 1.1 absolute BLEU points on the NIST 2005 Chinese-English test set. As a parser, our joint decoder achieves an F_1 score of 80.6% on the Penn Chinese Treebank.

1 Introduction

Recent several years have witnessed the rapid development of syntax-based translation models (Chiang, 2007; Galley et al., 2006; Shen et al., 2008; Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006; Eisner, 2003; Zhang et al., 2008; Chiang, 2010), which incorporate formal or linguistic syntax into translation process. Depending on whether modeling the linguistic syntax of source language or not, we divide them into two categories: *string-based* and *tree-based* models. [1]

[1]Mi et al. (2008) also distinguish between string-based and tree-based models but depending on the type of input.

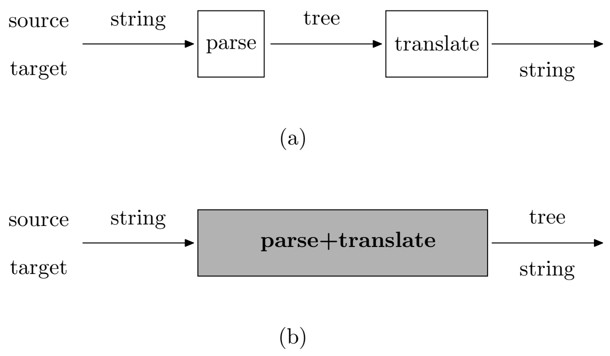

Figure 1: Tree-based decoding: (a) separate parsing and translation versus (b) joint parsing and translation.

String-based models include *string-to-string* (Chiang, 2007) and *string-to-tree* (Galley et al., 2006; Shen et al., 2008). Regardless of the syntactic information on the source side, they treat decoding as a parsing problem: the decoder parses a source-language sentence using the source projection of a synchronous grammar while building the target sub-translations in parallel.

Tree-based models include *tree-to-string* (Liu et al., 2006; Huang et al., 2006) and *tree-to-tree* (Quirk et al., 2005; Eisner, 2003; Zhang et al., 2008; Chiang, 2010). These models explicitly use source parse trees and divide decoding into two separate steps: parsing and translation. A parser first parses a source-language sentence into a parse tree, and then a decoder converts the tree to a translation on the target side (see Figure 1(a)).

Figure 2 gives a training example for tree-to-string translation, which consists of a Chinese tree, an English sentence, and the word alignment between them. Romanized Chinese words are given to facilitate identification. Table 1 shows

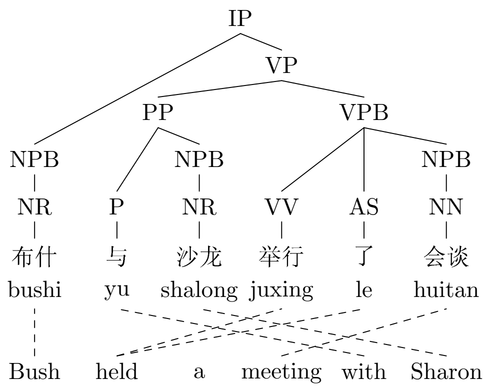

Figure 2: A training example that consists of a Chinese parse, an English sentence, and the word alignment between them.

a set of tree-to-string rules obtained from Figure 2. The source side of a rule is a tree fragment and the target side is a string. We use x to denote non-terminals and the associated subscripts indicate the correspondence between non-terminals on both sides.

Conventionally, decoding for tree-to-string translation is cast as a *tree parsing* problem (Eisner, 2003). The tree parsing algorithm visits each node in the input source tree in a top-down order and tries to match each translation rule against the local sub-tree rooted at the node. For example, the first rule in Table 1 matches a sub-tree rooted at $\mathrm{IP}_{0,6}$ in Figure 2. The descendent nodes of $\mathrm{IP}_{0,6}$ (i.e., $\mathrm{NPB}_{0,1}$, $\mathrm{PP}_{1,3}$, and $\mathrm{VPB}_{3,6}$) can be further matched by other rules in Table 1. The matching procedure runs recursively until the entire tree is covered. Finally, the output on the target side can be taken as a translation.

Compared with its string-based counterparts, tree-based decoding is simpler and faster: there is no need for *synchronous binarization* (Huang et al., 2009b; Zhang et al., 2006) and tree parsing generally runs in linear time (Huang et al., 2006).

