Journal of Artificial Intelligence Research 48 (2013) 733-782 Submitted 04/13; published 11/13

Unsupervised Sub-tree Alignment for Tree-to-Tree Translation

Tong Xiao XIAOTONG@MAIL.NEU.EDU.CN
Jingbo Zhu ZHUJINGBO@MAIL.NEU.EDU.CN
College of Information Science and Engineering
Northeastern University
No 3-11, Wenhua Road, Heping District
Shenyang, China

Abstract

This article presents a probabilistic sub-tree alignment model and its application to tree-to-tree machine translation. Unlike previous work, we do not resort to surface heuristics or expensive annotated data, but instead derive an unsupervised model to infer the syntactic correspondence between two languages. More importantly, the developed model is syntactically-motivated and does not rely on word alignments. As a by-product, our model outputs a sub-tree alignment matrix encoding a large number of diverse alignments between syntactic structures, from which machine translation systems can efficiently extract translation rules that are often filtered out due to the errors in 1-best alignment. Experimental results show that the proposed approach outperforms three state-of-the-art baseline approaches in both alignment accuracy and grammar quality. When applied to machine translation, our approach yields a +1.0 BLEU improvement and a -0.9 TER reduction on the NIST machine translation evaluation corpora. With tree binarization and fuzzy decoding, it even outperforms a state-of-the-art hierarchical phrase-based system.

1. Introduction

Recent years have witnessed increasing interest in syntax-based methods for many Artificial Intelligence (AI) and Natural Language Processing (NLP) applications ranging from text summarization to Machine Translation (MT). In particular, syntax-based models have been intensively investigated in Statistical Machine Translation (SMT). Approaches include string-to-tree MT (Galley, Hopkins, Knight, & Marcu, 2004; Galley, Graehl, Knight, Marcu, DeNeefe, Wang, & Thayer, 2006), tree-to-string MT (Liu, Liu, & Lin, 2006; Huang, Kevin, & Joshi, 2006) and tree-to-tree MT (Eisner, 2003; Zhang, Jiang, Aw, Li, Tan, & Li, 2008; Liu, Lü, & Liu, 2009a; Chiang, 2010), all of which train on tree-string/tree-tree pairs and seek to model the translation equivalency relations learned from parsed data. As a part of the focus on syntax-based MT, tree-to-tree models that use synchronous context free grammars or synchronous tree substitution grammars have received growing interest, showing very promising results on several well-established evaluation tasks (Zhang et al., 2008; Liu et al., 2009a; Chiang, 2010). For example, recent studies (Chiang, 2010) have demonstrated that modern tree-to-tree systems can significantly outperform the hierarchical phrase-based counterpart in large scale Chinese-English and Arabic-English translation.

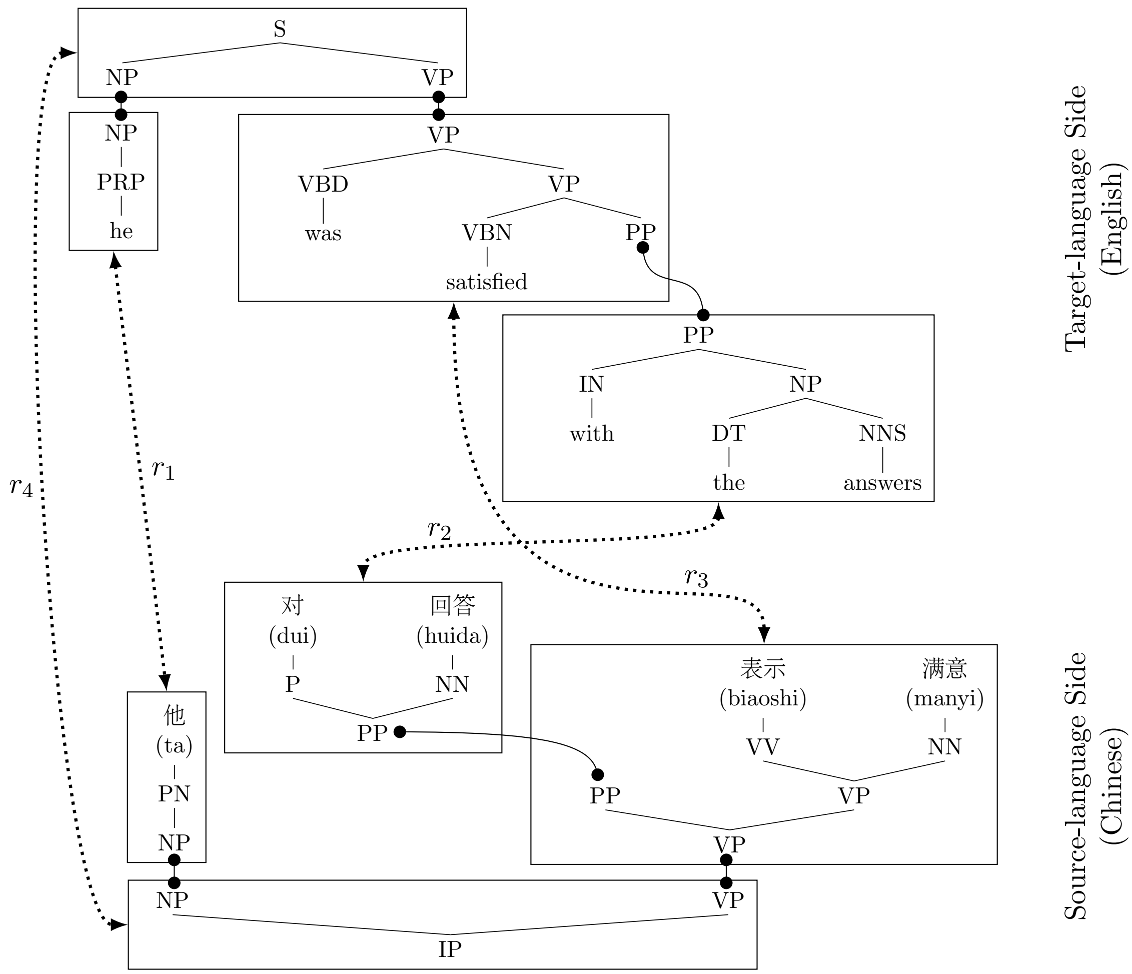
In tree-to-tree MT, the translation problem can be broadly regarded as transformation from a source-language syntax tree to a target-language syntax tree. To model this process, most tree-to-tree systems resort to the general framework of synchronous grammars, where a pair of trees is generated with derivations of synchronous grammar rules (or *translation rules*). In such a model, the goal of translation is to build the underlying derivations for all pairs of trees and output the target string encoded in the most likely derivation. Figure 1 shows an intuitive example to illustrate the generation process of a tree pair using a sample grammar, where both the source and target-language sentences are associated with the phrase structure trees generated using automatic parsers.[1]

Previous work has shown that the acquisition of good translation rules is one of the essential factors contributing to the success of syntax-based systems (DeNeefe, Knight, Wang, & Marcu, 2007). To date, several research groups have addressed the issue of rule acquisition and designed effective algorithms to extract high-coverage grammars from bilingual parsed data (Zhang et al., 2008; Liu et al., 2009a; Chiang, 2010). Despite their differences in detailed modeling, all these approaches rely on *syntactic alignments* that align tree nodes in the syntactic parse tree in one language to tree nodes in the other, and these alignments could be employed by standard tree-to-tree rule extraction algorithms (Liu et al., 2009a; Chiang, 2010).

While current tree-to-tree models heavily depend on syntactic alignments between two languages, all these alignments are induced indirectly from word alignments and tree-to-tree systems are very sensitive to the word alignment behavior. Unfortunately, word alignments are in general far from perfect from the viewpoint of syntactic alignment (Fossum, Knight, & Abney, 2008). In some cases, even one spurious word alignment can prevent a large number of desirable rules from extraction. For example, Figure 2(a) shows some tree-to-tree translation rules extracted using the word alignment produced by GIZA++. This alignment incorrectly aligns the source word "了" (a past tense marker in Chinese) to the target word "the". This spurious word alignment produces an incorrect rule "AS(了) $\rightarrow$ DT(the)" and blocks the extraction of more high-level syntactic transfer rules, such as "IP(NN_1 VP_2) $\rightarrow$ S(NP_1 VP_2)".

Obviously, a more desirable solution is to directly infer node correspondences from the source and target parse trees, namely *sub-tree alignment*. As syntactic parse trees can explain the underlying structure of sentences well, performing alignment in sub-tree level can make more benefits from the high-level structural information and syntactic categorization. For example, consider the alignment in Figure 2(b). It links up the nodes in the two parse trees (in Chinese and English), rather than aligning them in word level. In this example, it is very confident to align the VP sub-tree (spanning "大幅度 减少 了") in the source tree to the VP sub-tree (spanning "have drastically fallen") in the target tree.[2] We therefore

1. In a phrase structure tree, the leaf nodes are words of the sentence. The internal tree nodes followed by leaf nodes are labeled with Part-Of-Speech (POS) tags, while other tree nodes are labeled with syntactic categories defined in treebanks (see appendix for meanings of the POS tags and syntactic categories used in this work). In NLP, many well-developed parsers are available for automatic parsing. Also, several good-quality phrase structure treebanks across languages can be used to train parsing models, such as the Penn English and Chinese Treebanks (Marcus, Santorini, & Marcinkiewicz, 1993; Xue, Xia, Chiou, & Palmer, 2005). Note that in addition to phrase structure syntax, there are other popular formalisms (e.g., dependency syntax) can be used in syntax-based MT. But the discussion on different formalisms of syntactic parsing is beyond the scope of this article. We instead focus on tree-to-tree MT based on phrase structure trees throughout this work.
2. Both Chinese and English follow the subject-verb-object structure. The verb phrases in a Chinese sentence are frequently aligned to the verb phrases in its English translation.

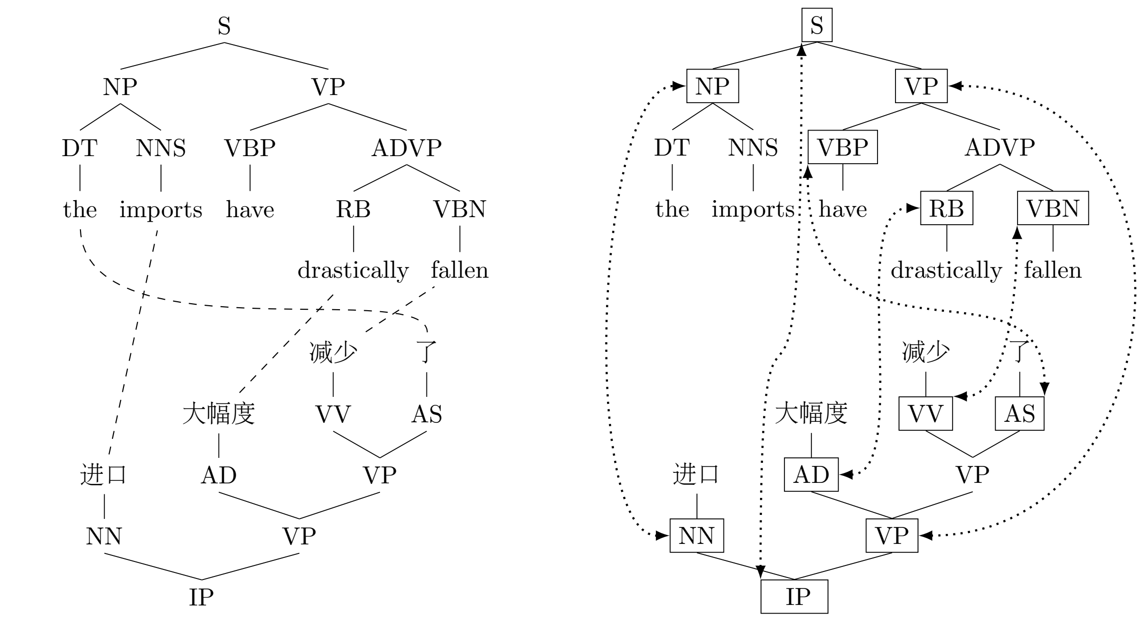


Synchronous Grammar Used

ID	Source-language Side	Target-language Side
r_1	NP(PN(他))	NP(PRP(he))
r_2	PP(P(对) NN(回答))	PP(IN(with) NP(DT(the) NNS(answers)))
r_3	VP(PP$_1$ VP(VV(表示) NN(满意)))	VP(VBD(was) VP(VBN(satisfied) PP$_1$))
r_4	IP(NP$_1$ VP$_2$)	S(NP$_1$ VP$_2$)

Figure 1: Example derivation of tree-to-tree translation rules. All the rules are represented as aligned pairs of tree-fragments (linked with dotted lines). The subscripts on both language sides of the grammar rules indicate the alignments of frontier non-terminals. On each language side of the derivation, the round-head lines link up the frontier non-terminals that are rewritten during translation.

know that the child nodes in the source-language VP are likely to be aligned with the child nodes in the target-language VP. This means that once the two VPs are aligned, their children should not be aligned outside the VP sub-tree structure, i.e., we can prevent the alignment between the Chinese tree node "AS" and the English tree node "DT" due to its inconsistency with the VP-VP alignment. In this case, "AS" is correctly aligned to "VBP".



(Minimal) Rules Extracted		**(Minimal) Rules Extracted**	
r_1	AS(了) → DT(the)	r_3	AD(大幅度) → RB(drastically)
r_2	NN(进口) → NNS(imports)	r_4	VV(减少) → VBN(fallen)
r_3	AD(大幅度) → RB(drastically)	r_6	AS(了) → VBP(have)
r_4	VV(减少) → VBN(fallen)	r_7	NN(进口) → NP(DT(the) NNS(imports))
r_5	IP(NN$_1$ VP(AD$_2$ VP(VV$_3$ AS$_4$))) → S(NP(DT$_4$ NNS$_1$) VP(VBP(have) ADVP(RB$_2$ VBN$_3$)))	r_8	VP(AD$_1$ VP(VV$_2$ AS$_3$)) → VP(VBP$_3$ ADVP(RB$_1$ VBN$_2$))
		r_9	IP(NN$_1$ VP$_2$) → S(NP$_1$ VP$_2$)

(a) word alignment and extracted rules (b) sub-tree alignment and extracted rules

Figure 2: Tree-to-tree translation rules extracted via word alignment (a) or sub-tree alignment (b). The dashed lines represent word alignment links, and the dotted lines represent sub-tree alignment (or node alignment) links.

As a result, the bad rule "AS(了) → DT(the)" is ruled out, and a few more desirable rules are extracted using the sub-tree alignment (including the desirable rules that are blocked in Figure 2(a)).

Actually, researchers have been aware of the sub-tree alignment problem and tried to explore solutions (Tinsley, Zhechev, Hearne, & Way, 2007; Sun, Zhang, & Tan, 2010b, 2010a). For example, they proposed to judge whether two nodes should be aligned or not. In their work, the alignment confidence is first calculated using lexical translation probabilities or classifiers trained on labeled data, and then the final alignment is determined according to node-level alignment score. However, the inference of sub-tree alignment in these approaches relies on heuristic algorithms, and their models are essentially not optimized within a unified probabilistic framework.

Moreover, when the alignment result is applied to tree-to-tree translation, most systems suffer from another problem that translation rules are extracted using the 1-best alignment only (Zhang et al., 2008; Liu et al., 2009a; Chiang, 2010). This problem significantly affects the rule-set coverage rate due to alignment errors. A simple solution to this issue is to use k-best alignments instead. However, k-best alignments often have few variations and many redundancies. Most of them differ in only a few alignment links. It is obviously inefficient to extract rules from those similar alignments.

In this article we address the sub-tree alignment issue in a principled way and investigate methods to effectively apply the sub-tree alignment result to tree-to-tree MT. In particular,

- We develop an unsupervised approach to learning a probabilistic sub-tree alignment model from bi-lingual parsed data.
- We investigate different methods for integrating sub-tree alignment to tree-to-tree machine translation. Specifically, we develop a *sub-tree alignment matrix* encoding an exponentially large number of diverse sub-tree alignments, and extract multiple alternative translation rules using alignment posteriors from the sub-tree alignment matrix.

The advantages of our approach are three-fold. First, our approach does not rely on heuristic algorithms or labeled data. Second, the developed sub-tree alignment model has the same structure as the model used in MT, i.e., both are based on synchronous tree substitution grammars. It means that MT systems can directly make benefits from the sub-tree alignment model, especially for rule extraction and MT parameter estimation. Third, by accessing the sub-tree alignment matrix which encodes a large number of alignments, we can efficiently obtain rules that are often filtered out due to the errors within the 1-best/k-best alignment result. We experiment with our approach in Chinese-English sub-tree alignment and translation tasks. For sub-tree alignment, it significantly outperforms three state-of-the-art baselines. For machine translation, our approach obtains significant improvements for a tree-to-tree system in both rule quality and translation quality. For example, it yields a +1.0 BLEU improvement and a -0.9 TER reduction on the NIST MT evaluation corpora. Finally, our system even outperforms a state-of-the-art hierarchical phrase-based system when equipped with tree binarization (Wang, Knight, & Marcu, 2007b) and fuzzy decoding (Chiang, 2010) techniques.

The rest of the article is structured as follows. Section 2 briefly introduces the sub-tree alignment task. Section 3 describes our unsupervised approach to sub-tree alignment. Then, Section 4 investigates effective methods for applying our alignment model to tree-to-tree translation. Then, Section 5 presents experimental evaluation of our approach. After reviewing the related work in Section 6, some interesting issues are discussed in Section 7. Finally, the article is concluded with a summary in Section 8.

2. Problem Statement

In general, sub-tree alignment can be defined as a task that we find an alignment from the nodes in a tree to the nodes of another tree.[3] While we restrict ourselves to machine translation in this article, sub-tree alignment is actually not a task that must be tightly coupled with specific applications. For example, in addition to machine translation, there are other

3. In this work term *tree* refers to a data structure that can be defined recursively as a collection of nodes starting at a root node. Each node has a list of edges pointing to nodes (or its children), with the constraint that no edge is duplicated or points to the root (Knuth, 1997).

NLP tasks which can make benefits from sub-tree alignment, including sentence simplification (Cohn & Lapata, 2009; Woodsend & Lapata, 2011), paraphrasing (Das & Smith, 2009), question answering (Wang, Smith, & Mitamura, 2007a), and parser adaptation and projection (Smith & Eisner, 2009).

Ideally, we would like a sub-tree alignment system that is language independent and application independent. Given a parallel corpus of training examples, we should be able to learn an alignment model and use it to infer the syntactic correspondence for any tree pairs. Broadly speaking, any alignments between paired linguistic tree structures can be regarded as instances of sub-tree alignment. For example, the alignment can be performed between dependency trees (Eisner, 2003; Nakazawa & Kurohashi, 2011) or phrase structure trees (Tinsley et al., 2007; Sun et al., 2010b).

Although the sub-tree alignment problem includes a number of tasks that seek alignments between syntactic tree structures, we are particularly interested in aligning the tree nodes of phrase structure trees in this work. We focus on phrase structure sub-tree alignment because: 1) phrase structure parsing is one of the most popular syntactic analysis formalisms. Several state-of-the-art full parsing models/tools have been developed for many languages; 2) phrase structure trees are the basis of many successful syntax-based MT systems. While other alternatives, such as dependency trees, can also benefit MT systems, the constituency-based models are of interest to a relatively larger portion of the MT community and show state-of-the-art performance in recent tree-to-tree systems (Zhang et al., 2008; Liu et al., 2009a; Chiang, 2010).

