Improving Decoding Generalization for Tree-to-String Translation

Jingbo Zhu
Natural Language Processing Laboratory
Northeastern University, Shenyang, China
zhujingbo@mail.neu.edu.cn

Tong Xiao
Natural Language Processing Laboratory
Northeastern University, Shenyang, China
xiaotong@mail.neu.edu.cn

Abstract

To address the parse error issue for tree-to-string translation, this paper proposes a similarity-based decoding generation (SDG) solution by reconstructing similar source parse trees for decoding at the decoding time instead of taking multiple source parse trees as input for decoding. Experiments on Chinese-English translation demonstrated that our approach can achieve a significant improvement over the standard method, and has little impact on decoding speed in practice. Our approach is very easy to implement, and can be applied to other paradigms such as tree-to-tree models.

1 Introduction

Among linguistically syntax-based statistical machine translation (SMT) approaches, the tree-to-string model (Huang *et al.* 2006; Liu *et al.* 2006) is the simplest and fastest, in which parse trees on source side are used for grammar extraction and decoding. Formally, given a source (e.g., Chinese) string *c* and its auto-parsed tree $T_{1\text{-}best}$, the goal of typical tree-to-string SMT is to find a target (e.g., English) string *e** by the following equation as

$$e^* = \arg\max_e \Pr(e \mid c, T_{1-best}) \quad (1)$$

where $\Pr(e|c,T_{1\text{-}best})$ is the probability that *e* is the translation of the given source string *c* and its $T_{1\text{-}best}$. A typical tree-to-string decoder aims to search for the best derivation among all consistent derivations that convert source tree into a target-language string. We call this set of consistent derivations the *tree-to-string search space.* Each derivation in the search space respects the source parse tree.

Parsing errors on source parse trees would cause negative effects on tree-to-string translation due to decoding on incorrect source parse trees. To address the parse error issue in tree-to-string translation, a natural solution is to use *n*-best parse trees instead of 1-best parse tree as input for decoding, which can be expressed by

$$e^* = \arg\max_e \Pr(e \mid c, \langle T_{n-best} \rangle) \quad (2)$$

where $<T_{n\text{-}best}>$ denotes a set of *n*-best parse trees of *c* produced by a state-of-the-art syntactic parser. A simple alternative (Xiao *et al.* 2010) to generate $<T_{n\text{-}best}>$ is to utilize multiple parsers, which can improve the diversity among source parse trees in $<T_{n\text{-}best}>$. In this solution, the most representative work is the forest-based translation method (Mi *et al.* 2008; Mi and Huang 2008; Zhang *et al.* 2009) in which a packed forest (forest for short) structure is used to effectively represent $<T_{n\text{-}best}>$ for decoding. Forest-based approaches can increase the tree-to-string search space for decoding, but face a non-trivial problem of high decoding time complexity in practice.

In this paper, we propose a new solution by reconstructing new similar source parse trees for decoding, referred to as *similarity-based decoding generation* (SDG), which is expressed as

$$\begin{aligned} e^* &= \arg\max_e \Pr(e \mid c, T_{1-best}) \\ &\cong \arg\max_e \Pr(e \mid c, \{T_{1-best}, \langle T_{sim} \rangle\}) \end{aligned} \quad (3)$$

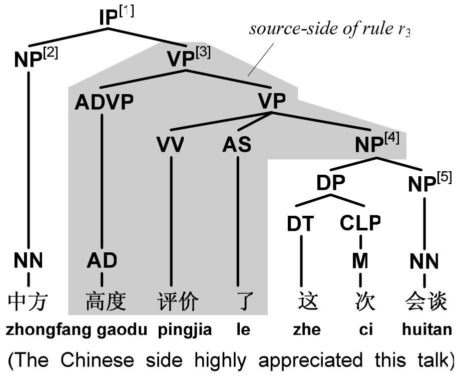
where $<T_{sim}>$ denotes a set of similar parse trees of $T_{1\text{-}best}$ that are dynamically reconstructed at the decoding time. Roughly speaking, <$T_{n\text{-}best}$> is a subset of {$T_{1\text{-}best}$, <T_{sim}>}. Along this line of thinking, Equation (2) can be considered as a special case of Equation (3).

In our SDG solution, given a source parse tree $T_{1\text{-}best}$, the key is how to generate its <T_{sim}> at the decoding time. In practice, it is almost intractable to directly reconstructing <T_{sim}> in advance as input for decoding due to too high computation complexity. To address this crucial challenge, this paper presents a simple and effective technique based on *similarity-based matching constraints* to construct new similar source parse trees for decoding at the decoding time. Our SDG approach can explicitly increase the tree-to-string search space for decoding without changing any grammar extraction and pruning settings, and has little impact on decoding speed in practice.