While separating parsing and translation makes tree-based decoding simple and efficient, its search space is limited by the number of parse trees offered by parser. Studies reveal that tree-based systems are prone to produce degenerate translations due to the propagation of parsing mistakes (Quirk and Corston-Oliver, 2006). This problem can be alleviated by offering more alternatives to the pipeline. An elegant solution is to replace 1-best trees with packed forests that encode exponentially many trees (Mi et al., 2008; Liu et al., 2009). Mi et al. (2008) present an efficient algorithm to match tree-to-string rules against packed forests that encode millions of trees. They prove that offering more alternatives to tree parsing improves translation performance substantially.

(1)	IP(x_1:NPB VP(x_2:PP x_3:VPB))→x_1 x_3 x_2
(2)	NPB(NR(*bushi*))→*Bush*
(3)	PP(P(*yu*) x_1:NPB)→*with* x_1
(4)	NPB(NR(*shalong*))→*Sharon*
(5)	VPB(VV(*juxing*) AS(*le*) x_1:NPB)→*held a* x_1
(6)	NPB(NN(*huitan*))→*meeting*

Table 1: Tree-to-string rules extracted from Figure 2.

In this paper, we take a further step towards the direction of offering multiple parses to translation by proposing *joint parsing and translation*. As shown in Figure 1(b), our approach parses and translates jointly as it finds a parse tree and a translation of a source-language sentence simultaneously. We integrate the tree-to-string model (Liu et al., 2006; Huang et al., 2006), n-gram language model, probabilistic context-free grammar (PCFG), and Collins' Model 1 (Collins, 2003) in a discriminative framework (Och, 2003). Allowing parsing and translation to interact with each other, our approach obtains an absolute improvement of 1.1 BLEU points over a forest-based tree-to-string translation system (Mi et al., 2008) on the 2005 NIST Chinese-English test set. As a parser, our joint decoder achieves an F_1 score of 80.6% on the Penn Chinese Treebank.

2 Joint Parsing and Translation

2.1 Decoding as Parsing

We propose to integrate parsing and translation into a single step. To achieve joint parsing and translation, we cast tree-to-string decoding as a monolingual parsing problem (Melamed, 2004; Chiang, 2007; Galley et al., 2006): the decoder takes a source-language string as input and parses it using the source-projection of SCFG while building the corresponding sub-translations simultaneously.

For example, given the Chinese sentence *bushi yu sha long juxing le huitan* in Figure 2, the derivation in Table 1 explains how a Chinese tree, an English string, and the word alignment between them are generated synchronously. Unlike the string-based systems as described in (Chiang, 2007; Galley et al., 2006; Shen et al., 2008), we exploit the linguistic syntax on the source side explicitly. Therefore, the source parse trees produced by our decoder are meaningful from a linguistic point of view.

As tree-to-string rules usually have multiple non-terminals that make decoding complexity generally exponential, synchronous binarization (Huang et al., 2009b; Zhang et al., 2006) is a key technique for applying the CKY algorithm to parsing with tree-to-string rules. [2] Huang et al. (2009b) factor each tree-to-string rule into two SCFG rules: one from the root nonterminal to the subtree, and the other from the subtree to the leaves. In this way, one can uniquely reconstruct the original tree using a two-step SCFG derivation.

For example, consider the first rule in Table 1:

$$\mathrm{IP}(x_1\text{:NPB VP}(x_2\text{:PP } x_3\text{:VPB}))\rightarrow x_1\ x_3\ x_2$$

We use a specific non-terminal, say, T, to uniquely identify the left-hand side subtree and produce two SCFG rules: [3]

$$\mathrm{IP} \rightarrow \langle \mathrm{T}_{\boxed{1}}, \mathrm{T}_{\boxed{1}} \rangle \quad (1)$$

$$\mathrm{T} \rightarrow \langle \mathrm{NPB}_{\boxed{1}}\mathrm{PP}_{\boxed{2}}\mathrm{VPB}_{\boxed{3}}, \mathrm{NPB}_{\boxed{1}}\mathrm{VPB}_{\boxed{3}}\mathrm{PP}_{\boxed{2}} \rangle \quad (2)$$

where the boxed numbers indicate the correspondence between nonterminals.

