Forest-based Tree Sequence to String Translation Model

Hui Zhang[1, 2] **Min Zhang**[1] **Haizhou Li**[1] **Aiti Aw**[1] **Chew Lim Tan**[2]
[1]Institute for Infocomm Research [2]National University of Singapore
zhangh1982@gmail.com {mzhang, hli, aaiti}@i2r.a-star.edu.sg tancl@comp.nus.edu.sg

Abstract

This paper proposes a forest-based tree sequence to string translation model for syntax-based statistical machine translation, which automatically learns tree sequence to string translation rules from word-aligned source-side-parsed bilingual texts. The proposed model leverages on the strengths of both tree sequence-based and forest-based translation models. Therefore, it can not only utilize forest structure that compactly encodes exponential number of parse trees but also capture non-syntactic translation equivalences with linguistically structured information through tree sequence. This makes our model potentially more robust to parse errors and structure divergence. Experimental results on the NIST MT-2003 Chinese-English translation task show that our method statistically significantly outperforms the four baseline systems.

1 Introduction

Recently syntax-based statistical machine translation (SMT) methods have achieved very promising results and attracted more and more interests in the SMT research community. Fundamentally, syntax-based SMT views translation as a structural transformation process. Therefore, structure divergence and parse errors are two of the major issues that may largely compromise the performance of syntax-based SMT (Zhang et al., 2008a; Mi et al., 2008).

Many solutions have been proposed to address the above two issues. Among these advances, forest-based modeling (Mi et al., 2008; Mi and Huang, 2008) and tree sequence-based modeling (Liu et al., 2007; Zhang et al., 2008a) are two interesting modeling methods with promising results reported. Forest-based modeling aims to improve translation accuracy through digging the potential better parses from *n*-bests (i.e. forest) while tree sequence-based modeling aims to model non-syntactic translations with structured syntactic knowledge. In nature, the two methods would be complementary to each other since they manage to solve the negative impacts of monolingual parse errors and cross-lingual structure divergence on translation results from different viewpoints. Therefore, one natural way is to combine the strengths of the two modeling methods for better performance of syntax-based SMT. However, there are many challenges in combining the two methods into a single model from both theoretical and implementation engineering viewpoints. In theory, one may worry about whether the advantage of tree sequence has already been covered by forest because forest encodes implicitly a huge number of parse trees and these parse trees may generate many different phrases and structure segmentations given a source sentence. In system implementation, the exponential combinations of tree sequences with forest structures make the rule extraction and decoding tasks much more complicated than that of the two individual methods.

In this paper, we propose a forest-based tree sequence to string model, which is designed to integrate the strengths of the forest-based and the tree sequence-based modeling methods. We present our solutions that are able to extract translation rules and decode translation results for our model very efficiently. A general, configurable platform was designed for our model. With this platform, we can easily implement our method and many previous syntax-based methods by simple parameter setting. We evaluate our method on the NIST MT-2003 Chinese-English translation tasks. Experimental results show that our method significantly outperforms the two individual methods and other baseline methods. Our study shows that the proposed method is able to effectively combine the strengths of the forest-based and tree sequence-based methods, and thus having great potential to address the issues of parse errors and non-syntactic transla-

tions resulting from structure divergence. It also indicates that tree sequence and forest play different roles and make contributions to our model in different ways.

The remainder of the paper is organized as follows. Section 2 describes related work while section 3 defines our translation model. In section 4 and section 5, the key rule extraction and decoding algorithms are elaborated. Experimental results are reported in section 6 and the paper is concluded in section 7.

2 Related work

As discussed in section 1, two of the major challenges to syntax-based SMT are structure divergence and parse errors. Many techniques have been proposed to address the structure divergence issue while only fewer studies are reported in addressing the parse errors in the SMT research community.

To address structure divergence issue, many researchers (Eisner, 2003; Zhang et al., 2007) propose using the Synchronous Tree Substitution Grammar (STSG) grammar in syntax-based SMT since the STSG uses larger tree fragment as translation unit. Although promising results have been reported, STSG only uses one single sub-tree as translation unit which is still committed to the syntax strictly. Motivated by the fact that non-syntactic phrases make non-trivial contribution to phrase-based SMT, the tree sequence-based translation model is proposed (Liu et al., 2007; Zhang et al., 2008a) that uses tree sequence as the basic translation unit, rather than using single sub-tree as in the STSG. Here, a tree sequence refers to a sequence of consecutive sub-trees that are embedded in a full parse tree. For any given phrase in a sentence, there is at least one tree sequence covering it. Thus the tree sequence-based model has great potential to address the structure divergence issue by using tree sequence-based non-syntactic translation rules. Liu et al. (2007) propose the tree sequence concept and design a tree sequence to string translation model. Zhang et al. (2008a) propose a tree sequence-based tree to tree translation model and Zhang et al. (2008b) demonstrate that the tree sequence-based modelling method can well address the structure divergence issue for syntax-based SMT.