In natural language processing, a phrase structure parse tree is an ordered and rooted tree. It represents the syntactic structure of a sentence according to some phrase structure grammars (or constituency grammars) which describe the way words combine to form phrases and sentences (Chiswell & Hodges, 2007). Generally, phrase structure parse trees distinguish between *terminal* and *non-terminal* nodes. The leaf nodes are labeled by terminal categories (or words), while the internal nodes are labeled by non-terminal categories of the grammar (or phrasal categories). For example, in the English parse tree in Figure 2(b), "imports" is a terminal, while nodes "NP" and "NNS" are two non-terminals indicating the noun phrase and the plural form of nouns respectively.[4] In the following description and experiments, we take the Penn Treebank for the standard of tree annotation. Here we choose the Penn Treebank because it is one of the most popular tree-annotated corpora used in syntactic parsing and of good quality and quantity for several languages, such as Chinese and English.

Based on the above definition, a sub-tree alignment can be defined as the alignments between non-terminals in the source and target-language (phrase structure) parse trees.[5] More formally, given a source-language parse tree S and a target-language parse tree T, a sub-tree alignment (denoted as $A(S,T)$ or A for short) is a set of node-to-node links between S and T. For any node pair (u,v) in (S,T), a good alignment should follow three criteria (Tinsley et al., 2007):

1. u (or v) can only be aligned once (indicating 1-to-1 alignment).

4. Note that the non-terminals that are always followed by leaf nodes are also called *pre-terminals* and labeled by part-of-speech tags. E.g., the "NNS" node is followed by the terminal node "imports" and thus is a pre-terminal.
5. In contrast, a word alignment can be regarded as the alignments between terminals in two languages.

2. if u is aligned to v, the descendants of u can only be aligned to the descendants of v.
3. if u is aligned to v, the ancestors of u can only be aligned to the ancestors of v.

Such criteria prevent from aligning constituents that cross each other. This property is very similar to that of some bi-parsing formalisms, such as synchronous context free grammars and synchronous tree substitution grammars. Its advantage is that it enables the use of powerful synchronous grammars in modeling the sub-tree alignment problem. As is shown in the very next section, based on the above constraints we can take synchronous tree substitution grammars as the basis of our proposed model.

According to Tinsley et al.'s (2007) work, the alignments satisfying the above criteria are called *well-formed* alignments. An alignment is *ill-formed* when it violates any of these criteria. In this work we focus on the well-formed alignments. Hence the sub-tree alignment task can be stated as: given a pair of parse trees (S, T), we search for the most likely well-formed alignment between S and T

$$\hat{A} = \underset{A \in \Omega(S,T)}{\arg\max} \, \mathrm{P}(A \mid S, T) \tag{1}$$

where $\Omega(S, T)$ is the set of well-formed alignments, and $\mathrm{P}(A \mid S, T)$ can be viewed as an alignment model which predicts the probability for every alignment A given S and T.

In what follows, we describe our approach to sub-tree alignment for tree-to-tree translation, including the alignment model, the training and inference methods, and the effective use of our model in tree-to-tree MT systems.

3. Unsupervised Sub-tree Alignment

In this section we present our unsupervised sub-tree alignment model. We first define the base model of sub-tree alignment in the framework of synchronous tree substitution grammars, and then describe the model parameterization, training and inference methods.

3.1 Base Model

The fundamental question of sub-tree alignment is how to define the correspondence between nodes of the source-language parse tree and nodes of the target-language parse tree. Here we address the issue using Synchronous Tree Substitution Grammars (STSGs) which have been widely adopted to model the transformation process between source and target-language parse trees in MT (Zhang et al., 2008; Liu et al., 2009a; Chiang, 2010). In the general framework of STSGs (Chiang & Knight, 2006), it is assumed that a pair of source and target parse trees can be simultaneously generated using a derivation of STSG rules (or tree-to-tree transfer rules). For example, the grammar in Figure 1 is an STSG and the rules in it can be used to generate a pair of sentences. More formally, an STSG is a system $\langle N_s, N_t, W_s, W_t, \Psi \rangle$, where N_s and N_t are sets of non-terminals in the source and target languages, W_s and W_t are sets of terminals (or words) in the source and target languages, Ψ is a finite set of productions. Each production is an STSG rewrite rule (denoted as r) for a pair of source and target-language non-terminals (s_{nt}, t_{nt}):

$$\langle s_{nt}, t_{nt} \rangle \rightarrow \langle s_r, t_r, \phi_r \rangle$$

where s_r is a source-language tree-fragment, whose frontier nodes are either words in W_s or non-terminals in N_s (labeled by x); t_r is the corresponding target-language tree-fragment; and ϕ_r is a set of 1-to-1 alignments that connect the frontier non-terminals of s_r to the frontier non-terminals of t_r. For example, for r_5 in Figure 2(a), we have

$$\begin{aligned}
s_{nt} &= \text{IP} \\
t_{nt} &= \text{S} \\
s_r &= \text{IP(NN:}x \text{ VP(AD:}x \text{ VP(VV:}x \text{ AS:}x\text{)))} \\
t_r &= \text{S(NP(DT:}x \text{ NNS:}x\text{) VP(VBP(have) ADVP(RB:}x \text{ VBN:}x\text{)))} \\
\phi_r &= \{\text{1-2, 2-3, 3-4, 4-1}\}
\end{aligned}$$

Note that the non-terminals on the left-hand side of the rule are actually the roots of the corresponding tree-fragments on the right hand side. This means that the rule contains exactly the same information no matter whether the root nodes (s_{nt}, t_{nt}) are explicitly represented or not. So in the following parts of this article we use $\langle s_r, t_r, \phi_r \rangle$ for a simpler representation of STSG rules. Beyond this, STSG rules can be written in a more compact form where the alignment ϕ_r is encoded in the numbers assigned to the frontier non-terminals of s_r and t_r. For example, in Figures 1 and 2, the subscripts on both language sides of the STSG rules indicate the aligned pairs of frontier non-terminals.

In the STSG model, frontier non-terminals are also called substitution nodes. When applying STSGs, we can rewrite an aligned pair of substitution nodes with the tree-fragment pair encoded in an STSG rule. The only constraint in this operation is that the labels of the substituted non-terminals must match the root labels of the rewrite rules. For example, the round-head lines in Figure 1 show the substitution operations used in a derivation.

By using STSG rules, we can parse any tree pair and generate the corresponding derivations. The generation process is trivial: we start with the pair of root symbols and repeatedly rewrite pairs of non-terminal symbols using STSG rules. For example, for the tree pair in Figure 2(b), we start with the root labels of the source and target-language parse trees (the superscript indicates the node index in the tree)

$$\langle \text{ IP}^{[1]}, \text{ S}^{[1]} \rangle$$

Then we apply rule r_9.

$$\xrightarrow[r_9]{\text{IP}^{[1]} \Leftrightarrow \text{S}^{[1]}} \langle \text{ IP(NN}^{[2]} \text{ VP}^{[3]}), \text{ S(NP}^{[2]} \text{ VP}^{[3]}) \rangle$$

where $\xrightarrow[r_9]{\text{IP}^{[1]} \Leftrightarrow \text{S}^{[1]}}$ represents the operation that rewrites the aligned node pair $\text{IP}^{[1]}$ and $\text{S}^{[1]}$ with r_9 (denoted as $\text{IP}^{[1]} \Leftrightarrow \text{S}^{[1]}$). This process proceeds by repeatedly rewriting the remaining frontier non-terminals until we get the complete source and target-language trees, like so:

$$\xrightarrow[r_7]{\mathrm{NN}^{[2]} \Leftrightarrow \mathrm{NP}^{[2]}} \langle\ \text{IP(NN(进口) VP}^{[3]}\text{)},\ \text{S(NP(DT(the) NNS(imports)) VP}^{[3]}\text{)}\ \rangle$$

$$\xrightarrow[r_8]{\mathrm{VP}^{[3]} \Leftrightarrow \mathrm{VP}^{[3]}} \langle\ \text{IP(NN(进口) VP(AD}^{[4]}\text{ VP(VV}^{[5]}\text{ AS}^{[6]}\text{))),}$$
$$\text{S(NP(DT(the) NNS(imports)) VP(VBP}^{[6]}\text{ ADVP(RB}^{[4]}\text{ VBN}^{[5]}\text{)))}\ \rangle$$

$$\xrightarrow[r_3]{\mathrm{AD}^{[4]} \Leftrightarrow \mathrm{RB}^{[4]}} \langle\ \text{IP(NN(进口) VP(AD(大幅度) VP(VV}^{[5]}\text{ AS}^{[6]}\text{))),}$$
$$\text{S(NP(DT(the) NNS(imports)) VP(VBP}^{[6]}\text{ ADVP(RB(drastically) VBN}^{[5]}\text{)))}\ \rangle$$

$$\xrightarrow[r_4]{\mathrm{VV}^{[5]} \Leftrightarrow \mathrm{VBN}^{[5]}} \langle\ \text{IP(NN(进口) VP(AD(大幅度) VP(VV(减少) AS}^{[6]}\text{))),}$$
$$\text{S(NP(DT(the) NNS(imports)) VP(VBP}^{[6]}$$
$$\text{ADVP(RB(drastically) VBN(fallen))))}\ \rangle$$

$$\xrightarrow[r_6]{\mathrm{AS}^{[6]} \Leftrightarrow \mathrm{VBP}^{[6]}} \langle\ \text{IP(NN(进口) VP(AD(大幅度) VP(VV(减少) AS(了)))),}$$
$$\text{S(NP(DT(the) NNS(imports)) VP(VBP(have)}$$
$$\text{ADVP(RB(drastically) VBN(fallen))))}\ \rangle$$

In the above process, each rewrite rule indicates a node alignment. More importantly, derivations from this model have two very nice properties: first, for each node u in the source-language (or target-language) parse tree, there is at most one node of the target-language (or source-language) parse tree which is aligned with u; second, the hierarchical structure behind the alignment avoids links between constituents that cross each other. Consequently, for any well-formed sub-tree alignment A, we can always find a derivation d that encodes the alignment A. It means that the sub-tree alignment problem is essentially the same as the problem of finding the most likely STSG derivation. Thus the sub-tree alignment task (see Equation (1)) can be restated as finding the most likely derivation for a given pair of parse trees.

To model the derivation probability, we follow the formulation adopted in statistical word alignment (Brown, Pietra, Pietra, & Mercer, 1993; Vogel, Ney, & Tillmann, 1996). The transformation from a source-language tree S to a target-language tree T is described by the following equation.

$$\mathrm{P}(T \mid S) = \sum_{d \in D(S,T)} \mathrm{P}_{\theta}(T, d|S) \tag{2}$$

where $D(S,T)$ is the set of all derivations transforming S into T (say, aligning the nodes of S to the nodes of T), $\mathrm{P}_{\theta}(T,d|S)$ is the probability of transforming from S to T using a derivation $d \in D(S,T)$, and θ is the parameters of the model. Here we use the notation $\mathrm{P}_{\theta}(\cdot)$ to express the dependence of the model on the parameters. In general, the optimal value of θ is learned from parsed parallel data by some training criteria. For example, in the context of unsupervised learning, we can optimize the model parameters by maximizing the probability of the observed data (known as maximum likelihood training).

Given a set of optimal parameters $\hat{\theta}$, the best sub-tree alignment for (S,T) is determined by choosing a derivation for which $\mathrm{P}_{\hat{\theta}}(d \mid S,T)$ is greatest. Since $\mathrm{P}_{\hat{\theta}}(d \mid S,T) = \frac{\mathrm{P}_{\hat{\theta}}(T,d|S)}{\mathrm{P}_{\hat{\theta}}(T|S)}$

and $\mathrm{P}_{\hat{\theta}}(T \mid S)$ is a constant for given (S, T) and $\hat{\theta}$, finding the best derivation $\hat{d}$ is the same as finding a derivation so as to make $\mathrm{P}_{\hat{\theta}}(T, d \mid S)$ as large as possible. Hence we reach the fundamental equation of sub-tree alignment.

$$\hat{d} = \arg\max_{d \in D(S,T)} \mathrm{P}_{\hat{\theta}}(T, d \mid S) \tag{3}$$

The above formulation implies three fundamental issues of sub-tree alignment, including modeling of derivation probability (i.e., $\mathrm{P}_{\theta}(T, d|S)$), learning of model parameters (i.e., $\hat{\theta}$) and finding the best alignment given the learned model (i.e., the $\arg\max$ operation). In the following parts of this section, we describe our solutions to these issues.

3.2 Parameterization

In the simplest case, our alignment model has one parameter for each instance of derivation. However, this model would have an unmanageable set of parameters since the number of derivations is exponential in the length of the input sentences. Here we choose a simple solution to this issue which decomposes the base model into a product of trainable sub-models. We start with an assumption that rules are conditionally independent for the given source-language parse tree S, then the probability $\mathrm{P}(T, d \mid S)$ can be defined as a product of rule probabilities (for conciseness, we will drop the subscript θ from now on).

$$\mathrm{P}(T, d \mid S) \equiv \prod_{r \in d} \mathrm{P}(r \mid S) \tag{4}$$

Nevertheless complex tree-to-tree mappings still result in an extremely large number of rules, which causes both the computational problem and the degenerate analysis of the data.[6] To control the number of parameters at a reasonable level, we further decompose the rule probability into simpler probability factors under independence assumptions.

First we assume that the generation of a rule r is independent of the input tree S, when conditioned on the source-language side of the rule, that is,

$$\mathrm{P}(r \mid S) \equiv \mathrm{P}(r \mid s_r) \tag{5}$$

Note that this is a strong assumption that the generation of a synchronous grammar rule depends only on its source-language side. It is similar to those used in statistical modeling of machine translation (Brown et al., 1993; Koehn, Och, & Marcu, 2003; Galley et al., 2004; Chiang, 2005) where the generation of atomic alignment/translation units are conditioned on the associated source-language words or tree-fragments, rather than the whole input sentence or tree. In SMT, the independence assumptions based on phrases or translation rules are generally used to decompose the parallel corpus into manageable units for parameter estimation. As they have been successfully used in most of modern SMT systems, we adopt a similar assumption here to ease the parameter estimation process of our model.

Then we further decompose $\mathrm{P}(r \mid s_r)$ with additional assumptions. Since $r = \langle s_r, t_r, \phi_r \rangle$, $\mathrm{P}(r \mid s_r)$ can be written into another form using the chain rule:

6. Here *degenerate analysis* refers to the case where using models that are too complex results in overfitting and a poor generalization ability on unseen data.

$$\begin{aligned} \mathrm{P}(r \mid s_r) &= \mathrm{P}(s_r, t_r, \phi_r \mid s_r) \\ &= \mathrm{P}(\phi_r \mid s_r, t_r) \times \mathrm{P}(t_r \mid s_r) \end{aligned} \tag{6}$$

Equation (6) indicates two sub-models, including a *reordering model* of frontier non-terminals $\mathrm{P}(\phi_r \mid s_r, t_r)$, and a *tree-fragment translation model* $\mathrm{P}(t_r \mid s_r)$.

To model $\mathrm{P}(\phi_r \mid s_r, t_r)$, we view frontier non-terminal reordering as a problem of aligning the elements between two vectors of non-terminals. Let $vnt(\cdot)$ be a function that returns the vector of leaf non-terminals for a given tree-fragment. ϕ_r defines a 1-to-1 alignment between the non-terminals in $vnt(s_r)$ and $vnt(t_r)$. For example, for r_5 in Figure 2(a), the frontier non-terminal vectors of s_{r_5} and t_{r_5} are:

$$\begin{aligned} vnt(s_{r_5}) &= (\text{NN, AD, VV, AS}) \\ vnt(t_{r_5}) &= (\text{DT, NNS, RB, VBN}) \end{aligned}$$

Then $\phi_{r_5} = \{$1-2, 2-3, 3-4, 4-1$\}$ indicates an alignment between $vnt(s_{r_5})$ and $vnt(t_{r_5})$, say, NN is aligned to NNS, AD is aligned to RB and so on. Here we opt for a simple model for selecting ϕ_r. It models the non-terminal reordering probability on the condition of the frontier non-terminal vectors on both language sides, as follows[7]

$$\mathrm{P}(\phi_r \mid t_r, s_r) \equiv \mathrm{P}_{reorder}(\phi_r \mid vnt(s_r), vnt(t_r)) \tag{7}$$

We then turn to the problem of modeling the tree-fragment translation $\mathrm{P}(t_r \mid s_r)$ (i.e., the second sub-model defined in Equation (6)). We define that a tree-fragment α consists of two parts: words $lex(\alpha)$ (i.e., terminals of α) and a tree structure $tree(\alpha)$ without lexicons involved. For example, for r_5 in Figure 2(a), the target-language tree-fragment contains two elements $lex(t_{r_5})$ and $tree(t_{r_5})$:

$$\begin{aligned} lex(t_{r_5}) &= \text{have} \\ tree(t_{r_5}) &= \text{S(NP(DT:}x \text{ NNS:}x\text{) VP(VBP ADVP(RB:}x \text{ VBN:}x\text{)))} \end{aligned}$$

Let $root(\cdot)$ be a function that returns the root for a given tree-fragment. We can write $\mathrm{P}(t_r \mid s_r)$ as:

$$\begin{aligned} \mathrm{P}(t_r \mid s_r) &= \mathrm{P}(lex(t_r), tree(t_r) \mid s_r) \\ &= \mathrm{P}(root(t_r) \mid s_r) \times \\ &\quad\ \mathrm{P}(tree(t_r) \mid root(t_r), s_r) \times \\ &\quad\ \mathrm{P}(lex(t_r) \mid tree(t_r), root(t_r), s_r) \end{aligned} \tag{8}$$

It is worth noting that Equation (8) is not an approximation. Here we just choose one of the many ways in which $\mathrm{P}(t_r \mid s_r)$ can be written as the product of a series of

7. The reordering model defined here ensures that arbitrary 1-to-1 alignments can be handled. But it might result in a very large model with sparse parameter distributions if big tree-fragments are involved. In considering this issue, we choose several pruning methods for better control of rule size in our sub-tree alignment system. See Section 5.2.2 for the pruning settings in this work.

conditional probabilities. We simply assert this equation that when generating a target-language tree-fragment for a source-language tree-fragment, first we can choose the root symbol of the target-language tree-fragment given the source-language tree-fragment (in probability of $\mathrm{P}(root(t_r) \mid s_r)$). Then we can choose the tree-structure of the target-language tree-fragment given its root symbol and the source-language tree-fragment (in probability of $\mathrm{P}(tree(t_r) \mid root(t_r), s_r)$). Then we can choose the target-language terminals associated with the tree-fragment given the target-language tree-structure, target-language root symbol and source-language tree-fragment (in probability of $\mathrm{P}(lex(t_r) \mid tree(t_r), root(t_r), s_r)$).

Another note is that Equation (8) actually does not reduce the model complexity. For example, $\mathrm{P}(lex(t_r) \mid tree(t_r), root(t_r), s_r)$ essentially indicates all the combinations of source and target-language tree-fragments. A simpler model is required for a feasible solution for parameter estimation. To do this, we introduce additional assumptions to relax the conditions of these probabilities and reduce the number of parameters to a reasonable level.

