Machine Translation with Lattices and Forests

Haitao Mi[†‡] **Liang Huang**[‡] **Qun Liu**[†]

[†]Key Lab. of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
{htmi,liuqun}@ict.ac.cn

[‡]Information Sciences Institute
Viterbi School of Engineering
University of Southern California
{lhuang,haitaomi}@isi.edu

Abstract

Traditional 1-best translation pipelines suffer a major drawback: the errors of 1-best outputs, inevitably introduced by each module, will propagate and accumulate along the pipeline. In order to alleviate this problem, we use compact structures, *lattice* and *forest*, in each module instead of 1-best results. We integrate both lattice and forest into a single tree-to-string system, and explore the algorithms of lattice parsing, lattice-forest-based rule extraction and decoding. More importantly, our model takes into account all the probabilities of different steps, such as segmentation, parsing, and translation. The main advantage of our model is that we can make global decision to search for the best segmentation, parse-tree and translation in one step. Medium-scale experiments show an improvement of +0.9 BLEU points over a state-of-the-art forest-based baseline.

1 Introduction

Statistical machine translation (SMT) has witnessed promising progress in recent years. Typically, conventional SMT is characterized as a 1-best pipeline system (Figure 1(a)), whose modules are independent of each other and only take as input 1-best results from the previous module. Though this assumption is convenient to reduce the complexity of SMT systems. It also bring a major drawback of error propagation. The errors of 1-best outputs, introduced inevitably in each phase, will propagate and accumulate along the pipeline. Not recoverable in the final decoding step. These errors will severely hurt the translation quality. For example, if the accuracy of each module is 90%, the final accuracy will drop to 73% after three separate phases.

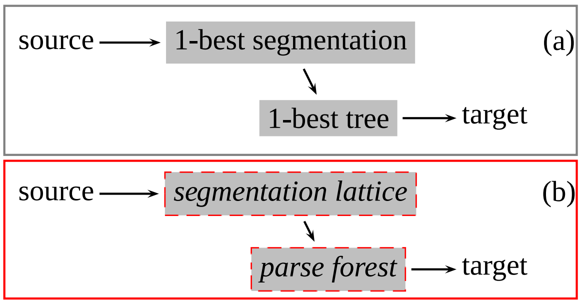


Figure 1: The pipeline of tree-based system: (a) 1-best (b) lattice-forest.

To alleviate this problem, an obvious solution is to widen the pipeline with *k*-best lists rather than 1-best results. For example Venugopal et al. (2008) use *k*-best alignments and parses in the training phase. However, with limited scope and too many redundancies, it is inefficient to search separately on each of these similar lists (Huang, 2008).

Another efficient method is to use compact data structures instead of *k*-best lists. A *lattice* or *forest*, compactly encoded exponentially many derivations, have proven to be a promising technique. For example, Mi and Huang (2008), Mi et al. (2008), Liu et al. (2009) and Zhang et al. (2009) use forests in rule extraction and decoding phases to extract more general rules and weaken the influence of parsing errors; Dyer et al. (2008) use word lattice in Chinese word segmentation and Arabic morphological variation phases to weaken the influence of segmentation errors; Huang (2008) and

Coling 2008: Poster Volume, pages 837–845,
Beijing, August 2010

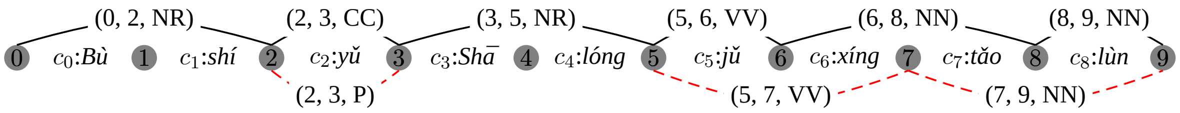


Figure 2: The lattice of the example:“ *Bù shí yǔ Shā lóng jǔ xíng tǎo lùn*.” The solid lines show the 1-best result, which is wrong.

Jiang et al. (2008b) stress the problems in reranking phase. Both lattices and forests have become popular in machine translation literature.

However, to the best of our knowledge, previous work only focused on one module at a time. In this paper, we investigate the combination of lattice and forest (Section 2), as shown in Figure 1(b). We explore the algorithms of lattice parsing (Section 3.2), rule extraction (Section 4) and decoding (Section 5). More importantly, in the decoding step, our model can search among not only more parse-trees but also more segmentations encoded in the lattice-forests and can take into account all the probabilities of segmentations and parse-trees. In other words, our model postpones the disambiguition of segmentation and parsing into the final translation step, so that we can do global search for the best segmentation, parse-tree and translation in one step. When we integrate a lattice into a forest system, medium-scale experiments (Section 6) show another improvement of +0.9 BLEU points over a state-of-the-art forest-based system.