To overcome the parse errors for SMT, Mi et al. (2008) propose a forest-based translation method that uses forest instead of one best tree as translation input, where a forest is a compact representation of exponentially number of n-best parse trees. Mi and Huang (2008) propose a forest-based rule extraction algorithm, which learn tree to string rules from source forest and target string. By using forest in rule extraction and decoding, their methods are able to well address the parse error issue.

From the above discussion, we can see that traditional tree sequence-based method uses single tree as translation input while the forest-based model uses single sub-tree as the basic translation unit that can only learn tree-to-string (Galley et al. 2004; Liu et al., 2006) rules. Therefore, the two methods display different strengths, and which would be complementary to each other. To integrate their strengths, in this paper, we propose a forest-based tree sequence to string translation model.

3 Forest-based tree sequence to string model

In this section, we first explain what a packed forest is and then define the concept of the tree sequence in the context of forest followed by the discussion on our proposed model.

3.1 Packed Forest

A packed forest (forest in short) is a special kind of hyper-graph (Klein and Manning, 2001; Huang and Chiang, 2005), which is used to represent all derivations (i.e. parse trees) for a given sentence under a context free grammar (CFG). A forest F is defined as a triple $< V, E, S >$, where V is non-terminal node set, E is hyper-edge set and S is leaf node set (i.e. all sentence words). A forest F satisfies the following two conditions:

1) Each node n in V should cover a phrase, which is a continuous word sub-sequence in S.

2) Each hyper-edge e in E is defined as $v_f \Rightarrow v_1 \dots v_i \dots v_n, (v_i \in (V \cup S), v_f \in V)$, where $v_1 \dots v_i \dots v_n$ covers a sequence of continuous and non-overlap phrases, v_f is the father node of the children sequence $v_1 \dots v_i \dots v_n$. The phrase covered by v_f is just the sum of all the phrases covered by each child node v_i.

We here introduce another concept that is used in our subsequent discussions. A complete forest *CF* is a general forest with one additional condition that there is only one root node *N* in *CF*, i.e., all nodes except the root *N* in a *CF* must have at least one father node.

Fig. 1 is a complete forest while Fig. 7 is a non-complete forest due to the virtual node "VV+VV" introduced in Fig. 7. Fig. 2 is a hyper-edge (IP => NP VP) of Fig. 1, where NP covers

the phrase “Xinhuashe”, VP covers the phrase “shengming youguan guiding” and IP covers the entire sentence. In Fig.1, only root IP has no father node, so it is a complete forest. The two parse trees T1 and T2 encoded in Fig. 1 are shown separately in Fig. 3 and Fig. 4[1].

Different parse tree represents different derivations and explanations for a given sentence. For example, for the same input sentence in Fig. 1, T1 interprets it as “XNA (Xinhua News Agency) declares some regulations.” while T2 interprets it as “XNA declaration is related to some regulations.”.

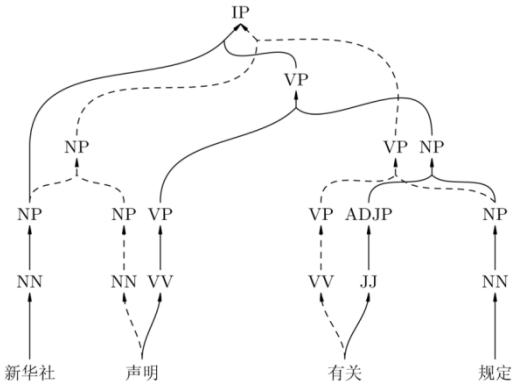

Figure 1. A packed forest for sentence “新华社/Xinhuashe 声明/shengming 有关/youguan 规定/guiding”

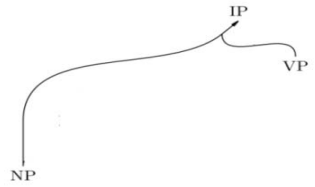

Figure 2. A hyper-edge used in Fig. 1

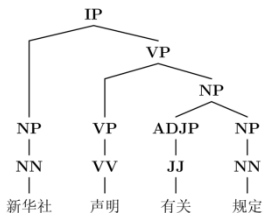

Figure 3. Tree 1 (T1)

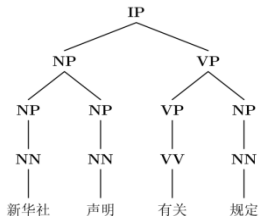

Figure 4. Tree 2 (T2)

3.2 Tree sequence in packed forest

Similar to the definition of tree sequence used in a single parse tree defined in Liu et al. (2007) and Zhang et al. (2008a), a tree sequence in a forest also refers to an ordered sub-tree sequence that covers a continuous phrase without overlapping. However, the major difference between them lies in that the sub-trees of a tree sequence in forest may belongs to different single parse trees while, in a single parse tree-based model, all the sub-trees in a tree sequence are committed to the same parse tree.