1. $\mathrm{P}(root(t_r) \mid s_r)$ depends only on $root(s_r)$, i.e.,

$$\mathrm{P}(root(t_r) \mid s_r) \equiv \mathrm{P}_{nt}(root(t_r) \mid root(s_r)) \tag{9}$$

   This assumption implies the node correspondence between the source and target-language parse trees.

2. $\mathrm{P}(tree(t_r) \mid root(t_r), s_r)$ depends only on $root(t_r)$, i.e.,

$$\mathrm{P}(tree(t_r) \mid root(t_r), s_r) \equiv \mathrm{P}_{tree}(tree(t_r) \mid root(t_r)) \tag{10}$$

   The second assumption results in a monolingual model of generating target-language tree-structures, where the generation of a tree-fragment is only conditioned on its root. It can be viewed as an analogy to the generative model used in standard TSGs.

3. $\mathrm{P}(lex(t_r) \mid tree(t_r), root(t_r), s_r)$ only depends on source words $lex(s_r)$, i.e.,

$$\mathrm{P}(lex(t_r) \mid tree(t_r), root(t_r), s_r) \equiv \mathrm{P}_{lex}(lex(t_r) \mid lex(s_r)) \tag{11}$$

   This allows us to directly model the terminal correspondence between the two languages.

Then, we substitute Equations (9)-(11) into Equation (8), and get

$$\begin{aligned}\mathrm{P}(t_r \mid s_r) \;\equiv\; & \mathrm{P}_{nt}(root(t_r) \mid root(s_r)) \times \\ & \mathrm{P}_{tree}(tree(t_r) \mid root(t_r)) \times \\ & \mathrm{P}_{lex}(lex(t_r) \mid lex(s_r))\end{aligned} \tag{12}$$

By using Equations (6), (7) and (12), Equation (4) can be finally written as:

id	rule	probability
r_3	AD(大幅度) $\rightarrow$ RB(drastically)	P_{nt}(RB | AD) $\cdot$ P_{lex}(drastically | 大幅度)
r_4	VV(减少) $\rightarrow$ VBN(fallen)	P_{nt}(VBN | VV) $\cdot$ P_{lex}(fallen | 减少)
r_6	AS(了) $\rightarrow$ VBP(have)	P_{nt}(VBP | AS) $\cdot$ P_{lex}(have | 了)
r_7	NN(进口) $\rightarrow$ NP(DT(the) NNS(imports))	P_{nt}(NP | NN) $\cdot$ P_{lex}(the imports | 进口)$\cdot$ P_{tree}("NP(DT NNS)" | NP)
r_8	VP(AD_1 VP(VV_2 AS_3)) $\rightarrow$ VP(VBP_3 ADVP(RB_1 VBN_2))	P_{nt}(VP | VP) $\cdot$ P_{tree}("VP(VBP ADVP(RB VBN))" | VP)$\cdot$ $\mathrm{P}_{reorder}$("1-2, 2-3, 3-1" | (AD, VV, AS), (VBP, RB, VBN))
r_9	IP(NN_1 VP_2) $\rightarrow$ S(NP_1 VP_2)	P_{nt}(S | IP) $\cdot$ P_{tree}("S(NP VP)" | S)$\cdot$ $\mathrm{P}_{reorder}$("1-1, 2-2" | (NN, VP), (NP, VP))

Table 1: Rule probabilities for the sample derivation $d = \{r_3, r_4, r_6, r_7, r_8, r_9\}$ in Figure 2(b)

$$\begin{aligned}\mathrm{P}(T, d \mid S) \equiv \prod_{r \in d} & \mathrm{P}_{nt}(root(t_r) \mid root(s_r)) \times \\ & \mathrm{P}_{tree}(tree(t_r) \mid root(t_r)) \times \\ & \mathrm{P}_{lex}(lex(t_r) \mid lex(s_r)) \times \\ & \mathrm{P}_{reorder}(\phi_r \mid vnt(s_r), vnt(t_r)) \end{aligned} \qquad (13)$$

This is a simplified model for the generative story described in this section. It takes the rule generation probability as a product of four probability factors: $\mathrm{P}_{nt}(\cdot)$ is the non-terminal mapping probability, which roughly captures the syntactic correspondence of sub-trees between the two languages; $\mathrm{P}_{tree}(\cdot)$ is the probability of generating the tree structure of T; $\mathrm{P}_{lex}(\cdot)$ is the probability of the terminal mappings between the two language sides of a rule; and $\mathrm{P}_{reorder}(\cdot)$ is the probability of the frontier non-terminal reordering encoded in a rule. See Table 1 for rule probabilities of a sample derivation.

In our model all parameters are assumed to be multinomial distributions. The calculation of $\mathrm{P}_{nt}(\cdot)$, $\mathrm{P}_{tree}(\cdot)$ and $\mathrm{P}_{reorder}(\cdot)$ is straightforward: they can be directly used without any further decompositions and assumptions. To calculate $\mathrm{P}_{lex}(\cdot)$, we choose the form adopted in the popular models of word alignment (Och & Ney, 2004; Thayer, Ettelaie, Knight, Marcu, Munteanu, Och, & Tipu, 2004), where the probability is defined as a product of word-based translation probabilities:

$$\mathrm{P}_{lex}(t_1 ... t_l \mid s_1 ... s_m) \equiv \mathrm{P}_{length}(l \mid m) \prod_{i=1}^{l} \frac{1}{m} \sum_{j=1}^{m} \mathrm{P}_w(t_i \mid s_j) \qquad (14)$$

where t_i is a target word, and s_j is a source word. $\mathrm{P}_{length}(\cdot)$ is used to control the number of target words produced from a given number of source words. $\mathrm{P}_w(\cdot)$ is the word translation probability. This sub-model is in principle doing something rather similar to conventional word-based translation tables, as in IBM Models (Brown et al., 1993).

3.3 Node Deletion and Insertion

Word (or sub-tree) deletion/insertion is common in real-world alignment and translation tasks. To add flexibility to modeling this problem, we allow for the production of empty sub-trees on either source or target-language side of a rule in our model. More formally, for a rule whose target-language side is an empty sub-tree, its probability is defined as:

$$
\begin{aligned}
\mathrm{P}(r \mid S) \equiv\ & \mathrm{P}_{nt}(root(\varepsilon) \mid root(s_r)) \times \\
& \mathrm{P}_{tree}(tree(\varepsilon) \mid root(\varepsilon)) \times \\
& \mathrm{P}_{lex}(lex(\varepsilon) \mid lex(s_r)) \times \\
& \mathrm{P}_{reorder}(\phi_\varepsilon \mid vnt(s_r), vnt(\varepsilon))
\end{aligned} \tag{15}
$$

where ε is a special symbol that indicates *nothing*. Factors $\mathrm{P}_{nt}(root(\varepsilon) \mid root(s_r))$ and $\mathrm{P}_{lex}(lex(\varepsilon) \mid lex(s_r))$ model the deletion probability at different levels of a tree-fragment. $\mathrm{P}_{tree}(tree(\varepsilon) \mid root(\varepsilon))$ is the probability of generating an empty tree-fragment. Factor $\mathrm{P}_{reorder}(\phi_\varepsilon \mid vnt(s_r), vnt(\varepsilon))$ regards ε as a special reordering pattern ϕ_ε which aligns all frontier non-terminals on the source side to a virtual node NULL. Obviously, the values of $\mathrm{P}_{tree}(tree(\varepsilon) \mid root(\varepsilon))$ and $\mathrm{P}_{reorder}(\phi_\varepsilon \mid vnt(s_r), vnt(\varepsilon))$ are both simply 1.

Similarly, for a rule whose source side is an empty sub-tree, its probability is defined as:

$$
\begin{aligned}
\mathrm{P}(r \mid S) \equiv\ & \mathrm{P}_{nt}(root(t_r) \mid root(\varepsilon)) \times \\
& \mathrm{P}_{tree}(tree(t_r) \mid root(t_r)) \times \\
& \mathrm{P}_{lex}(lex(t_r) \mid lex(\varepsilon)) \times \\
& \mathrm{P}_{reorder}(\phi_\varepsilon \mid vnt(\varepsilon), vnt(t_r))
\end{aligned} \tag{16}
$$

where the value of $\mathrm{P}_{reorder}(\phi_\varepsilon \mid vnt(\varepsilon), vnt(t_r))$ is also 1.

It is worthwhile to note that word deletion and insertion problems are very important in MT in spite of relatively less discussion in recent studies on tree-to-tree translation. What we are doing here is actually an analogy to the NULL-alignment used in IBM Models (Brown et al., 1993). For word/phrase-based models, removing some words from alignment can leave more space for correctly aligning other words in the sentence.[8] This is even more necessary for (1-to-1) sub-tree alignment because the alignment has to respect the syntactic constraints on both language sides, e.g., sub-tree alignments are not allowed to break the constraints imposed by neighbouring parts of the tree. In some cases, we cannot obtain a correct 1-to-1 alignment for the tree pair due to only one or two "bad" nodes which are not necessarily to be aligned with a valid node in the counterpart tree. Instead, some nodes can be "skipped" in alignment and thus do not impose "bad" constraints to other parts of the tree if node deletion/insertion is allowed. This is especially true when we align sentence pairs with very flat tree structures or free translations. In this work we found that node deletion and insertion operations were necessary to achieve satisfactory sub-tree alignment result. We therefore used them in our implementation by default.

3.4 Training

We now turn to the training problem. As discussed in Section 3.1, we focus on unsupervised learning of model parameters, that is, the optimal values of parameters are estimated given

8. Note that current phrase-based approaches (Koehn et al., 2003; Och & Ney, 2004) allow NULL-aligned words to appear at the boundary of a phrase, which can be viewed as a way of implicit modeling of the word insertion/deletion problem

a collection of tree pairs without any annotation of sub-tree level alignment. In this work we choose two approaches for estimating parameters of the sub-tree alignment model, including the Maximum Likelihood Estimation (MLE) approach and the Bayesian approach.

3.4.1 Maximum Likelihood Training

MLE is one of the most popular methods of parameter estimation for statistical models. Its basic idea is that, given a model and a set of parameters, the MLE method selects the values of parameters to generate a distribution that gives the highest probability to the observed data. MLE is a general approach to parameter estimation which has been widely adopted in many AI and NLP tasks, such as part-of-speech tagging. In the case of sub-tree alignment, MLE can be simply described as finding the optimal values of parameters that lead to the maximum probability of aligning the tree nodes from a source-language parse tree to a target-language parse tree. More formally, given a set of tree pairs $\{(S_1, T_1), ..., (S_n, T_n)\}$, the objective of the MLE-based training is defined to be:

$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{n} \sum_{d \in D(S_i, T_i)} \mathrm{P}_{\theta}(T_i, d \mid S_i) \tag{17}$$

We choose here Expectation Maximization (EM, Dempster, Laird, & Rubin, 1977) as the algorithm to solve the above optimization problem. Basically the EM algorithm is an iterative training method for finding the maximum likelihood estimates of model parameters, where it is assumed that the observed data depends on some latent variables. The algorithm performs by iteratively calling two sub-routines, namely the Expectation (E)-step and the Maximization (M)-step. In the E-step, it calculates the expected value of the likelihood function associated with the parameters and observed data, with respect to the distribution of latent variables given the observed data and current estimates of parameters. In the M-step, it seeks for the parameters that maximize the expected likelihood found in the E-step.

When applying the EM algorithm to our case, we can view the input pairs of parse trees $\{(S_1, T_1), ..., (S_n, T_n)\}$ as observed data, the underlying derivations of rules as latent variables, and the distributions $\mathrm{P}_{nt}(\cdot)$, $\mathrm{P}_{tree}(\cdot)$, $\mathrm{P}_{reorder}(\cdot)$ and $\mathrm{P}_{lex}(\cdot)$ (i.e., $\mathrm{P}_{length}(\cdot)$ and $\mathrm{P}_{w}(\cdot)$) as unknown parameters. See Figure 3 for the pseudo-code of the training algorithm for $\mathrm{P}_{nt}(\cdot)$ (denoted as $\theta_{t_{nt}|s_{nt}}$). Because this algorithm is directly applicable to the estimation of all parameters in our model, we skip the description of learning the remaining parameters here. For a more detailed description of the EM-based training of all model parameters we refer the reader to the appendix.

In the algorithm, s_{nt} and t_{nt} represent a source-language non-terminal symbol and a target-language non-terminal symbol, u and v represent a source-language tree node and a target-language tree node, $EC(\cdot)$ represents the expected count of the given variable, $\mathrm{P}_{\theta^{(k)}}(T, d \mid S)$ represents the derivation probability based on the parameters obtained in the k-th round. In each EM iteration, the E-step of the algorithm accumulates the expected count over all pairs of parse trees. Then, the M-step finds the maximum likelihood estimate using this quantity. The only nontrivial part in this algorithm is the computation of the expected count in the E-step. Roughly speaking, the physical meaning of the right-hand side of line 9 is the relative probability that a derivation contains rule r (with root node

Function TrainModelWithEM ($\{(S_1, T_1), ..., (S_n, T_n)\}$)
  Set $\{\theta^{(0)}_{t_{nt}|s_{nt}}\}$ = an initial model
  For $k = 0$ to $K - 1$ **do**
    Foreach non-terminal symbol pair (s_{nt}, t_{nt}) **do**
      $EC(\theta_{t_{nt}|s_{nt}}) = 0$
E-step:
    Foreach tree pair (S, T) in sequence $\{(S_1, T_1), ..., (S_n, T_n)\}$ **do**
      Foreach node pair (u, v) with symbol pair (t_{nt}, s_{nt}) in (S, T) **do**
        Foreach rule r rooted at (u, v) **do**
          $EC(\theta_{t_{nt}|s_{nt}}) += \frac{\sum_{d:\, r \in d \,\wedge\, r \text{ is rooted at } (u,v)} \mathrm{P}_{\theta^{(k)}}(T,d|S)}{\sum_{d'} \mathrm{P}_{\theta^{(k)}}(T,d'|S)}$
M-step:
    Foreach non-terminal symbol pair (s_{nt}, t_{nt}) **do**
      $\theta^{(k+1)}_{t_{nt}|s_{nt}} = \frac{EC(\theta_{t_{nt}|s_{nt}})}{\sum_{t'_{nt}} EC(\theta_{t'_{nt}|s_{nt}})}$
  return $\{\theta^{(K)}_{t_{nt}|s_{nt}}\}$

Figure 3: The EM-based training algorithm (for $\mathrm{P}_{nt}(\cdot)$)

pair (u, v)). The numerator $\sum_{d:\, r \in d \,\wedge\, r \text{ is rooted at } (u,v)} \mathrm{P}_{\theta^{(k)}}(T, d \mid S)$ is the probability sum over all the derivations that involve r , while the denominator $\sum_{d'} \mathrm{P}_{\theta^{(k)}}(T, d' \mid S)$ is the overall probability of the alignment from S to T. However, the brute-force computation of expected counts is very inefficient because it requires the sum over all possible derivations whose number is exponential in the length of the input sentences.

In this work we use the bilingual version of the inside and outside probabilities (Manning & Schütze, 1999) to avoid the naive enumeration of all possible derivations in computing various probabilities. The inside probability of (u, v) (denoted as $\beta(u, v)$) measures how likely we generate the sub-tree pair inside the node pair (u, v). The outside probability (denoted as $\alpha(u, v)$) is a dual of the inside probability. It measures how likely we generate the remaining parts of the tree pair (S, T) from the start symbols. Like the formulation used in monolingual parsing (Manning & Schütze, 1999), $\beta(u, v)$ and $\alpha(u, v)$ can be defined using the following recursive forms:

$$\beta(u, v) = \sum_{\forall r: root(r)=(u,v)} \Big(\mathrm{P}(r \mid S) \times \prod_{(p,q) \in yield(r)} \beta(p, q) \Big) \qquad (18)$$

$$\alpha(u, v) = \sum_{\forall r: (u,v) \in yield(r)} \Big(\alpha(root(r)) \times \mathrm{P}(r \mid S) \times \prod_{\substack{(p,q) \in yield(r) \\ \wedge\ (p,q) \neq (u,v)}} \beta(p, q) \Big) \qquad (19)$$

where $root(r)$ is the abbreviation of the node pair $(root(s_r), root(t_r))$, and $yield(r)$ is the set of the aligned frontier non-terminal pairs yielded by r. Based on the above recursive definitions, both $\beta(u, v)$ and $\alpha(u, v)$ can be efficiently computed with dynamic programming.

By using the inside and outside probabilities, it is easy to address the computation problem mentioned above. Let $\gamma(u, v)$ denote the probability that the tree node u is aligned with the tree node v. This probability can be expressed in an inside-outside fashion:

$$\begin{aligned}\gamma(u,v) &= \sum_{\substack{\forall d:\,(u,v)\text{ is}\\ \text{aligned in } d}} \mathrm{P}(T,d \mid S) \\ &= \beta(u,v) \times \alpha(u,v) \end{aligned} \tag{20}$$

In this way, the overall alignment probability from S to T (i.e., the denominator of the right-hand side of line 9) can be simply written as:

$$\sum_{d} \mathrm{P}(T,d \mid S) = \beta(root(S), root(T)) \times \alpha(root(S), root(T)) \tag{21}$$

For the numerator of the right-hand side of line 9, let us view it in another angle. In the E-Step of the algorithm, the expected count is accumulated over rules whose root is (u,v). As all the rules rooted at (u,v) indicate the same node alignment between u and v, lines 8-9 in principle imply the probability of all derivations aligning u to v, or more precisely the node alignment probability of (u,v). This probability can be written in a very simple form using the inside and outside probabilities:

$$\begin{aligned}\sum_{\substack{r:\,r\text{ is rooted}\\ \text{at } (u,v)}} \sum_{d:\,r\in d} \mathrm{P}(T,d \mid S) &= \gamma(u,v) \\ &= \beta(u,v) \times \alpha(u,v) \end{aligned} \tag{22}$$

Together with the result in Equation (21), the E-step can be efficiently implemented by replacing lines 8-9 with the following equation

$$EC(\theta_{t_{nt}|s_{nt}}) \mathrel{+}= \frac{\beta(u,v) \times \alpha(u,v)}{\beta(root(S), root(T)) \times \alpha(root(S), root(T))} \tag{23}$$

where (s_{nt}, t_{nt}) is the symbol pair of (u,v). Note that for each (s_{nt}, t_{nt}), the E-step step increases $EC(\theta_{t_{nt}|s_{nt}})$ by the sum of $\frac{\beta(u,v)\times\alpha(u,v)}{\beta(root(S),root(T))\times\alpha(root(S),root(T))}$ over all node pairs (u,v) whose symbols are s_{nt} and t_{nt}. This means that if (s_{nt}, t_{nt}) is aligned at different positions in the input tree pair, the above method considers the alignment of (s_{nt}, t_{nt}) for multiple times and updates $EC(\theta_{t_{nt}|s_{nt}})$ accordingly.

It is also worth noting that there are several methods for initializing the model parameters before the EM-style training begins. For example, the model can be initialized by uniform or random distributions. In this work we initialize all parameters in our sub-tree alignment model by the model obtained using the word alignment result. This is a standard way adopted in many unsupervised models where a "simpler" model is used for a good starting point in the training process. It is very helpful when the optimization procedure is sensitive to the initial setting of model parameters (e.g., EM for non-convex objective functions). In our experiments we found that using the GIZA++ word alignment for parameter initialization resulted in better performance and fewer iterations for convergence than the uniform initial distributions. As the word alignment can be obtained in an unsupervised manner, it does not change the training condition of our approach. Thus we chose this method for initializing the model parameters in our implementation.

