Augmenting String-to-Tree Translation Models with Fuzzy Use of Source-side Syntax

Jiajun Zhang, Feifei Zhai and Chengqing Zong
Institute of Automation, Chinese Academy of Sciences
Beijing, China
{jjzhang, ffzhai, cqzong}@nlpr.ia.ac.cn

Abstract

Due to its explicit modeling of the grammaticality of the output via target-side syntax, the string-to-tree model has been shown to be one of the most successful syntax-based translation models. However, a major limitation of this model is that it does not utilize any useful syntactic information on the source side. In this paper, we analyze the difficulties of incorporating source syntax in a string-to-tree model. We then propose a new way to use the source syntax in a fuzzy manner, both in source syntactic annotation and in rule matching. We further explore three algorithms in rule matching: 0-1 matching, likelihood matching, and deep similarity matching. Our method not only guarantees grammatical output with an explicit target tree, but also enables the system to choose the proper translation rules via fuzzy use of the source syntax. Our extensive experiments have shown significant improvements over the state-of-the-art string-to-tree system.

1 Introduction

In recent years, statistical translation models based upon linguistic syntax have shown promising progress in improving translation quality. It appears that encoding syntactic annotations on either side or both sides in translation rules can increase the expressiveness of rules and can produce more accurate translations with improved reordering.

One of the most successful syntax-based models is the string-to-tree model (Galley et al., 2006; Marcu et al., 2006; Shen et al., 2008; Chiang et al., 2009). Since it explicitly models the grammaticality of the output via target-side syntax, the string-to-tree model (Xiao et al., 2010) significantly outperforms both the state-of-the-art phrase-based system Moses (Koehn et al., 2007) and the formal syntax-based system Hiero (Chiang, 2007). However, there is a major limitation in the string-to-tree model: it does not utilize any useful source-side syntactic information, and thus to some extent lacks the ability to distinguish good translation rules from bad ones.

The source syntax is well-known to be helpful in improving translation accuracy, as shown especially by tree-to-string systems (Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006; Mi et al., 2008; Zhang et al., 2009). The tree-to-string systems are simple and efficient, but they also have a major limitation: they cannot guarantee the grammaticality of the translation output because they lack target-side syntactic constraints.

Thus a promising solution is to combine the advantages of the tree-to-string and string-to-tree approaches. A natural idea is the tree-to-tree model (Ding and Palmer, 2005; Cowan et al., 2006; Liu et al., 2009). However, as discussed by Chiang (2010), while tree-to-tree translation is indeed promising in theory, in practice it usually ends up over-constrained. Alternatively, Mi and Liu (2010) proposed to enhance the tree-to-string model with target dependency structures (as a language model). In this paper, we explore in the other direction: based on the strong string-to-tree model which builds an explicit target syntactic tree during decoding rather than apply only a syntactic language model, we aim to find a useful way to incorporate the source-side syntax.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 204–215,
Edinburgh, Scotland, UK, July 27–31, 2011.

First, we give a motivating example to show the importance of the source syntax for a string-to-tree model. Then we discuss the difficulties of integrating the source syntax into the string-to-tree model. Finally, we propose our solutions.

Figure 1 depicts a standard process that transforms a Chinese string into an English tree using several string-to-tree translation rules. The tree with solid lines is produced by the baseline string-to-tree system. Although the yield is grammatical, the translation is not correct since the system mistakenly applies rule r_2, thus translating the Chinese preposition 和(hé) in the example sentence into the English conjunction *and*. As a result, the Chinese prepositional phrase '和 恐怖 组织 网' ("with terrorist networks") is wrongly translated as a part of the relevant noun phrase ("[Hussein] and terrorists networks"). Why does this happen? We find that r_2 occurs 103316 times in our training data, while r_3 occurs only 1021 times. Thus, without source syntactic clues, the Chinese word 和(hé) is converted into the conjunction *and* in most cases. In general, this conversion is correct when the word 和(hé) is used as a conjunction. But 和(hé) is a preposition in the source sentence. If we are given this source syntactic clue, rule r_3 will be preferred. This example motivates us to provide a moderate amount of source-side syntactic information so as to obtain the correct English tree with dotted lines (as our proposed system does).

A natural question may arise that is it easy to incorporate source syntax in the string-to-tree model? To the best of our knowledge, no one has studied this approach before. In fact, it is not a trivial question if we look into the string-to-tree model. We find that the difficulties lie in at least three problems: 1) For a string-to-tree rule such as r_6 in figure 1, how should we syntactically annotate its source string? 2) Given the source-annotated string-to-tree rules, how should we match these rules according to the test source tree during decoding? 3) How should we binarize the source-annotated string-to-tree rules for efficient decoding?

For the first problem, one may require the source side of a string-to-tree rule to be a constituent. However, such excessive constraints will exclude many good string-to-tree rules whose source strings are not constituents. Inspired by Chiang (2010), we adopt a fuzzy way to label every source string with the complex syntactic categories of SAMT (Zollmann and Venugopal, 2006). This method leads to a one-to-one correspondence between the new rules and the string-to-tree rules. We will detail our fuzzy labeling method in Section 2.