The forest-based tree sequence enables our model to have the potential of exploring additional parse trees that may be wrongly pruned out by the parser and thus are not encoded in the forest. This is because that a tree sequence in a forest allows its sub-trees coming from different parse trees, where these sub-trees may not be merged finally to form a complete parse tree in the forest. Take the forest in Fig. 1 as an example, where ((*VV shengming*) (*JJ youguan*)) is a tree sequence that all sub-trees appear in T1 while ((*VV shengming*) (*VV youguan*)) is a tree sequence whose sub-trees do not belong to any single tree in the forest. But, indeed the two sub-trees (*VV shengming*) and (*VV youguan*) can be merged together and further lead to a complete single parse tree which may offer a correct interpretation to the input sentence (as shown in Fig. 5). In addition, please note that, on the other hand, more parse trees may introduce more noisy structures. In this paper, we leave this problem to our model and let the model decide which sub-structures are noisy features.

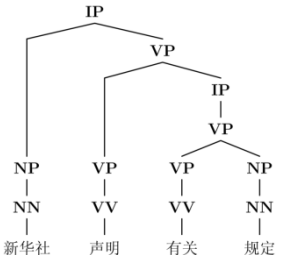

Figure 5. A parse tree that was wrongly pruned out

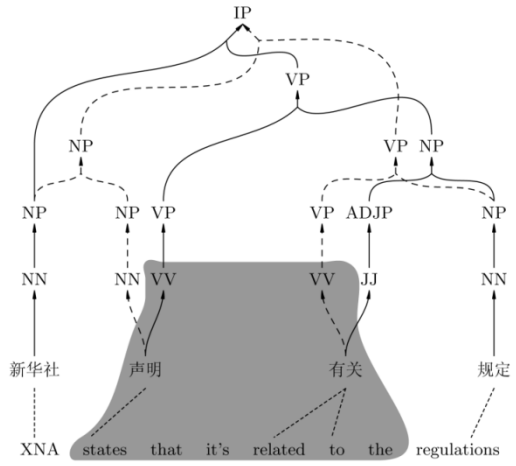

Figure 6. A tree sequence to string rule

[1] Please note that a single tree (as T1 and T2 shown in Fig. 3 and Fig. 4) is represented by edges instead of hyper-edges. A hyper-edge is a group of edges satisfying the 2nd condition as shown in the forest definition.

A tree-sequence to string translation rule in a forest is a triple <*L*, *R*, *A*>, where *L* is the tree sequence in source language, *R* is the string containing words and variables in target language, and *A* is the alignment between the leaf nodes of *L* and *R*. This definition is similar to that of (Liu et al. 2007, Zhang et al. 2008a) except our tree-sequence is defined in forest. The shaded area of Fig. 6 exemplifies a tree sequence to string translation rule in the forest.

3.3 Forest-based tree-sequence to string translation model

Given a source forest *F* and target translation T_S as well as word alignment A, our translation model is formulated as:

$$\Pr(F, T_S, A) = \sum_{\theta_i \in \Theta, C(\Theta)=(F,T_S,A)} \prod_{r_i \in \theta_i} p(r_i)$$

By the above Eq., translation becomes a tree sequence structure to string mapping issue. Given the *F*, T_S and A, there are multiple derivations that could map *F* to T_S under the constraint A. The mapping probability $\Pr(F, T_S, A)$ in our study is obtained by summing over the probabilities of all derivations Θ. The probability of each derivation θ_i is given as the product of the probabilities of all the rules $p(r_i)$ used in the derivation (here we assume that each rule is applied *independently* in a derivation).

Our model is implemented under log-linear framework (Och and Ney, 2002). We use seven basic features that are analogous to the commonly used features in phrase-based systems (Koehn, 2003): 1) bidirectional rule mapping probabilities, 2) bidirectional lexical rule translation probabilities, 3) target language model, 4) number of rules used and 5) number of target words. In addition, we define two new features: 1) number of leaf nodes in auxiliary rules (the auxiliary rule will be explained later in this paper) and 2) product of the probabilities of all hyper-edges of the tree sequences in forest.