2 Compact Structures

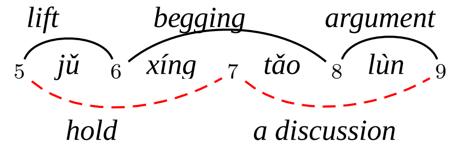
A **word lattice** (Figure 2) is a compact representation of all the possible of segmentations and POS tags, while a **parse forest** (Figure 5) is a compact representation of all parse trees.

2.1 Word Lattice

For a given input sentence $C = c_0..c_{n-1}$, where c_i denotes a character at position i, and n is the length of the sentence.

A word lattice (Figure 2), or **lattice** in short, is a set of **edges** L, where each edge is in the form of (i, j, X), which denotes a word of tag X, covering characters c_i through c_{j-1}. For example, in Figure 2, (7, 9, NN) is a noun “*tǎolùn*” of two characters.

The lattice in Figure 2 shows result of the example:“ *Bù shí yǔ Shā lóng jǔ xíng tǎo lùn* ”. One ambiguity comes from the POS tag of word “*yǔ*” (preposition (P) or conjunction (CC)). The other one is the segmentation ambiguity of the last four characters, we can segment into either “*jǔ xíngtǎo lùn*” (solid lines), which means *lift*, *begging* and *argument* separately for each word or “*jǔxíng tǎolùn*” (dashed lines), which means *hold a discussion*.



The solid lines above (and also in Figure 2) show the 1-best result, which is obviously wrong. If we feed it into the next modules in the SMT pipeline, parsing and translation will be become much more difficult, since the segmentation is not recoverable. So it is necessary to postpone error segmentation decisions to the final translation step.

2.2 Parse Forest

In parsing scenario, a parse forest (Figrure 5), or **forest** for short, can be formalized as a hypergraph H, a pair $\langle V, E\rangle$, where node $v \in V$ is in the form of $X_{i,j}$, which denotes the recognition of nonterminal X spanning the substring $c_{i:j-1}$ from positions c_i through c_{j-1}. Each hyperedge $e \in E$ is a pair $\langle tails(e), head(e)\rangle$, where $head(e) \in V$ is the **consequent node** in an instantiated deductive step, and $tails(e) \in (V)^*$ is the list of **antecedent nodes**.

For the following deduction:

$$\frac{\mathrm{NR}_{0,2} \quad \mathrm{CC}_{2,3} \quad \mathrm{NR}_{3,5}}{\mathrm{NP}_{0,5}} \qquad (*)$$

its hyperedge e^* is notated:

$$\langle(\mathrm{NR}_{0,2},\ \mathrm{CC}_{2,3},\ \mathrm{NR}_{3,5}),\ \ \mathrm{NP}_{0,5}\rangle.$$

where

$$head(e^*) = \{\mathrm{NP}_{0,5}\}, \text{ and}$$
$$tails(e^*) = \{\mathrm{NR}_{0,2}, \mathrm{CC}_{2,3}, \mathrm{NR}_{3,5}\}.$$

We also denote $IN(v)$ to be the set of **incoming hyperedges** of node v, which represents the different ways of deriving v. For simplicity, we only show a tree in Figure 5(a) over 1-best segmentation and POS tagging result in Figure 2. So the $IN(\mathrm{NP}_{0,5})$ is $\{e^*\}$.

