Unsupervised Tree Induction for Tree-based Translation

Feifei Zhai, Jiajun Zhang, Yu Zhou and Chengqing Zong
National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, China
{ffzhai,jjzhang,yzhou,cqzong}@nlpr.ia.ac.cn

Abstract

In current research, most tree-based translation models are built directly from parse trees. In this study, we go in another direction and build a translation model with an unsupervised tree structure derived from a novel non-parametric Bayesian model. In the model, we utilize synchronous tree substitution grammars (STSG) to capture the bilingual mapping between language pairs. To train the model efficiently, we develop a Gibbs sampler with three novel Gibbs operators. The sampler is capable of exploring the infinite space of tree structures by performing local changes on the tree nodes. Experimental results show that the string-to-tree translation system using our Bayesian tree structures significantly outperforms the strong baseline string-to-tree system using parse trees.

1 Introduction

In recent years, tree-based translation models[1] are drawing more and more attention in the community of statistical machine translation (SMT). Due to their remarkable ability to incorporate context structure information and long distance reordering into the translation process, tree-based translation models have shown promising progress in improving translation quality (Liu et al., 2006, 2009; Quirk et al., 2005; Galley et al., 2004, 2006; Marcu et al., 2006; Shen et al., 2008; Zhang et al., 2011b).

However, tree-based translation models always suffer from two major challenges: 1) They are usually built directly from parse trees, which are generated by supervised linguistic parsers. However, for many language pairs, it is difficult to acquire such corresponding linguistic parsers due to the lack of Tree-bank resources for training. 2) Parse trees are actually only used to model and explain the monolingual structure, rather than the bilingual mapping between language pairs. This indicates that parse trees are usually not the optimal choice for training tree-based translation models (Wang et al., 2010).

Based on the above analysis, we can conclude that the tree structure that is independent from Tree-bank resources and simultaneously considers the bilingual mapping inside the bilingual sentence pairs would be a good choice for building tree-based translation models.

Therefore, complying with the above conditions, we propose an unsupervised tree structure for tree-based translation models in this study. In the structures, tree nodes are labeled by combining the word classes of their boundary words rather than by syntactic labels, such as NP, VP. Furthermore, using these node labels, we design a generative Bayesian model to infer the final tree structure based on synchronous tree substitution grammars (STSG)[2]. STSG is derived from the word alignments and thus can grasp the bilingual mapping effectively.

Training the Bayesian model is difficult due to the exponential space of possible tree structures for each training instance. We therefore develop an efficient Gibbs sampler with three novel Gibbs operators for training. The sampler is capable of exploring the infinite space of tree structures by performing local changes on the tree nodes.

[1] A tree-based translation model is defined as a model using tree structures on one side or both sides.

[2] We believe it is possible to design a model to infer the node label and tree structure jointly. We plan this as future work, and here, we focus only on inferring the tree structure in terms of the node labels derived from word classes.

Transactions of the Association for Computational Linguistics, 1 (2013) 243–254. Action Editor: Philipp Koehn.
Submitted 12/2012; Revised 3/2013; Published 5/2013.

The tree structure formed in this way is independent from the Tree-bank resources and simultaneously exploits the bilingual mapping effectively. Experiments show that the proposed unsupervised tree (U-tree) is more effective and reasonable for tree-based translation than the parse tree.

The main contributions of this study are as follows:

1) Instead of the parse tree, we propose a Bayesian model to induce a U-tree for tree-based translation. The U-tree exploits the bilingual mapping effectively and does not rely on any Tree-bank resources.
2) We design a Gibbs sampler with three novel Gibbs operators to train the Bayesian model efficiently.

The remainder of the paper is organized as follows. Section 2 introduces the related work. Section 3 describes the STSG generation process, and Section 4 depicts the adopted Bayesian model. Section 5 describes the Gibbs sampling algorithm and Gibbs operators. In Section 6, we analyze the achieved U-trees and evaluate their effectiveness. Finally, we conclude the paper in Section 7.

2 Related Work

In this study, we move in a new direction to build a tree-based translation model with effective unsupervised U-tree structures.

For unsupervised tree structure induction, DeNero and Uszkoreit (2011) adopted a parallel parsing model to induce unlabeled trees of source sentences for syntactic pre-reordering. Our previous work (Zhai et al., 2012) designed an EM-based method to construct unsupervised trees for tree-based translation models. This work differs from the above work in that we design a novel Bayesian model to induce unsupervised U-trees, and prior knowledge can be encoded into the model more freely and effectively.

Blunsom et al. (2008, 2009, 2010) utilized Bayesian methods to learn synchronous context free grammars (SCFG) from a parallel corpus. The obtained SCFG is further used in a phrase-based and hierarchical phrase-based system (Chiang, 2007). Levenberg et al. (2012) employed a Bayesian method to learn discontinuous SCFG rules. This study differs from their work because we concentrate on constructing tree structures for tree-based translation models. Our U-trees are learned based on STSG, which is more appropriate for tree-based translation models than SCFG.

Burkett and Klein (2008) and Burkett et al. (2010) focused on joint parsing and alignment. They utilized the bilingual Tree-bank to train a joint model for both parsing and word alignment. Cohn and Blunsom (2009) adopted a Bayesian method to infer an STSG by exploring the space of alignments based on parse trees. Liu et al. (2012) re-trained the linguistic parsers bilingually based on word alignment. Burkett and Klein (2012) utilized a transformation-based method to learn a sequence of monolingual tree transformations for translation. Compared to their work, we do not rely on any Tree-bank resources and focus on generating effective unsupervised tree structures for tree-based translation models.

Zollmann and Venugopal (2006) substituted the non-terminal X in hierarchical phrase-based model by extended syntactic categories. Zollmann and Vogel (2011) further labeled the SCFG rules with POS tags and unsupervised word classes. Our work differs from theirs in that we present a Bayesian model to learn effective STSG translation rules and U-tree structures for tree-based translation models, rather than designing a labeling strategy for translation rules.