Then, the rule (2) can be further binarized into two rules that have at most two non-terminals:

$$\mathrm{T} \rightarrow \langle \mathrm{NPB}_{\boxed{1}}\ \text{PP-VPB}_{\boxed{2}}, \mathrm{NPB}_{\boxed{1}}\ \text{PP-VPB}_{\boxed{2}} \rangle \quad (3)$$

$$\text{PP-VPB} \rightarrow \langle \mathrm{PP}_{\boxed{1}}\mathrm{VPB}_{\boxed{2}}, \mathrm{VPB}_{\boxed{2}}\mathrm{PP}_{\boxed{1}} \rangle \quad (4)$$

where PP-VPB is an intermediate *virtual non-terminal*.

We call rules the tree roots of which are virtual non-terminals *virtual rules* and others *natural rules*. For example, the rule (1) is a natural rule and the rules (3) and (4) are virtual rules. We follow Huang et al. (2009b) to keep the probabilities of a natural rule unchanged and set those of a virtual rule to 1. [4]

After binarizing tree-to-string rules into SCFG rules that have at most two non-terminals, we can use the CKY algorithm to parse a source sentence and produce its translation simultaneously as described in (Chiang, 2007; Galley et al., 2006).

2.2 Adding Parsing Models

As our decoder produces "genuine" parse trees during decoding, we can integrate parsing models as features together with translation features such as the tree-to-string model, n-gram language model, and word penalty into a discriminative framework (Och, 2003). We expect that parsing and translation could interact with each other: parsing offers linguistically motivated reordering to translation and translation helps parsing resolve ambiguity.

2.2.1 PCFG

We use the probabilistic context-free grammar (PCFG) as the first parsing feature in our decoder. Given a PCFG, the probability for a tree is the product of probabilities for the rules that it contains. That is, if a tree π is a context-free derivation that involves K rules of the form $\alpha_k \rightarrow \beta_k$, its probability is given by

$$\mathcal{P}(\pi) = \prod_{k=1...K} \mathcal{P}_{pcfg}(\alpha_k \rightarrow \beta_k) \quad (5)$$

For example, the probability for the tree in Figure 2 is

$$\begin{aligned}\mathcal{P}(\pi) &= \mathcal{P}_{pcfg}(\mathrm{IP} \rightarrow \mathrm{NPB\ VP}) \times \\ &\quad \mathcal{P}_{pcfg}(\mathrm{NPB} \rightarrow \mathrm{NR}) \times \\ &\quad \mathcal{P}_{pcfg}(\mathrm{NR} \rightarrow \textit{bushi}) \times \\ &\quad \ldots \end{aligned} \quad (6)$$

[2]But CKY is not the only choice. The Earley algorithm can also be used to parse with tree-to-string rules (Zhao and Al-Onaizan, 2008). As the Earley algorithm binarizes multi-nonterminal rules implicitly, there is no need for synchronous binarization.

[3]It might look strange that the node VP disappears. This node is actually stored in the monolithic node T. Please refer to page 573 of (Huang et al., 2009b) for more details about how to convert tree-to-string rules to SCFG rules.

[4]This makes the scores of hypotheses in the same chart cell hardly comparable because some hypotheses are covered by a natural non-terminal and others covered by a virtual non-terminal. To alleviate this problem, we follow Huang et al. (2009b) to separate natural and virtual hypotheses in different beams.

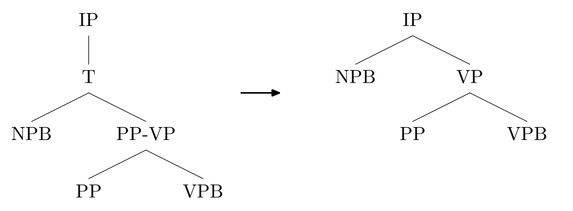

Figure 3: Reconstructing original tree from virtual rules. We first construct the tree on the left by substituting the trees of the rules (1), (3), and (4) and then restore the original tree on the right via the monolithic node T.

There are 13 PCFG rules involved. We omit the remaining 10 rules.