4 Training

This section discusses how to extract our translation rules given a triple $< F, T_S, A >$. As we know, the traditional tree-to-string rules can be easily extracted from $< F, T_S, A >$ using the algorithm of Mi and Huang (2008)[2]. We would like to leverage on their algorithm in our study. Unfortunately, their algorithm is not directly applicable to our problem because tree rules have only one root while tree sequence rules have multiple roots. This makes the tree sequence rule extraction very complex due to its interaction with forest structure. To address this issue, we introduce the concepts of virtual node and virtual hyper-edge to convert a complete parse forest *F* to a non-complete forest *F* which is designed to encode all the tree sequences that we want. Therefore, by doing so, the tree sequence rules can be extracted from a forest in the following two steps:

1) Convert the complete parse forest *F* into a non-complete forest *F* in order to cover those tree sequences that cannot be covered by a single tree node.

2) Employ the forest-based tree rule extraction algorithm (Mi and Huang, 2008) to extract our rules from the non-complete forest.

To facilitate our discussion, here we introduce two notations:

- Alignable: A consecutive source phrase is an alignable phrase if and only if it can be aligned with at least one consecutive target phrase under the word-alignment constraint. The covered source span is called alignable span.
- Node sequence: a sequence of nodes (either leaf or internal nodes) in a forest covering a consecutive span.

Algorithm 1 illustrates the first step of our rule extraction algorithm, which is a CKY-style Dynamic Programming (DP) algorithm to add virtual nodes into forest. It includes the following steps:

1) We traverse the forest to visit each span in bottom-up fashion (line 1-2),
   1.1) for each span [*u*,*v*] that is covered by single tree nodes[3], we put these tree nodes into the set *NSS*(*u*,*v*) and go back to step 1 (line 4-6).
   1.2) otherwise we concatenate the tree sequences of sub-spans to generate the set of tree sequences covering the current larger span (line 8-13). Then, we prune the set of node sequences (line 14). If this span is alignable, we create virtual father nodes and corresponding virtual hyper-edges to link the node sequences with the virtual father nodes (line 15-20).

[2] Mi and Huang (2008) extend the tree-based rule extraction algorithm (Galley et al., 2004) to forest-based by introducing non-deterministic mechanism. Their algorithm consists of two steps, minimal rule extraction and composed rule generation.

[3] Note that in a forest, there would be multiple single tree nodes covering the same span as shown Fig.1.

2) Finally we obtain a forest with each alignable span covered by either original tree nodes or the newly-created tree sequence virtual nodes.

Theoretically, there is exponential number of node sequences in a forest. Take Fig. 7 as an example. The *NSS* of span [1,2] only contains "NP" since it is alignable and covered by the single tree node NP. However, span [2,3] cannot be covered by any single tree node, so we have to create the *NSS* of span[2,3] by concatenating the *NSS*s of span [2,2] and span [3,3]. Since *NSS* of span [2,2] contains 4 element {"NN", "NP", "VV", "VP"} and *NSS* of span [3, 3] also contains 4 element {"VV", "VP", "JJ", "ADJP"}, *NSS* of span [2,3] contains 16=4*4 elements. To make the *NSS* manageable, we prune it with the following thresholds:

- each node sequence should contain less than *n* nodes
- each node sequence set should contain less than *m* node sequences
- sort node sequences according to their lengths and only keep the *k* shortest ones

Each virtual node is simply labeled by the concatenation of all its children's labels as shown in Fig. 7.

Algorithm 1. *add virtual nodes into forest*
Input: *packed forest F, alignment A*
Notation:
L: length of source sentence
NSS(*u*,*v*): the set of node sequences covering span [*u*,*v*]
VN(*ns*): virtual father node for node sequence *ns*.
Output: *modified forest F with virtual nodes*

```
1.  for length := 0 to L - 1 do
2.      for start := 1 to L - length do
3.          stop := start + length
4.          if span[start, stop] covered by tree nodes then
5.              for each node n of span [start, stop] do
6.                  add n into NSS(start, stop)
7.          else
8.              for pivot := start to stop - 1
9.                  for each ns1 in NSS(start, pivot) do
10.                     for each ns2 in NSS(pivot+1, stop) do
11.                         create ns := ns1 ⊕ ns2
12.                         if ns is not in NSS(start, stop) then
13.                             add ns into NSS(start, stop)
14.             do pruning on NSS(start, stop)
15.             if the span[start, stop] is alignable then
16.                 for each ns of NSS(start, stop) do
17.                     if node VN(ns) is not in F then
18.                         add node VN(ns) into F
19.                     add a hyper-edge h into F,
20.                     let lhs(h) := VN(ns), rhs(h) := ns
```

Algorithm 1 outputs a non-complete forest *CF* with each alignable span covered by either tree nodes or virtual nodes. Then we can easily extract our rules from the *CF* using the tree rule extraction algorithm (Mi and Huang, 2008).