3.4.2 The Bayesian Approach

While MLE is one of the standard approaches to training unsupervised models, it is well known for its tendency to overfit the data. The overfitting problem becomes more severe for complex models since they have more parameters and fit the training data better. In the case of STSGs, this is likely to result in the degenerate analysis of the data, i.e., rare and big rules dominate the ML solution of STSGs, while they are considered to be noisy and generalize poorly on the unseen data (Cohn & Blunsom, 2009; Liu & Gildea, 2009). A natural solution to this problem is to incorporate constraints or proper priors into the training process. Here we take the Bayesian approach as an alternative solution for the training problem.

Unlike MLE, the Bayesian approach does not plug a single optimum point estimate of the parameter into the distribution of a data point, but instead account for any uncertainty in the value of the parameter. In Bayesian models, the parameters are assumed to be drawn from some probability distributions or priors. The parameters of these extra prior distributions are called *hyperparameters*, and are denoted by ω. As the parameters of our model are viewed mathematically as multinomials, we choose Dirichlet distributions (Ferguson, 1973) for the prior over model parameters. The advantage of using Dirichlet distributions is that they are conjugate to multinomial distributions and inference with such priors is easier.

Following the previous description, we use θ to denote the model parameters which are multinomial with outcomes $\{1, ..., K\}$ (i.e., θ_k is the probability of outcome $k \in \{1, ..., K\}$). From this multinomial distribution we sample a set of outcomes $\{x_1, ..., x_n\}$ with probability $\mathrm{P}(x_i = k) = \theta_k$. As the Dirichlet prior is a distribution over multinomials, each sample from the prior is actually a set of parameter values θ. Therefore the distribution can be modeled as:

$$x_i \mid \theta \sim \text{Multinomial}(\theta) \tag{24}$$

$$\theta \mid \omega \sim \text{Dirichlet}(\omega) \tag{25}$$

Here Equation (24) means that x_i is distributed according to a multinomial with parameters θ. Similarly, Equation (25) can be read as θ is distributed according to a Dirichlet distribution with parameters ω. $\omega = \{\omega_1, ..., \omega_K\}$ is the hyperparameter vector corresponding to the outcomes. In this work we use a symmetric Dirichlet prior (i.e., $\omega_1, ..., \omega_K$ share the same value), and use ω to represent the single hyperparameter instead of the hyperparameter vector.

Using this model, we can compute the conditional distribution of a new observation x_{n+1} given previous observations $\{x_1, ..., x_n\}$ and the hyperparameter ω, as follows:

$$\mathrm{P}(x_{n+1} \mid x_1, ..., x_n, \omega) = \int \mathrm{P}(x_{n+1} \mid x_1, ..., x_n, \theta)\, \mathrm{P}(\theta \mid \omega)\, \mathrm{d}\theta \tag{26}$$

The big advantage of the Bayesian approach is to introduce a prior distribution over the unknown parameters of the model, which is meant to capture the knowledge and beliefs about the model before seeing the data (Neal, 1998). It is especially important in our case where we need a "bias" towards some preferred situations. For example, we expect that our model can favor high frequency rules and dislike rare and big rules. This goal can be easily achieved by using the Bayesian approach and an appropriate choice of priors, say, a Dirichlet prior with a low concentration parameter ω. However, the introduction of priors generally makes it intractable to estimate the posterior analytically. In practical systems based on the Bayesian approach, a widely-used solution is to use approximate methods to seek a compromise between exact inference and computational resources. In this work we choose *Variational Bayes* for approximate inference. Variational Bayes is a good method that preserves the benefits of introducing the prior but with a tractable inference procedure (Attias, 2000; Beal, 2003). It has been successfully applied to several NLP-related models, such as Hidden Markov Models (HMMs) and IBM Models (Beal, 2003; Riley & Gildea, 2010). One more good thing is that Variational Bayes can be seen as an extension of the EM algorithm and resembles the usual forms used in EM. The resulting procedure looks a lot like the EM algorithm with a modified M-step, which is very convenient for implementation. Here we follow the approach presented in previous work (Beal, 2003; Riley & Gildea, 2010) where variational Bayesian algorithms are applied to similar tasks. All we need is only a very slight change to the M-step of the original EM algorithm presented in Section 3.4.1. In the original EM algorithm (see Figure 3), the M-step normalizes the expected counts collected in the E-step as standard MLE. The variational Bayesian version of the M-step slightly modifies this formula and performs an inexact normalization by passing counts through function $f(x) = \exp(\psi(x))$.

$$\theta_{t_{nt}|s_{nt}} = \frac{f(EC(\theta_{t_{nt}|s_{nt}}) + \omega)}{f(\sum_{t'_{nt}}(EC(\theta_{t'_{nt}|s_{nt}}) + \omega))} \tag{27}$$

where $\psi(x)$ is the digamma function (Johnson, 2007). It has an approximate effect of subtracting 0.5 from its argument. The choice of ω controls the behavior of this estimation. When ω is set to a low value, it performs estimation as a way of "anti-smoothing". As about 0.5 is subtracted from the rule counts, small counts corresponding to rare events are penalized heavily, while large counts corresponding to frequent events are not be affected very much. For example, low values of ω make Equation (27) favor the non-terminal pairs which are aligned frequently and distrust the non-terminal pairs which are aligned rarely. In this way, the variational Bayesian method could control the overfitting caused by abusing rare events. On the other hand, a larger ω can be used when smoothing is required.

The above method is applicable to training all the parameters in our model. It only requires a replacement of the M-step in Figure 3 with the variational Bayesian M-step (as in Equation (27)). In our implementation, after the variational Bayes-based training, we perform an additional round of normalization without variational Bayes to normalize rule probabilities to sum to one.[9]

9. The additional normalization process makes the posterior probabilities directly comparable with those obtained in other training methods, such as the EM-based training. Note that here we convert the result of Bayesian inference into some probability distributions for a good explanation of various probability factors in our model. On the other hand, this technical trick results in a pseudo-Bayesian procedure that is not doing Bayesian inference exactly though it shows good results in our empirical study. One can remove the additional round of normalization for a pure Bayesian approach. But all these changes do not affect the overall pipeline of our approach (from a practical standpoint).

Function DECODE (S, T)
  $(\beta[\cdot], \alpha[\cdot])$ = GETINSIDEOUTSIDEPROBABILITIES (S, T)
  Foreach node u in S in bottom-up order **do**
    Foreach node v in T in bottom-up order **do**
      $\gamma[u, v] = \beta[u, v] \times \alpha[u, v]$
      Foreach tree-fragment s_r rooted at u **do**
        Foreach tree-fragment t_r rooted at v **do**
          Foreach frontier non-terminal alignment a between s_r and t_s **do**
            r = CREATERULE(s_r, t_r, a)
            $score = \mathrm{P}(r \mid S) \cdot \prod_{(p,q) \in yield(r)} \mathrm{P}(d[p, q])$
            If $score > \mathrm{P}(d[u, v])$ **then**
              $d[u, v]$ = CREATEDERIVATION(r, $\{d[p, q] : (p, q) \in yield(r)\}$)
  return $(d[\cdot], \gamma[\cdot])$
Function GETINSIDEOUTSIDEPROBABILITIES (S, T)
  Foreach node u in S in bottom-up order **do**
    Foreach node v in T in bottom-up order **do**
      Set $\beta[u, v]$ according to Equation (18)
  Foreach node u in S in top-down order **do**
    Foreach node v in T in top-down order **do**
      Set $\alpha[u, v]$ according to Equation (19)
  return $(\beta[\cdot], \alpha[\cdot])$

Figure 4: Decoding algorithm of the proposed sub-tree alignment model for both 1-best and posterior-based outputs

3.5 Decoding

Inference with our model is straightforward. The simplest case is inferring the 1-best sub-tree alignment. Given a set of learned parameters, we first visit every node pair (u, v) in a bottom-up fashion, and compute the posterior probability of aligning the sub-tree pair rooting at (u, v). This procedure is the same as the dynamic program used in the trainer. We then select the derivation with the maximum sub-tree alignment probability for the input tree pair. Also, we can generate a list of k-best derivations in a similar manner.

In addition to the 1-best/k-best output, our model is able to output the alignment posterior probability for every pair of tree nodes. To do this, we only need to record the probability $\gamma(u, v)$ for each node pair after we obtain the inside and outside probabilities. Note that outputting alignment posterior probabilities is also commonly used in statistical word and phrasal aligners. It provides a flexible way of making use of the alignment result for the downstream components, such as the rule extraction system. As is presented in the very next sections, tree-to-tree MT systems can make great benefits from the posterior-based alignment output, which results in a very effective rule extraction method as well as better translation results.

Figure 4 depicts the pseudo-code of the decoding algorithm for both 1-best and posterior-based outputs. In this algorithm, $d[x, y]$ is a data structure that records the best derivation rooted at (x, y). $\beta[x, y]$, $\alpha[x, y]$ and $\gamma[x, y]$ are data structures that record the inside probabilities, output probabilities and alignment posterior probabilities, respectively. CREateRule($\cdot$) creates a rule with a pair of tree-fragments (s_r, t_r) and a frontier non-terminal alignment a, and calculates the rule probability. CreateDerivation($\cdot$) builds a derivation using the input rules. For output, we can access the 1-best alignment by traversing from $d[root(S), root(T)]$, and access the alignment posterior through $\gamma[\cdot]$.

Given a pair of trees (S, T), the outer two loops of the algorithm iterates over each pair of nodes in the two trees, resulting in time complexity of $O(|S| \cdot |T|)$ where $|\cdot|$ represents the size of the input tree. Generating all pairs of tree-fragments requires $O(N_{tree}^2)$, where N_{tree} is the maximum number of tree-fragments given a tree node. Computing the alignment between (s_r, t_r) requires $O(L!)$ where L is the maximum number of leaf non-terminals in a rule. Therefore the time complexity of this algorithm is $O(|S| \cdot |T| \cdot N_{tree}^2 \cdot L!)$, quadratic in the size of the input trees. Note that the actual time complexity of this algorithm could be very high if all potential alignments are considered. For example, N_{tree} is generally an exponential function of the depth of the input tree-fragment, and a very deep tree could results in an extremely large space of alignments. To make practical sub-tree alignment systems, pruning techniques are taken into account in this work. For example, in our implementation, we restrict the depth of tree-fragment to a reasonable number (see Section 5.2.2). In addition, as is commonly-used in phrasal alignment and related tasks, we consider word alignments in pruning and discard the sub-tree alignments which violate a certain number of word alignments. For example, we throw away the sub-tree alignments if there are more than two word alignment links outside the spans covered by the aligned sub-trees.

4. Applying Sub-tree Alignment to Tree-to-Tree Translation

Once sub-tree alignment is obtained, current tree-to-tree systems can directly learn translation rules from node-aligned tree pairs. In this section we investigate methods of applying sub-tree alignment to tree-to-tree rule extraction.

4.1 Rule Extraction Using 1-best/k-best Sub-tree Alignments

To data, several methods have been developed for tree-to-tree rule extraction (Zhang et al., 2008; Liu et al., 2009a; Chiang, 2010). The most popular of these is the GHKM-like method which extends the idea of extracting syntactic translation rules from string-tree pairs (Galley et al., 2004). In GHKM-like extraction, we first compute the set of the minimally-sized translation rules that can explain the mappings between the source-language tree and the target-language tree while respecting the alignment and reordering between the two languages. Larger rules are then learned by composing two or more minimal rules. For example, in Figure 2(b), r_7 and r_9 are two minimal rules extracted according to the sub-tree alignment. We can compose these rules to form a larger rule, like this:

$$\text{IP(NN(进口) VP}_1) \ \rightarrow \ \text{S(NP(DT(the) NNS(imports)) VP}_1)$$

In this work we use the tree-to-tree version of GHKM-like extraction which is described in Liu et al.'s (2009a) work. See Figure 5(a) for the pseudo-code of rule extraction with 1-best sub-tree alignment. We choose this method because it has been widely used in most tree-to-tree systems. Note that rule extraction in tree-to-tree translation is generally not restricted to be performed on the 1-best sub-tree alignment result. The GHKM-like

```
Function OneBestExtract (S, T, A)
  Foreach node u in S do
    Foreach node v in T do
      Foreach tree-fragment pair (s_r, t_r)
            rooted at (u, v) do
        a = OneToOneAlign(s_r, t_r, A)
        If a is not empty then
          r = CreateRule(s_r, t_r, a)
          rules.Add(r)
  return rules
Function OneToOneAlign(s_r, t_r, A)
  If frontier non-terminals of (s_r, t_r) have
     1-to-1 alignments in A then
    return frontier alignment of (s_r, t_r)
  Else
    return φ
```

(a) 1-best Extraction

```
Function MatrixExtract (S, T, M)
  Foreach node u in S do
    Foreach node v in T do
      If IsExtractable({(u, v)}, M) do
        next loop
      Foreach tree-fragment pair (s_r, t_r)
                rooted at (u, v) do
        Foreach frontier alignment a
                between (s_r, t_r) do
          If IsExtractable(a, M) then
            r = CreateRule(s_r, t_r, a)
            rules.Add(r)
  return rules
Function IsExtractable(a, M)
  Foreach alignment (p, q) in a do
    If probability of (p, q) in M < P_min then
      return false
  return true
```

(b) Matrix-based Extraction

Figure 5: The 1-best and matrix-based rule extraction algorithms

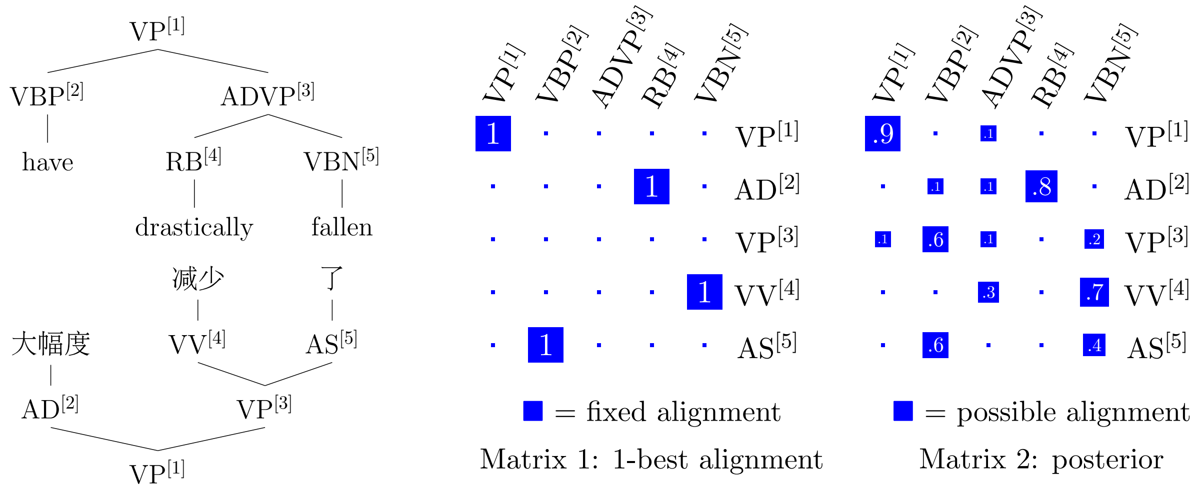
extraction method can be employed when a list of k-best sub-tree alignments is provided. In k-best extraction we only need to repeat the procedure of 1-best extraction for each sub-tree alignment in the k-best list.

4.2 Rule Extraction Using Sub-tree Alignment Matrices

Previous work has pointed out that current MT systems suffer from error propagation due to the alignment errors made within 1-best alignment (Venugopal, Zollmann, Smith, & Stephan, 2008). As sub-tree alignment is an early-stage step in the training pipeline, the errors in 1-best alignment are likely to be propagated to translation rule extraction and parameter estimation of the translation model. Though this problem can be alleviated by using k-best alignments, the limited scope of k-best alignments still results in inefficient learning of translation rules. For example, our preliminary experiment shows that 95.8% of the extracted rules are redundant when 100-best alignments are involved.

Here we instead present a simple but efficient method, namely *matrix-based rule extraction*. In this method, we use the posterior-based output of our aligner and represent the sub-tree alignment with a compact structure - call it *sub-tree alignment matrix* or *alignment matrix* for short (Liu, Xia, Xiao, & Liu, 2009b; de Gispert, Pino, & Byrne, 2010).

See Figure 6(a) for two example sub-tree alignment matrices made from a pair of sentence segments. In the matrices, each entry is indexed by a pair of source and target nodes. The score in an entry is the posterior probability of the alignment between the corresponding node pair, i.e., the $\gamma(u, v)$ probability defined in Equation (20). This probability is straightforwardly accessible in the output of the inference algorithm described in Section 3.5. In principle $\gamma(u, v)$ can be viewed a measure of sub-tree alignment confidence: a higher value indicates a more confident alignment between the two nodes. In this way we can



(a) Sub-tree alignment matrices for a sample sub-tree pair

Minimal Rules Extracted from Matrix 1 (1-best)		**Minimal Rules Extracted from Matrix 2 (posterior)**	
r_3	AD(大幅度) → RB(drastically)	r_3	AD(大幅度) → RB(drastically)
r_4	VV(减少) → VBN(fallen)	r_4	VV(减少) → VBN(fallen)
r_6	AS(了) → VBP(have)	r_6	AS(了) → VBP(have)
r_8	VP(AD_1 VP(VV_2 AS_3)) → VP(VBP_3 ADVP(RB_1 VBN_2))	r_8	VP(AD_1 VP(VV_2 AS_3)) → VP(VBP_3 ADVP(RB_1 VBN_2))
		r_{10}	VP(VV(减少) AS(了)) → VBN(fallen)
		r_{11}	VP(AD_1 VP_2) → VP(VBP_1 ADVP_2)
		...	

(b) Rules extracted using 1-best alignment and alignment posterior

Figure 6: Matrix-based representation of sub-tree alignment and sample rules extracted. Matrix 1 shows the case of 1-best sub-tree alignment, and Matrix 2 shows the case of sub-tree alignment posterior.

access all possible sub-tree alignments (with different probabilities), rather than a limited number of them.

We then extract rules using the sub-tree alignment matrix. The method is simple: we collect the rules associated with each entry from the matrix. The core algorithm of this method is in essential the same as that used in 1-best/k-best extraction. The only difference from 1-best/k-best extraction is that the matrix-based method considers all possible node pairs for extraction, rather than visiting some of them only. See Figure 5(b) for the pseudo-code of the sub-tree alignment matrix-based rule extraction algorithm, where M represents a sub-tree alignment matrix for the pair of trees (S, T). Compared to extracting rules from k-best alignments, this method can efficiently obtain additional rules whose extraction is blocked in k-best extraction. For example, on the right side of Figure 6(b) two new rules r_{10} and r_{11} are extracted, which cannot be obtained from the 1-best alignment result. To prevent the extraction of a great number of noisy rules with low alignment probabilities, we prune away those rules whose alignment probabilities are below a pre-specified threshold. More formally, given a pair of nodes (u, v), rule extraction can be executed at (u, v) only when it satisfies:

$$\frac{\gamma(u,v)}{\gamma(root(S), root(T))} < P_{min} \tag{28}$$

This expression measures the relative probability of alignment (u, v) with respect to the sum of probabilities over all possible derivations. P_{min} is an empirical threshold to control how often rules to be pruned (a larger P_{min} means more rules are thrown away). In this work, it is set to 10^{-7} by default. Therefore, all those entries with a zero score in Figure 6(a) (denoted as a dot) are excluded in rule extraction.