We formalize the decoding process as a deductive system to illustrate how to include a PCFG. Given a natural rule

$$\mathrm{VP} \to \langle \mathrm{PP}_{\boxed{1}}\ \mathrm{VPB}_{\boxed{2}}, \mathrm{VPB}_{\boxed{2}}\ \mathrm{PP}_{\boxed{1}} \rangle \qquad (7)$$

the following deductive step grows an item in the chart by the rule

$$\frac{(\mathrm{PP}_{1,3}) : (w_1, e_1) \quad (\mathrm{VPB}_{3,6}) : (w_2, e_2)}{(\mathrm{VP}_{1,6}) : (w, e_2 e_1)} \qquad (8)$$

where $\mathrm{PP}_{1,3}$ denotes the recognition of the non-terminal PP spanning from the substring from position 1 through 3 (i.e., *yu shalong* in Figure 2), w_1 and e_1 are the score and translation of the first antecedent item, respectively, and the resulting item score is calculated as: [5]

$$w = w_1 + w_2 + \log \mathcal{P}_{pcfg}(\mathrm{VP} \to \mathrm{PP}\ \mathrm{VPB}) \quad (9)$$

As the PCFG probabilities of natural rules are fixed during decoding, they can be pre-computed and stored in the rule table. Therefore, including PCFG for natural rules hardly increases decoding complexity.

However, calculating the PCFG probabilities for virtual rules is quite different due to the presence of virtual non-terminals. For instance, using the rule (4) in Section 2.1 to generate an item leads to the following deductive step:

$$\frac{(\mathrm{PP}_{1,3}) : (w_1, e_1) \quad (\mathrm{VPB}_{3,6}) : (w_2, e_2)}{(\text{PP-VPB}_{1,6}) : (w, e_2 e_1)} \qquad (10)$$

[5]The logarithmic form of probability is used to avoid manipulating very small numbers for practical reasons. w_1 and w_2 take the PCFG probabilities of the two antecedent items into consideration.

As PP-VPB is a virtual non-terminal, the subtree it dominates is a virtual tree, for which we cannot figure out its PCFG probability. Therefore, we have to postpone the calculation of PCFG probabilities until reaching a natural non-terminal such as IP. In other words, only when using the rule (1) to produce an item, the decoding algorithm can update PCFG probabilities because the original tree can be restored from the special node T now. Figure 3 shows how to reconstruct the original tree from virtual rules. We first construct the tree on the left by substituting the trees of the rules (1), (3), and (4) and then restore the original tree on the right via T. Now, we can calculate the PCFG probability of the original tree. [6] In practice, we pre-compute this PCFG probability and store it in the rule (1) to reduce computational overhead.

2.2.2 Lexicalized PCFG

Although widely used in natural language processing, PCFGs are often criticized for the lack of lexicalization, which is very important to capture the lexical dependencies between words. Therefore, we use Collins' Model 1 (Collins, 2003), a simple and effective lexicalized parsing model, as the second parsing feature in our decoder.

Following Collins (2003), we first lexicalize a tree by associating a *headword* h with each non-terminal. Figure 4 gives the lexicalized tree corresponding to Figure 2. The left-hand side of a rule in a lexicalized PCFG is $P(h)$ and the right-hand side has the form:

$$L_n(l_n) \ldots L_1(l_1) H(h) R_1(\tau_1) \ldots R_m(\tau_m) \quad (11)$$

where H is the head-child that inherits the headword h from its parent P, $L_1 \ldots L_n$ and $R_1 \ldots R_m$ are left and right modifiers of H, and $l_1 \ldots l_n$ and $\tau_1 \ldots \tau_m$ are the corresponding headwords. Either n or m may be zero, and $n = m = 0$ for unary rules. Collins (2003) extends the left and right sequences to include a terminating STOP symbol. Thus, $L_{n+1} = R_{m+1} = \mathrm{STOP}$.

[6]Postponing the calculation of PCFG probabilities also leads to the "hard-to-compare" problem mentioned in footnote 4 due to the presence of virtual non-terminals. We still maintain multiple beams for natural and virtual hypotheses (i.e., items) to alleviate this prblem.

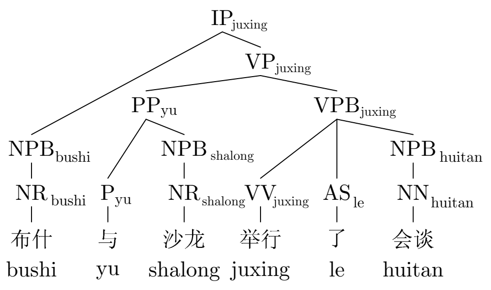

Figure 4: The lexicalized tree corresponding to Figure 2.