Finally, to calculate rule feature probabilities for our model, we need to calculate the fractional counts (it is a kind of probability defined in Mi and Huang, 2008) of each translation rule in a parse forest. In the tree case, we can use the inside-outside-based methods (Mi and Huang 2008) to do it. In the tree sequence case, since the previous method cannot be used directly, we provide another solution by making an independent assumption that each tree in a tree sequence is independent to each other. With this assumption, the fractional counts of both tree and tree sequence can be calculated as follows:

$$c(r) = \frac{\alpha\beta(lhs(r))}{\alpha\beta(TOP)}$$

$$\alpha\beta(frag) = \prod_{v \in root(frag)} \alpha(v) * \prod_{h \in frag} P(h) * \prod_{v \in leaves(frag)} \beta(v)$$

where $c(r)$ is the fractional counts to be calculated for rule *r*, a *frag* is either *lhs*(*r*) (excluding virtual nodes and virtual hyper-edges) or any tree node in a forest, *TOP* is the root of the forest, $\alpha(.)$ and $\beta(.)$ are the outside and inside probabilities of nodes, $root(.)$ returns the root nodes of a tree sequence fragment, $leaves(.)$ returns the leaf nodes of a tree sequence fragment, $p(h)$ is the hyper-edge probability.

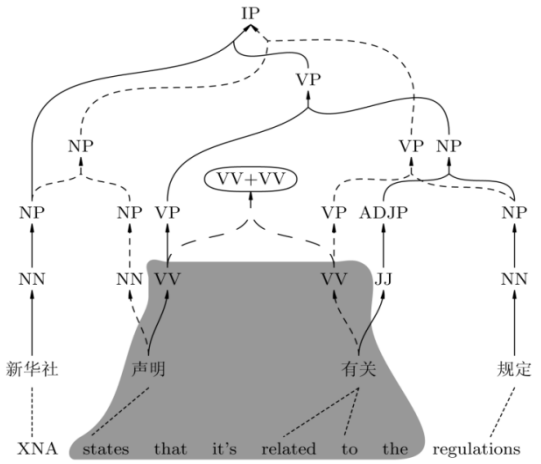

Figure 7. A virtual node in forest

5 Decoding

We benefit from the same strategy as used in our rule extraction algorithm in designing our decoding algorithm, recasting the forest-based tree sequence-to-string decoding problem as a forest-based tree-to-string decoding problem. Our decoding algorithm consists of four steps:

1) Convert the complete parse forest to a non-complete one by introducing virtual nodes.

2) Convert the non-complete parse forest into a translation forest[4] *TF* by using the translation rules and the pattern-matching algorithm presented in Mi et al. (2008).

3) Prune out redundant nodes and add auxiliary hyper-edge into the translation forest for those nodes that have either no child or no father. By this step, the translation forest *TF* becomes a complete forest.

4) Decode the translation forest using our translation model and a dynamic search algorithm.

The process of step 1 is similar to Algorithm 1 except no alignment constraint used here. This may generate a large number of additional virtual nodes; however, all redundant nodes will be filtered out in step 3. In step 2, we employ the tree-to-string pattern match algorithm (Mi et al., 2008) to convert a parse forest to a translation forest. In step 3, all those nodes not covered by any translation rules are removed. In addition, please note that the translation forest is already not a complete forest due to the virtual nodes and the pruning of rule-unmatchable nodes. We, therefore, propose Algorithm 2 to add auxiliary hyper-edges to make the translation forest complete.

In Algorithm 2, we travel the forest in bottom-up fashion (line 4-5). For each span, we do:

1) generate all the *NSS* for this span (line 7-12)

2) filter the *NSS* to a manageable size (line 13)

3) add auxiliary hyper-edges for the current span (line 15-19) if it can be covered by at least one single tree node, otherwise go to step 1 . This is the key step in our Algorithm 2. For each tree node and each node sequences covering the same span (stored in the current *NSS*), if the tree node has no children or at least one node in the node sequence has no father, we add an auxiliary hyper-edge to connect the tree node as father node with the node sequence as children. Since Algorithm 2 is DP-based and traverses the forest in a bottom-up way, all the nodes in a node sequence should already have children node after the lower level process in a small span. Finally, we re-build the *NSS* of current span for upper level *NSS* combination use (line 20-22).