3 Lattice Parsing

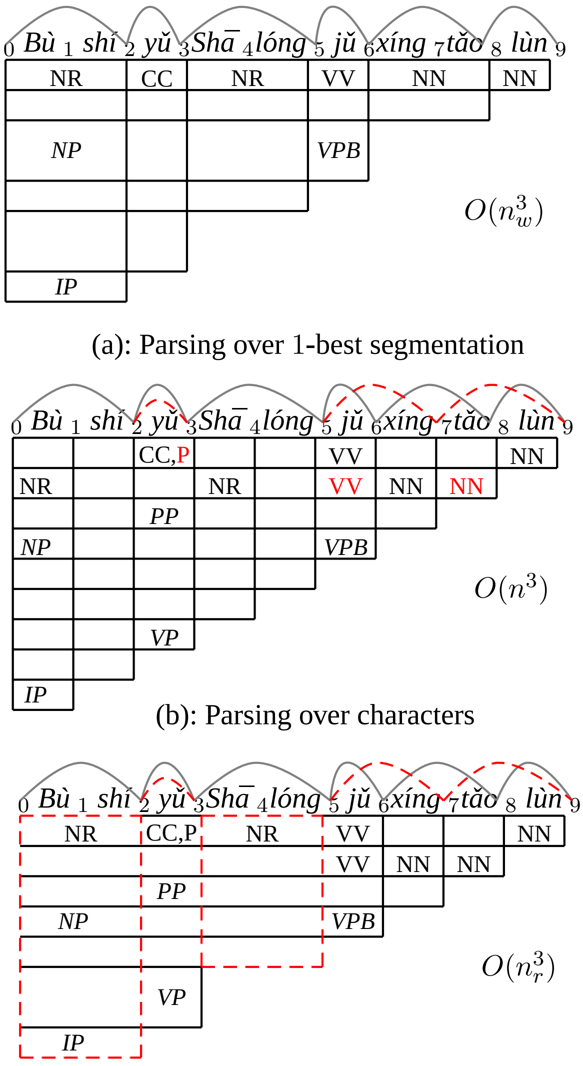
In this section, we first briefly review the conventional CYK parsing, and then extend to lattice parsing. More importantly, we propose a more efficient parsing paradigm in Section 3.3.

3.1 Conventional Parsing

The conventional CYK parsing algorithm in Figure 3(a) usually takes as input a single sequence of words, so the CYK cells are organized over words. This algorithm consists of two steps: initialization and parsing. The first step is to initialize the CYK cells, whose span size is one, with POS tags produced by a POS tagger or defined by the input string[1]. For example, the top line in Figure 3(a) is initialized with a series of POS tags in 1-best segmentation. The second step is to search for the best syntactic tree under a context-free grammar. For example, the tree composed by the solid lines in Figure 5(a) shows the parsing tree for the 1-best segmentation and POS tagging results.

3.2 Lattice Parsing

The main differences of our lattice parsing in Figure 3(b) from conventional approach are listed in following: First, the CYK cells are organized over characters rather than words. Second, in the initialization step, we only initialize the cells with all edges L in the lattice. Take the edge (7, 9, NN) in Figure 2 for example, the corresponding cell should be $(7, 9)$, then we add a leaf node $v = \mathrm{NN}_{7,9}$ with a word *tǎolùn*. The final initialization is shown in Figure 3(b), which shows that


0 Bù 1 shí 2 yǔ 3 Shā 4 lóng 5 jǔ 6 xíng 7 tǎo 8 lùn 9
NR
CC
NR
VV
NN
NN
NP
VPB
IP
$O(n_w^3)$
0 Bù 1 shí 2 yǔ 3 Shā 4 lóng 5 jǔ 6 xíng 7 tǎo 8 lùn 9
CC,P
VV
NN
NR
NR
VV
NN
NN
PP
NP
VPB
VP
IP
$O(n^3)$
0 Bù 1 shí 2 yǔ 3 Shā 4 lóng 5 jǔ 6 xíng 7 tǎo 8 lùn 9
NR
CC,P
NR
VV
NN
VV
NN
NN
PP
NP
VPB
VP
IP
$O(n_r^3)$

(a): Parsing over 1-best segmentation

(b): Parsing over characters

(c): Parsing over most-refined segmentation

Figure 3: CKY parsing charts (a): Conventional parsing over 1-best segmentation. (b): Lattice parsing over characters of input sentence. (c): Lattice parsing over most-refined segmentation of lattice. n_w and n_r denotes the number of tokens over the 1-best segmentation and the most-refined segmenation respectively, and $n_w \le n_r \le n$.

lattice parsing can initialize the cells, whose span size is larger than one. Third, in the deduction step of the parsing algorithm *i*, *j*, *k* are the indexes between characters rather than words.

We formalize our lattice parser as a deductive proof system (Shieber et al., 1994) in Figure 4.

Following the definitions of the previous Sec-

[1] For simplicity, we assume the input of a parser is a segmentation and POS tagging result