2 Tree-to-String Derivation

We choose the tree-to-string paradigm in our study because this is the simplest and fastest among syntax-based models, and has been shown to be one of the state-of-the-art syntax-based models. Typically, by using the GHKM algorithm (Galley *et al.* 2004), translation rules are learned from word-aligned bilingual texts whose source side has been parsed by using a syntactic parser. Each rule consists of a syntax tree in the source language having some words (terminals) or variables (nonterminals) at leaves, and sequence words or variables in the target language. With the help of these learned translation rules, the goal of tree-to-string decoding is to search for the best derivation that converts the source tree into a target-language string. A derivation is a sequence of translation steps (i.e., the use of translation rules).

Figure 1 shows an example derivation *d* that performs translation over a Chinese source parse tree, and how this process works. In the first step, we can apply rule r_1 at the root node that matches a subtree {IP[1] (NP[2] VP[3])}. The corresponding target side {x_1 x_2} means to preserve the top-level word-order in the translation, and results in two unfinished subtrees with root labels NP[2] and VP[3], respectively. The rule r_2 finishes the translation on the subtree of NP[2], in which the Chinese word "*中方*" is translated into an English string "*the Chinese side*". The rule r_3 is applied to perform translation on the subtree of VP[3], and results in an


IP[1]
source-side of rule r3
NP[2]
VP[3]
ADVP
VP
VV
AS
NP[4]
DP
NP[5]
DT
CLP
NN
AD
M
NN
中方 高度 评价 了 这 次 会谈
zhongfang gaodu pingjia le zhe ci huitan
(The Chinese side highly appreciated this talk)

An example tree-to-string derivation *d* consisting of five translation rules is given as follows:
r_1: IP[1] (x_1:NP[2] x_2:VP[3]) → x_1 x_2
r_2: NP[2] (NN (中方)) → the Chinese side
r_3: VP[3] (ADVP(AD(高度)) VP(VV(评价) AS(了) x_1:NP[4])) → highly appreciated x_1
r_4: NP[4] (DP(DT(这) CLP(M(次))) x_1:NP[5]) → this x_1
r_5: NP[5] (NN(会谈)) → talk
Translation results: *The Chinese side highly appreciated this talk.*

Figure 1. An example derivation performs translation over the Chinese parse tree *T*.

unfinished subtree of NP[4]. Similarly, rules r_4 and r_5 are sequentially used to finish the translation on the remaining. This process is a depth-first search over the whole source tree, and visits every node only once.

3 Decoding Generalization

3.1 Similarity-based Matching Constraints

In typical tree-to-string decoding, an ordered sequence of rules can be reassembled to form a derivation *d* whose source side matches the given source parse tree *T*. The source side of each rule in *d* should match one of subtrees of *T*, referred to as *matching constraint*. Before discussing how to apply our *similarity-based matching constraints* to reconstruct new similar source parse trees for decoding at the decoding time, we first define the similarity between two tree-to-string rules.

Definition 1 *Given two tree-to-string rules t and u, we say that t and u are similar such that their source sides t_s and u_s have the same root label and frontier nodes, written as $t \cong u$, otherwise not.*

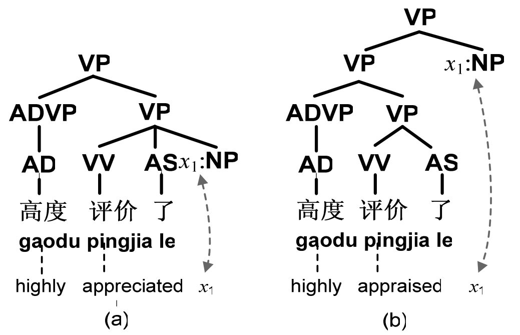


Figure 2: Two similar tree-to-string rules. (a) rule r_3 used by the example derivation d in Figure 1, and (b) a similar rule τ_3 of r_3.

Here we use an example figure to explain our similarity-based matching constraint scheme (similarity-based scheme for short).

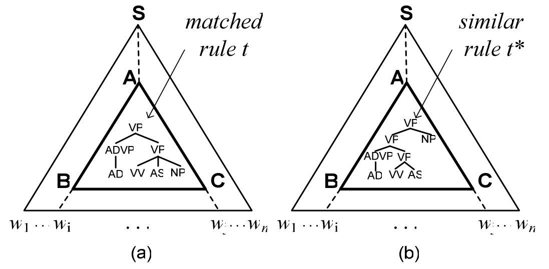

S
matched rule t
A
B
C
$w_1 \cdots w_i$ ... $w_j \cdots w_n$
(a)
S
similar rule t*
A
B
C
$w_1 \cdots w_i$ ... $w_j \cdots w_n$
(b)

Figure 3: (a) a typical tree-to-string derivation d using rule t, and (b) a new derivation d^* is generated by the similarity-based matching constraint scheme by using rule t^* instead of rule t, where $t^* \cong t$.