3 The STSG Generation Process

In this work, we induce effective U-trees for the string-to-tree translation model, which is based on a synchronous tree substitution grammar (STSG) between source strings and target tree fragments. We take STSG as the generation grammar to match the translation model. Typically, such an STSG[3] is a 5-tuple as follows:

$$G = (\sum_s, \sum_t, N_t, S_t, P)$$

where:

- $\sum_s$ and $\sum_t$ represent the set of source and target words, respectively,
- N_t is the set of target non-terminals,
- $S_t \in N_t$ is the start root non-terminal, and
- P is the production rule set.

[3] Generally, an STSG involves tree fragments on both sides. Here we only consider the special case where the source side is actually a string.

Apart from the start non-terminal S_t, we define all the other non-terminals in N_t by word classes. Inspired by (Zollmann and Vogel, 2011), we divide these non-terminals into three categories: one-word, two-word and multi-word non-terminals. The one-word non-terminal is a word class, such as *C*, meaning that it dominates a word whose word class is *C*. Two-word non-terminals are used to stand for two word strings. They are labeled in the form of C_1+C_2, where C_1 and C_2 are the word classes of the two words separately. Accordingly, multi-word non-terminals represent the strings containing more than two words. They are labeled as $C_1 \ldots C_n$, demanding that the word classes of the leftmost word and the rightmost word are C_1 and C_n, respectively.

We use POS tag to play the role of word class[4]. For example, the head node of the rule in Figure 1 is a multi-word non-terminal *PRP...RB*. It requires that the POS tags of the leftmost and rightmost word must be *PRP* and *RB*, respectively. Xiong et al. (2006) showed that the boundary word is an effective indicator for phrase reordering. Thus, we believe that combining the word class of boundary words can denote the whole phrase well.

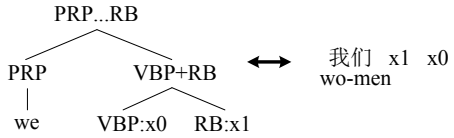

Figure 1. An example of an STSG production rule.

Each production rule in *P* consists of a source string and a target tree fragment. In the target tree fragment, each internal node is labeled with a non-terminal in N_t, and each leaf node is labeled with either a target word in $\sum_t$ or a non-terminal in N_t. The source string in a production rule comprises source words and *variables*. Each variable corresponds to a leaf non-terminal in the target tree fragment. In the STSG, the production rule is used to rewrite the root node into a string and a tree fragment. For example, in Figure 1, the rule rewrites the head node *PRP...RB* into the corresponding string and fragment.

An STSG *derivation* refers to the process of generating a specific source string and target tree structure by production rules. This process begins with the start non-terminal S_t and an empty source string. We repeatedly choose production rules to rewrite the leaf non-terminals and expand the string until no leaf non-terminal is left. Finally, we acquire a source string and a target tree structure defined by the derivation. The probability of a derivation is given as follows:

$$p(d) = \prod_{i=1}^{n} p(r_i \mid N_i) \quad (1)$$

where the derivation comprises a sequence of rules $d=(r_1,\ldots,r_n)$, and N_i represents the root node of rule r_i. Hence, for a specific bilingual sentence pair, we can generate the best target-side tree structure based on the STSG, independent from the Tree-bank resources. The STSG used in the above process is learned by the Bayesian model that is detailed in the next section.

Actually, SCFG can also be used to build the U-trees. We do not use SCFG because most of the tree-based models are based on STSG. In our Bayesian model, the U-trees are optimized through selecting a set of STSG rules. These STSG rules are consistent with the translation rules used in the tree-based models.

Another reason is that STSG has a stronger expressive power on tree construction than SCFG. In a STSG-based U-tree or a STSG rule, although not linguistically informed, the nodes labeled by POS tags are also effective on distinguishing different ones. However, with SCFG, we have to discard all the internal nodes (i.e., flattening the U-trees or rules) to express the same sequence, leading to a poor ability of distinguishing different U-trees and production rules. Thus, using STSG, we can build more specific U-trees for translation.

In addition, we find that the Bayesian SCFG grammar cannot even significantly outperform the heuristic SCFG grammar (Blunsom et al. 2009)[5]. This would indicate that the SCFG-based derivation tree as by-product is also not such good for tree-based translation models. Considering the above reasons, we believe that the STSG-based learning procedure would result in a better translation grammar for tree-based models.

[4] The demand of a POS tagger impairs the independence from manual resources to some extent. In future, we plan to design a method to learn effective unsupervised labels for the non-terminals.

[5] In (Blunsom et al., 2009), for Chinese-to-English translation, the Bayesian SCFG grammar only outperform the heuristic SCFG grammar by 0.1 BLEU points on NIST MT 2004 and 0.6 BLEU points on NIST MT 2005 in the NEWS domain.

4 Bayesian Model

In this section, we present a Bayesian model to learn STSG defined in section 3. In the model, we use θ_N to denote the probability distribution $p(r \mid N)$ in Equation (1). θ_N follows a multinomial distribution and we impose a Dirichlet prior (DP) on it:

$$\begin{aligned} r \mid N \quad &\sim Multi(\theta_N) \\ \theta_N \mid \alpha_N, P_0 \ &\sim DP(\alpha_N, P_0(\bullet \mid N)) \end{aligned} \tag{2}$$

where $P_0(\bullet \mid N)$ (*base distribution*) is used to assign prior probabilities to the STSG production rules. α_N controls the model's tendency to either reuse existing rules or create new ones using the base distribution $P_0(\bullet \mid N)$.