Collins (2003) breaks down the generation of the right-hand side of a rule into a sequence of smaller steps. The probability of a rule is decomposed as:

$$\mathcal{P}_h(H|P(h)) \times \prod_{i=1...n+1} \mathcal{P}_l(L_i(l_i)|P(h),H,t,\Delta) \times \prod_{j=1...m+1} \mathcal{P}_r(R_j(\tau_j)|P(h),H,t,\Delta) \quad (12)$$

where t is the POS tag of of the headword h and Δ is the distance between words that captures head-modifier relationship.

For example, the probability of the lexicalized rule $\mathrm{IP}(\textit{juxing}) \rightarrow \mathrm{NPB}(\textit{bushi})\ \mathrm{VP}(\textit{juxing})$ can be computed as [7]

$$\begin{aligned} &\mathcal{P}_h(\mathrm{VP}|\mathrm{IP},\ \textit{juxing}) \times \\ &\mathcal{P}_l(\mathrm{NPB}(\textit{bushi})|\mathrm{IP},\mathrm{VP},\ \textit{juxing}) \times \\ &\mathcal{P}_l(\mathrm{STOP}|\mathrm{IP},\mathrm{VP},\ \textit{juxing}) \times \\ &\mathcal{P}_r(\mathrm{STOP}|\mathrm{IP},\mathrm{VP},\ \textit{juxing}) \end{aligned} \quad (13)$$

We still use the deductive system to explain how to integrate the lexicalized PCFG into the decoding process. Now, Eq. (8) can be rewritten as:

$$\frac{(\mathrm{PP}^{\mathrm{yu}}_{1,3}) : (w_1, e_1) \quad (\mathrm{VPB}^{\mathrm{juxing}}_{3,6}) : (w_2, e_2)}{(\mathrm{VP}^{\mathrm{juxing}}_{1,6}) : (w, e_2e_1)} \quad (14)$$

where *yu* and *juxing* are the headwords attached to $\mathrm{PP}_{1,3}$, $\mathrm{VPB}_{3,6}$, and $\mathrm{VP}_{1,6}$. The resulting item score is given by

$$\begin{aligned} w \;=\; & w_1 + w_2 + \log\mathcal{P}_h(\mathrm{VPB}|\mathrm{VP},\ \textit{juxing}) + \\ & \log\mathcal{P}_l(\mathrm{PP}(\textit{yu})|\mathrm{VP},\mathrm{VPB},\ \textit{juxing}) + \\ & \log\mathcal{P}_l(\mathrm{STOP}|\mathrm{VP},\mathrm{VPB},\ \textit{juxing}) + \\ & \log\mathcal{P}_r(\mathrm{STOP}|\mathrm{VP},\mathrm{VPB},\ \textit{juxing}) \end{aligned} \quad (15)$$

Unfortunately, the lexicalized PCFG probabilities of most natural rules cannot be pre-computed because the headword of a non-terminal must be determined on the fly during decoding. Consider the third rule in Table 1

$$\mathrm{PP}(\mathrm{P}(\textit{yu})\ x_1{:}\mathrm{NPB}) \rightarrow \textit{with}\ x_1$$

It is impossible to know what the headword of NPB is in advance, which depends on the actual sentence being translated. However, we could safely say that the headword attached to PP is always *yu* because PP should have the same headword with its child P.

Similar to the PCFG scenario, calculating lexicalized PCFG for virtual rules is different from natural rules. Consider the rule (4) in Section 2.1, the corresponding deductive step is

$$\frac{(\mathrm{PP}^{\mathrm{yu}}_{1,3}) : (w_1, e_1) \quad (\mathrm{VPB}^{\mathrm{juxing}}_{3,6}) : (w_2, e_2)}{(\text{PP-VPB}^{-}_{1,6}) : (w, e_2e_1)} \quad (16)$$

where "$-$" denotes that the headword of $\text{PP-VPB}_{1,6}$ is undefined.