For the second problem, it appears simple and intuitive to match rules by requiring a rule's source syntactic category to be the same as the category of the test string. However, this hard constraint will greatly narrow the search space during decoding. Continuing to pursue the fuzzy methodology, we adopt a fuzzy matching procedure to enable matching of all the rules whose source strings match the test string, and then determine the degree of matching between the test source tree and each rule. We will discuss three fuzzy matching algorithms, from simple to complex, in Section 3.

The third question is a technical problem, and we will give our solution in Section 4.

Our method not only guarantees the grammaticality of the output via the target tree structure, but also enables the system to choose appropriate translation rules during decoding through source syntactic fuzzy labeling and fuzzy matching.

The main contributions of this paper are as follows:

1) We propose a fuzzy method for both source syntax annotation and rule matching for augmenting string-to-tree models.
2) We design and investigate three fuzzy rule matching algorithms: 0-1 matching, likelihood matching, and deep similarity matching.

We hope that this paper will demonstrate how to effectively incorporate both source and target syntax into a translation model with promising results.

2 Rule Extraction

Since we annotate the source side of each string-to-tree rule with source parse tree information in a fuzzy way, we will henceforward denote the source-syntax-decorated string-to-tree rule as a **fuzzy-tree to exact-tree rule**. We first briefly review issues of string-to-tree rule extraction; then we discuss how to augment the string-to-tree rules to yield fuzzy-tree to exact-tree rules.

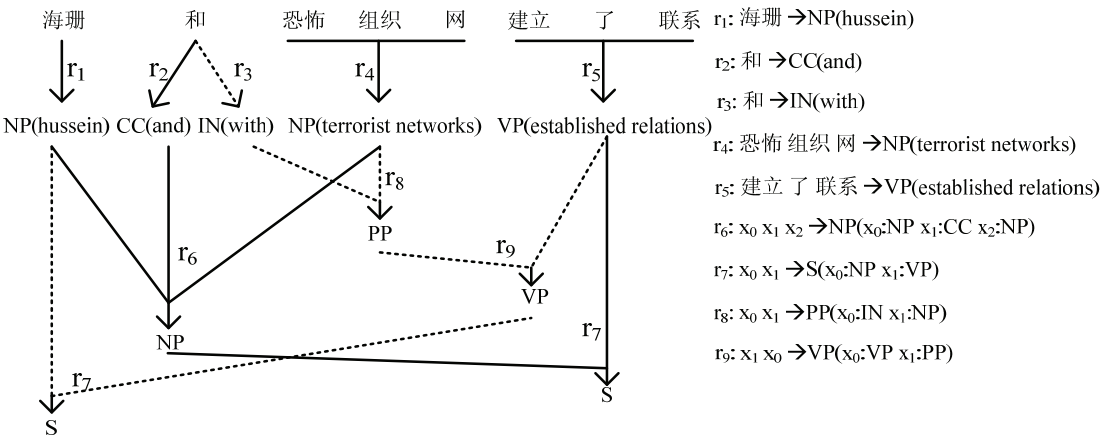

Figure 1: Two alternative derivations for a sample string-to-tree translation. The rules used are listed on the right. The target yield of the tree with solid lines is *hussein and terrorist networks established relations*. The target yield of the tree with dotted lines is *hussein established relations with terrorist networks*.

2.1 String-to-Tree Rule Extraction

Galley et al. (2004) proposed the GHKM algorithm for extracting (minimal) string-to-tree translation rules from a triple (*f, e_t, a*), where *f* is the source-language sentence, e_t is a target-language parse tree whose yield *e* is the translation of *f*, and *a* is the set of word alignments between *e* and *f*. The basic idea of GHKM is to obtain the set of minimally-sized translation rules which can explain the mappings between source string and target parse tree. The minimal string-to-tree rules are extracted in three steps: (1) frontier set computation; (2) fragmentation; and (3) extraction.

The frontier set (**FS**) is the set of potential points at which to cut the graph *G* constructed by the triple (*f, e_t, a*) into fragments. A node satisfying the word alignment is a frontier. ***Bold italic*** nodes in the English parse tree in Figure 2 are all frontiers.

Given the frontier set, a well-formed fragmentation of *G* is generated by restricting each fragment to take only nodes in **FS** as the root and leaf nodes.

With fragmentation completed, the rules are extracted through a depth-first traversal of e_t: for each frontier being visited, a rule is extracted. These extracted rules are called *minimal rules* (Galley et al., 2004). For example, rules $r_a \sim r_i$ in Figure 2 are part of the total of 13 minimal rules.

To improve the rule coverage, SPMT models can be employed to obtain *phrasal rules* (Marcu et at., 2006). In addition, the minimal rules which share the adjacent tree fragments can be connected together to form composed rules (Galley et al., 2006). In Figure 2, r_j is a rule composed by combining r_c and r_g.

2.2 Fuzzy-tree to Exact-tree Rule Extraction

Our fuzzy-tree to exact-tree rule extraction works on word-aligned tree-to-tree data (Figure 2 illustrates a Chinese-English tree pair). Basically, the extraction algorithm includes two parts:

(1) String-to-tree rule extraction (without considering the source parse tree);
(2) Decoration of the source side of the string-to-tree rules with syntactic annotations.

We use the same algorithm introduced in the previous section for extracting the base string-to-tree rules. The source-side syntactic decoration is much more complicated.