Instead of denoting the multinomial distribution explicitly with a specific θ_N, we integrate over all possible values of θ_N to achieve the probabilities of rules. This integration results in the following conditional probability for rule r_i given the previously observed rules $r^{-i} = r_1, \ldots, r_{i-1}$:

$$p(r_i \mid r^{-i}, N, \alpha_N, P_0) = \frac{n_{r_i}^{-i} + \alpha_N P_0(r_i \mid N)}{n_N^{-i} + \alpha_N} \tag{3}$$

Where $n_{r_i}^{-i}$ denotes the number of r_i in r^{-i}, and n_N^{-i} represents the total count of rules rewriting non-terminal N in r^{-i}. Thanks to the exchangeability of the model, all permutations of the rules are actually equiprobable. This means that we can compute the probability of each rule based on the previous and subsequent rules (i.e. consider each rule as the last one). This characteristic allows us to design an efficient Gibbs sampling algorithm to train the Bayesian model.

4.1 Base Distribution

The base distribution $P_0(r \mid N)$ is designed to assign prior probabilities to the STSG production rules. Because each rule r consists of a target tree fragment *frag* and a source string *str* in the model, we follow Cohn and Blunsom (2009) and decompose the prior probability $P_0(r \mid N)$ into two factors as follows:

$$P_0(r \mid N) = P(frag \mid N) \cdot P(str \mid frag) \tag{4}$$

where $P(frag \mid N)$ is the probability of producing the target tree fragment *frag*. To generate *frag*, Cohn and Blunsom (2009) used a geometric prior to decide how many child nodes to assign each node. Differently, we require that each multi-word non-terminal node must have two child nodes. This is because the binary structure has been verified to be very effective for tree-based translation (Wang et al., 2007; Zhang et al., 2011a).

The generation process starts at root node *N*. At first, root node *N* is expanded into two child nodes. Then, each newly generated node will be checked to expand into two new child nodes with probability p_{expand}. This process repeats until all the new non-terminal nodes are checked. Obviously, p_{expand} controls the scale of tree fragments, where a large p_{expand} corresponds to large fragments[6]. The new terminal nodes (words) are drawn uniformly from the target-side vocabulary, and the non-terminal nodes are created by asking two questions as follows:

1) What type is the node, one-word, two-word or multi-word non-terminal?
2) What tag is used to label the node?

The answer to question 1) is chosen from a uniform distribution, i.e., the probability is 1/3 for each type of non-terminal. The entire generation process is in a top-down manner, i.e., generating a parent node first and then its children.

With respect to question 2), because the father node has determined the POS tags of boundary words, we only need one POS tag to generate the label of the current node. For example, in Figure 1, as the father node *PRP...RB* demands that the POS tag of the rightmost word is *RB*, the right child of *PRP...RB* must also satisfy this condition. Therefore, we choose a POS tag *VBP* and obtain the label *VBP+RB*. The POS tag is drawn uniformly from the POS tag set. If the current node is a one-word non-terminal, question 2) is unnecessary. Similarly, with respect to the two-word non-terminal node, questions 1) and 2) are both unnecessary for its two child nodes because they have already been defined by their father node.

As an example of the generative process, the tree fragment in Figure 1 is created as follows:

a. Determine that the left child of *PRP...RB* is a one-word non-terminal (labeled with *PRP*);
b. Expand *PRP* and generate the word "we" for *PRP*;

[6] In our experiment, we set p_{expand} to 1/3 to encourage small tree fragments.

c. Determine that the right child of *PRP…RB* is a two-word non-terminal;
d. Utilize the predetermined *RB* and a POS tag *VBP* to form the tag of the two-word non-terminal: *VBP+RB*;
e. Expand *VBP+RB* (to *VBP* and *RB*);
f. Do not expand *VBP* and *RB*.

$P(str \mid frag)$ in Equation (4) is the probability of generating the source string, which contains several source words and variables. Inspired by (Blunsom et al., 2009) and (Cohn and Blunsom, 2009), we define $P(str \mid frag)$ as follows:

$$P(str \mid frag) = P_{poisson}(c_{sw};1) \times \frac{1}{|\sum_s|^{c_{sw}}} \times \prod_{i=1}^{c_{var}} \frac{1}{c_{sw}+i} \quad (5)$$

where c_{sw} is the number of words in the source string. $\sum_s$ means the source vocabulary set. Further, c_{var} denotes the number of variables, which is determined by the tree fragment *frag*.

As shown in Equation(5), we first determine how many source words to generate using a Poisson distribution $P_{poisson}(c_{sw};1)$, which imposes a stable preference for short source strings. Then, we draw each source word from a uniform distribution over $\sum_s$. Afterwards, we insert the variables into the string. The variables are inserted one at a time using a uniform distribution over the possible positions. This factor discourages more variables.

For the example rule in Figure 1, the generative process of the source string is:

a. Decide to generate one source word;
b. Generate the source word "我们 (wo-men) ";
c. Insert the first variable after the word;
d. Insert the second variable between the word and the first variable.

Intuitively, a good translation grammar should carry both small translation rules with enough generality and large rules with enough context information. DeNero and Klein (2007) proposed this statement, and Cohn and Blunsom (2009) has verified it in their experiments with parse trees.

Our base distribution is also designed based on this intuition. Considering the two factors in our base distribution, we penalize both large target tree fragments with many nodes and long source strings with many words and variables. The Bayesian model tends to select both small and frequent STSG production rules to construct the U-trees. With these types of trees, we can extract small rules with good generality and simultaneously obtain large rules with enough context information by composition. We will show the effectiveness of our U-trees in the verification experiment.

5 Model Training by Gibbs Sampling

In this section, we introduce a collapsed Gibbs sampler, which enables us to train the Bayesian model efficiently.

5.1 Initialization State

At first, we use random binary trees to initialize the sampler. To get the initial U-trees, we recursively and randomly segment a sentence into two parts and simultaneously create a tree node to dominate each part. The created tree nodes are labeled by the non-terminals described in section 3.