tion, given a set of edges L of a lattice for an input sentence $C = c_0..c_{n-1}$ and a PCFG grammar: a 4-tuple $\langle N, \Sigma, P, S\rangle$, where N is a set of nonterminals, Σ is a set of terminal symbols, P is a set of inference rules, each of which is in the form of $X \to \alpha : p$ for $X \in N$, $\alpha \in (N \cup \Sigma)^*$ and p is the probability, and $S \in N$ is the start symbol. The deductive proof system (Figure 4) consists of **axioms**, **goals** and **inference rules**. The axioms are converted by edges in L. Take the (5, 7, NN) associated with a weight p_1 for example, the corresponding axiom is NN $\to$ *tǎolùn* : p_1. All axioms converted from the lattice are shown in Figure 3(b) exclude the italic non-terminals. Please note that all the probabilities of the edges L in a lattice are taken into account in the parsing step. The goals are the recognition X$_{0,n} \in S$ of the whole sentence. The inference rules are the deductions in parsing. Take the deduction (*) for example, it will prove a new item NP$_{0,5}$ (italic NP in Figure 3(b)) and generate a new hyper-edge e^* (in Figure 5(b)). So the parsing algorithm starts with the axioms, and then applies the inference rules to prove new items until a goal item is proved. The final whole forest for the input lattice (Figure 2) is shown in Figure 5(b). The extra hyper-edges of lattice-forest are highlighted with dashed lines, which can inference the input sentence correctly. For example: "*yǔ*" is tagged into P rather than CC.

3.3 Faster Parsing with Most-refined Lattice

However, our statistics show that the average number of characters n in a sentence is 1.6 times than the number of words n_w in its 1-best segmentation. As a result, the parsing time over the characters will grow more than 4 times than parsing over the 1-best segmentation, since the time complexity is $O(n^3)$. In order to alleviate this problem, we reduce the parsing time by using **most-refined segmentation** for a lattice, whose number of tokens is n_r and has the property $n_w \leq n_r \leq n$.

Given a lattice with its edges L over indexes $(0, .., n)$, a index i is a **split point**, if and only if there exists some edge $(i, j, X) \in L$ or $(k, i, X) \in L$. The **most-refined segmentation**, or **ms** for short, is the segmentation result by using all split points in a lattice. For example, the corresponding **ms** of the example is "*Bùshí yǔ Shālóng jǔ xíng tǎo lùn*" since points 1 and 4 are *not* split points.

Item form: $X_{i,j}$

Axioms: $$\frac{}{X_{i,j} : p(i,j,X)} \; (i,j,X) \in L$$

Infer. rules: $$\frac{X_{i,k} : p_1 \; Y_{k,j} : p_2}{Z_{i,j} : pp_1p_2} Z \to XY : p \in P$$

Goals: $X_{0,n}$

Figure 4: Lattice parsing as deductive proof system. The i, j, k are the indexes between characters.

Figure 3(c) shows the CKY parsing cells over most-refined segmentation, the average number of tokens n_r is reduced by combining columns, which are shown with red dashed boxes. As a result, the search space is reduced without losing any derivations. Theoretically, the parsing over *fs* will speed up in $O((n/n_r)^3)$. And our experiments in Section 6 show the efficiency of our new approach.

It turns out that the parsing algorithm developed in lattice-parsing Section 3.2 can be used here without any change. The non-terminals inducted are also shown in Figure 3(c) in italic style.

4 Rule Extraction with Lattice & Forest

We now explore the extraction algorithm from aligned source lattice-forest and target string[2], which is a tuple $\langle F, \tau, a\rangle$ in Figure 5(b). Following Mi and Huang (2008), we extract minimal rules from a lattice-forest also in two steps:

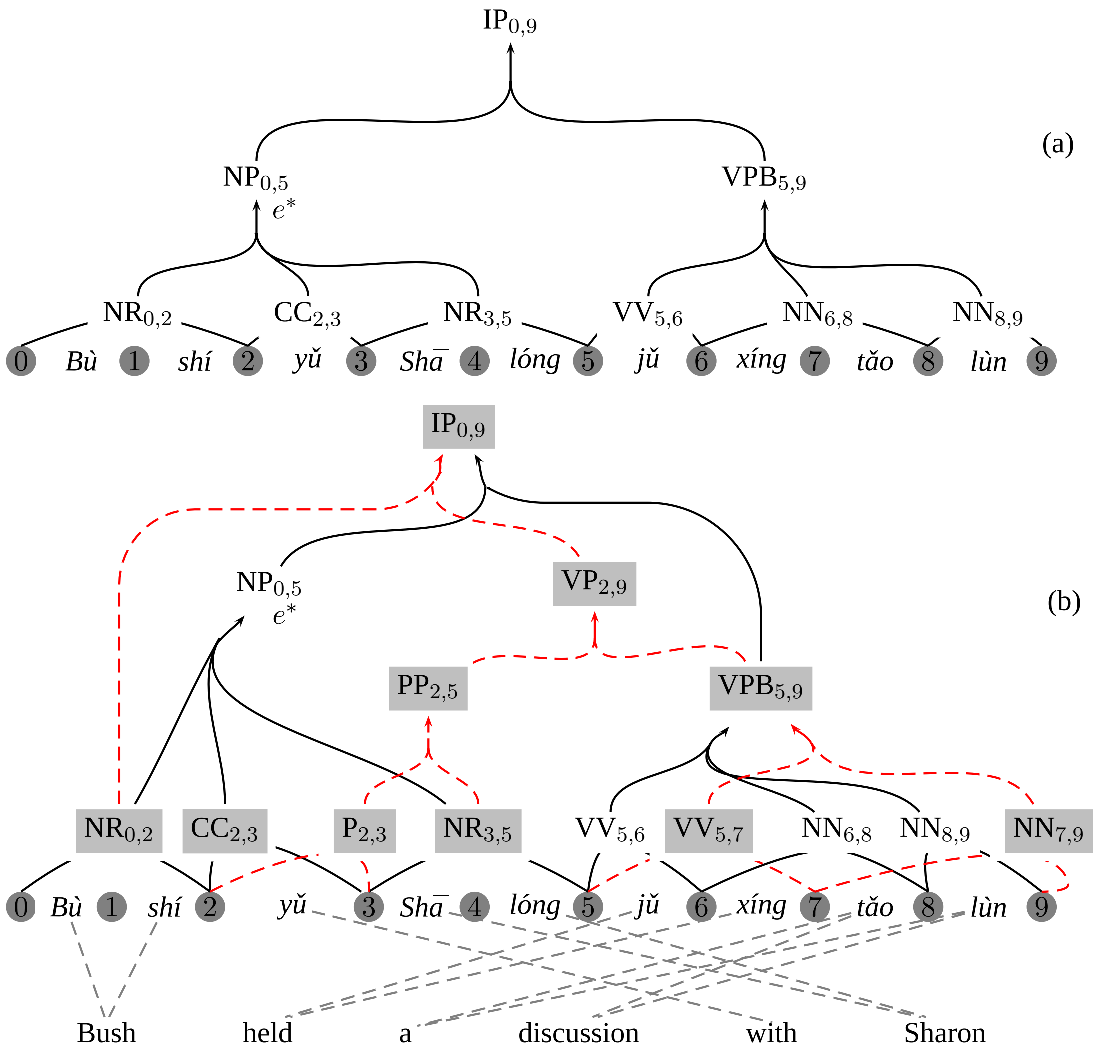
(1) frontier set computation

(2) fragmentation

Following the algorithms developed by Mi and Huang (2008) in Algorithm 1, all the nodes in *frontier set* (*fs*) are highlighted with gray in Figure 5(b).

Our process of fragmentation (lines 1- 13) is to visit each frontier node v and initial a queue ($open$) of growing fragments with a pair of empty fragment and node v (line 3). Each fragment is associated with a list of *expansion sites* ($front$) being

[2] For simplicity and consistency, we use character-based lattice-forest for the running example. The "*Bù*" and "*shí*" are aligned to the same word "*Bush*". In our experiment, we use most-refined segmentation to run lattice-parsing and word alignment.



Forest only (Minimal rules)	Lattice & forest (Extra minimal rules)
IP(NP(x_1:NR x_2:CC x_3:NR) x_4:VPB)	IP(x_1:NR x_2:VP) → x_1 x_2
→ x_1 x_4 x_2 x_3	VP(x_1:PP x_2:VPB) → x_2 x_1
CC(*yǔ*) →*with*	PP(x_1:P x_2:NR) → x_1 x_2
NR(*Shālóng*) →*Sharon*	P(*yǔ*) →*with*
NR(*Bùshí*) →*Bush*	VPB(x_1:VV x_2:NN) → x_1 x_2
VPB(VV(*jǔ*) NN(*xíngtǎo*) NN(*lùn*))	VV(*jǔxíng*) →*held*
→*held a discussion*	NN(*tǎolùn*) →*a discussion*

(c)

Figure 5: (a): The parse forest over the 1-best segmentation and POS tagging result. (b): Word-aligned tuple $\langle F, \tau, a\rangle$: the lattice-forest F, the target string τ and the word alignment a. The solid hyperedges form the forest in (a). The dashed hyperedges are the extra hyperedges introduced by the lattice-forest. (c): The minimal rules extracted on forest-only (left column), and the extra minimal rules extracted on lattice-forest (right column).

the subset of leaf nodes of the current fragment that are *not* in the *fs* except for the initial node v. Then we keep expanding fragments in *open* in following way. If current fragment is complete, whose expansion sites is empty, we extract rule corresponding to the fragment and its target string