However, discarding rules with relatively low probabilities in turn results in an incompleteness problem, that is, the extracted rules might be unable to transform a given source parse-tree, even in the training set. Nonetheless, this problem is not very severe in our case. In our experiments we observed that most parse-tree pairs (over 90%) in the training corpus could be recovered by the extracted rules when P_{min} chose its default value, and the contribution to translation accuracy from those low confidence rules was very limited (generally less than 0.1 BLEU points).

Another note on sub-tree alignment matrix-based extraction. The advantage of this method is that it follows the general and well-developed framework of syntax-based MT, i.e., word/syntactic alignment + rule extraction/parameter estimation + MT decoding. All we need is to replace the rule extraction component with our sub-tree alignment matrix-based system, while preserve other components of the pipeline. This means that we can still use some heuristics to obtain additional useful rules from the result of sub-tree alignment matrix-based extraction, such as rule composing (Galley et al., 2006) and SPMT extraction (Marcu, Wang, Echihabi, & Knight, 2006). Also, the posterior probability encoded in the matrix can be used for better estimation of various MT-oriented features.[10]

Note that the basis of our approach is the STSG model, and all rules in the sub-tree alignment model resemble the general forms of the translation rules used in tree-to-tree MT systems. So, as an alternative but simple way of rule induction, we can directly infer translation rules from our sub-tree alignment model and take the corresponding rule probabilities as features of the translation model for MT decoding. However, in tree-to-tree MT this method suffers from several problems. First, our sub-tree alignment model requires computation of all possible aligned tree-fragments, which results in very high time complexity for both training and decoding procedures. As a result, aggressive pruning has to be used for a reasonable size of search space, e.g., we consider only relatively small tree-fragments in our implementation for acceptable running speed. As a side effect, many relatively large rules (e.g., composed rules and SPMT rules) are absent in our sub-tree alignment model, while they are available if we use the traditional *alignment + extraction heuristics* pipeline. From an engineering standpoint, it is not efficient to directly infer translation rules from the sub-tree alignment model, compared to inferring rules using a pruned and fixed sub-tree alignment matrix plus some heuristics. Second, the rule probability and optimization objective for sub-tree alignment is very different from those used in MT systems. For example, we use a generative model and a maximum-likelihood/Bayesian approach in sub-tree

10. See Section 4.3 for more detailed discussion of the parameter estimation issue.

alignment, but use a discriminative model and minimum error rate training in MT. Many features employed in the MT decoder are not considered in our sub-tree alignment model. All the above issues might lead to unsatisfactory MT performance. As shown in our experiments (see Section 5.3.5), directly inferring translation rules from the sub-tree alignment model does not achieve promising results.

4.3 Learning Features for Machine Translation

In previous work on syntax-based MT, it is proved that syntax-based systems can make great benefits from MT-oriented features, even some of them are not necessarily well explained in the syntactic parsing viewpoint (e.g., phrase-based translation probabilities). However, most of these features are not available in the word/sub-tree alignment model. Instead we need to learn these features using an additional step of parameter estimation for MT. To do this, we follow the commonly-used framework which estimates values of various MT-oriented features on the extracted rule set using MLE. This procedure is simple: once all translation rules are extracted, we obtain the maximum-likelihood (or relative-frequency) estimate of parameters according to the definition of each feature function.

However, in traditional tree-to-tree systems each rule extracted from a tree pair has a count of unit one, which is then used to calculate the values of various features. Such a approach might enlarge the influence of noisy rules extracted from sub-tree alignment matrices. E.g., a rule with a high alignment probability has an equal weight as a rule with a low alignment probability, and thus has a unreasonably large impact on MT systems. A more desired solution is that a rule extracted from a derivation having a low probability is penalized accordingly in feature learning. Motivated by this idea, we use fractional counts to estimate the appearance for each rule (Mi & Huang, 2008). Given a node pair (u, v) in (S, T), the alignment probability of a rule r rooted at (u, v) is defined to be (denoted as $\gamma(r; u, v)$):

$$\gamma(r; u, v) = \sum_{\substack{d \in D(S,T) \\ \wedge\, r \in d}} \mathrm{P}(T, d \mid S) \tag{29}$$

where $\gamma(r; u, v)$ is regarded as the probability sum over all derivations involving r at (u, v). Also, we can rewrite Equation (29) in an inside-outside fashion:

$$\gamma(r; u, v) = \alpha(u, v) \times \prod_{(p,q) \in yield(r)} \beta(p, q) \times \mathrm{P}(r \mid S) \tag{30}$$

Then we define the probability that r is involved in the derivations of (S, T) as:

$$\gamma(r) = \sum_{u,v} \gamma(r; u, v) \tag{31}$$

Equation (31) is the sum of probabilities of r over all node pairs. This means that the rule probability can be considered for multiple times if some particular derivations contain r more than once. By using $\gamma(r)$, the fractional count of r is defined to be:

$$c(r) = \frac{\gamma(r)}{\gamma(root(S), root(T))} \tag{32}$$

Equation (32) reflects the probability how likely r is involved in a derivation given a pair of trees. For a set of bilingual parse trees, $c(r)$ can be accumulated over each tree pair.

Obviously, $c(r)$ can be used to estimate the parameters of the MT model, that is, once translation rules are weighted, the parameter estimation procedure can proceed as usual, but with weight counts. In this work $c(r)$ is employed to learn five features used in the MT decoder, including the bi-directional phrase-based conditional translation probabilities (Marcu et al., 2006) and three syntax-based conditional probabilities (Mi & Huang, 2008). Let $\varphi(\cdot)$ be a function that returns the sequence of frontier nodes for an input tree-fragment. These probabilities can be computed by the following equations:

$$\mathrm{P}_{phrase}(t_r \mid s_r) = \frac{\sum_{r'':\varphi(s_{r''})=\varphi(s_r)\wedge\varphi(t_{r''})=\varphi(t_r)} c(r'')}{\sum_{r':\varphi(s_{r'})=\varphi(s_r)} c(r')} \tag{33}$$

$$\mathrm{P}_{phrase}(s_r \mid t_r) = \frac{\sum_{r'':\varphi(s_{r''})=\varphi(s_r)\wedge\varphi(t_{r''})=\varphi(t_r)} c(r'')}{\sum_{r':\varphi(t_{r'})=\varphi(t_r)} c(r')} \tag{34}$$

$$\mathrm{P}(r \mid root(r)) = \frac{c(r)}{\sum_{r':root(r')=root(r)} c(r')} \tag{35}$$

$$\mathrm{P}(r \mid s_r) = \frac{c(r)}{\sum_{r':s_{r'}=s_r} c(r')} \tag{36}$$

$$\mathrm{P}(r \mid t_r) = \frac{c(r)}{\sum_{r':t_{r'}=t_r} c(r')} \tag{37}$$

5. Experiments

For evaluation, we first experimented with our approach on a Chinese-English sub-tree alignment task, then tested its effectiveness in a state-of-the-art tree-to-tree MT system.

5.1 Baselines

Three unsupervised sub-tree alignment methods were chosen as baselines in our experiments.

- *WordAlign-1*: WordAlign-1 is based on a GHKM-like method (Galley et al., 2004) that uses word alignments to infer syntactic correspondences. In our implementation, both the GIZA++ toolkit and the "grow-diag-final-and" method were used to obtain the symmetric word alignment from sentence pairs. The sub-tree alignments were then heuristically induced by selecting those node correspondences that are consistent with the word alignment result (i.e., sub-tree alignments that do not violate any word alignments). We chose this method because it has been widely adopted in modern tree-to-tree systems.

- *WordAlign-2*: the second baseline is essentially the same as WordAlign-1. The only difference from WordAlign-1 is that we improved the word alignment system using link-deletion techniques (Fossum et al., 2008). The basic idea is to delete harmful alignment links from an initial word alignment result (e.g., deleting the link between "了" and "the" in Figure 2(a)). In our experiments we only considered the most likely deletion between the top-10 most common Chinese words (including {的, 在, 和, 是,

了, 一, 对, 有, 中, 也}) and the top-10 most common English words (including {the, of, and, to, in, a, is, that, for, on}).

- *HeuristicAlgin*: HeuristicAlgin is a re-implementation of the approach proposed in Tinsley et al.'s (2007) work. In this method the alignment confidence of every node pair is first computed with lexical translation probabilities, and then used to obtain node correspondences via a heuristic algorithm. Because this method does not require a training process and has been successfully adopted in several translation tasks, such as French-English translation, it was chosen as another baseline for comparison.

5.2 Experimental Setup

The settings of our experiments are described as follows.

5.2.1 Data Preparation

Our bilingual corpus consists of 1.06 million sentence pairs.[11] As mentioned above, we used GIZA++ and the "grow-diag-final-and" heuristics to generate 1-best/k-best word alignments, which were then used as our baseline word alignment results. The parse trees in both Chinese and English were generated using the Berkeley Parser.[12] A publicly available corpus was used to evaluate the sub-tree alignment result.[13] It consists of 736 node-aligned sentence pairs (with gold-standard parse trees on both language sides) in LDC2003E07 which is also included in our bilingual data. This corpus was divided into two parts: a held-out set used for finding an appropriate setting of hyperparameters (99 sentences in articles 301-309), and a test set used for evaluating the sub-tree alignment systems (637 sentences in articles 001-066). For MT experiments, a 5-gram language model was trained on the Xinhua portion of the Gigaword corpus in addition to the English part of the LDC bilingual training data.[14] We used the NIST 2003 MT evaluation corpus as our development set (919 sentences) and the newswire portion of the NIST 2004-2006 MT evaluation corpora as our test set (3,486 sentences).

5.2.2 Sub-tree Alignment

All parameters of the sub-tree alignment model were initialized with the add-one smoothing on the rule-set extracted using word alignments (i.e., the WordAlign-1 baseline). Then, the model was trained on the parse trees of the bilingual corpus using the EM algorithm or the Variational Bayes (VB) approach. In our implementation of the VB-based training, all hyperparameters are assumed to share the same value.[15] This leads to a setting of $\omega = 0.01$

11. LDC category: LDC2003E14, LDC2005T10, LDC2003E07, LDC2005T06, LDC2005E83, LDC2006E26, LDC2006E34, LDC2006E85, LDC2006E92 and LDC2004T08. See http://www.ldc.upenn.edu/ for more details.
12. Note that for the LDC2003E07 corpus we reused the gold-standard parse trees provided in the Chinese and English treebanks.
13. Available from http://www.nlplab.com/resources/nodealigned-bitreebank.html
14. LDC category of the English Gigaword corpus: LDC2003T05
15. Although we could adopt different hyperparameters for finer control over the priors of model parameters, we found that setting those hyperparameters to the same value could also lead to satisfactory performance.

which is an optimal value on the held-out set. By default, we trained our model for 5 EM or Variational EM iterations. To speed-up the training process and further avoid degenerate analysis caused by too large rules, we restricted ourselves to rules with reasonable sizes - rules with at most five frontier non-terminals and depth of three. For rules having more than five frontier non-terminals, we only considered the tree-fragments of depth one but did not restrict the number of frontier non-terminals involved, that is, for flat tree structures, we only used the associated height-one tree-fragments. Besides, we discarded the sub-tree alignment between every node pair whose terminals are aligned outside the corresponding spans for more than two times in WordAlign-1.

5.2.3 Machine Translation

We used the NiuTrans open-source toolkit (Xiao, Zhu, Zhang, & Li, 2012) to build our tree-to-tree MT system. For rule extraction, we used an extension of the GHKM method to extract minimal tree-to-tree transformation rules (Liu et al., 2009a) and obtained larger rules by composing two or three minimal rules (Galley et al., 2006). We used a CKY-style decoder with cube pruning (Huang & Chiang, 2005) and beam search to decode Chinese sentences. By default the beam size was set to 50. In addition to the features described in Equations (33)-(37), we also used several other features in our MT system, including the 5-gram language model, the rule number bonus, the target length bonus and two binary features - lexicalized rule and low frequency rule (Marcu et al., 2006). These features are combined in a log-linear fashion and optimized using Minimum Error Rate Training (MERT, Och, 2003).

5.3 Results

In the following part of this section, we present our experimental results, including evaluations of sub-tree alignments, extracted rules, and MT systems. Also, we show results of several improved methods for the effective use of our approach in tree-to-tree MT.

5.3.1 Evaluation of Alignments

First we evaluated the alignment quality of various sub-tree alignment approaches in terms of precision (P), recall (R) and F-1 score.[16] See Table 2 for results of the three baseline systems and our sub-tree alignment system. By those measures, our VB-based system significantly improves both the overall recall and F-1 score, slightly degrading in precision compared to WordAlign-1/2. Also, the VB-based training outperforms the EM-based counterpart due to the priors introduced into the learning process. The interesting observation here is that, though the EM training of our model suffers from the degenerate analysis of the data, it does not show extremely bad results in our experiment. This phenomenon is due to our restriction on the size of tree-fragment in training. As described in Section 5.2.2, we restricted the translation rules to those reasonable-size tree-fragments in several ways (e.g.,

16. Let *predicted* be the number of alignments in system output, *correct* be the number of correct alignments in system output, *gold* be the number of alignments in gold-standard. The measure of precision, recall and F-β score can be defined as: precision $= \frac{correct}{predicted}$, recall $= \frac{correct}{gold}$ and F-$\beta = \frac{(1+\beta^2)\times\text{precision}\times\text{recall}}{\beta^2\times\text{precision}+\text{recall}}$. Here β is a parameter that controls the preference for recall (i.e., $\beta > 1$) or precision (i.e., $0 \leq \beta < 1$). In most NLP tasks β is set to 1, indicating equal weights of recall and precision.

Entry	WordAlign-1			WordAlign-2			HeuristicAlgin			Ours (EM)			Ours (VB)		
	P	R	F-1	P	R	F-1	P	R	F-1	P	R	F-1	P	R	F-1
Overall	75.4	63.6	69.0	76.3	64.5	69.9	65.7	67.7	66.7	79.8	46.2	58.5	72.6	75.1	73.8
NP $\leftrightarrow$ NP	86.9	59.0	70.3	87.0	62.1	72.5	79.1	73.6	76.3	84.2	48.2	61.3	88.7	75.3	81.4
NN $\leftrightarrow$ NN	83.9	75.6	79.5	83.6	77.0	80.2	76.2	74.1	75.1	81.8	63.7	71.6	81.1	79.9	80.5
VP $\leftrightarrow$ VP	75.3	61.9	68.0	74.9	61.9	67.8	71.3	71.8	71.6	80.5	49.0	60.9	75.7	75.8	75.7
PU $\leftrightarrow$,	84.7	76.5	80.4	84.7	76.5	80.4	69.6	71.3	70.3	82.7	67.7	74.5	82.5	80.7	81.6
IP $\leftrightarrow$ S	92.5	87.5	89.9	92.3	87.7	89.9	90.6	90.7	90.6	90.4	76.8	83.0	90.0	92.4	91.2
PU $\leftrightarrow$.	98.5	98.7	98.6	98.5	98.7	98.6	98.5	98.7	98.6	94.7	86.8	90.6	98.5	98.5	98.6
NP $\leftrightarrow$ NN	77.6	71.5	74.4	83.0	71.5	76.8	59.3	77.5	67.2	75.4	62.3	68.2	75.9	78.9	77.4
NP $\leftrightarrow$ PP	46.8	63.8	54.0	49.7	63.5	55.7	53.1	31.9	39.8	43.2	42.1	42.6	54.5	70.1	61.3
NN $\leftrightarrow$ NNS	84.2	76.0	79.9	83.8	76.5	80.0	77.8	75.5	76.7	83.7	58.3	68.7	81.0	77.6	79.2
NR $\leftrightarrow$ NNP	69.4	41.2	51.8	69.9	41.7	52.3	63.4	57.4	60.2	57.7	41.5	48.3	67.2	56.3	61.3
NN $\leftrightarrow$ NP	65.1	59.8	62.4	65.9	61.0	63.3	71.8	50.4	59.2	70.5	48.1	57.2	71.1	67.7	69.4
PP $\leftrightarrow$ PP	84.6	68.4	75.7	85.1	69.2	76.3	79.3	72.6	75.8	85.6	56.9	68.4	85.4	82.5	83.9
NN $\leftrightarrow$ JJ	90.7	76.5	83.0	90.4	76.3	82.7	83.9	81.7	82.8	84.2	69.1	75.9	85.9	81.2	83.5
P $\leftrightarrow$ IN	81.5	69.8	75.2	82.3	68.3	74.7	81.2	72.2	76.4	79.9	57.9	67.1	84.7	76.1	80.2
QP $\leftrightarrow$ NP	72.2	64.9	68.3	72.6	65.2	68.7	67.1	65.6	66.4	78.3	42.2	54.8	74.9	71.7	73.3

Table 2: Evaluation results of sub-tree alignment for our system and the baselines. All measures are reported in percentage.

we set a parameter of maximum depth). While such constraints reduce the number of rules involved in training, it prevents the use of rare and large rules. Our result here indicates the fact that the tree-fragment size constraint is actually not only important for efficiency but also crucial for learning. As discussed in previous work, without these constraints or imposing a proper prior, the solution of EM is degenerate (Marcu & Wong, 2002; DeNero, Gillick, Zhang, & Klein, 2006).

In addition, Table 2 shows the result for the 15 most common types of sub-tree alignment. As expected, the VB-based system achieves the best F-1 score in most cases. More interestingly, it is observed that our approach obtains significantly better performance in handling the PP (Prepositional Phrase) alignment that seems to be a difficult problem for baselines due to the unclear boundary indicators in aligning the PP structures. We attribute this to the better use of syntactic information on both language sides in our model, which are generally ignored in traditional models based on surface heuristics and word alignments.

5.3.2 Evaluation of Extracted Rules

We then applied the sub-tree alignment result to our tree-to-tree system to study the impact of sub-tree alignment on MT. As discussed in Section 4, rule extraction is a downstream component of sub-tree alignment in the current tree-to-tree MT pipeline. We therefore chose to evaluate the quality of the rules obtained from various sub-tree alignment results. To determine the goodness of extracted grammars, we computed the rule precision, recall, and F-1 scores for our approach and the baseline approaches on the same test set used in the (1-best) alignment quality evaluation. To make a gold-standard grammar, we chose the method used in Fossum et al.'s (2008) work where the grammar was automatically generated from the manually-annotated alignment result, that is, the rules extracted using the annotated sub-tree alignments were regarded as the gold-standard in computing various evaluation scores. Table 3 shows the evaluation result for the grammars extracted with

	Entry	Rule P	Rule R	Rule F-1
1best	WordAlign-1	51.9	60.8	55.9
	WordAlign-2	52.3	61.8	56.6
	HeuristicAlgin	55.8	55.3	55.5
	Ours (EM)	61.9	49.2	54.8
	Ours (VB)	54.9	65.2	59.6
matrix	Ours (VB + $P_{min} = 10^{-5}$)	79.6	34.5	48.2
	Ours (VB + $P_{min} = 10^{-6}$)	53.0	70.0	60.3
	Ours (VB + $P_{min} = 10^{-7}$)	41.3	75.6	53.4
	Ours (VB + $P_{min} = 10^{-8}$)	34.9	79.5	48.5

Table 3: Evaluation results of rules obtained from various sub-tree alignment approaches. All measures are reported in percentage.

different sub-tree alignment approaches. We see that the improvements persist when our sub-tree alignments are employed in translation rule extraction. Our VB-based approach produces grammars with a higher rule F-1 score than all three of the baselines.