Using the initial target U-trees, source sentences and word alignment, we extract minimal GHKM translation rules[7] in terms of frontier nodes (Galley et al., 2004). Frontier nodes are the tree nodes that can map onto contiguous substrings on the source side via word alignment. For example, the bold italic nodes with shadows in Figure 2 are frontier nodes. In addition, it should be noted that the word alignment is fixed[8], and we only explore the entire space of tree structures in our sampler. Differently, Cohn and Blunsom (2009) designed a sampler to infer an STSG by fixing the tree structure and exploring the space of alignment. We believe that it is possible to investigate the space of both tree structure and alignment simultaneously. This subject will be one of our future work topics.

For each training instance (a pair of source sentence and target U-tree structure), the extracted GHKM minimal translation rules compose a unique STSG derivation[9]. Moreover, all the rules developed from the training data constitute an initial STSG for the Gibbs sampler.

[7] We attach the unaligned word to the lowest frontier node that can cover it in terms of word alignment.

[8] The sampler might reinforce the frequent alignment errors (AE), which would harm the translation model (TM). Actually, the frequent AEs also greatly impair the conventional TM. Besides, our sampler encourages the correct alignments and simultaneously discourages the infrequent AEs. Thus, compared with the conventional TMs, we believe that our final TM would not be worse due to AEs. Our final experiments verify this point and we will conduct a much detailed analysis in future.

[9] We only use the minimal GHKM rules (Galley et al., 2004) here to reduce the complexity of the sampler.

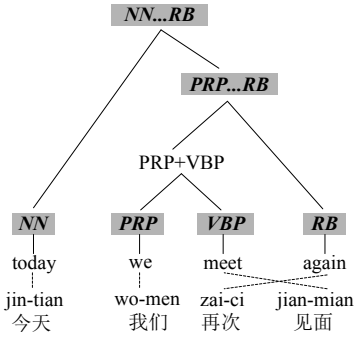

Figure 2. Illustration of an initial U-tree structure. The bold italic nodes with shadows are frontier nodes.

Under this initial STSG, the sampler modifies the initial U-trees (initial sample) to create a series of new ones (new samples) by the Gibbs operators. Consequently, new STSGs are created based on the new U-trees simultaneously and used for the next sampling operation. Repeatedly and after a number of iterations, we can obtain the final U-trees for building translation models.

5.2 The Gibbs Operators

In this section, we develop three novel Gibbs operators for the sampler. They explore the entire space of the U-tree structures by performing local changes on the tree nodes.

For a U-tree of a given sentence, we define ***s-node*** as the non-root node covering at least two words. Thus, the set of s-node contains all the tree nodes except the root node, the pre-terminal nodes and leaf nodes, which we call ***non-s-node***. For example, in Figure 2, *PRB…RB* and *PRP+VBP* are s-nodes, while *NN* and *NN…RB* are non-s-nodes. Since the POS tag sequence of the sentence is fixed, all non-s-nodes would stay unchanged in all possible U-trees of the sentence. Based on this fact, our Gibbs operators work only on s-nodes.

Further, we assign 3 descendant candidates (DC) for each s-node: its left child, right child and its sibling. For example, in Figure 3, the 3 DCs for the s-node are node *PRP*, *VBP* and *RB* respectively. According to the different DCs it governs, every s-node might be in one of the two different states:

1) ***Left state***: as Figure 3(a) shows, the s-node governs the left two DCs, *PRP* and *VBP*, and is labeled *PRP+VBP*.
2) ***Right state***: as Figure 3(b) shows, the s-node governs the right two DCs, *VBP* and *RB*, and is labeled *VBP+RB*.

For a specific U-tree, the states of s-nodes are fixed. Thus, by changing an s-node's state, we can easily transform this U-tree to another one, i.e., from the current sample to a new one.

To formulate the U-tree transformation process, we associate a binary variable $\Psi \in \{0,1\}$ with each s-node, indicating whether the s-node is in the left (Ψ=0) or right state (Ψ=1). Then we can change the U-tree by changing value of the Ψ parameters.

Our first Gibbs operator, ***Rotate***, just works by sampling value of the Ψ parameters, one at a time, and changing the U-tree accordingly. For example, in Figure 3(a), the s-node is currently in the left state (Ψ=0). We sample the Ψ of this node, and if the sampled value of Ψ is 0, we keep the structure unchanged, i.e., in the left state. Otherwise, we change its state to the right state (Ψ=1), and transform the U-tree to Figure 3(b) accordingly.

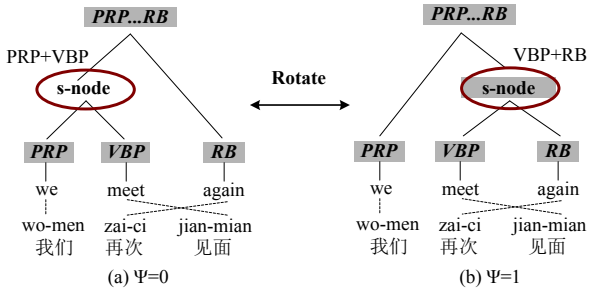

Figure 3. Illustration of the *Rotate* operator. In the figure, (a) and (b) denote the s-node's left state and right state respectively. The bold italic nodes with shadows in the figure are frontier nodes.