The simplest way to decorate, as mentioned in the Introduction, is to annotate the source-side of a string-to-tree rule with the syntactic tag that exactly covers the source string. This is what the exact tree-to-tree procedure does (Liu et al., 2009). However, many useful string-to-tree rules will become invalid if we impose such a tight restriction. For example, in Figure 2, the English phrase *discuss ... them* is a VP, but its Chinese counterpart is not a constituent. Thus we will miss the rule r_h although it is a useful reordering rule. According to the analysis of our training data, the rules with rigid source-side syntactic constraints account for only about 74.5% of the base string-to-tree rules. In this paper, we desire more general applicability.

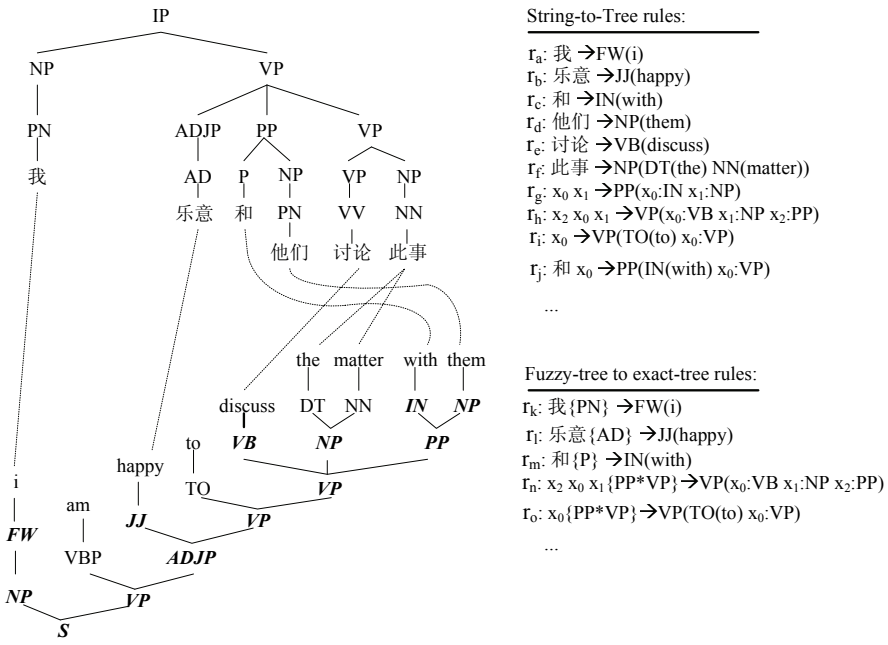

Figure 2: A sample Chinese-English tree pair for rule extraction. The ***bold italic*** nodes in the target English tree are frontiers. **Note that** string-to-tree rules are extracted without considering source-side syntax (upper-right). The new fuzzy-tree to exact-tree rules are extracted with both-side parse trees (bottom-right).

Inspired by (Zollmann and Venugopal, 2006; Chiang, 2010), we resort to SAMT-style syntactic categories in the style of categorial grammar (Bar-Hillel, 1953). The annotation of the source side of string-to-tree rules is processed in three steps: (1) If the source-side string corresponds to a syntactic category C in the source parse tree, we label the source string with C. (2) Otherwise, we check if there exists an extended category of the forms C_1*C_2, C_1/C_2 or $C_2\backslash C_1$[1], indicating respectively that the source string spans two adjacent syntactic categories, a partial syntactic category C_1 missing a C_2 on the right, or a partial C_1 missing a C_2 on the left. (3) If the second step fails, we check if there is an extended category of the forms C_1*C_2*C_3 or $C_1..C_2$, showing that the source string spans three adjacent syntactic categories or a partial category with C_1 and C_2 on each side. In the worst case, $C_1..C_2$ can denote every source string, thus all of the decorations in our training data can be explained within the above three steps. Using the SAMT-style grammar, each source string can be associated with a syntactic category. Thus our fuzzy-tree to exact-tree extraction does not lose any rules as compared with string-to-tree extraction. For example, rule r_o in Figure 2 uses the product category $PP*VP$ on the source side.

A problem may arise: How should we handle the situation where several rules are observed which only differ in their source-side syntactic categories? For example, besides the rule r_m in Figure 2, we encountered rules like $和\{CC\} \rightarrow IN(with)$ in the training data. Which source tag should we retain? We do not make a partial choice in the rule extraction phase. Instead, we simply make a union of the relevant rules and retain the respective tag counts. Applying this strategy, the rule takes the form of $和\{P:6, CC:4\} \rightarrow IN(with)$[2], indicating that the source-side preposition tag appears six times while the conjunction occurs four times. Note that the final rule format used in translation depends on the specific fuzzy rule matching algorithm adopted.

3 Fuzzy Rule Matching Algorithms

The extracted rules will ultimately be applied to derive translations during decoding. One way to apply the fuzzy-tree to exact-tree rules is to narrow the rule search space. Given a test source sentence

[1] The kinds of categories are checked in order. This means that if C_1*C_2, C_1/C_2 can both describe the same source string, we will choose C_1*C_2.

[2] 6 and 4 are not real counts. They are used for illustration only.

with its parse tree, we can according to this strategy choose only the rules whose source syntax matches the test source tree. However, this restriction will rule out many potentially correct rules. In this study, we keep the rule search space identical to that of the string-to-tree setting, and postpone the use of source-side syntax until the derivation stage. During derivation, a fuzzy matching algorithm will be adopted to compute a score to measure the compatibility between the rule and the test source syntax. The translation model will learn to distinguish good rules from bad ones via the compatibility scores.