In Fig. 8, the hyper-edge "IP=>NP VV+VV NP" is an auxiliary hyper-edge introduced by Algorithm 2. By Algorithm 2, we convert the translation forest into a complete translation forest. We then use a bottom-up node-based search algorithm to do decoding on the complete translation forest. We also use Cube Pruning algorithm (Huang and Chiang 2007) to speed up the translation process.

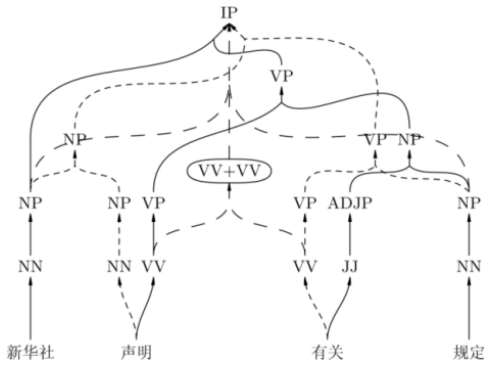

Figure 8. Auxiliary hyper-edge in a translation forest

Algorithm 2. *add auxiliary hyper-edges into mt forest F*
Input: *mt forest F*
Output: *complete forest F with auxiliary hyper-edges*

```
1.  for i := 1 to L do
2.      for each node n of span [i, i] do
3.          add n into NSS(i, i)
4.  for length := 1 to L - 1 do
5.      for start := 1 to L - length do
6.          stop := start + length
7.          for pivot := start to stop-1 do
8.              for each ns1 in NSS (start, pivot) do
9.                  for each ns2 in NSS (pivot+1,stop) do
10.                     create ns := ns1 ⊕ ns2
11.                     if ns is not in NSS(start, stop) then
12.                         add ns into NSS (start, stop)
13.         do pruning on NSS(start, stop)
14.         if there is tree node cover span [start, stop] then
15.             for each tree node n of span [start,stop] do
16.                 for each ns of NSS(start, stop) do
17.                     if node n have no children or
                            there is node in ns with no father
                            then
18.                         add auxiliary hyper-edge h into F
19.                         let lhs(h) := n, rhs(h) := ns
20.             empty NSS(start, stop)
21.             for each node n of span [start, stop] do
22.                 add n into NSS(start, stop)
```

6 Experiment

6.1 Experimental Settings

We evaluate our method on Chinese-English translation task. We use the FBIS corpus as training set, the NIST MT-2002 test set as development (dev) set and the NIST MT-2003 test set as test set. We train Charniak's parser (Charniak 2000) on CTB5 to do Chinese parsing, and modify it to output packed forest. We tune the parser on section 301-325 and test it on section 271-300. The F-measure on all sentences is 80.85%. A 3-gram language model is trained on the Xin-

[4] The concept of translation forest is proposed in Mi et al. (2008). It is a forest that consists of only the hyper-edges induced from translation rules.

hua portion of the English Gigaword3 corpus and the target side of the FBIS corpus using the SRILM Toolkits (Stolcke, 2002) with modified Kneser-Ney smoothing (Kenser and Ney, 1995). GIZA++ (Och and Ney, 2003) and the heuristics "grow-diag-final-and" are used to generate *m-to-n* word alignments. For the MER training (Och, 2003), Koehn's MER trainer (Koehn, 2007) is modified for our system. For significance test, we use Zhang et al.'s implementation (Zhang et al, 2004). Our evaluation metrics is case-sensitive BLEU-4 (Papineni et al., 2002).

For parse forest pruning (Mi et al., 2008), we utilize the Margin-based pruning algorithm presented in (Huang, 2008). Different from Mi et al. (2008) that use a static pruning threshold, our threshold is sentence-depended. For each sentence, we compute the Margin between the *n*-th best and the top 1 parse tree, then use the Margin-based pruning algorithm presented in (Huang, 2008) to do pruning. By doing so, we can guarantee to use at least all the top *n* best parse trees in the forest. However, please note that even after pruning there is still exponential number of additional trees embedded in the forest because of the sharing structure of forest. Other parameters are set as follows: maximum number of roots in a tree sequence is 3, maximum height of a translation rule is 3, maximum number of leaf nodes is 7, maximum number of node sequences on each span is 10, and maximum number of rules extracted from one node is 10000.

6.2 Experimental Results

We implement our proposed methods as a general, configurable platform for syntax-based SMT study. Based on this platform, we are able to easily implement most of the state-of-the-art syntax-based x-to-string SMT methods via simple parameter setting. For training, we set forest pruning threshold to 1 best for tree-based methods and 100 best for forest-based methods. For decoding, we set:

1) TT2S: tree-based tree-to-string model by setting the forest pruning threshold to 1 best and the number of sub-trees in a tree sequence to 1.