Obviously, towards an s-node for sampling, the two values of Ψ would define two different U-trees. Using the GHKM algorithm (Galley et al. 2004), we can get two different STSG derivations from the two U-trees based on the fixed word alignment. Each derivation carries a set of STSG rules (i.e., minimal GHKM translation rules) of its own. In the two derivations, the STSG rules defined by the two states include the one rooted at the s-node's lowest ancestor frontier node, and the one rooted at the s-node if it is a frontier node. For instance, in Figure 3(a), as the s-node is not a frontier node, the left state (Ψ=0) defines only one rule:

$$r_{left} : x_0\ x_2\ x_1 \rightarrow$$
$$PRP...RB(PRP+VBP(x_0 : PRP\ x_1 : VBP)\ x_2 : RB)$$

Differently, in Figure 3(b), the s-node is a frontier node and thus the right state (Ψ=1) defines two rules:

$$r_{right-0} : x_0\ x_1 \rightarrow PRP...RB(x_0 : PRP\ x_1 : VBP + RB)$$
$$r_{right-1} : x_1\ x_0 \rightarrow VBP + RB(x_0 : VBP\ x_1 : RB)$$

Using these STSG rules, the two derivations are evaluated as follows (We use the value of Ψ to denote the corresponding STSG derivation):

$$p(\Psi = 0) \propto p(r_{left} \mid r^-)$$
$$p(\Psi = 1) \propto p(r_{right-0}, r_{right-1} \mid r^-)$$
$$= p(r_{right-0} \mid r^-)\, p(r_{right-1} \mid r_{right-0}, r^-)$$

Where r^- refers to the conditional context, i.e., the set of all other rules in the training data. All the probabilities in the above formulas are computed by Equation(3). We then normalize the two scores and sample a value of Ψ based on them. With the Bayesian model described in section 4, the sampler will prefer the Ψ that produces small and frequent STSG rules. This tendency results in more frontier nodes in the U-tree (i.e., the s-node tends to be in the state that is a frontier node), which will factor the training instance into more small STSG rules. In this way, the overall likelihood of the bilingual data is improved by the sampler.

Theoretically, the *Rotate* operator is capable of arriving at any possible U-tree from the initial U-tree. This is because we can first convert the initial U-tree to a left branch tree by the *Rotate* operator, and then transform it to any other U-tree. However, it may take a long time to do so. Thus, to speed up the structure transformation process, we employ a ***Two-level-Rotate*** operator, which takes a pair of s-nodes in a parent-child relationship as a unit for sampling. Similar to the *Rotate* operator, we also assign a binary variable ξ∈{0,1} to each unit and update the U-tree by sampling the value of ξ. The method of sampling ξ is similar to the one used for Ψ. Figure 4 shows an example of the operator. As shown in Figure 4(a), the unit *NN...VBP* and *PRP+VBP* is in the left state (ξ=0), and governs the left three descendants: *NN*, *PRP*, and *VBP*. By the *Two-level-Rotate* operator, we can convert the unit to Figure 4(b), i.e., the right state (ξ=1). Just as Figure 4(b) shows, the governed descendants of the unit are turned to *PRP*, *VBP*, and *RB*.

It may be confusing when choosing the parent-child s-node pair for sampling because the parent node always faces two choices: combining the left child or right child for sampling. To avoid confusion, we split the *Two-level-Rotate* operator into two operators: ***Two-level-left-Rotate*** operator, which works with the parent node and its left child, and ***Two-level-right-Rotate*** operator, which only considers the parent node and its right child[10]. Therefore, the operator used in Figure 4 is a *Two-level-right-Rotate* operator.

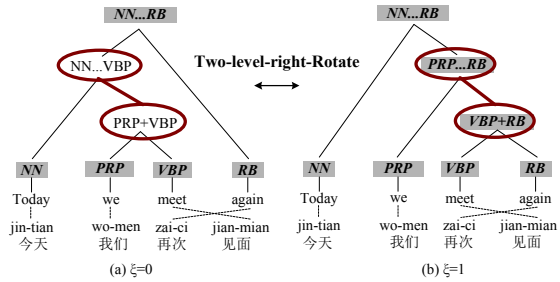

Figure 4. Illustration of the *Two-level-Rotate* operator. The bold italic nodes with shadows in the Figure are frontier nodes.

During sampling, for each training instance, the sampler first applies the *Two-level-left-Rotate* operator to all candidate pairs of s-nodes (parent s-node and its left child s-node) in the U-tree. After that, the *Two-level-right-Rotate* operator is applied to all the candidate pairs of s-nodes (parent s-node and its right child s-node). Then, we use the *Rotate* operator on every s-node in the U-tree. By utilizing the operators separately, we can guarantee that our sampler satisfies detailed balance. We visit all the training instances in a random order (one iteration). After a number of iterations, we can obtain the final U-tree structures and build the tree-based translation model accordingly.

6 Experiments

6.1 Experimental Setup

The experiments are conducted on Chinese-to-English translation. The training data are the FBIS corpus with approximately 7.1 million Chinese words and 9.2 million English words. We obtain the bidirectional word alignment with GIZA++, and then adopt the *grow-diag-final-and* strategy to obtain the final symmetric alignment. We train a 5-gram language model on the Xinhua portion of the English Gigaword corpus and the English part of

[10] We can also take more nodes as a unit for sampling, but this would make the algorithm much more complex.

the training data. For tuning and testing, we use the NIST MT 2003 evaluation data as the development set, and use the NIST MT04 and MT05 data as the test set. We use MERT (Och, 2004) to tune parameters. Since MERT is prone to search errors, we run MERT 5 times and select the best tuning parameters in the tuning set. The translation quality is evaluated by case-insensitive BLEU-4 with the shortest length penalty. The statistical significance test is performed by the re-sampling approach (Koehn, 2004).

To create the baseline system, we use the open-source Joshua 4.0 system (Ganitkevitch et al., 2012) to build a hierarchical phrase-based (HPB) system, and a syntax-augmented MT (SAMT) [11] system (Zollmann and Venugopal, 2006) respectively.

The translation system used for testing the effectiveness of our U-trees is our in-house string-to-tree system (abbreviated as ***s2t***). The system is implemented based on (Galley et al., 2006) and (Marcu et al. 2006). In the system, we extract both the minimal GHKM rules (Galley et al., 2004), and the rules of SPMT Model 1 (Galley et al., 2006) with phrases up to length $L=5$ on the source side. We then obtain the composed rules by composing two or three adjacent minimal rules.

To build the above *s2t* system, we first use the parse tree, which is generated by parsing the English side of the bilingual data with the Berkeley parser (Petrov et al., 2006). Then, we binarize the English parse trees using the head binarization approach (Wang et al., 2007) and use the resulting binary parse trees to build another *s2t* system.