Code 1 Rule Extraction (Mi and Huang, 2008).

```
    Input: lattice-forest $F$, target sentence $\tau$, and alignment $a$
    Output: minimal rule set $\mathcal{R}$
 1: $fs \leftarrow \textsc{FroSet}(F, \tau, a)$                ▷ frontier set
 2: for each $v \in fs$ do
 3:     $open \leftarrow \{\langle\varnothing, \{v\}\rangle\}$          ▷ initial queue
 4:     while $open \neq \varnothing$ do
 5:         $\langle frag, front\rangle \leftarrow open.pop()$
 6:         if $front = \varnothing$ then              ▷ finished?
 7:             generate a rule $r$ using $frag$
 8:             $\mathcal{R}$.append($r$)
 9:         else          ▷ incomplete: further expand
10:             $u \leftarrow front$.pop() ▷ expand frontier
11:             for each $e \in IN(u)$ do
12:                 $f \leftarrow front \cup (tails(e) \setminus fs)$
13:                 $open$.append($\langle frag \cup \{e\}, f\rangle$)
```

(line 7) . Otherwise we pop one expansion node u to grow and spin-off new fragments by $IN(u)$, adding new expansion sites (lines 11- 13), until all active fragments are complete and $open$ queue is empty.

The extra minimal rules extracted on lattice-forest are listed at the right bottom of Figure 5(c). Compared with the forest-only approach, we can extract smaller and more general rules.

After we get all the minimal rules, we compose two or more minimal rules into composed rules (Galley et al., 2006), which will be used in our experiments.

For each rule r extracted, we also assign a fractional count which is computed by using inside-outside probabilities:

$$c(r) = \frac{\alpha(root(r)) \;\cdot\; P(lhs(r)) \;\cdot\; \prod_{v \in yield(root(r))} \beta(v)}{\beta(\text{TOP})}, \tag{1}$$

where $root(r)$ is the root of the rule, $lhs(r)$ is the left-hand-side of rule, $rhs(r)$ is the right-hand-side of rule, $P(lhs(r))$ is the product of all probabilities of hyperedges involved in $lhs(r)$, $yield(root(r))$ is the leave nodes, TOP is the root node of the forest, $\alpha(v)$ and $\beta(v)$ are outside and inside probabilities, respectively.

Then we compute three conditional probabilities for each rule:

$$P(r \mid lhs(r)) = \frac{c(r)}{\sum_{r':lhs(r')=lhs(r)} c(r')} \tag{2}$$

$$P(r \mid rhs(r)) = \frac{c(r)}{\sum_{r':rhs(r')=rhs(r)} c(r')} \tag{3}$$

$$P(r \mid root(r)) = \frac{c(r)}{\sum_{r':root(r')=root(r)} c(r')}. \tag{4}$$

All these probabilities are used in decoding step (Section 5). For more detail, we refer to the algorithms of Mi and Huang (2008).

5 Decoding with Lattice & Forest

Given a source-side lattice-forest F, our decoder searches for the best derivation d^* among the set of all possible derivation D, each of which converts a tree in lattice-forest into a target string τ:

$$\begin{aligned} d^* = \operatorname*{argmax}_{d \in D, T \in F} & P(d|T)^{\lambda_0} \cdot e^{\lambda_1 |d|} \\ & \cdot LM(\tau(d))^{\lambda_2} \cdot e^{\lambda_3 |\tau(d)|}, \end{aligned} \tag{5}$$

where $|d|$ is the penalty term on the number of rules in a derivation, $LM(\tau(d))$ is the language model and $e^{\lambda_3|\tau(d)|}$ is the length penalty term on target translation. The $P(d|T)$ decomposes into the product of rule probabilities $P(r)$, each of which is decomposed further into

$$P(d|T) = \prod_{r \in d} P(r). \tag{6}$$

Each $P(r)$ in Equation 6 is decomposed further into the production of five probabilities:

$$\begin{aligned} P(r) = & P(r|lhs(r))^{\lambda_4} \\ & \cdot P(r|rhs(r))^{\lambda_5} \\ & \cdot P(r|root(lhs(r))^{\lambda_6} \\ & \cdot P_{lex}(lhs(r)|rhs(r))^{\lambda_7} \\ & \cdot P_{lex}(rhs(r)|lhs(r))^{\lambda_8}, \end{aligned} \tag{7}$$

where the last two are the lexical probabilities between the terminals of $lhs(r)$ and $rhs(r)$. All the weights of those features are tuned by using Minimal Error Rate Training (Och, 2003).