In this section, three fuzzy matching algorithms, from simple to complex, are investigated in order.

3.1 0-1 Matching

0-1 matching is a straightforward approach that rewards rules whose source syntactic category exactly matches the syntactic category of the test string and punishes mismatches. It has mainly been employed in hierarchical phrase-based models for integrating source or both-side syntax (Marton and Resnik, 2008; Chiang et al., 2009; Chiang, 2010). Since it is verified to be very effective in hierarchical models, we borrow this idea in our source-syntax-augmented string-to-tree translation.

In 0-1 matching, the rule's source side must contain only one syntactic category, but a rule may have been decorated with more than one syntactic category on the source side. Thus we have to choose the most reliable category and discard the others. Here, we select the one with the highest frequency. For example, the tag *P* in the rule $和\{P:6, CC:4\} \rightarrow IN(with)$ appears more frequently, so the final rule used in 0-1 matching will be $和\{P\} \rightarrow IN(with)$. Accordingly, we design two features:

1. *match_count* calculates in a derivation the number of rules whose source-side syntactic category matches the syntactic category of the test string.
2. *unmatch_count* counts the number of mismatches.

For example, in the derivations of Figure 1, we know the Chinese word 和(hé) is a preposition in this sentence (and thus can be written as P(和)), therefore, *match_count += 1* if the above rule $和\{P\} \rightarrow IN(with)$ is employed.

These two features are integrated into the log-linear translation model and the corresponding feature weights will be tuned along with other model features to learn which rules are preferred.

3.2 Likelihood Matching

It appears intuitively that the 0-1 matching algorithm does not make full use of the source-side syntax because it keeps only the most-frequent syntactic label and discards some potentially useful information. Therefore, it runs the risk of treating all the discarded source syntactic categories of the rule as equally likely. For example, there is an extracted rule as follows:

$的\{DEC:11233, DEG:11073, DEV:65\} \rightarrow IN(of)$

0-1 matching converts it into $的\{DEC\} \rightarrow IN(of)$. The use of this rule will be penalized if the syntactic category of the test string 的(dē) is parsed as *DEG* or *DEV*. On one hand, the frequency of the tag *DEG* is just slightly less than that of *DEC*, but the 0-1 matching punishes the former while rewarding the latter. On the other hand, the frequency of *DEG* is much more than that of *DEV*, but they are penalized equally. It is obvious that the syntactic categories are not finely distinguished.

Considering this situation, we propose the likelihood matching algorithm. First, we compute the likelihood of the rule's source syntactic categories. Since we need to deal with the potential problem that the rule is hit by the test string but the syntactic category of the test string is not in the category set of the rule's source side, we apply the *m*-estimate of probability (Mitchell, 1997) to calculate a smoothed likelihood

$$likelihood_c = \frac{n_c + mp}{n + m} \quad (1)$$

in which n_c is the count of each syntactic category *c* in a specific rule, *n* denotes the total count of the rule, *m* is a constant called the equivalent sample size, and *p* is the prior probability of the category *c*. In our work, we set the constant *m=1* and the prior *p* to 1/12599 where 12599 is the total number of source-side syntactic categories in our training data. For example, the rule $和\{P:6, CC:4\} \rightarrow IN(with)$ becomes $和\{P:0.545, CC:0.364, 7.2e\text{-}6\} \rightarrow IN(with)$ after likelihood computation. Then, if we apply likelihood matching in the derivations in Figure 1 where the test string is 和 and its syntax is P(和),

the matching score with the above rule will be 0.545. When the test Chinese word 和 is parsed as a category other than *P* or *CC*, the matching score with the above rule will be 7.2e-6.

Similar to 0-1 matching, likelihood matching will serve as an additional model feature representing the compatibility between categories and rules.

3.3 Deep Similarity Matching

Considering the two algorithms above, we can see that the purpose of fuzzy matching is in fact to calculate a similarity. 0-1 matching assigns similarity 1 for exact matches and 0 for mismatch, while likelihood matching directly utilizes the likelihood to measure the similarity. Going one step further, we adopt a measure of deep similarity, computed using latent distributions of syntactic categories. Huang et al. (2010) proposed this method to compute the similarity between two syntactic tag sequences, used to impose soft syntactic constraints in hierarchical phrase-based models. Analogously, we borrow this idea to calculate the similarity between two SAMT-style syntactic categories, and then apply it to calculate the degree of matching between a translation rule and the syntactic category of a test source string for purposes of fuzzy matching. We call this procedure **deep similarity matching**.

Instead of directly using the SAMT-style syntactic categories, we represent each category by a real-valued feature vector. Suppose there is a set of *n* latent syntactic categories $V=(v_1,\cdots,v_n)$ (*n=16* in our experiments). For each SAMT-style syntactic category, we compute its distribution of latent syntactic categories $\vec{P}_c(V)=(P_c(v_1),\cdots,P_c(v_n))$. For example, $\vec{P}_{VP*NP}(V)=(0.4, 0.2, 0.3, 0.1)$ means that the latent syntactic categories v_1, v_2, v_3, v_4 are distributed as $p(v_1)$=0.4, $p(v_2)$=0.2, $p(v_3)$=0.3 and $p(v_4)$=0.1 for the SAMT-style syntactic category *VP*NP*. Then we further transform the distribution to a normalized feature vector $\vec{F}(c)=\vec{P}_c(V)/\left\|\vec{P}_c(V)\right\|$ to represent the SAMT-style syntactic category *c*.