In addition to the 1-best extraction, we studied how rule extraction behaves under the sub-tree alignment matrix-based extraction method. Table 3 also shows the result of the sub-tree matrix-based extraction method with different choices of the pruning parameter P_{min}. We see that smaller values of P_{min} result in grammars with higher rule recall. Also, better rule F-1 scores can be achieved by adjusting P_{min} and seeking a good balance between rule precision and rule recall , e.g., $P_{min} = 10^{-6}$ or 10^{-7}.

The above scores are informative as a measure of grammar quality, but we also investigated some of the differences in the rule sets obtained from our model compared to the baseline approaches, following Levenberg, Dyer, and Blunsom's (2012) method. Figure 7 shows the most probable rules (frequency ≥ 2) obtained from our bilingual corpus using the VB-based alignment approach that do not appear in the model from the WordAlign-2 alignment and vice versa. We asked two annotators of sub-tree alignment to estimate the rule quality based on the syntactic correspondence and adequacy of frontier node sequence between the two languages sides. A rule was labeled as "good" only if both judges considered it to be of good quality. From the figure, we see that eight of the top-10 rules extracted using our approach but absent in the WordAlign-2 grammar are good rules. In contrast, only four of the top-10 rules in the baseline model are of good quality in a sense of human preference. Furthermore, we examined the top-100 most probable rules that appear in the two grammars individually. Again, the top-100 rules extracted using our proposed model are of better quality. It results in 61 good rules. By contrast, only 44% of the top-ranking rules induced using the WordAlign-2 alignment are good translation rules.

5.3.3 Evaluation of Translations

We also evaluated the translations generated by the MT system for different sub-tree alignment approaches. Since the VB-based training shows the best performance in the previous experiments, we chose it as a default setting of our approach in the following experiments. Table 4 shows the evaluation result where the translation quality is estimated using case-insensitive IBM-version BLEU4 (Papineni, Roukos, Ward, & Zhu, 2002) and TER (Snover,

The top-10 highest probability rules (for MT) in our approach but absent in WordAlign-2	
1*	NP(DNP$_1$ NN(提高)) → VP(ADVP$_1$ VP(VBD(improved)))
2*	NP(PU(“) NP(CD(两) NN(会)) PU(”)) → NP(DT(the) CD(two) NNS(sessions))
3*	NP(NP(QP$_1$ NP(NN(代表))) NP$_2$) → NP(X$_2$ NP(CD$_1$ NP(NNS(represents))))
4	NP(NN(渴望) NP$_1$) → VP(VB(desire) NNP$_1$)
5*	NP(PU(“) NP(NR$_1$ NN(独立)) PU(”)) → NP(”(”) NP(NNP$_1$ NN(independence)) ”(”))
6*	NP(VP$_1$ DEC(的) NP(NN(思想) NN(斗争))) → NP(ADJP(ADJP$_1$ JJ(ideological)) NN(struggle))
7*	NP(NP(QP$_1$ NP$_2$) NP(ADJP(JJ(重要)) NP(NN(思想)))) → NP(NP(DT(the) JJ(important) NN(thinking)) IN(of) SBAR(WHNP$_1$ S$_2$))
8*	NP(IP$_1$ DEG(的) NP(NN(现实) NN(意义))) → NP(ADJP(ADJP$_1$ JJ(practical)) NN(significance))
9*	NP(NP(PU(“) NP(QP$_1$ NN(代表)) PU(”)) NP(ADJP$_2$ NN(思想))) → NP(NP(DT(the) JJ$_2$ NN(idea)) PP(IN(of) NP(DT(the) CD$_1$ NNS(represents))))
10	NP(PU(“) NN(入市) PU$_1$) → VP(VBG(joining) NP(DT(the) NN$_1$))

The top-10 highest probability rules (for MT) in WordAlign-2 but absent in our approach	
1	LCP(QP$_1$ LC(以上)) → ADJP(JJ$_1$)
2*	NP(DNP$_1$ NN(变化)) → VP(ADVP$_1$ VBP(changes))
3*	NP(NN(三通) NN$_1$) → NP(CD(three) NNS(links) X$_1$)
4	NP(DNP(IP$_1$ DEC(的)) NP(NN(意义))) → NP(ADJP$_1$ NN(significance))
5	VP(ADVP$_1$ VP(VV$_2$ NP(NN(群众) NN$_3$))) → VP(ADVP$_1$ VP(VP(VV$_2$) NP(DT(the) JJ(mass) NN$_3$)))
6	IP(NP$_1$ VP(VV(换) NP(NN(和平) NN$_2$))) → NP(NP(NNS$_1$) PP(IN(for) NP$_2$))
7	NP(VP$_1$ DEG(的) NN(合作)) → ADJP(JJ$_1$)
8	NP(NP(PU(“) NT$_1$ PU(”)) NP(NN(讲话)) NR(中)) → NP(NP($\mathrm{PRP\$}$(his)) QP(CD$_1$) NN(speech))
9*	VP(VP(ADVP$_1$ VP(VV(加强) CC(和) VV(改进))) NP$_2$) → VP(ADVP$_1$ VP(VP(VB(strengthen) CC(and) VB(improve)) NP$_2$))
10*	NP(NP(PU(“) NN(台独) PU(”)) NP$_1$) → NP(NP(”(”) NP(NN(taiwan) NN(independence)) ”(”)) NNS$_1$)

Figure 7: The top-10 highest probability rules built from the proposed sub-tree alignment approach that are not in the WordAlign-2 baseline grammar, and the top-10 rules in the WordAlign-2 baseline grammar that are not obtained using the proposed sub-tree alignment approach. * = a good translation rule.

Dorr, Schwartz, Makhoul, Micciula, & Weischedel, 2005), and the significance test is performed using the bootstrap resampling method (Koehn, 2004). Moreover, the efficiency of rule extraction is reported in terms of *rule-set-size/extraction-time*. For comparison, we also report the result of rule extraction using word alignment matrices (Liu et al., 2009b) on WordAlign-1 and WordAlign-2.

Table 4 indicates that our approach outperforms the baselines by the BLEU and TER measures for both 1-best and 30-best extraction. In addition, the matrix-based method is much more efficient than the k-best method. For example, compared with 30-best extraction, extracting rules from sub-tree alignment matrices is 9 times more efficient. However, when all rules are counted as unit one in parameter estimation of the translation model, using alignment matrices does not show significant BLEU improvements or TER reductions in comparison with the 30-best counterpart (see rows marked with *unitcount*). This is because many of those additionally extracted rules are not utilized in real translation. For example, we observed that only 7.3% of the rules used in generating the final (1-best)

Entry	Dev		Test		Rule-set	Efficiency
	BLEU4[%]	TER[%]	BLEU4[%]	TER[%]	size	(rule/sec)
WordAlign-1 (1-best)	36.2	57.0	34.2	58.3	24.8M	75.4
WordAlign-2 (1-best)	36.2	56.9	34.2	58.1	25.3M	75.9
HeuristicAlgin (1-best)	35.7	57.2	33.8	58.3	22.7M	72.0
WordAlign-1 (30-best)	36.3	57.2	34.4	58.2	32.7M	3.8
WordAlign-2 (30-best)	36.4	57.0	34.6*	58.0	33.0M	3.9
HeuristicAlgin (30-best)	35.4	57.2	33.9	58.0	32.4M	3.8
WordAlign-1 (matrix)	36.9*	56.5	35.0*	57.9*	50.2M	35.8
WordAlign-2 (matrix)	36.8*	56.6	35.1*	57.7*	53.8M	37.9
Ours (1-best + unitcount)	36.7*	56.6	34.9*	57.8	27.0M	78.8
Ours (30-best + unitcount)	36.8*	56.6	35.0*	57.6*	37.4M	4.1
Ours (matrix + unitcount)	36.9*	56.3*	35.3**	57.5*	54.9M	37.7
Ours (1-best + posterior)	36.9*	56.4*	35.0*	57.4*	27.0M	78.8
Ours (30-best + posterior)	37.0*	56.2**	35.2**	57.0**	37.4M	4.1
Ours (matrix + posterior)	37.4**	55.9**	35.6**	57.1**	54.9M	37.7

Table 4: Evaluation of translations for different alignment approaches. For BLEU, higher is better. For TER, lower is better. *unitcount* means that we take each rule occurrence as unit one in parameter estimation, and *posterior* means that we use rule posterior probabilities as fractional counts in parameter estimation. * or ** = significantly better than all three of the 1-best baselines ($p < 0.05$ or 0.01).

translations were indeed extracted from the alignments that were not seen in the 30-best alignments. It thus indicates the fact that naively increasing the number of rules might not be effective for improving the translation quality.

The last three rows in Table 4 show the result of using alignment posterior probabilities in parameter estimation (i.e., the method described in Section 4.3). We see that alignment posterior probabilities are very helpful in improving translation quality because the system can weight more on those rules with more confidence (entries with *unitcount* vs. entries with *posterior*). By using sub-tree alignment matrices in rule extraction and alignment posterior probabilities in parameter estimation, our approach finally achieves a +1.0 BLEU improvement and a -0.9 TER reduction over the 30-best case of the baselines. It even outperforms the word alignment matrix-based counterpart by +0.5 BLEU points and -0.6 TER points (both are significant at $p < 0.05$).

Further, the effectiveness of the proposed approach is demonstrated in terms of BLEU and TER scores under the same rule-set size. Figure 8 compares our approach and the baseline approaches in different numbers of unique rules extracted.[17] Clearly, in the same number of unique rules, the proposed sub-tree alignment approach leads to better translations than those of the baselines.

5.3.4 The Impact of Alignment and Grammar Quality on MT Performance

The above experiments demonstrate the effectiveness of the proposed approach in terms of different measures individually. The next natural question is how sub-tree alignment

17. To do this, we adjusted P_{min} to obtain grammars with different sizes for our approach. For other approaches, we used different k-best lists for rule extraction.

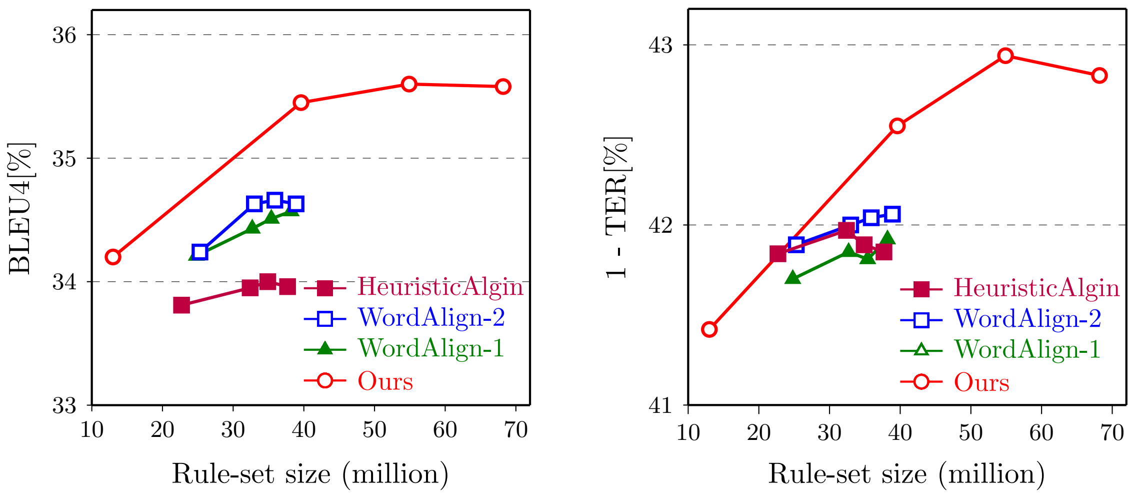


Figure 8: BLEU and 1-TER against rule-set size

and rule extraction affect the translation quality. The study of this issue is very important when we optimize the upstream systems of MT decoding and select appropriate evaluation metrics for a good prediction of MT performance.

We therefore carried out another set of experiments which compares the translation quality in different sub-tree alignment and rule extraction settings. To generate diverse sub-tree alignment and rule extraction results, we varied the values of ω and P_{min} for sub-tree alignment and rule extraction respectively. In this way, we obtained ensembles of sub-tree alignments and grammars with different precision and recall scores.[18] We chose F-β score as the evaluation metric for both the sub-tree alignment system and the rule extraction system. Instead of fixing β to be 1, we varied the β value from 0.5 to 3. Since the parameter β can control the bias towards precision or recall, choosing different values for β is very helpful in seeking a good tradeoff between precision and recall. Then we can find an appropriate evaluation measure of sub-tree alignment and rule extraction for predicting MT performance well.

Figures 9 and 10 plot the F-β scores as measures of MT performance for sub-tree alignment and rule extraction. From Figure 10, we see that the rule F-3 score correlates best with the translation quality measures, which indicates that the MT system prefers rule recall-biased metrics. This agrees with our observation in Figure 8 that the MT system can make benefits from more rules. On the other hand, the curves in Figure 9 show a better correlation between sub-tree alignment F-2/F-3 score and translation quality measures, implying a preference for relatively higher sub-tree alignment recall. This result is reasonable because in our framework more node alignment links can result in more aligned tree-fragments (or rules) extracted. A high-recall sub-tree alignment generally results in a big grammar with high rule recall, and thus better BLEU and TER results. We also com-

18. For example, a larger value of ω generally results in higher alignment precision, while a small value prefers higher alignment recall. For rule extraction, a larger value of P_{min} generally leads to a grammar with higher rule precision, while choosing a smaller P_{min} can generate a grammar with higher rule recall.

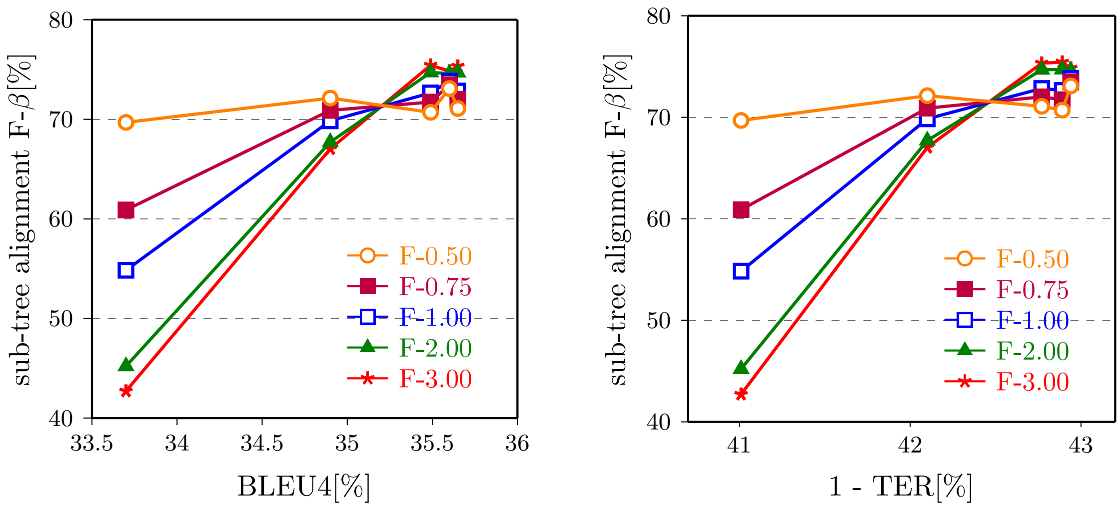


Figure 9: BLEU and 1-TER against sub-tree alignment F-β measure

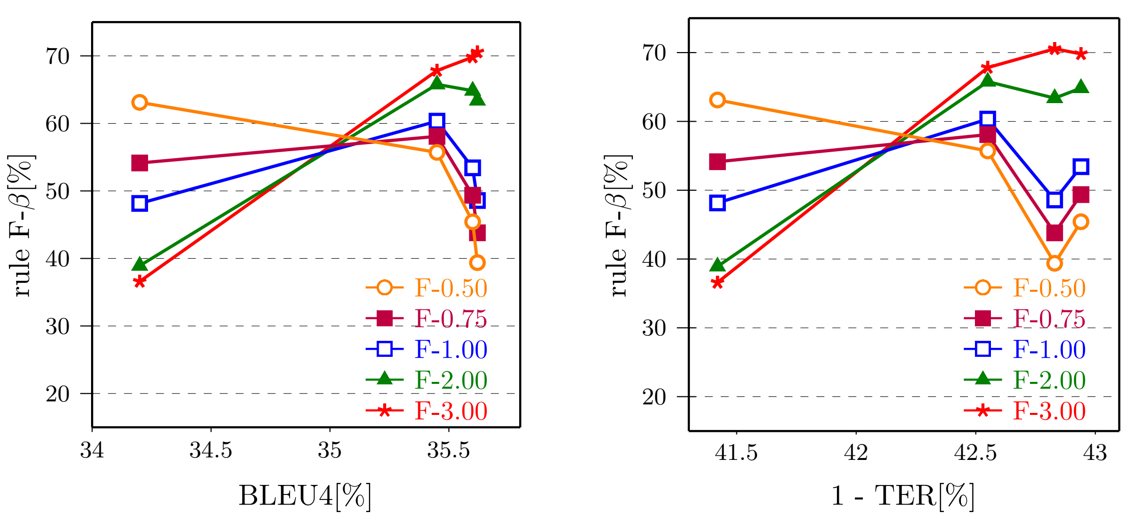


Figure 10: BLEU and 1-TER against rule F-β measure

puted the Pearson's correlation coefficients between sub-tree alignment/rule F-3 score and BLEU/TER. For sub-tree alignment F-3, its correlation coefficients with BLEU and TER are 0.971 and -0.962 respectively. For rule F-3, its correlation coefficients with BLEU and TER are 0.983 and -0.963 respectively. Both of them show good correlations with the translation quality measures. Another interesting observation here is that the MT performance is more sensitive to the change of rule F-β score than to the change of sub-tree alignment F-β score. This may lie in that rule extraction is a direct upstream step of decoding and impacts more on the output of MT systems. In contrast, sub-tree alignment is a front-end step of the MT pipeline and has an indirect effect on the actual translation process.

System		Dev		Test	
		BLEU4[%]	TER[%]	BLEU4[%]	TER[%]
Hierarchical phrase-based		37.2	57.3	35.2	58.0
Tree	1-best word alignment (WordAlign-2)	36.8	56.4*	35.3	57.7
to	Word alignment matrix	37.0	56.3*	35.2	57.5*
tree	Sub-tree alignment matrix	37.9*	55.8**	36.0**	56.8**

Table 5: MT Evaluation results of rules obtained from various alignment approaches. For BLEU, higher is better. For TER, lower is better. * or ** = significantly better than the hierarchical phrase-based baseline ($p < 0.05$ or 0.01).