Following Mi et al. (2008), we first convert the lattice-forest into *lattice translation forest* with the conversion algorithm proposed by Mi et al. (2008), and then the decoder finds the best derivation on the lattice translation forest. For 1-best search, we use the *cube pruning* technique (Chiang, 2007; Huang and Chiang, 2007) which approximately intersects the translation forest with the LM. For k-best search after getting 1-best derivation, we use the lazy Algorithm 3 of Huang and Chiang (2005) to incrementally compute the second, third, through the kth best alternatives.

For more detail, we refer to the algorithms of Mi et al. (2008).

6 Experiments

6.1 Data Preparation

Our experiments are on Chinese-to-English translation. Our training corpus is FBIS corpus with about 6.9M/8.9M words in Chinese/English respectively.

We use SRI Language Modeling Toolkit (Stolcke, 2002) to train a 4-gram language model with Kneser-Ney smoothing on the first 1/3 of the Xinhua portion of Gigaword corpus.

We use the 2002 NIST MT Evaluation test set as development set and the 2005 NIST MT Evaluation test set as test set. We evaluate the translation quality using the case-insensitive BLEU-4 metric (Papineni et al., 2002). We use the standard MERT (Och, 2003) to tune the weights.

6.1.1 Baseline Forest-based System

We first segment the Chinese sentences into the 1-best segmentations using a state-of-the-art system (Jiang et al., 2008a), since it is not necessary for a conventional parser to take as input the POS tagging results. Then we parse the segmentation results into forest by using the parser of Xiong et al. (2005). Actually, the parser will assign multiple POS tags to each word rather than one. As a result, our baseline system has already postponed the POS tagging disambiguition to the decoding step. Forest is pruned by using a marginal probability-based pruning algorithm similar to Huang (2008). The pruning threshold are $p_f = 5$ and $p_f = 10$ at rule extraction and decoding steps respectively.

We word-align the strings of 1-best segmentations and target strings with GIZA++ (Och and Ney, 2000) and apply the refinement method "grow-diag-final-and" (Koehn et al., 2003) to get the final alignments. Following Mi and Huang (2008) and Mi et al. (2008), we also extract rules from forest-string pairs and translate forest to string.

6.1.2 Lattice-forest System

We first segment and POS tag the Chinese sentences into word lattices using the same system (Jiang et al., 2008a), and prune each lattice into a reasonable size using the marginal probability-based pruning algorithm.

Then, as current GIZA++ (Och and Ney, 2000) can only handle alignment between string-string pairs, and word-alingment with the pairs of Chinese characters and target-string will obviously result in worse alignment quality. So a much better way to utilize GIZA++ is to use the most-refined segmentation for each lattice instead of the character sequence. This approach can be viewed as a compromise between character-string and lattice-string word-alignment paradigms. In our experiments, we construct the most-refined segmentations for lattices and word-align them against the English sentences. We again apply the refinement method "grow-diag-final-and" (Koehn et al., 2003) to get the final alignments.

In order to get the lattice-forests, we modified Xiong et al. (2005)'s parser into a lattice parser, which produces the pruned lattice forests for both training, dev and test sentences. Finally, we apply the rule extraction algorithm proposed in this paper to obtain the rule set. Both lattices and forests are pruned using a marginal probability-based pruning algorithm similar to Huang (2008). The pruning threshold of lattice is $p_l = 20$ at both the rule extraction and decoding steps, the thresholds for the latice-forests are $p_f = 5$ and $p_f = 10$ at rule extraction and decoding steps respectively.

6.2 Results and Analysis

Table 1 shows results of two systems. Our lattice-forest (LF) system achieves a BLEU score of 29.65, which is an absolute improvement of 0.9 points over the forest (F) baseline system, and the improvement is statistically significant at $p < 0.01$ using the *sign-test* of Collins et al. (2005).

The average number of tokens for the 1-best and most-refined segmentations are shown in second column. The average number of characters is 46.7, which is not shown in Table 1. Com-

Sys	Avg # of		Rules		BLEU
	tokens	links	All	dev&tst	
F	28.7	35.1	29.6M	3.3M	28.75
LF	37.1	37.1	23.5M	3.4M	29.65

Table 1: Results of forest (F) and lattice-forest (LF) systems. Please note that lattice-forest system only extracts 23.5M rules, which is only 79.4% of the rules extracted by forest system. However, in decoding step, lattice-forest system can use more rules after filtered on dev and test sets.

pared with the characters-based lattice parsing, our most-refined lattice parsing speeds up parsing by $(37.1/46.7)^3 \approx 2$ times, since parsing complexity is $O(n^3)$.