With the real-valued vector representation for each SAMT-style syntactic category, the degree of similarity between two syntactic categories can be simply computed as a dot-product of their feature vectors:

$$\vec{F}(c)\cdot\vec{F}(c')=\sum_{1\le i\le n} f_i(c)f_i(c') \quad (2)$$

This computation yields a similarity score ranging from 0 (totally different syntactically) to 1 (totally identical syntactically).

Since we can now compute the similarity of any syntactic category pair, we are currently ready to compute the matching degree between the syntactic category of a test source string and a fuzzy-tree to exact-tree rule. To do this, we first convert the original fuzzy-tree to exact-tree rule to the rule of likelihood format without any smoothing. For example, the rule $和\{P:6, CC:4\}\rightarrow IN(with)$ becomes $和\{P:0.6, CC:0.4\}\rightarrow IN(with)$ after conversion. We then denote the syntax of a rule's source-side ***RS*** by weighting all the SAMT-style categories in RS

$$\vec{F}(RS)=\sum_{c\in RS} P_{RS}(c)\vec{F}(c) \quad (3)$$

where $P_{RS}(c)$ is the likelihood of the category *c*. Finally, the deep similarity between a SAMT-style syntactic category *tc* of a test source string and a fuzzy-tree to exact-tree rule is computed as follows:

$$DeepSim(tc, RS)=\vec{F}(tc)\cdot\vec{F}(RS) \quad (4)$$

This deep similarity score will serve as a useful feature in the string-to-tree model which will enable the model to learn how to take account of the source-side syntax during translation.

We have ignored the details of latent syntactic category induction in this paper. In brief, the set of latent syntactic categories is automatically induced from a source-side parsed, word-aligned parallel corpus. The EM algorithm is employed to induce the parameters. We simply follow the algorithm of (Huang et al., 2010), except that we replace the tag sequence with SAMT-style syntactic categories.

4 Rule Binarization

In the baseline string-to-tree model, the rules are not in Chomsky Normal Form. There are several ways to ensure cubic-time decoding. One way is to prune the extracted rules using a scope-3 grammar and do SCFG decoding without binarization (Hopkins and Lengmead, 2010). The other, and most popular way is to binarize the translation rules (Zhang et al., 2006). We adopt the latter approach for efficient decoding with integrated *n*-gram language models since this binarization technique has been well studied in string-to-tree

translation. However, when the rules' source string is decorated with syntax (fuzzy-tree to exact-tree rules), how should we binarize these rules?

We use the rule r_n in Figure 2 for illustration:

$r_n:\ x_2\ x_0\ x_1\ \{PP*VP\} \rightarrow VP(x_0:VB\ \ x_1:NP\ \ x_2:PP)$.

Without regarding the source-side syntax, we obtain the following two binarized rules:

$$B1:\quad x_2\ x_{0*1} \rightarrow VP\left(x_{0*1}:V_{x_0*x_1}\ \ x_2:PP\right)$$

$$B2:\quad x_0\ x_1 \rightarrow V_{x_0*x_1}\left(x_0:VB\ \ x_1:NP\right)$$

Since the source-side syntax *PP*VP* in rule r_n only accounts for the entire source side, it is unclear how to annotate the source side of a partial rule such as the second binary rule *B2*.

Analyzing the derivation process, we observe that a partial rule such as binary rule *B2* never appears in the final derivation unless the rooted binary rule *B1* also appears in the derivation. Based on this observation, we design a heuristic[3] strategy: we simply attach the syntax *PP*VP* in the rooted binary rule *B1*, and do not decorate other binary rules with source syntax. Thus rule r_n will be binarized as:

$$(1)\quad x_2\ x_{0*1}\ \{PP*VP\} \rightarrow VP\left(x_{0*1}:V_{x_0*x_1}\ \ x_2:PP\right)$$

$$(2)\quad x_0\ x_1 \rightarrow V_{x_0*x_1}\left(x_0:VB\ \ x_1:NP\right)$$

5 Translation Model and Decoding

The proposed translation system is an augmentation of the string-to-tree model. In the baseline string-to-tree model, the decoder searches for the optimal derivation d^* that parses a source string f into a target tree e_t from all possible derivations D:

$$d^* = \arg\max_{d\in D} \lambda_1 \log p_{LM}\left(\tau(d)\right) + \lambda_2\left|\tau(d)\right| + \lambda_3|d| + R(d\mid f) \quad (5)$$

where the first element is a language model score in which $\tau(d)$ is the target yield of derivation d; the second element is the translation length penalty; the third element is used to control the derivation length; and the last element is a translation score that includes six features:

$$\begin{aligned} R(d\mid f) = &\sum_{r\in d} \lambda_4 \log p\left(r \mid root(r)\right) + \lambda_5 \log p\left(r \mid lhs(r)\right) \\ &+ \lambda_6 \log p\left(r \mid rhs(r)\right) + \lambda_7 \log p_{lex}\left(lhs(r) \mid rhs(r)\right) \\ &+ \lambda_8 \log p_{lex}\left(rhs(r) \mid lhs(r)\right) + \lambda_9 \delta\left(is\_comp\right) \end{aligned} \quad (6)$$

In equation (6), the first three elements denote the conditional probability of the rule given the root, the source-hand side, and the target-hand side. The next two elements are bidirectional lexical translation probabilities. The last element is the preferred binary feature for learning: either the composed rule or the minimal rule.