We still need to postpone the calculation of lexicalized PCFG probabilities until reaching a natural non-terminal such as IP. In other words, only when using the rule (1) to produce an item, the decoding algorithm can update the lexicalized PCFG probabilities. After restoring the original tree from T, we need to visit backwards to frontier nodes of the tree to find headwords and calculate lexicalized PCFG probabilities. More specifically, updating lexicalized PCFG probabilities for the rule the rule (1) involves the following steps:

1. Reconstruct the original tree from the rules (1), (3), and (4) as shown in Figure 3;

2. Attach headwords to all nodes;

3. Calculate the lexicalized PCFG probabilities according to Eq. (12).

[7]For simplicity, we omit POS tag and distance in the presentation. In practice, we implemented the Collins' Model 1 exactly as described in (Collins, 2003).

Back-off level	$\mathcal{P}_h(H\|\dots)$	$\mathcal{P}_l(L_i(l_i)\|\dots)$ $\mathcal{P}_r(R_j(\tau_j)\|\dots)$
1	P, h, t	P, H, h, t, Δ
2	P, t	P, H, t, Δ
3	P	P, H, Δ

Table 2: The conditioning variables for each level of back-off.

As suggested by Collins (2003), we use back-off smoothing for sub-model probabilities during decoding. Table 2 shows the various levels of back-off for each type of parameter in the lexicalized parsing model we use. For example, $\mathcal{P}_h$ estimation p interpolates maximum-likelihood estimates $p_1 = \mathcal{P}_h(H|P, h, t)$, $p_2 = \mathcal{P}_h(H|P, t)$, and $p_3 = \mathcal{P}_h(H|P)$ as follows:

$$p_1 = \lambda_1 p_1 + (1 - \lambda_1)(\lambda_2 p_2 + (1 - \lambda_2) p_3) \quad (17)$$

where λ_1, λ_2, and λ_3 are smoothing parameters.

3 Experiments

In this section, we try to answer two questions:

1. Does tree-based translation by parsing outperform the conventional tree parsing algorithm? (Section 3.1)
2. How about the parsing performance of the joint decoder? (Section 3.2)

3.1 Translation Evaluation

We used a bilingual corpus consisting of 251K sentences with 7.3M Chinese words and 9.2M English words to extract tree-to-string rules. The Chinese sentences in the bilingual corpus were parsed by an in-house parser (Xiong et al., 2005), which obtains an F_1 score of 84.4% on the Penn Chinese Treebank. After running GIZA++ (Och and Ney, 2003) to obtain word alignments, we used the GHKM algorithm (Galley et al., 2004) and extracted 11.4M tree-to-string rules from the source-side parsed, word-aligned bilingual corpus. Note that the bilingual corpus does not contain the bilingual version of Penn Chinese Treebank. In other words, all tree-to-string rules were learned from noisy parse trees and alignments. We used the SRILM toolkit (Stolcke, 2002) to train a 4-gram language model on the Xinhua portion of the GIGAWORD corpus, which contains 238M English words. We trained PCFG and Collins' Model 1 on the Penn Chinese Treebank.

We used the 2002 NIST MT Chinese-English test set as the development set and the 2005 NIST test set as the test set. Following Huang (2008), we modified our in-house parser to produce and prune packed forests on the development and test sets. There are about 105M parse trees encoded in a forest of a sentence on average. We also extracted 1-best trees from the forests.

As the development and test sets have many long sentences ($\geq$ 100 words) that make our decoder prohibitively slow, we divided long sentences into short sub-sentences simply based on punctuation marks such as comma and period. The source trees and target translations of sub-sentences were concatenated to form the tree and translation of the original sentence.

We compared our parsing-based decoder with the tree-to-string translation systems based on the tree parsing algorithm, which match rules against either 1-best trees (Liu et al., 2006; Huang et al., 2006) or packed forests (Mi et al., 2008). All the three systems used the same rule set containing 11.4M tree-to-string rules. Given the 1-best trees of the test set, there are 1.2M tree-to-string rules that match fragments of the 1-best trees. For the forest-based system (Mi et al., 2008), the number of filtered rules increases to 1.9M after replacing 1-best trees with packed forests, which contain 105M trees on average. As our decoder takes a string as input, 7.7M tree-to-string rules can be used to parse and translate the test set. We binarized 99.6% of tree-to-string rules into 16.2M SCFG rules and discarded non-binarizable rules. As a result, the search space of our decoder is much larger than those of the tree parsing counterparts.