More interestingly, our lattice-forest model only extracts 23.5M rules, which is 79.4% percent of the rules extracted by the baseline system. The main reason lies in the larger average number of words for most-refined segmentations over lattices being 37.1 words vs 28.7 words over 1-best segmentations. With much finer granularity, more word aligned links and restrictions are introduced during the rule extraction step by GIZA++. However, more rules can be used in the decoding step for the lattice-forest system, since the lattice-forest is larger than the forest over 1-best segmentation.

We also investigate the question of how often the non 1-best segmentations are picked in the final translation. The statistic on our dev set suggests 33% of sentences choose non 1-best segmentations. So our lattice-forest model can do global search for the best segmentation and parse-tree to direct the final translation. More importantly, we can use more translation rules in the translation step.

7 Related Works

Compactly encoding exponentially many derivations, lattice and forest have been used in some previous works on SMT. To alleviate the problem of parsing error in 1-best tree-to-string translation model, Mi et al. (2008) first use forest to direct translation. Then Mi and Huang (2008) use forest in rule extraction step. Following the same direction, Liu et al. (2009) use forest in tree-to-tree model, and improve 1-best system by 3 BLEU points. Zhang et al. (2009) use forest in tree-sequence-to-string model and also achieve a promising improvement. Dyer et al. (2008) combine multiple segmentations into word lattice and then use lattice to direct a phrase-based translation decoder. Then Dyer (2009) employ a single Maximum Entropy segmentation model to generate more diverse lattice, they test their model on the hierarchical phrase-based system. Lattices and forests can also be used in Minimal Error Rate Training and Minimum Bayes Risk Decoding phases (Macherey et al., 2008; Tromble et al., 2008; DeNero et al., 2009; Kumar et al., 2009; Li and Eisner, 2009). Different from the works listed above, we mainly focus on how to combine lattice and forest into a single tree-to-string system.

8 Conclusion and Future Work

In this paper, we have proposed a lattice-forest based model to alleviate the problem of error propagation in traditional single-best pipeline framework. Unlike previous works, which only focus on one module at a time, our model successfully integrates lattice into a state-of-the-art forest tree-to-string system. We have explored the algorithms of lattice parsing, rule extraction and decoding. Our model postpones the disambiguition of segmentation and parsing into the final translation step, so that we can make a more global decision to search for the best segmentation, parse-tree and translation in one step. The experimental results show that our lattice-forest approach achieves an absolute improvement of +0.9 points in term of BLEU score over a state-of-the-art forest-based model.

For future work, we would like to pay more attention to word alignment between lattice pairs and forest pairs, which would be more principled than our current method of word alignment between most-refined segmentation and string.

Acknowledgement

We thank Steve DeNeefe and the three anonymous reviewers for comments. The work is supported by National Natural Science Foundation of China, Contracts 90920004 and 60736014, and 863 State Key Project No. 2006AA010108 (H. M and Q. L.), and in part by DARPA GALE Contract No. HR0011-06-C-0022, and DARPA under DOI-NBC Grant N10AP20031 (L. H and H. M).

References

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

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540, Ann Arbor, Michigan, June.

John DeNero, David Chiang, and Kevin Knight. 2009. Fast consensus decoding over translation forests. In *Proceedings of ACL/IJCNLP*.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio, June.

C. Dyer. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proceedings of NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL*, pages 961–968, Sydney, Australia, July.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of IWPT*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151, June.

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

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008a. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008b. Word lattice reranking for chinese word segmentation and part-of-speech tagging. In *Proceedings of Coling 2008*.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133, Edmonton, Canada, May.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the ACL/IJCNLP 2009*.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of EMNLP*, pages 40–51, Singapore, August. Association for Computational Linguistics.

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

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of EMNLP 2008*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP 2008*, pages 206–214, Honolulu, Hawaii, October.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08:HLT*, pages 192–199, Columbus, Ohio, June.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL*, pages 440–447.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadephia, USA, July.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1994. Principles and implementation of deductive parsing.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of EMNLP 2008*.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2008. Wider pipelines: N-best alignments and parses in MT training. In *Proceedings of AMTA*.

Deyi Xiong, Shuanglong Li, Qun Liu, and Shouxun Lin. 2005. Parsing the Penn Chinese Treebank with Semantic Knowledge. In *Proceedings of IJCNLP 2005*, pages 70–81.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw, and Chew Lim Tan. 2009. Forest-based tree sequence to string translation model. In *Proceedings of the ACL/IJCNLP 2009*.