In our source-syntax-augmented model, the decoder also searches for the best derivation. With the help of the source syntactic information, the derivation rules in our new model are much more distinguishable than that in the string-to-tree model:

$$d^* = \arg\max_{d\in D} \lambda_1 \log p_{LM}\left(\tau(d)\right) + \lambda_2\left|\tau(d)\right| + \lambda_3|d| + \overline{R}(d\mid f) \quad (7)$$

Here, all elements except the last one are the same as in the string-to-tree model. The last item is:

$$\begin{aligned} \overline{R}(d\mid f) = &R(d\mid f) \\ &+ \sum_{r\in d} \delta(DeepSim)\lambda_{10} \log\left(DeepSim(tag, r)\right) \\ &+ \delta(likelihood)\lambda_{11} \log\left(likelihood(tag, r)\right) \\ &+ \delta(01)\left\{\lambda_{12}\delta(match) + \lambda_{13}\delta(unmatch)\right\} \end{aligned} \quad (8)$$

The 0-1 matching[4] is triggered only when we set $\delta(01)=1$. The other two fuzzy matching algorithms are triggered in a similar way.

During decoding, we use a CKY-style parser with beam search and cube-pruning (Huang and Chiang, 2007) to decode the new source sentences.

6 Experiments

6.1 Experimental Setup

The experiments are conducted on Chinese-to-English translation, with training data consisting of about 19 million English words and 17 million Chinese words[5]. We performed bidirectional word alignment using GIZA++, and employed the *grow-diag-final* balancing strategy to generate the final

[3] We call it heuristic because there may be other syntactic annotation strategies for the binarized rules. It should be noted that our strategy makes the annotated binarized rules equivalent to the original rule.

[4] In theory, the features *unmatch_count*, *match_count* and *derivation_length* are linearly dependent, so the *unmatch_count* is redundant. In practice, since the derivation may include glue rules which are not scored by fuzzy matching. Thus, "*unmatch_count* + *match_count* + *glue_rule_number* = *derivation_length*".

[5] LDC catalog number: LDC2002E18, LDC2003E14, LDC2003E07, LDC2004T07 and LDC2005T06.

symmetric word alignment. We parsed both sides of the parallel text with the Berkeley parser (Petrov et al., 2006) and trained a 5-gram language model with the target part of the bilingual data and the Xinhua portion of the English Gigaword corpus.

For tuning and testing, we use NIST MT evaluation data for Chinese-to-English from 2003 to 2006 (MT03 to MT06). The development data set comes from MT06 in which sentences with more than 20 words are removed to speed up MERT[6] (Och, 2003). The test set includes MT03 to MT05.

We implemented the baseline string-to-tree system ourselves according to (Galley et al., 2006; Marcu et al., 2006). We extracted minimal GHKM rules and the rules of SPMT Model 1 with source language phrases up to length *L=4*. We further extracted composed rules by composing two or three minimal GHKM rules. We also ran the state-of-the-art hierarchical phrase-based system Joshua (Li et al., 2009) for comparison. In all systems, we set the beam size to 200. The final translation quality is evaluated in terms of case-insensitive BLEU-4 with shortest length penalty. The statistical significance test is performed using the re-sampling approach (Koehn, 2004).

6.2 Results

Table 1 shows the translation results on development and test sets. First, we investigate the performance of the strong baseline string-to-tree model (*s2t* for short). As the table shows, *s2t* outperforms the hierarchical phrase-based system *Joshua* by more than 1.0 BLEU point in all translation tasks. This result verifies the superiority of the baseline string-to-tree model.

With the *s2t* system providing a baseline, we further study the effectiveness of our source-syntax-augmented string-to-tree system with fuzzy-tree to exact-tree rules (we use *FT2ET* to denote our proposed system). The last three lines in Table 1 show that, for each fuzzy matching algorithm, our new system *TF2ET* performs significantly better than the baseline *s2t* system, with an improvement of more than 0.5 absolute BLEU points in all tasks. This result demonstrates the success of our new method of incorporating source-side syntax in a string-to-tree model.

[6] The average decoding speed is about 50 words per minute in the baseline string-to-tree system and our proposed systems.

System		MT06 (dev)	MT03	MT04	MT05
Joshua		29.42	28.62	31.52	31.39
s2t		30.84	29.75	32.68	32.41
FT2ET	*0-1*	31.61**	30.60**	33.45**	33.37**
	LH	31.35*	30.34*	33.21*	33.05*
	DeepSim	31.77**	30.82**	33.69**	33.50**

Table 1: Results (in BLEU scores) of different translation models in multiple tasks. *LH*=likelihood. *or**=significantly better than *s2t* system (p<0.05 or 0.01 respectively).