For the U-trees, we run the Gibbs sampler for 1000 iterations on the whole corpus. The sampler uses 1,087s per iteration, on average, using a single core, 2.3 GHz Intel Xeon machine. For the hyperparameters, we set α to 0.1 and p_{expand} = 1/3 to give a preference to the rules with small fragments. We built an *s2t* translation system with the achieved U-trees after the 1000th iteration. We only use one sample to extract the translation grammar because multiple samples would result in a grammar that would be too large.

[11] From (Zollmann and Vogel, 2011), we find that the performance of SAMT system is similar with the method of labeling SCFG rules with POS tags. Thus, to be convenient, we only conduct experiments with the SAMT system.

6.2 Analysis of The Gibbs Sampler

To evaluate the effectiveness of the Gibbs sampler, we explore the change of the training data's likelihood with increasing sampling iterations.

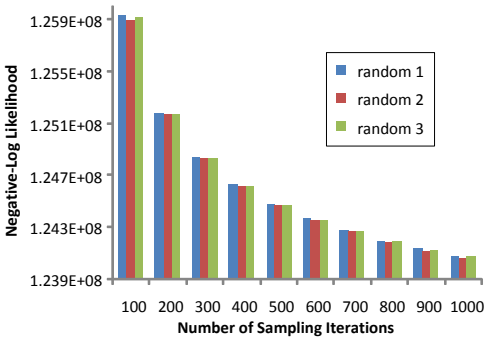

Figure 5. Histograms of the training data's likelihood vs. the number of sampling iterations. In the figure, random 1 to 3 refers to three independent runs of the sampler with different initial U-trees as initialization states.

Figure 5 depicts the negative-log likelihood of the training data after several sampling iterations. The results show that the overall likelihood of the training data is improved by the sampler. Moreover, comparing the three independent runs, we see that although the sampler begins with different initial U-trees, the training data's likelihood is always similar during sampling. This demonstrates that our sampler is not sensitive to the random initial U-trees and can always arrive at a good final state beginning from different initialization states. Thus, we only utilize the U-trees from random 1 for further analysis hereafter.

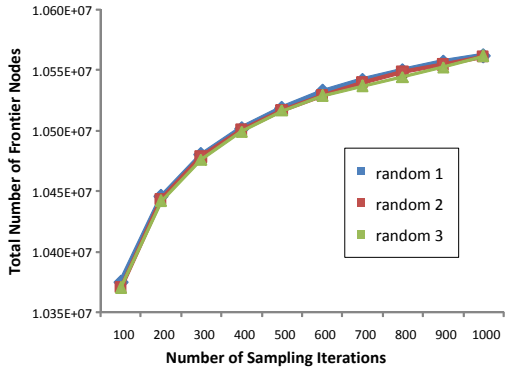

Figure 6. The total number of frontier nodes for the three independent runs.

6.3 Analysis of the U-tree Structure

Acquiring better U-trees for translation is our final purpose. However, are the U-trees achieved by the

Gibbs sampler appropriate for the tree-based translation model?

To answer this question, we first analyze the effect of the sampler on the U-trees. Figure 6 shows the total number of frontier nodes in the training data during sampling. The results show that the number of frontier nodes increases with increased sampling. This tendency indicates that our sampler prefers the tree structure with more frontier nodes. Consequently, the final U-tree structures can always be factored into many small minimal translation rules. Just as we have argued in section 4.1, this is beneficial for a good translation grammar.

To demonstrate the above analysis, Figure 7 shows a visual comparison between our U-tree (from random 1) and the binary parse tree (found by head binarization). Because the traditional parse tree is not binarized, we do not consider it for this analysis. Figure 7 shows that whether it is the target tree fragment or the source string of the rule, our U-trees always tend to obtain the smaller ones[12]. This comparison verifies that our Bayesian tree induction model is effective in shifting the tree structures away from complex minimal rules, which tend to negatively affect translation.

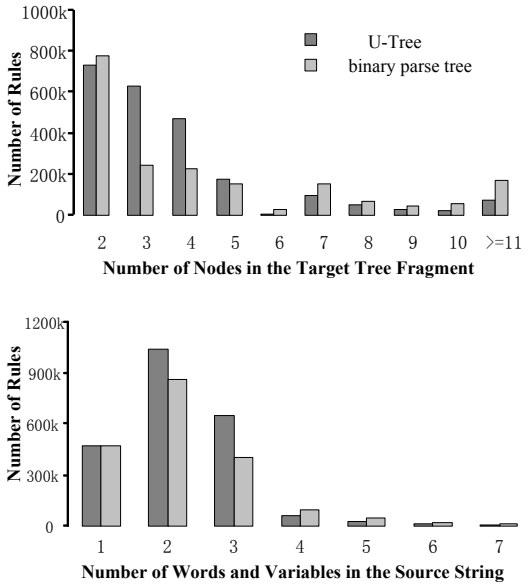

Figure 7. Histograms over minimal translation rule statistics comparing our U-trees and binary parse trees.

[12] Binary parse trees get more tree fragments with two nodes than U-trees. This is because there are many unary edges in the binary parse trees, while no unary edge exists in our U-trees.

Specifically, we show an example of a binary parse tree and our U-tree in Figure 8. The example U-tree is more conducive to extracting effective translation rules. For example, to translate the Chinese phrase "仅 为", we can extract a rule (R2 in Figure 9) directly from the U-tree because the phrase "仅 为" is governed by a frontier node, i.e., node "VBD+RB". However, because no node governs "仅 为" in the binary parse tree, we can only obtain a rule (R1 in Figure 9) with many extra nodes and edges, such as node CD in R1. Due to these extra things, R1 is too large to show good generality.