Table 3 shows the results. All the three systems used the conventional translation features such as relative frequencies, lexical weights, rule count, n-gram language model, and word count. Without any parsing models, the tree-based system achieves a BLEU score of 29.8. The forest-based system outperforms the tree-based system by $+1.8$ BLEU points. Note that each hyperedge

Algorithm	Input	Parsing model	# of rules	BLEU (%)	Time (s)
tree parsing	tree	-	1.2M	29.8	0.56
	forest	PCFG	1.9M	31.6	9.49
parsing	string	-	7.7M	32.0	51.41
		PCFG		32.4	55.52
		Lex		32.6	89.35
		PCFG + Lex		**32.7**	91.72

Table 3: Comparison of tree parsing and parsing for tree-to-string translation in terms of *case-insensitive* BLEU score and average decoding time (second per sentence). The column "parsing model" indicates which parsing models were used in decoding. We use "-" to denote using only translation features. "Lex" represents the Collins' Model 1. We excluded the extra parsing time for producing 1-best trees and packed forests.

Forest size	Exact match (%)	Precision (%)
1	0.55	41.5
390	0.74	47.7
5.8M	0.92	54.1
66M	1.48	62.0
105M	2.22	65.9

Table 4: Comparison of 1-best trees produced by our decoder and the parse forests produced by the monolingual Chinese parser. Forest size represents the average number of trees stored in a forest.

in a parse forest is assigned a PCFG probability. Therefore, the forest-based system actually includes PCFG as a feature (Mi et al., 2008). Without incorporating any parsing models as features, our joint decoder achieves a BLEU score of 32.0. Adding PCFG and Collins' Model 1 (i.e., "Lex" in Table 2) increases translation performance. When both PCFG and Collins' Model 1 are used, our joint decoder outperforms the tree parsing systems based on 1-best trees ($+2.9$) and packed forests ($+1.1$) significantly ($p < 0.01$). This result is also better than that of using only translation features significantly (from 32.0 to 32.7, $p < 0.05$).

Not surprisingly, our decoder is much slower than pattern matching on 1-best trees and packed forests (with the same beam size). In particular, including Collins' Model 1 increases decoding time significantly because its sub-model probabilities requires back-off smoothing on the fly.

How many 1-best trees produced by our decoder are included in the parse forest produced by a standard parser? We used the Chinese parser to generate five pruned packed forests with different sizes (average number of trees stored in a forest). As shown in Table 4, only 2.22% of the trees produced by our decoder were included in the biggest forest. One possible reason is that we used sub-sentence division to reduce decoding complexity. To further investigate the matching rate, we also calculated labeled precision, which indicates how many brackets in the parse match those in the packed forest. The labeled precision on the biggest forest is 65.9%, suggesting that the 1-best trees produced by our decoder are significantly different from those in the packed forests produced by a standard parser. [8]

3.2 Parsing Evaluation

We followed Petrov and Klein (2007) to divide the Penn Chinese Treebank (CTB) version 5 as follows: Articles 1-270 and 400-1151 as the training set, Articles 301-325 as the development set, and Articles 271-300 as the test set. We used max-F_1 training (Och, 2003) to train the feature weights. We did not use sub-sentence division as the sentences in the test set have no more than 40 words.

[8]The packed forest produced by our decoder ("rule" forest) might be different from the forest produced by a monolingual parser ("parser" forest). While tree-based and forest-based decoders search in the intersection of the two forests (i.e., matched forest), our decoder directly explores the "rule" forest, which represents the true search space of tree-to-string translation. This might be the key difference of our approach from forest-based translation (Mi et al., 2008). As sub-sentence division makes direct comparison of the two forests quite difficult, we leave this to future work.

Parsing model	F_1 (%)	Time (s)
-	62.7	23.9
PCFG	65.4	24.7
Lex	79.8	48.8
PCFG + Lex	$\mathbf{80.6}$	50.4

Table 5: Effect of parsing models on parsing performance ($\leq$ 40 words) and average decoding time (second per sentence). We use "-" to denote only using translation features.

Table 5 shows the results. Translation features were used for all configurations. Without parsing models, the F_1 score is 62.7. Adding Collins' Model 1 results in much larger gains than adding PCFG. With all parsing models integrated, our joint decoder achieves an F_1 score of 80.6 on the test set. Although lower than the F_1 score of the in-house parser that produces the noisy training data, this result is still very promising because the tree-to-string rules that construct trees in the decoding process are learned from noisy training data.