5.3.5 Further Improvements

Previous work has pointed out that the straightforward implementation of tree-to-tree MT suffers from the problems of too few rules and too few derivations in either rule extraction or decoding process (Chiang, 2010). To further advance the tree-to-tree system and compare it with the state-of-the-art, we employed tree binarization (Wang et al., 2007b) and fuzzy decoding (Chiang, 2010) in our system. As our alignment approach can be equipped with the general framework of tree-to-tree translation, it is trivial to conduct another set of experiments to investigate the effectiveness of our approach in a stronger system.[19] Table 5 shows the BLEU and TER scores of our system enhanced with the above methods.[20] For comparison, we also report the result of a state-of-the-art MT system that implements the hierarchical phrase-based model (Chiang, 2007) and our tree-to-tree system that extracts rules using the word alignment matrices (Liu et al., 2009b). Table 5 indicates the superiority of our approach when tree binarization and fuzzy decoding are involved. It significantly outperforms the hierarchical phrase-based system (+0.7 BLEU points and -1.2 TER points) and the tree-to-tree system based on the word alignment matrices (+0.8 BLEU points and -0.7 TER points).

As discussed in Section 4, the transfer rules in our sub-tree alignment model resembles the general form of STSGs which can be directly used in MT. Instead of resorting to an explicit step of rule extraction, we can use the rules in the sub-tree alignment model for MT decoding, i.e., sub-tree alignment is cast as a grammar induction step. We therefore built another system which directly acquires translation rules from the sub-tree alignment step. To do this, we only need to output all the rules from the derivation forest generated by our alignment model. All rule probabilities can be obtained using the inside and output probabilities, and pruning is performed by throwing away rules whose probability is below P_{min}. In addition to rule probability, we reused the n-gram language model, the rule number bonus, the target length bonus, the lexicalized rule and low frequency rule indicators in our base tree-to-tree system as additional features for a fair comparison. To obtain a good and reasonable result, we employed both fuzzy decoding and tree binarizaiton in the experiment.

19. We did not choose this setting in the previous experiments because our gold-standard alignment annotation was on the Penn Treebank-style trees only. It was difficult to evaluate the alignment and grammar quality on binarized trees due to the lack of benchmark data. In our experiments we first conducted the experiments on each individual tasks (see Sections 5.3.1-5.3.3), and studied their correlations in a simple but reasonable setting for a consistent result of sub-tree alignment and MT (see Section 5.3.4). Then we investigated the effectiveness of our approach on an advanced tree-to-tree system (see Section 5.3.5).

20. In our implementation all parse trees were binarized in a head-out fashion.

Entry	Dev		Test	
	BLEU4[%]	TER[%]	BLEU4[%]	TER[%]
Baseline (explicit rule extraction)	37.9	55.8	36.0	56.8
Rules from the sub-tree alignment model	36.2	57.9	34.3	58.8
Rules from the sub-tree alignment model + MERT	36.7	57.3	34.9	58.0
Baseline + sub-tree alignment features	38.2	55.8	36.1	56.9

Table 6: MT Evaluation results of obtaining rules from the sub-tree alignment model and obtaining rules with the traditional rule extraction pipeline

Table 6 compares the results of sub-tree alignment matrix-based rule extraction and inducing rules from the alignment model (Row 1 vs. Row 2). Unfortunately, straightforwardly inferring rules and their probabilities from the sub-tree alignment model underperforms the baseline. This might be attributed to several reasons. First, due to the large derivation space, we cannot enumerate all relatively large tree-fragments in the sub-tree alignment step, instead we only access the tree-fragments with limited depths. By contrast, our baseline system extracts basic rules using sub-tree alignment matrices and then obtains more large rules with some heuristics (e.g., rule composing). The additional rules obtained in our baseline framework of rule extraction are in general very useful to modern syntax-based systems (Galley et al., 2006; Marcu et al., 2006; DeNeefe et al., 2007). Second, the rule probability in our sub-tree alignment model is defined as a product of probability factors for a good generation story. However, MT systems usually use features which are not required to form a generative model, such as the features shown in Equations (33)-(37). In consequence, many well-developed features can be used by the baseline system but they are not available in our sub-tree alignment model. Third, our sub-tree alignment model is trained by maximizing the likelihood or other criteria, which is not consistent with that adopted by the MT system (i.e., minimizing an evaluation-related error rate function). To further study these issues, we improved the system in two ways. First, we treated all four of the probability factors in the sub-tree alignment model (See Equation (13)) as different features in the MT decoder, and tuned their weights using MERT. Row 3 in Table 6 shows that this method achieves better results than the system employing unweighted probability factors. However, its performance is still worse than that of the baseline, which indicates that the MT-oriented features and rule extraction heuristics are crucial to the success of our tree-to-tree system. Finally we added the probabilistic factors of our sub-tree alignment model to the baseline system as additional features. As shown in the last row of Table 6, the enhanced system yields modest BLEU improvements over the baseline, but no TER improvement is achieved. All the above results give us two interesting messages - 1) rule extraction heuristics, MT-oriented features and objectives of learning are key factors contributing to a good tree-to-tree system; 2) and it is better to use our sub-tree alignment model as an upstream module of rule extraction and decoding, rather than using it as a simple step for grammar induction.

The last issue we investigate here is whether our sub-tree alignment model can make benefits from labeled data. Although we focus on unsupervised learning in this work, the proposed model does not require a strictly unsupervised condition. Instead it can be enhanced with the use of labeled data. The idea is simple: we combine the probability factors of our sub-tree alignment model in a log-linear weighted fashion. It means that

Entry	Dev		Test	
	BLEU4[%]	TER[%]	BLEU4[%]	TER[%]
Unweighted	37.9	55.8	36.0	56.8
Weighted (weights are learned on labeled data)	38.3	55.6	36.1	56.9

Table 7: Comparison of the unweighted and weighted sub-tree alignment models

all probability factors of our sub-tree alignment model are taken as real valued feature functions, and their feature weights can be learned on labeled data by supervised methods. In this way, our unweighted generative model (i.e., each factor has a weight of one) is transformed into a weighted model (i.e., each factor has an individual weight). Note that this weighted model has almost the same form as those used in SMT systems. The only difference from the SMT model is that no language model is needed because the target-language side is fixed in the sub-tree alignment step. To avoid bias towards too few or too many rules, we also added the rule number as an additional feature to our new model. For training and test, we divided our node-aligned gold-standard data into two parts - the first 310 sentences were selected for weight training, and the remaining 327 sentences were selected for testing the system. To learn feature weights in a supervised manner, we chose MERT which is one of the most powerful tools for training log-linear models. The error function used in MERT is defined by one minus sub-tree alignment F-1 score.

On the 327-sentence test data (with the tree annotation of the Penn Treebanks), the weighted model achieves an alignment F-1 score of 75.4% and a rule F-1 score of 60.0%, respectively. This result is better than that of the unweighed (and unsupervised) model which obtains an alignment F-1 score of 72.4% and a rule F-1 score of 59.2% on the same data set. Finally we tested the MT performance in our best setting (i.e., sub-tree alignment matrix-based rule extraction + tree binarization + fuzzy decoding).[21] Table 7 shows that the weighted sub-tree alignment model leads to a better BLEU score on the tuning set but does not show promising improvements on the test data. As the size of our labeled corpus is small, we expect better results if more labeled data is available. Also it is worth studying more sophisticated supervised methods to learn better weights, such as the kernel-based methods (Sun et al., 2010b). As supervised/semi-supervised learning is not the focus of this work, we leave these interesting issues to future investigations.

6. Related Work

Syntax-based approaches have been widely adopted in machine translation over the last ten years. Many successful syntactic MT systems have been developed and shown good results on several translation tasks (Eisner, 2003; Galley et al., 2004, 2006; Liu et al., 2006; Huang et al., 2006; Zhang et al., 2008; Liu et al., 2009a; Chiang, 2010). Despite differences in modeling and implementation details, all these models require an "alignment" step to acquire the syntactic correspondence between the source and target languages. As is standard in SMT, most syntax-based MT systems use word alignments to infer syntactic alignments from string-tree/tree-tree pairs. However, word alignments are generally not of good quality from the viewpoint of syntactic alignment. It makes more sense to directly

21. As we did not have sub-tree-aligned data for binarized trees, we reused the weights learned on the Penn Treebank-style trees for all four of the probability factors in our sub-tree alignment model.

induce sub-tree level alignments from pairs of sentences with the syntactic information on either language side or both. This is especially true for tree-to-tree MT where what we actually need is the alignment between sub-trees in two languages, rather than the surface alignment of words. There are several lines of work that address the syntactic alignment problem and make better use of various alignment results for tree-to-tree translation.

6.1 Word and Sub-tree Alignment for Machine Translation

The earliest efforts in syntactic alignment focus on enhancing word alignment models with syntactic information. To date, several research groups (Fraser & Marcu, 2007; DeNero & Klein, 2007; May & Knight, 2007; Fossum et al., 2008; Haghighi, Blitzer, DeNero, & Klein, 2009; Burkett, Blitzer, & Klein, 2010; Riesa, Irvine, & Marcu, 2011) have proposed syntax-augmented models to advance their word alignment systems. Although these models achieved promising improvements, they still address the alignment problem in word level. As discussed in Section 1, such methods might not be desirable choices for learning the correspondence between tree nodes in two languages. As an alternative and more straightforward solution, researchers tried to infer sub-tree level alignments for pairs of syntactic trees. For example, Imamura (2001), Groves, Hearne, and Way (2004), and Tinsley et al. (2007) defined several scoring functions to measure the similarity between the source and target sub-trees, and aligned the tree nodes with greedy algorithms. Their approaches, though simple to implement, are not derived in a principled way. For example, all these models do not have an explicit optimization procedure, as in a general framework of statistical learning. Instead, the model parameters are obtained using additional alignment models or lexicons. In another line of research, Sun et al. (2010a, 2010b) attempted to address the sub-tree alignment problem with supervised/semi-supervised models. They used tree kernels and various syntactic features to advance their sub-tree alignment system and showed promising results on Chinese-English translation tasks. However, this approach still relies on heuristic algorithms for inferring node correspondences between two parse trees. Beyond this, to train the tree kernels, their approach requires additional labeled data which is generally very expensive to build. Unlike these studies, we derive our sub-tree model in a principled way and develop an unsupervised sub-tree alignment framework for tree-to-tree MT.

6.2 Unsupervised Syntactic Alignment

There are also previous studies that do not resort to labeled data for sub-tree alignment. The earliest of these was Eisner's (2003) work. He designed an unsupervised approach to modeling the sub-tree alignment problem in the STSG formalism. However, since no detailed derivation and model decomposition is provided, this model is computationally expensive, even difficult to be applied to current tree-to-tree systems where complex tree structures are involved. Gildea (2003) also applied STSGs to tree-to-tree/tree-to-string alignment. He developed a "loosely" tree-based alignment method to address the issue of parse-tree isomorphism in bitext. But his work only targets the word alignment rather than modern syntactic MT systems. Recently Nakazawa and Kurohashi (2011) proposed a Bayesian approach for sub-tree alignment between dependency trees, and tested it in a Japanese-English MT system. Actually their model has much in common with the model presented in this work. For example, they both apply unsupervised learning methods and Bayesian models to sub-tree alignment. On the other hand, the two studies differ in some important aspects. First, Nakazawa and Kurohashi (2011) restricted themselves to sub-tree alignment between dependency trees, which is very different from aligning tree nodes in phrase structure trees. Since phrase structure trees involve more complex structures and syntactic categories, the alignment problem in phrase structure trees are relatively more difficult than the dependency-based counterpart. Second, our model makes benefits from the recent advances in STSGs and is directly applicable to current state-of-the-art tree-to-tree systems.

Another related work to what is presented here is Pauls, Klein, Chiang, and Knight's (2010) work. They factored a node-to-string alignment model over components that each generates a target side of a synchronous rule from a source side. Moreover, the probability of a rule fragment was factored into a lexical and structural component in their work. Actually, their model and our proposed model are two variants on a theme. But there appear to be obvious differences between them. First, our focus is sub-tree alignment for tree-to-tree translation, while Pauls et al. (2010) addressed the alignment issue for tree-to-string/string-to-tree translation. In our model, we parse both language sides independently, rather than parsing only one side and projecting syntactic categories. As a result, inference is faster in this work since we do not need to consider all possible parse trees of the unparsed side during alignment. Second, the permutation model presented in this work is more general in order to handle non-ITG trees. Third, we investigate methods for the effective use of sub-tree alignment in MT. In particular, we present a rule extraction approach to obtaining additional translation rules using sub-tree alignment posteriors, rather than learning rules only from the 1-best sub-tree alignment.

6.3 Rule Extraction Using Various Alignment Results

In machine translation, word and syntactic alignments are used to extract translation rules or phrases. In the traditional pipeline of rule and phrase extraction, only the 1-best alignment result is considered, which suffers from the limited scope of the single alignment. To efficiently obtain diverse alignment/parsing results, packed data structures were adopted to improve 1-best pipeline MT systems in recent years (Mi & Huang, 2008; Liu et al., 2009a; Zhang, Zhang, Li, Aw, & Tan, 2009). For example, Liu et al. (2009b) and de Gispert et al. (2010) used alignment posterior probabilities for phrase or hierarchical phrase extraction. The development of sub-tree alignment matrices is actually motivated by a similar idea from word alignment matrices. The difference from the above work is that we use sub-tree on both language sides to infer alignment posterior probabilities, while these probabilities are calculated in word/phrase-level in previous work (Liu et al., 2009b; de Gispert et al., 2010). Moreover, to our knowledge, the effectiveness of sub-tree alignment matrix has not been systematically studied in the case of tree-to-tree translation.

Note that the approach presented in this work is also doing something similar to synchronous grammar induction. For example, our model results in an STSG which has the same formalism as that used in MT. Recent studies on Bayesian models (Blunsom, Cohn, Dyer, & Osborne, 2009; Cohn & Blunsom, 2009; Levenberg et al., 2012) have shown very promising results in directly learning synchronous grammars from bilingual data for hierarchical phrase-based and string-to-tree systems, rather than extracting synchronous grammar rules based on an explicit word/syntactic alignment step. However it is rare to see related work on tree-to-tree MT. In principle this article is different from previous work on synchronous grammar induction. For example, the aim of this work is to learn a sub-tree alignment model, which can be applied to many potential applications except MT, such as sentence compression for paraphrasing and test summarization (Jing, 2000; Cohn & Lapata, 2009). Unlike synchronous grammar induction where the alignment is implicitly encoded in the learning process, we treat sub-tree alignment as a separate task. This eases the development and tuning of the alignment system because we actually do not resort to MT systems which are "slow" and difficult to optimize. Another advantage is that our approach can make benefits from more compact models, rather than those used in MT where a great number of rules are involved. Take our implementation as instance. The alignment model is learned on a relatively small set of grammar rules (rules with limited depths), while the MT system accesses a much larger grammar where many additional rules are involved by rule composing. Such a method can result in an efficient alignment system and is likely to alleviate the degenerate analysis of the data, with no cost of degrading in MT performance.

7. Discussion

The underlying assumption of our proposed model is that 1-to-1 sub-tree alignments can be achieved based on the constraints imposed by neighboring parts of the tree (see Section 2). This makes sense in the standpoint of linguistically-motivated models, yet it in turn faces a problem that the constraints make it difficult to align some sentences/trees correctly, particularly if they are free translations. There are several reasons that can explain why our approach works nice with tree-to-tree MT and why it does not suffer greatly from the structure divergence between languages. First, our model is "flexible" and allows for node deletion/insertion in alignment. It means that in some levels of the tree it is not necessary to require that every node is aligned to a valid node in the other language side, instead nodes can be "dropped" as needed. The advantage of such a method is that if we cannot confidently align a node to any node in the counterpart tree, we can align it to a virtual node and do not enforce bad constraints to aligning other parts of the tree. This is very useful when there are very flat tree structures or partial translations which are not syntactically well-formed. Second, the main purpose of our approach is to infer sub-tree alignment probabilities which can be used in pruning sub-tree alignment matrices and extracting rules for MT systems. Though 1-to-1 alignment is required for training the sub-tree alignment model, what we actually do is to access a large number of alignment alternatives for rule extraction, even if some of them cannot appear in the same derivation due to our alignment constraints. Third, our model can work with any phrase structure trees. Instead of the Penn Treebank-style trees which are difficult for alignment in some cases, our sub-tree alignment system works well with binarized trees and shows promising improvements over various baselines. Note that tree binarization is a very effective method to alleviate the structure divergence problem, especially for Chinese-English translation. Also, it might be interesting to investigate other methods for dealing with differences in syntactic structures between languages, such as the forest-based methods (Mi & Huang, 2008; Liu et al., 2009a).

Another note on our approach. If implemented naively, the speed of the sub-tree alignment system will be very slow since our model needs the calculation of the alignment probability over all pairs of tree-fragments. Fortunately, with the thought of computation, several optimizations can make the system much more efficient in practice. First, as is described in this work, pruning methods can be employed to restrict the number of tree-fragments to a reasonable level. In our experiments, the system with pruning achieved a speed of 1.8∼2.2 sentence/second on a single core of the Intel Xeon 3.16-GHz CPU. Another way for speed improvement is parallel processing. A very good property of the EM-style algorithms is that the E-step can be easily implemented in parallel computation environment. What we need is to divide the training data set into a number of "smaller" parts, and then run the inside-outside algorithm on these parts in parallel (i.e., a Map procedure). The expected counts of model parameters can be accumulated over the results of all the parts (i.e., a Reduce procedure). Then the M-step can be performed as usual. In our implementation we used 40 threads for parallel training. The running time of one training iteration over our 1-million-sentence corpus was about 14-17 hours. Note that further system speed-up can be expected if more powerful distributed infrastructures are available (e.g., clusters + Hadoop), and it is not difficult to scale up our approach to handle millions of sentence pairs using the current training framework.

8. Conclusions

We have proposed an unsupervised probabilistic sub-tree alignment approach for tree-to-tree translation. By factoring the alignment model over several components, the resulting model can be easily learned using the EM algorithm and the variational Bayesian approach. Also, we have investigated different ways of applying the proposed model to tree-to-tree translation. In particular, we developed a sub-tree alignment matrix which encodes an exponentially large number of alignments. From this representation of sub-tree alignment, desirable rules can be extracted more efficiently than using the k-best sub-tree alignment result. Our experiments showed that the proposed model achieved significant improvements in both alignment quality and grammar quality over several baselines. On the NIST Chinese-English evaluation corpora, it achieved a +1.0 BLEU improvement and a -0.9 TER reduction on top of a state-of-the-art tree-to-tree system. The improved MT system even significantly outperformed a state-of-the-art hierarchical phrase-based system when equipped with tree binarization and fuzzy decoding.

Acknowledgments

This work was supported in part by the National Science Foundation of China (Grants 61073140 and 61272376), the Natural Science Foundation for the Youth of China (Grant 61300097), the China Postdoctoral Science Foundation (Grant 2013M530131), the Specialized Research Fund for the Doctoral Program of Higher Education (Grant 20100042110031), and the Fundamental Research Funds for the Central Universities (Grant N100204002). The authors would like to thank the anonymous reviewers for their pertinent and insightful comments, Keh-Yih Su for his great help in improving the early version of this article, Ji Ma for helpful discussions, and Chunliang Zhang and Tongran Liu for language refinement. The corresponding author of this article is Jingbo Zhu.