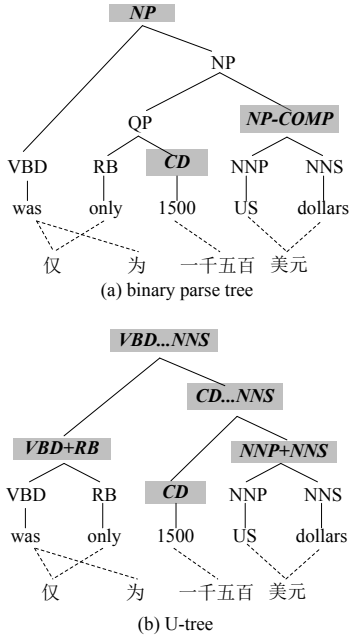

Figure 8. Example of different tree structures. The node NP-COMP is achieved by head binarization. The bold italic nodes with shadows denote frontier nodes.

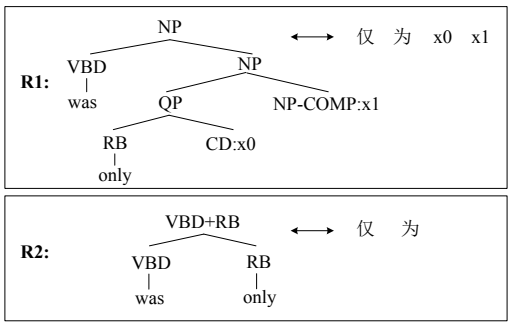

Figure 9. Example rules to translate the Chinese phrase "仅 为." R1 is extracted from Figure 8(a), i.e., the binary parse tree. R2 is from Figure 8(b), i.e., the U-tree.

Based on the above analysis, we can conclude that our proposed U-tree structures are conducive to extracting small, minimal translation rules. This indicates that the U-trees are more consistent with the word alignment and are good at capturing bilingual mapping information. Therefore, because parse trees are always constrained by cross-lingual structure divergence, we believe that the proposed U-trees would result in a better translation grammar. We demonstrate this conclusion in the next sub-section.

6.4 Final Translation Results

The final translation results are shown in Table 1. In the table, lines 3-6 refer to the string-to-tree systems built with different types of tree structures.

Table 1 shows that all our *s2t* systems outperform the *Joshua (HPB)* and *Joshua (SAMT)* system significantly. This comparison verifies the superiority of our in-house *s2t* system. Moreover, the results shown in Table 1 also demonstrate the effectiveness of head binarization, which helps to improve the *s2t* system using parse trees in all translation tasks.

To test the effectiveness of our U-trees, we give the *s2t* translation system using the U-trees (from random 1). The results show that the system using U-trees achieves the best translation result from all of the systems. It surpasses the *s2t* system using parse trees by 1.47 BLEU points on MT04 and 1.44 BLEU points on MT05. Moreover, even using the binary parse trees, the achieved *s2t* system is still lower than our U-tree-based *s2t* system by 0.97 BLEU points on the combined test set. From the translation results, we can validate our former analysis that the U-trees generated by our Bayesian tree induction model are more appropriate for string-to-tree translation than parse trees.

System	MT04	MT05	All
Joshua (HPB)	31.73	28.82	30.64
Joshua (SAMT)	32.48	29.77	31.56
***s2t** (parse-tree)*	33.73*	30.25*	32.75*
***s2t** (binary-parse-tree)*	34.09*	30.99*#	32.92*
***s2t** (U-tree)*	**35.20***#	**31.69***#	**33.89***#

Table 1. Results (in case-insensitive BLEU-4 scores) of *s2t* systems using different types of trees. The "*" and "#" denote that the results are significantly better than the Joshua (SAMT) system and the *s2t* system using parse trees (p<0.01).

6.5 Large Data

We also conduct an experiment on a larger bilingual training data from the LDC corpus[13]. The training corpus contains 2.1M sentence pairs with approximately 27.7M Chinese words and 31.9M English words. Similarly, we train a 5-gram language model using the Xinhua portion of the English Gigaword corpus and the English part of the training corpus. With the same settings as before, we run the Gibbs sampler for 1000 iterations and utilize the final U-tree structure to build a string-to-tree translation system.

The final BLEU score results are shown in Table 2. In the scenario with a large data, the string-to-tree system using our U-trees still significantly outperforms the system using parse trees.

System	MT04	MT05	All
Joshua (HPB)	34.55	33.11	34.01
Joshua (SAMT)	34.76	33.72	34.37
***s2t** (parse-tree)*	36.40*	34.53*	35.70*
***s2t** (binary-parse-tree)*	37.38*#	35.14*#	36.54*#
***s2t** (U-tree)*	**38.02***#	**36.12***#	**37.34***#

Table 2. Results (in case-insensitive BLEU-4 scores) for the large training data. The meaning of "*" and "#" are similar to Table 1.

7 Conclusion and Future Work

In this paper, we explored a new direction to build a tree-based model based on unsupervised Bayesian trees rather than supervised parse trees. To achieve this purpose, we have made two major efforts in this paper:

(1) We have proposed a novel generative Bayesian model to induce effective U-trees for tree-based translation. We utilized STSG in the model to grasp bilingual mapping information. We further imposed a reasonable hierarchical prior on the tree structures, encouraging small and frequent minimal rules for translation.

(2) To train the Bayesian tree induction model efficiently, we developed a Gibbs sampler with three novel Gibbs operators. The operators are designed specifically to explore the infinite space of tree structures by performing local changes on the tree structure.

[13] LDC category number : LDC2000T50, LDC2002E18, LDC2003E07, LDC2004T07, LDC2005T06, LDC2002L27, LDC2005T10 and LDC2005T34.

Experiments on the string-to-tree translation model demonstrated that our U-trees are better than the parse trees. The translation results verify that the well-designed unsupervised trees are actually more appropriate for tree-based translation than parse trees. Therefore, we believe that the unsupervised tree structure would be a promising research direction for tree-based translation.