	Very similar $\vec{F}(c)\cdot\vec{F}(c')$>0.9	Very dissimilar $\vec{F}(c)\cdot\vec{F}(c')$<0.1
ADJP	JJ; AD\ADJP	VP; ADVP\NP
NP	DT*NN; LCP*P*NP	CP; BA*CP

Table 2: Example of similar and dissimilar categories.

Specifically, the *FT2ET* system with *deep similarity matching* obtains the best translation quality in all tasks and surpasses the baseline *s2t* system by 0.93 BLEU points in development data and by more than 1.0 BLEU point in test sets. The *0-1 matching* algorithm is simple but effective, and it yields quite good performance (line 3). The contribution of *0-1 matching* as reflected in our experiments is consistent with the conclusions of (Marton and Resnik, 2008; Chiang, 2010). By contrast, the system with *likelihood matching* does not perform as well as the other two algorithms, although it also significantly improves the baseline *s2t* in all tasks.

6.3 Analysis and Discussion

We are a bit surprised at the large improvement gained by the 0-1 matching algorithm. This algorithm has several advantages: it is simple and easy to implement, and enhances the translation model by enabling its rules to take account of the source-side syntax to some degree. However, a major deficiency of this algorithm is that it does not make full use of the source side syntax, since it retains only the most frequent SAMT-style syntactic category to describe the rule's source syntax. Thus this algorithm penalizes all the other categories equally, although some may be more frequent than others, as in the case of *DEG* and *DEV* in the rule 的$\{DEC:11233, DEG:11073, DEV:65\}\rightarrow IN(of)$.

To some extent, the likelihood matching algorithm solves the main problem of 0-1 matching.

Instead of rewarding or penalizing, this algorithm uses the likelihood of the syntactic category to approximate the degree of matching between the test source syntactic category and the rule. For a category not in the rule's source syntactic category set, the likelihood algorithm computes a smoothed likelihood. However, the likelihood algorithm does not in fact lead to very promising improvement. We conjecture that this disappointing performance is due to the simple smoothing method we employed. Future work will investigate more fully.

Compared with the above two matching algorithms, the deep similarity matching algorithm based on latent syntactic distribution is much more beautiful in theory. This algorithm can successfully measure the similarity between any two SAMT-style syntactic categories (Table 2 gives some examples of similar and dissimilar category pairs). Then it can accurately compute the degree of matching between a test source syntactic category and a fuzzy-tree to exact-tree rule. Thus this algorithm obtains the best translation quality. However, the deep similarity matching algorithm has two practical shortcomings. First, it is not easy to determine the number of latent categories. We have to conduct multiple experiments to arrive at a number which can yield a tradeoff between translation quality and model complexity. In our work, we have tried the numbers *n=4, 8, 16, 32,* and have found *n=16* to give the best tradeoff. The second shortcoming is that the induction of latent syntactic categories has been very time consuming, since we have applied the EM algorithm to the entire source-parsed parallel corpus. Even with *n=8*, it took more than a week to induce the latent syntactic categories on our middle-scale training data when using a Xeon four-core computer ($2.5GHz \times 2CPU \times 16GB\ memory$). When the training data contains tens of millions of sentence pairs, the computation time may no longer be tolerable.

Table 3 shows some translation examples for comparison. In the first example, the Chinese preposition word *和* is mistakenly translated into English conjunction word *and* in *Joshua* and baseline string-to-tree system *s2t*, however, our source-syntax-augmented system *FT2ET-DeepSim* correctly converts the Chinese word *和* into English preposition *with* and finally yield the right translation. In the second example, our proposed system moves the prepositional phrase *at an early date* after the sibling verb phrase. It is more reasonable compared with the baseline system *s2t*. In the third example, the proposed system *FT2ET-DeepSim* successfully recognizes the Chinese long prepositional phrase *在 与 中国 总理 温家宝 举行 峰会 后 发布 的联合 声明 中* and short verb phrase *说*, and obtains the correct phrase reordering at last.

7 Related Work

Several studies have tried to incorporate source or target syntax into translation models in a fuzzy manner.

Zollmann and Venugopal (2006) augment the hierarchical string-to-string rules (Chiang, 2005) with target-side syntax. They annotate the target side of each string-to-string rule using SAMT-style syntactic categories and aim to generate the output more syntactically. Zhang et al. (2010) base their approach on tree-to-string models, and generate grammatical output more reliably with the help of tree-to-tree sequence rules. Neither of them builds target syntactic trees using target syntax, however. Thus they can be viewed as integrating target syntax in a fuzzy manner. By contrast, we base our approach on a string-to-tree model which does construct target syntactic trees during decoding.

(Marton and Resnik, 2008; Chiang et al., 2009 and Huang et al., 2010) apply fuzzy techniques for integrating source syntax into hierarchical phrase-based systems (Chiang, 2005, 2007). The first two studies employ 0-1 matching and the last tries deep similarity matching between two tag sequences. By contrast, we incorporate source syntax into a string-to-tree model. Furthermore, we apply fuzzy syntactic annotation on each rule's source string and design three fuzzy rule matching algorithms.