Appendix A. Part-of-speech Tags and Phrase Structure Labels

In this work the annotation of POS tagging and phrase structure parsing follows the standard defined in the Penn English and Chinese Treebanks (Marcus et al., 1993; Xue et al., 2005). See Tables 8-11 for lists of POS tags and constituent labels used in the example trees in this article.

POS Tag	Description
AD	Adverb
AS	Aspect Particle
NN	Noun (except proper nouns and temporal nouns)
NR	Proper Noun
P	Preposition
PN	Pronoun
PU	Punctuation
VV	Verb (except stative verbs, copulas, and the main verbs of ”有”, ”没” and ”无”)

Table 8: Chinese POS tags used in the examples

POS Tag	Description
DT	Determiner
IN	Preposition
JJ	Adjective
NNP	Proper noun (singular)
NNS	Noun (plural)
PRP	Personal Pronoun
RB	Adverb
VBD	Verb (past tense)
VBN	Verb (past participle)
VBP	Verb (non-3rd person singular present)
,	Comma
.	Period

Table 9: English POS tags used in the examples

Syntactic Label	Description
IP	Single Clause
NP	Noun Phrase
PP	Preposition Phrase
QP	Quantity Phrase
VP	Verb Phrase

Table 10: Chinese constituent labels used in the examples

Syntactic Label	Description
ADVP	Adverb Phrase
NP	Noun Phrase
PP	Preposition Phrase
S	Sentence
VP	Verb Phrase

Table 11: English constituent labels used in the examples

Distribution	Notation	Description
$\mathrm{P}_{nt}(\cdot)$	$\theta_{t_{nt}\|s_{nt}}$	s_{nt} and t_{nt} are source and target-language non-terminal symbols
$\mathrm{P}_{tree}(\cdot)$	$\theta_{t_{tree}\|t_{nt}}$	t_{tree} is a target-language tree-fragment
$\mathrm{P}_{reorder}(\cdot)$	$\theta_{t_{vnt}\|s_{vnt}}$	s_{vnt} and t_{vnt} are vectors of non-terminal symbols in source and target-languages
$\mathrm{P}_{length}(\cdot)$	$\theta_{l\|m}$	m and l are numbers of source and target terminals (or words)
$\mathrm{P}_{w}(\cdot)$	$\theta_{t_w\|s_w}$	s_w and t_w are source and target terminals (or words)

Table 12: Notations of model parameters

Appendix B. EM-based Training for the Sub-tree Alignment Model

As described in Section 3, the proposed sub-tree alignment model has five types of parameters, including the non-terminal mapping probability $\mathrm{P}_{nt}(\cdot)$, the target-language tree-fragment generation probability $\mathrm{P}_{tree}(\cdot)$, the frontier non-terminal reordering probability $\mathrm{P}_{reorder}(\cdot)$, the word number probability $\mathrm{P}_{length}(\cdot)$ and the word mapping probability $\mathrm{P}_{w}(\cdot)$. For convenience we use a new set of notations to denote these model parameters in the following description. See Table 12 for a symbol list.

Here we follow the same framework of the EM-based training as that described in Figure 3. See Figure 11 for a complete version of the EM algorithm for all parameters of the model. In this algorithm, $EC(\cdot)$ represents the expected count of the input variable. $\delta(X = x)$ is a 0-1 function that returns 1 if variable X takes a value of x, 0 otherwise. $\gamma^{(k)}(r; u, v)$ and $\gamma^{(k)}(S, T)$ are the rule probability (see Equation (29)) and the probability of the sub-tree alignment between S and T (see Equation (20)), where k indicates that all these probabilities are calculated based on the parameters in the k-th iteration. $tree(\cdot)$, $vnt(\cdot)$ and $lex(\cdot)$ are the functions that return the tree-structure, frontier non-terminal vector, and terminal sequence of the input tree-fragment, respectively (see Section 3.2).

The basic idea of the E-step is that we check each rule r (given a pair of tree nodes u and v) and update $EC(\cdot)$ by its relative probability $\frac{\gamma(r;u,v)}{\gamma(root(S),root(T))}$. This can be applied to the update rules for the parameters $\theta_{t_{nt}|s_{nt}}$, $\theta_{t_{tree}|t_{nt}}$, $\theta_{t_{vnt}|s_{vnt}}$ and $\theta_{l|m}$ (see lines 8-11). The only exception is $\theta_{t_w|s_w}$. As defined in Equation (14), $\mathrm{P}_w(t_i \mid s_j)$ is not a direct product factor, instead we use the sum over all terminals of the source-language tree-fragment (i.e., $\sum_{j=1}^{m} \mathrm{P}_w(t_i \mid s_j)$). Here we follow the result of IBM Model 1 and make the update magnitude proportional to $\mathrm{P}_w(t_i \mid s_j)/\sum_{j'=1}^{|lex(s_r)|} \mathrm{P}_w(t_i \mid s_{j'})$. We refer the reader to Brown et al.'s (1993) work for detailed derivation of the expected count in IBM Model 1. It is also worth noting that the above algorithm performs parameter update based on different choices of (u, v) and r in the E-step. This means that if a rule instance is involved in a particular derivation for more than one time (e.g., the same tree-fragment appears at

1: **Function** $\textsc{TrainModelWithEM}$ $(\{(S_1, T_1), ..., (S_n, T_n)\})$
2: **Initialize** $\{\theta^{(0)}_{t_{nt}|s_{nt}}, \theta^{(0)}_{t_{tree}|t_{nt}}, \theta^{(0)}_{t_{vnt}|s_{vnt}}, \theta^{(0)}_{l|m}, \theta^{(0)}_{t_w|s_w}\}$
3: **For** $k = 0$ to $K - 1$ **do**
4: **Set** $EC(\cdot) = 0$ for all model parameters
E-step:
5: **Foreach** tree pair (S, T) in sequence $\{(S_1, T_1), ..., (S_n, T_n)\}$ **do**
6: **Foreach** node pair (u, v) in (S, T) **do**
7: **Foreach** rule r rooted at (u, v) **do**
8: $EC(\theta_{t_{nt}|s_{nt}}) \mathrel{+}= \frac{\gamma^{(k)}(r;u,v)\cdot\delta(s_{nt}=u)\cdot\delta(t_{nt}=v)}{\gamma^{(k)}(root(S),root(T))}$
9: $EC(\theta_{t_{tree}|t_{nt}}) \mathrel{+}= \frac{\gamma^{(k)}(r;u,v)\cdot\delta(t_{nt}=v)\cdot\delta(t_{tree}=tree(t_r))}{\gamma^{(k)}(root(S),root(T))}$
10: $EC(\theta_{t_{vnt}|s_{vnt}}) \mathrel{+}= \frac{\gamma^{(k)}(r;u,v)\cdot\delta(s_{vnt}=vnt(s_r))\cdot\delta(t_{vnt}=vnt(t_r))}{\gamma^{(k)}(root(S),root(T))}$
11: $EC(\theta_{l|m}) \mathrel{+}= \frac{\gamma^{(k)}(r;u,v)\cdot\delta(m=|lex(s_r)|)\cdot\delta(l=|lex(t_r)|)}{\gamma^{(k)}(root(S),root(T))}$
12: **Foreach** word pair (s_j, t_i) in position (j, i) of $(lex(s_r), lex_(t_r))$ **do**
13: $EC(\theta_{t_w|s_w}) \mathrel{+}= \frac{\gamma^{(k)}(r;u,v)\cdot\frac{\mathrm{P}^{(k)}_w(t_i|s_j)}{\sum_{j'=1}^{|lex(s_r)|}\mathrm{P}^{(k)}_w(t_i|s_{j'})}\cdot\delta(s_w=s_j)\cdot\delta(t_w=t_i)}{\gamma^{(k)}(root(S),root(T))}$
M-step:
14: **Foreach** non-terminal symbol pair (s_{nt}, t_{nt}) **do**
15: $\theta^{(k+1)}_{t_{nt}|s_{nt}} = \frac{EC(\theta_{t_{nt}|s_{nt}})}{\sum_{t'_{nt}} EC\left(\theta_{t'_{nt}|s_{nt}}\right)}$
16: **Foreach** target non-terminal symbol t_{nt} and tree-fragment structure t_{tree} **do**
17: $\theta^{(k+1)}_{t_{tree}|t_{nt}} = \frac{EC(\theta_{t_{tree}|t_{nt}})}{\sum_{t'_{tree}} EC\left(\theta_{t'_{tree}|t_{nt}}\right)}$
18: **Foreach** pair of non-terminal symbol vectors (s_{vnt}, t_{vnt}) **do**
19: $\theta^{(k+1)}_{t_{vnt}|s_{vnt}} = \frac{EC(\theta_{t_{vnt}|s_{vnt}})}{\sum_{t'_{vnt}} EC\left(\theta_{t'_{vnt}|s_{vnt}}\right)}$
20: **Foreach** pair of word numbers (m, l) **do**
21: $\theta^{(k+1)}_{l|m} = \frac{EC(\theta_{l|m})}{\sum_{l'} EC\left(\theta_{l'|m}\right)}$
22: **Foreach** pair of words (s_w, t_w) **do**
23: $\theta^{(k+1)}_{t_w|s_w} = \frac{EC(\theta_{t_w|s_w})}{\sum_{t'_w} EC\left(\theta_{t'_w|s_w}\right)}$
24: **return** $\{\theta^{(K)}_{t_{nt}|s_{nt}}, \theta^{(K)}_{t_{tree}|t_{nt}}, \theta^{(K)}_{t_{vnt}|s_{vnt}}, \theta^{(K)}_{l|m}, \theta^{(K)}_{t_w|s_w}\}$

Figure 11: The EM-based training algorithm for all model parameters

different positions), the update of the corresponding parameters would be carried out for multiple times.

Another note on the EM algorithm. The expected counts of all the parameters can be efficiently calculated using the inside and outside probabilities according to lines 8-11 and

13. But for some parameters there are more efficient ways. For example, as is discussed in Section 3.4.1, the expected count of $\theta_{t_{nt}|s_{nt}}$ can be obtained without checking each individual rule, that is, we can omit the loop for r in this case. This technique can be considered for further speed-up of the sub-tree alignment system.

References

Attias, H. (2000). A variational bayesian framework for graphical models. In Solla, S. A., Leen, T. K., & K., M. (Eds.), *Advances in Neural Information Processing Systems 12*, pp. 209–215. MIT Press.

Beal, M. J. (2003). Variational algorithms for approximate bayesian inference. Master's thesis, University College London.

Blunsom, P., Cohn, T., Dyer, C., & Osborne, M. (2009). A gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pp. 782–790, Suntec, Singapore.

Brown, P. E., Pietra, S. A. D., Pietra, V. J. D., & Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, *19*, 263–311.

Burkett, D., Blitzer, J., & Klein, D. (2010). Joint parsing and alignment with weakly synchronized grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT:NAACL)*, pp. 127–135, Los Angeles, California, USA.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 263–270, Ann Arbor, Michigan, USA.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, *33*, 45–60.

Chiang, D. (2010). Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1443–1452, Uppsala, Sweden.

Chiang, D., & Knight, K. (2006). An introduction to synchronous grammars. In *Tutorials of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*.

Chiswell, I., & Hodges, W. (2007). *Mathematical Logic*. Oxford University Press.

Cohn, T., & Blunsom, P. (2009). A Bayesian model of syntax-directed tree to string grammar induction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 352–361, Singapore.

Cohn, T., & Lapata, M. (2009). Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, *34*, 637–674.

Das, D., & Smith, N. A. (2009). Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the*

ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP), pp. 468–476, Suntec, Singapore.

de Gispert, A., Pino, J., & Byrne, W. (2010). Hierarchical phrase-based translation grammars extracted from alignment posterior probabilities. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 545–554, Cambridge, MA, USA.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*, 1–38.

DeNeefe, S., Knight, K., Wang, W., & Marcu, D. (2007). What can syntax-based MT learn from phrase-based MT?. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 755–763, Prague, Czech Republic.

DeNero, J., Gillick, D., Zhang, J., & Klein, D. (2006). Why generative phrase models underperform surface heuristics. In *Proceedings on the Workshop on Statistical Machine Translation (WMT)*, pp. 31–38, New York city, USA.

DeNero, J., & Klein, D. (2007). Tailoring word alignments to syntactic machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pp. 17–24, Prague, Czech Republic.

Eisner, J. (2003). Learning non-isomorphic tree mappings for machine translation. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 205–208, Sapporo, Japan.

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, *1*, 209–230.

Fossum, V., Knight, K., & Abney, S. (2008). Using syntax to improve word alignment precision for syntax-based machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation (WMT)*, pp. 44–52, Columbus, Ohio, USA.

Fraser, A., & Marcu, D. (2007). Getting the structure right for word alignment: LEAF. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 51–60, Prague, Czech Republic.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., & Thayer, I. (2006). Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pp. 961–968, Sydney, Australia.

Galley, M., Hopkins, M., Knight, K., & Marcu, D. (2004). What's in a translation rule?. In Susan Dumais, D. M., & Roukos, S. (Eds.), *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT:NAACL)*, pp. 273–280, Boston, Massachusetts, USA.

Gildea, D. (2003). Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 80–87, Sapporo, Japan.

Groves, D., Hearne, M., & Way, A. (2004). Robust sub-sentential alignment of phrase-structure trees. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pp. 1072–1078, Geneva, Switzerland.

Haghighi, A., Blitzer, J., DeNero, J., & Klein, D. (2009). Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pp. 923–931, Suntec, Singapore.

Huang, L., & Chiang, D. (2005). Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology (IWPT)*, pp. 53–64, Vancouver, British Columbia, Canada.

Huang, L., Kevin, K., & Joshi, A. (2006). Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA)*, pp. 66–73, Cambridge, Massachusetts, USA.

Imamura, K. (2001). Hierarchical phrase alignment harmonized with parsing. In *Proceedings of the 6th NLP Pacific Rim Symposium*, pp. 377–384.

Jing, H. (2000). Sentence reduction for automatic text summarization. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pp. 310–315.

Johnson, M. (2007). Why doesn't EM find good HMM POS-taggers?. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 296–305, Prague, Czech Republic.

Knuth, D. (1997). *The Art of Computer Programming: Fundamental Algorithms*. Addison-Wesley.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In Lin, D., & Wu, D. (Eds.), *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 388–395, Barcelona, Spain.

Koehn, P., Och, F., & Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT:NAACL)*, pp. 48–54, Edmonton, Canada.

Levenberg, A., Dyer, C., & Blunsom, P. (2012). A bayesian model for learning scfgs with discontiguous rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 223–232, Jeju Island, Korea.

Liu, D., & Gildea, D. (2009). Bayesian learning of phrasal tree-to-string templates. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1308–1317, Singapore.

Liu, Y., Liu, Q., & Lin, S. (2006). Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pp. 609–616, Sydney, Australia.

Liu, Y., Lü, Y., & Liu, Q. (2009a). Improving tree-to-tree translation with packed forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pp. 558–566, Suntec, Singapore.

Liu, Y., Xia, T., Xiao, X., & Liu, Q. (2009b). Weighted alignment matrices for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1017–1026, Singapore.

Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

Marcu, D., Wang, W., Echihabi, A., & Knight, K. (2006). Spmt: Statistical machine translation with syntactified target language phrases. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 44–52, Sydney, Australia.

Marcu, D., & Wong, D. (2002). A phrase-based,joint probability model for statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 133–139.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, *19*, 313–330.

May, J., & Knight, K. (2007). Syntactic re-alignment models for machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 360–368, Prague, Czech Republic.

Mi, H., & Huang, L. (2008). Forest-based translation rule extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 206–214, Honolulu, Hawaii, USA.

Nakazawa, T., & Kurohashi, S. (2011). Bayesian subtree alignment model based on dependency trees. In *Proceedings of 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pp. 794–802, Chiang Mai, Thailand.

Neal, R. (1998). *Philosophy of Bayesian Inference*. http://www.cs.toronto.edu/~radford/res-bayes-ex.html.

Och, F. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 160–167, Sapporo, Japan.

Och, F., & Ney, H. (2004). The alignment template approach to statistical machine translation. *Computational Linguistics*, *30*, 417–449.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the*

Association for Computational Linguistics (ACL), pp. 311–318, Philadelphia, Pennsylvania, USA.

Pauls, A., Klein, D., Chiang, D., & Knight, K. (2010). Unsupervised syntactic alignment with inversion transduction grammars. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT:NAACL)*, pp. 118–126, Los Angeles, California, USA.

Riesa, J., Irvine, A., & Marcu, D. (2011). Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 497–507, Edinburgh, Scotland, UK.

Riley, D., & Gildea, D. (2010). Improving the performance of giza++ using variational bayes. Tech. rep., University of Rochester.

Smith, D. A., & Eisner, J. (2009). Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 822–831, Singapore.

Snover, M., Dorr, B., Schwartz, R., Makhoul, J., Micciula, L., & Weischedel, R. (2005). A Study of Translation Error Rate with Targeted Human Annotation. Tech. rep. LAMP-TR-126,CS-TR-4755,UMIACS-TR-2005-58, University of Maryland, College Park and BBN Technologies.

Sun, J., Zhang, M., & Tan, C. L. (2010a). Discriminative induction of sub-tree alignment using limited labeled data. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pp. 1047–1055, Beijing, China.

Sun, J., Zhang, M., & Tan, C. L. (2010b). Exploring syntactic structural features for sub-tree alignment using bilingual tree kernels. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 306–315, Uppsala, Sweden.

Thayer, I., Ettelaie, E., Knight, K., Marcu, D., Munteanu, D., Och, F., & Tipu, Q. (2004). The isi/usc mt system. In *Proceedings of International Workshop on Spoken Language Translation 2004*, pp. 59–60.

Tinsley, J., Zhechev, V., Hearne, M., & Way, A. (2007). Robust language pair-independent sub-tree alignment. In *Proceedings of Machine Translation Summit XI*, pp. 467–474, Copenhagen, Denmark.

Venugopal, A., Zollmann, A., Smith, N. A., & Stephan, V. (2008). Wider pipelines: n-best alignments and parses in mt training. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA)*, pp. 192–201.

Vogel, S., Ney, H., & Tillmann, C. (1996). Hmm-based word alignment in statistical translation. In *Proceedings of the 16rd International Conference on Computational Linguistics (COLING)*, pp. 836–841.

Wang, M., Smith, N. A., & Mitamura, T. (2007a). What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 22–32, Prague, Czech Republic.

Wang, W., Knight, K., & Marcu, D. (2007b). Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 746–754, Prague, Czech Republic.

Woodsend, K., & Lapata, M. (2011). Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 409–420, Edinburgh, Scotland, UK.

Xiao, T., Zhu, J., Zhang, H., & Li, Q. (2012). Niutrans: An open source toolkit for phrase-based and syntax-based machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics System Demonstrations (ACL)*, pp. 19–24, Jeju Island, Korea.

Xue, N., Xia, F., Chiou, F.-d., & Palmer, M. (2005). The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, *11*, 207–238.

Zhang, H., Zhang, M., Li, H., Aw, A., & Tan, C. L. (2009). Forest-based tree sequence to string translation model. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pp. 172–180, Suntec, Singapore.

Zhang, M., Jiang, H., Aw, A., Li, H., Tan, C. L., & Li, S. (2008). A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Techonologies (ACL:HLT)*, pp. 559–567, Columbus, Ohio, USA.