In future, we plan to testify our sampler with various initial trees, such as the tree structure formed by (Zhang et al., 2008). We also plan to perform a detailed empirical comparison between STST and SCFG under our settings. Moreover, we will further conduct experiments to compare our methods with other relevant works, such as (Cohn and Blunsom, 2009) and (Burkett and Klein, 2012).

Acknowledgments

We would like to thank Philipp Koehn and three anonymous reviewers for their valuable comments and suggestions. The research work has been funded by the Hi-Tech Research and Development Program ("863" Program) of China under Grant No. 2011AA01A207, 2012AA011101, and 2012AA011102.

References

Phil Blunsom, Trevor Cohn, Miles Osborne. 2008. Bayesian synchronous grammar induction. In *Advances in Neural Information Processing Systems*, volume 21, pages 161-168.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A gibbs sampler for phrasal synchronous grammar induction. In *Proc. of ACL 2009*, pages 782-790.

Phil Blunsom and Trevor Cohn. 2010. Inducing synchronous grammars with slice sampling. In *Proc. of NAACL 2010*, pages 238-241.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic Parsing). In *Proc. of EMNLP 2008*, pages 877-886.

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In Proc. of *NAACL 2010*, pages 127-135.

David Burkett and Dan Klein. 2012. Transforming trees to improve syntactic convergence. In *Proc. of EMNLP 2012*, pages 863-872.

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

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. of ACL 1996*, pages 152-158.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377-404.

Trevor Cohn and Phil Blunsom. 2009. A bayesian model of syntax-directed tree to string grammar induction. In *Proc. of EMNLP 2009*, pages 352-361.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *Journal of Machine Learning Research*, pages 3053-3096.

Brooke Cowan, Ivona Kucerova and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proc. of EMNLP 2006*, pages 232-241.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *Proc. of ACL 2007*, pages 17-24.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proc. of EMNLP 2011*, pages 193-203.

Chris Dyer. 2010. Two monolingual parses are better than one (synchronous parse). In *Proc. of NAACL 2010*, pages 263-266.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. of ACL 2003*, pages 205-208.

Michel Galley, Mark Hopkins, Kevin Knight and Daniel Marcu. 2004. What's in a translation rule. In *Proc. of HLT-NAACL 2004*, pages 273–280.

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 *Proc. of ACL-COLING 2006*, pages 961-968.

Jonathan Weese, Juri Ganitkevitch, Chris Callison-Burch, Matt Post and Adam Lopez. 2011. Joshua 3.0: syntax-based machine translation with the thrax Grammar Extractor. In *Proc of WMT11*, pages 478-484.

Liang Huang, Kevin Knight and Aravind Joshi. 2006. A syntax-directed translator with extended domain of locality. In *Proc. of AMTA 2006*, pages 65-73.

Philipp Koehn, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation, In *Proc. of HLT/NAACL 2003*, pages 48-54.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer and Ondřej Bojar. 2007. Moses: open source toolkit for statistical machine translation. In *Proc. of ACL 2007*, pages 177-180.

Abby Levenberg, Chris Dyer and Phil Blunsom. 2012. A bayesian model for learning SCFGs with discontiguous Rules. In *Proc. of EMNLP 2012*, pages 223-232.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N.G. Thornton, Jonathan Weese and Omar F. Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proc. of ACL 2009*, pages 135-139.

Shujie Liu, Chi-Ho Li, Mu Li, Ming Zhou. 2012. Re-training monolingual parser bilingually for syntactic SMT. In *Proc. of EMNLP 2012*, pages 854-862.

Yang Liu, Qun Liu and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL-COLING 2006*, pages 609-616.

Yang Liu, Yajuan Lv and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proc. of ACL-IJCNLP 2009*, pages 558-566.

Daniel Marcu, Wei Wang, Abdessamad Echihabi and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. of EMNLP 2006*, pages 44-52.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proc. of ACL 2002*, pages 311-318.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL 2006*, pages 433-440.

Chris Quirk, Arul Menezes and Colin Cherry. 2005. Dependency treelet translation: syntactically informed phrasal SMT. In *Proc. of ACL 2005*, pages 271-279.

Libin Shen, Jinxi Xu and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL-08*, pages 577-585.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proc. of EMNLP 2007*, pages 746-754.

Wei Wang, Jonathan May, Kevin Knight, and Daniel Marcu. 2010. Re-structuring, re-labeling, and re-aligning for syntax-based machine translation. *Computational Linguistics*, 36(2):247–277.

Feifei Zhai, Jiajun Zhang, Yu Zhou and Chengqing Zong. 2012. Tree-based translation without using parse trees. In *Proc. of COLING 2012*, pages 3037-3054.

Hao Zhang, Liang Huang, Daniel Gildea and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proc. of HLT-NAACL 2006*, pages 256-263.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting synchronous grammars rules from word level alignments in linear time. In *Proc. of COLING 2008*, pages 1081-1088.

Hao Zhang, Licheng Fang, Peng Xu, Xiaoyun Wu. 2011a. Binarized forest to string translation. In *Proc. of ACL 2011*, pages 835-845.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw, Chew Lim Tan. 2009. Forest-based tree sequence to string translation model. In *Proc. of ACL-IJCNLP 2009*, pages 172-180.

Jiajun Zhang, Feifei Zhai and Chengqing Zong. 2011b. Augmenting string-to-tree translation models with fuzzy use of source-side syntax. In *Proc. of EMNLP 2011*, pages 204-215.

Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Chew Lim Tan and Sheng Li. 2007. A tree-to-tree alignment-based model for statistical Machine translation. *MT-Summit-07*. pages 535-542

Min Zhang, Hongfei Jiang, Ai ti 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*, pages 559-567.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. of Workshop on Statistical Machine Translation 2006*, pages 138-141.

Andreas Zollmann and Stephan Vogel. 2011. A word-class approach to labeling PSCFG rules for machine translation. In *Proc. of ACL 2011*, pages 1-11.