Chiang (2010) proposes a method for learning to translate with both source and target syntax in the framework of a hierarchical phrase-based system. He not only executes 0-1 matching on both sides of rules, but also designs numerous features such as $root_{X,X'}$ which counts the number of rules whose source-side root label is X and target-side root label is X'. This fuzzy use of source and target syntax enables the translation system to learn which tree labels are similar enough to be compatible, which ones are harmful to combine, and which ones can be ignored. The differences between us are twofold: 1) his work applies fuzzy syntax in both sides, while ours bases on the string-

1	*Source sentence*	海珊 也 [和 恐怖 组织网] 建立 了 联系
	Reference	hussein also established ties with terrorist networks
	Joshua	hussein also and terrorist networks established relations
	s2t	hussein also and terrorist networks established relations
	FT2ET- DeepSim	hussein also established relations with terrorist networks
2	*Source sentence*	… [以 期] [早日] [结束] [以 巴 之间 多年 的 流血 冲突]
	Reference	.. to end years of bloody conflict between israel and palestine as soon as possible .. to end at an early date years of bloody conflict between israel and palestine
	Joshua	… in the early period to end years of blood conflict between israel and palestine
	s2t	… at an early date to end years of blood conflict between israel and palestine
	FT2ET- DeepSim	… to end years of blood conflict between israel and palestine at an early date
3	*Source sentence*	欧盟 [在 与 中国 总理 温家宝 举行 峰会 后 发布 的联合 声明 中] [说] …
	Reference	the europen union said in a joint statement issued after its summit meeting with china 's premier wen jiabao … in a joint statement released after the summit with chinese premier wen jiabao , the europen union said …
	Joshua	the europen union with chinese premier wen jiabao in a joint statement issued after the summit meeting said …
	s2t	the europen union in a joint statement issued after the summit meeting with chinese premier wen jiabao said …
	FT2ET- DeepSim	the europen union said in a joint statement issued after the summit meeting with chinese premier wen jiabao …

Table 3: Some translation examples produced by *Joshua*, string-to-tree system *s2t* and source-syntax-augmented string-to-tree system *FT2ET* with deep similarity matching algorithm

to-tree model and applies fuzzy syntax on source side; and 2) we not only adopt the 0-1 fuzzy rule matching algorithm, but also investigate likelihood matching and deep similarity matching algorithms.

8 Conclusion and Future Work

In this paper, we have proposed a new method for augmenting string-to-tree translation models with fuzzy use of the source syntax. We first applied a fuzzy annotation method which labels the source side of each string-to-tree rule with SAMT-style syntactic categories. Then we designed and explored three fuzzy rule matching algorithms: 0-1 matching, likelihood matching, and deep similarity matching. The experiments show that our new system significantly outperforms the strong baseline string-to-tree system. This substantial improvement verifies that our fuzzy use of source syntax is effective and can enhance the ability to choose proper translation rules during decoding while guaranteeing grammatical output with explicit target trees. We believe that our work may demonstrate effective ways of incorporating both-side syntax in a translation model to yield promising results.

Next, we plan to further study the likelihood fuzzy matching and deep similarity matching algorithms in order to fully exploit their potential. For example, we will combine the merits of 0-1 matching and likelihood matching so as to avoid the setting of parameter *m* in likelihood matching. We also plan to explore another direction: we will annotate the source side of each string-to-tree rule with subtrees or subtree sequences. We can then apply tree-kernel methods to compute a degree of matching between a rule and a test source subtree or subtree sequence.

Acknowledgments

The research work has been funded by the Natural Science Foundation of China under Grant No. 60975053, 61003160 and 60736014 and supported by the External Cooperation Program of the Chinese Academy of Sciences. We would also like to thank Mark Seligman and Yu Zhou for revising the early draft, and anonymous reviewers for their valuable suggestions.

References

Yehoshua Bar-Hillel, 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29 (1). pages 47-58.

David Chiang, 2005. A hiearchical phrase-based model for statistical machine translation. In *Proc. of ACL* 2005, pages 263-270.

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

David Chiang, 2010. Learning to translate with source and target syntax. In *Proc. of ACL 2010*, pages 1443-1452.

David Chiang, Kevin Knight and Wei Wang, 2009. 11,001 new features for statistical machine translation. In *Proc. of NAACL 2009*, pages 218-226.

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

Yuan Ding and Martha Palmer, 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of ACL 2005*, pages 541-548.

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

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer, 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of ACL-COLING 2006*.

Mark Hopkins and Greg Langmead, 2010. SCFG decoding without binarization. In *Proc. of EMNLP 2010*, pages 646-655.

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

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

Zhongqiang Huang, Martin Cmejrek and Bowen Zhou, 2010. Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distributions. In *Proc. of EMNLP 2010*, pages 138-147.

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

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

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

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

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

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

Yuval Marton and Philip Resnik, 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proc. of ACL-08: HLT*. pages 1003–1011.

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

Haitao Mi and Qun Liu, 2010. Constituency to dependency translation with forests. In *Proc. of ACL 2010*, pages 1433-1442.

Tom M. Mitchell, 1997. Machine learning. *Mac Graw Hill*.

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

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

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

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

Tong Xiao, Jingbo Zhu, Muhua Zhu and and Huizhen Wang, 2010. Boosting-based System Combination for Machine Translation. In *Proc. of ACL 2010*, pages 739-748.

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

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

Hui Zhang, Min Zhang, Haizhou Li and Chng Eng Siong, 2010. Non-isomorphic forest pair translation. In *Proc. of EMNLP 2010*, pages 440-450.

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