2) TTS2S: tree-based tree-sequence to string system by setting the forest pruning threshold to 1 best and the maximum number of sub-trees in a tree sequence to 3.

3) FT2S: forest-based tree-to-string system by setting the forest pruning threshold to 500 best, the number of sub-trees in a tree sequence to 1.

4) FTS2S: forest-based tree-sequence to string system by setting the forest pruning threshold to 500 best and the maximum number of sub-trees in a tree sequence to 3.

Model	BLEU(%)
Moses	25.68
TT2S	26.08
TTS2S	26.95
FT2S	27.66
FTS2S	**28.83**

Table 1. Performance Comparison

We use the first three syntax-based systems (TT2S, TTS2S, FT2S) and Moses (Koehn et al., 2007), the state-of-the-art phrase-based system, as our baseline systems. Table 1 compares the performance of the five methods, all of which are fine-tuned. It shows that:

1) FTS2S significantly outperforms ($p<0.05$) FT2S. This shows that tree sequence is very useful to forest-based model. Although a forest can cover much more phrases than a single tree does, there are still many non-syntactic phrases that cannot be captured by a forest due to structure divergence issue. On the other hand, tree sequence is a good solution to non-syntactic translation equivalence modeling. This is mainly because tree sequence rules are only sensitive to word alignment while tree rules, even extracted from a forest (like in FT2S), are also limited by syntax according to grammar parsing rules.

2) FTS2S shows significant performance improvement ($p<0.05$) over TTS2S due to the contribution of forest. This is mainly due to the fact that forest can offer very large number of parse trees for rule extraction and decoder.

3) Our model statistically significantly outperforms all the baselines system. This clearly demonstrates the effectiveness of our proposed model for syntax-based SMT. It also shows that the forest-based method and tree sequence-based method are complementary to each other and our proposed method is able to effectively integrate their strengths.

4) All the four syntax-based systems show better performance than Moses and three of them significantly outperforms ($p<0.05$) Moses. This suggests that syntax is very useful to SMT and translation can be viewed as a structure mapping issue as done in the four syntax-based systems.

Table 2 and Table 3 report the distribution of different kinds of translation rules in our model (training forest pruning threshold is set to 100 best) and in our decoding (decoding forest pruning threshold is set to 500 best) for one best translation generation. From the two tables, we can find that:

Rule Type	Tree to String	Tree Sequence to String
L	4,854,406	20,526,674
P	37,360,684	58,826,261
U	3,297,302	3,775,734
All	45,512,392	83,128,669

Table 2. # of rules extracted from training corpus. **L** means fully lexicalized, **P** means partially lexicalized, **U** means unlexicalized.

Rule Type	Tree to String	Tree Sequence to String
L	10,592	1,161
P	7,132	742
U	4,874	278
All	22,598	2,181

Table 3. # of rules used to generate one-best translation result in testing

1) In Table 2, the number of tree sequence rules is much larger than that of tree rules although our rule extraction algorithm only extracts those tree sequence rules over the spans that tree rules cannot cover. This suggests that the non-syntactic structure mapping is still a big challenge to syntax-based SMT.

2) Table 3 shows that the tree sequence rules is around 9% of the tree rules when generating the one-best translation. This suggests that around 9% of translation equivalences in the test set can be better modeled by tree sequence to string rules than by tree to string rules. The 9% tree sequence rules contribute 1.17 BLEU score improvement (28.83-27.66 in Table 1) to FTS2S over FT2S.

3) In Table 3, the fully-lexicalized rules are the major part (around 60%), followed by the partially-lexicalized (around 35%) and unlexicalized (around 15%). However, in Table 2, partially-lexicalized rules extracted from training corpus are the major part (more than 70%). This suggests that most partially-lexicalized rules are less effective in our model. This clearly directs our future work in model optimization.

	BLEU (%)	
N-best \ model	**FT2S**	**FTS2S**
100 Best	27.40	28.61
500 Best	27.66	28.83
2500 Best	27.66	**28.96**
5000 Best	27.79	28.89

Table 4. Impact of the forest pruning

Forest pruning is a key step for forest-based method. Table 4 reports the performance of the two forest-based models using different values of the forest pruning threshold for decoding. It shows that:

1) FTS2S significantly outperforms ($p<0.05$) FT2S consistently in all test cases. This again demonstrates the effectiveness of our proposed model. Even if in the 5000 Best case, tree sequence is still able to contribute 1.1 BLEU score improvement (28.89-27.79). It indicates the advantage of tree sequence cannot be covered by forest even if we utilize a very large forest.