4 Related Work

Charniak et al. (2003) firstly associate lexicalized parsing model with syntax-based translation. They first run a string-to-tree decoder (Yamada and Knight, 2001) to produce an English parse forest and then use a lexicalized parsing model to select the best translation from the forest. As the parsing model operates on the target side, it actually serves as a syntax-based language model for machine translation. Recently, Shen et al. (2008) have shown that dependency language model is beneficial for capturing long-distance relations between target words. As our approach adds parsing models to the source side where the source sentence is fixed during decoding, our decoder does parse the source sentence like a monolingual parser instead of a syntax-based language model. More importantly, we integrate translation models and parsing models in a discriminative framework where they can interact with each other directly.

Our work also has connections to joint parsing (Smith and Smith, 2004; Burkett and Klein, 2008) and bilingually-constrained monolingual parsing (Huang et al., 2009a) because we use another language to resolve ambiguity for one language. However, while both joint parsing and bilingually-constrained monolingual parsing rely on the target sentence, our approach only takes a source sentence as input.

Blunsom and Osborne (2008) incorporate the source-side parse trees into their probabilistic SCFG framework and treat every source-parse PCFG rule as an individual feature. The difference is that they parse the test set before decoding so as to exploit the source syntactic information to guide translation.

More recently, Chiang (2010) has shown that ("exact") tree-to-tree translation as parsing achieves comparable performance with Hiero (Chiang, 2007) using much fewer rules. Xiao et al. (2010) integrate tokenization and translation into a single step and improve the performance of tokenization and translation significantly.

5 Conclusion

We have presented a framework for joint parsing and translation by casting tree-to-string translation as a parsing problem. While tree-to-string rules construct parse trees on the source side and translations on the target side simultaneously, parsing models can be integrated to improve both translation and parsing quality.

This work can be considered as a final step towards the continuum of tree-to-string translation: from single tree to forest and finally to the integration of parsing and translation. In the future, we plan to develop more efficient decoding algorithms, analyze forest matching systematically, and use more sophisticated parsing models.

Acknowledgement

The authors were supported by National Natural Science Foundation of China, Contracts 60736014 and 60903138, and 863 State Key Project No. 2006AA010108. We are grateful to the anonymous reviewers for their insightful comments. We thank Liang Huang, Hao Zhang, and Tong Xiao for discussions on synchronous binarization, Haitao Mi and Hao Xiong for providing and running the baseline systems, and Wenbin Jiang for helping us train parsing models.

References

Blunsom, Phil and Miles Osborne. 2008. Probabilistic inference for machine translation. In *Proc. of EMNLP 2008*.

Burkett, David and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proc. of EMNLP 2008*.

Charniak, Eugene, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proc. of MT Summit IX*.

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

Chiang, David. 2010. Learning to translate with source and target syntax. In *Proc. of ACL 2010*.

Collins, Michael. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

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

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. of NAACL 2004*.

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 2006*.

Huang, Liang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA 2006*.

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

Huang, Liang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2009b. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4):559–595.

Huang, Liang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL 2008*.

Liu, Yang, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL 2006*.

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

Melamed, I. Dan. 2004. Statistical machine translation by parsing. In *Proc. of ACL 2004*.

Mi, Haitao, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. of ACL 2008*.

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

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

Petrov, Slav and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of NAACL 2007*.

Quirk, Chris and Simon Corston-Oliver. 2006. The impact of parsing quality on syntactically-informed statistical machine translation. In *Proc. of EMNLP 2006*.

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

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 2008*.

Smith, David and Noah Smith. 2004. Bilingual parsing with factored estimation: using english to parse korean. In *Proc. of EMNLP 2004*.

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

Xiao, Xinyan, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. 2010. Joint tokenization and translation. In *Proc. of COLING 2010*.

Xiong, Deyi, Shuanglong Li, Qun Liu, and Shouxun Lin. 2005. Parsing the penn chinese treebank with semantic knowledge. In *Proc. of IJCNLP 2005*.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. of ACL 2001*.

Zhang, Hao, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translatio. In *Proc. of NAACL 2007*.

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 2008*.

Zhao, Bing and Yaser Al-Onaizan. 2008. Generalizing local and non-local word-reordering patterns for syntax-based machine translation. In *Proc. of EMNLP 2008*.