2) The BLEU scores are very similar to each other when we increase the forest pruning threshold. Moreover, in one case the performance even drops. This suggests that although more parse trees in a forest can offer more structure information, they may also introduce more noise that may confuse the decoder.

7 Conclusion

In this paper, we propose a forest-based tree-sequence to string translation model to combine the strengths of forest-based methods and tree-sequence based methods. This enables our model to have the great potential to address the issues of structure divergence and parse errors for syntax-based SMT. We convert our forest-based tree sequence rule extraction and decoding issues to tree-based by introducing virtual nodes, virtual hyper-edges and auxiliary rules (hyper-edges). In our system implementation, we design a general and configurable platform for our method, based on which we can easily realize many previous syntax-based methods. Finally, we examine our methods on the FBIS corpus and the NIST MT-2003 Chinese-English translation task. Experimental results show that our model greatly outperforms the four baseline systems. Our study demonstrates that forest-based method and tree sequence-based method are complementary to each other and our proposed method is able to effectively combine the strengths of the two individual methods for syntax-based SMT.

Acknowledgement

We would like to thank Huang Yun for preparing the pictures in this paper; Run Yan for providing the java version modified MERT program and discussion on the details of MOSES; Mi Haitao for his help and discussion on re-implementing the FT2S model; Sun Jun and Xiong Deyi for their valuable suggestions.

References

Eugene Charniak. 2000. *A maximum-entropy inspired parser*. NAACL-00.

Jason Eisner. 2003. *Learning non-isomorphic tree mappings for MT*. ACL-03 (companion volume).

Michel Galley, Mark Hopkins, Kevin Knight and Daniel Marcu. 2004. *What's in a translation rule?* HLT-NAACL-04. 273-280.

Liang Huang. 2008. Forest Reranking: *Discriminative Parsing with Non-Local Features*. ACL-HLT-08. 586-594

Liang Huang and David Chiang. 2005. *Better k-best Parsing*. IWPT-05.

Liang Huang and David Chiang. 2007. *Forest rescoring: Faster decoding with integrated language models*. ACL-07. 144–151

Liang Huang, Kevin Knight and Aravind Joshi. 2006. *Statistical Syntax-Directed Translation with Extended Domain of Locality*. AMTA-06. (poster)

Reinhard Kenser and Hermann Ney. 1995. *Improved backing-off for M-gram language modeling*. ICASSP-95. 181-184

Dan Klein and Christopher D. Manning. 2001. *Parsing and Hypergraphs*. IWPT-2001.

Philipp Koehn, F. J. Och and D. Marcu. 2003. *Statistical phrase-based translation.* HLT-NAACL-03. 127-133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin and Evan Herbst. 2007. *Moses: Open Source Toolkit for Statistical Machine Translation.* ACL-07. 177-180. (poster)

Yang Liu, Qun Liu and Shouxun Lin. 2006. *Tree-to-String Alignment Template for Statistical Machine Translation.* COLING-ACL-06. 609-616.

Yang Liu, Yun Huang, Qun Liu and Shouxun Lin. 2007. *Forest-to-String Statistical Translation Rules.* ACL-07. 704-711.

Haitao Mi, Liang Huang, and Qun Liu. 2008. *Forest-based translation*. ACL-HLT-08. 192-199.

Haitao Mi and Liang Huang. 2008. *Forest-based Translation Rule Extraction*. EMNLP-08. 206-214.

Franz J. Och and Hermann Ney. 2002. *Discriminative training and maximum entropy models for statistical machine translation.* ACL-02. 295-302.

Franz J. Och. 2003. *Minimum error rate training in statistical machine translation.* ACL-03. 160-167.

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

Kishore Papineni, Salim Roukos, ToddWard and Wei-Jing Zhu. 2002. *BLEU: a method for automatic evaluation of machine translation.* ACL-02. 311-318.

Andreas Stolcke. 2002. *SRILM - an extensible language modeling toolkit.* ICSLP-02. 901-904.

Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Sheng Li and Chew Lim Tan. 2007. *A Tree-to-Tree Alignment-based Model for Statistical Machine Translation*. MT-Summit-07. 535-542.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, Sheng Li. 2008a. *A Tree Sequence Alignment-based Tree-to-Tree Translation Model.* ACL-HLT-08. 559-567.

Min Zhang, Hongfei Jiang, Haizhou Li, Aiti Aw, Sheng Li. 2008b. *Grammar Comparison Study for Translational Equivalence Modeling and Statistical Machine Translation.* COLING-08. 1097-1104.

Ying Zhang, Stephan Vogel, Alex Waibel. 2004. *Interpreting BLEU/NIST scores: How much improvement do we need to have a better system?* LREC-04. 2051-2054.