Given a source-language parse tree T, in typical tree-to-string matching constraint scheme shown in Figure 3(a), rule t used by the derivation d should match a substree ABC of T. In our similarity-based scheme, the similar rule t^* ($\cong t$) is used to form a new derivation d^* that performs translation over the same source sentence $\{w_1 \ldots w_n\}$. In such a case, this new derivation d^* can yield a new similar parse tree T^* of T.

Since an incorrect source parse tree might filter out good derivations during tree-to-string decoding, our similarity-based scheme is much more likely to recover the correct tree for decoding at the decoding time, and does not rule out good (potentially correct) translation choices. In our method, many new source-language trees T^* that are similar to but different from the original source tree T can be reconstructed at the decoding time. In theory our similarity-based scheme can increase the search space of the tree-to-string decoder, but we did not change any rule extraction and pruning settings.

In practice, our similarity-based scheme can effectively keep the advantage of fast decoding for tree-to-string translation because its implementation is very simple. Let's revisit the example derivation d in Figure 1, i.e., $d=r_1 \oplus r_2 \oplus r_3 \oplus r_4 \oplus r_5$[1]. In such a case, the decoder can effectively produce a new derivation d^* by simply replacing rule r_3 with its similar rule τ_3 ($\cong r_3$) shown in Figure 2, that is, $d^*=r_1 \oplus r_2 \oplus \tau_3 \oplus r_4 \oplus r_5$.

With beam search, typical tree-to-string decoding with an integrated language model can run in time[2] $O(ncb^2)$ in practice (Huang 2007). For our decoding time complexity computation, only the parameter c value can be affected by our similarity-based scheme. In other words, our similarity-based scheme would result in a larger c value at decoding time as many similar translation rules might be matched at each node. In practice, there are two feasible optimization techniques to alleviate this problem. The first technique is to limit the maximum number of similar translation rules matched at each node. The second one is to predefine a similarity threshold to filter out less similar translation rules in advance.

In the implementation, we add a new feature into the model: *similarity-based matching counting feature*. This feature counts the number of similar rules used to form the derivation. The weight λ_{sim} of this feature is tuned via minimal error rate training (MERT) (Och 2003) with other feature weights.

3.2 Pseudo-rule Generation

In the implementation of tree-to-string decoding, the first step is to load all translation rules matched at each node of the source tree T. It is possible that some nonterminal nodes do not have any matched rules when decoding some new sentences. If the root node of the source tree has no any matched rules, it would cause decoding failure. To tackle this problem, motivated by "*glue*" rules (Chiang 2005), for some node S without any matched rules, we introduce a special *pseudo-rule* which reassembles all child nodes with *local reordering* to form new translation rules for S to complete decoding.

[1] The symbol $\oplus$ denotes the composition (leftmost substitution) operation of two tree-to-string rules.

[2] Where n is the number of words, b is the size of the beam, and c is the number of translation rules matched at each node.

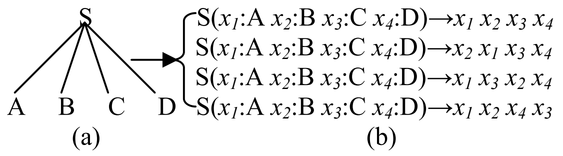

S
A B C D
(a)
S(x_1:A x_2:B x_3:C x_4:D)→x_1 x_2 x_3 x_4
S(x_1:A x_2:B x_3:C x_4:D)→x_2 x_1 x_3 x_4
S(x_1:A x_2:B x_3:C x_4:D)→x_1 x_3 x_2 x_4
S(x_1:A x_2:B x_3:C x_4:D)→x_1 x_2 x_4 x_3
(b)

Figure 4: (a) An example unseen substree, and (b) its four pseudo-rules.

Figure 4 (a) depicts an example unseen substree where no any rules is matched at its root node *S*. Its simplest pseudo-rule is to simply combine a sequence of *S*'s child nodes. To give the model more options to build partial translations, we utilize a local reordering technique in which any two adjacent frontier (child) nodes are reordered during decoding. Figure 4(b) shows four pseudo-rules in total generated from this example unseen substree.

In the implementation, we add a new feature to the model: *pseudo-rule counting feature*. This feature counts the number of pseudo-rules used to form the derivation. The weight λ_{pseudo} of this feature is tuned via MERT with other feature weights.

4 Evaluation

4.1 Setup

Our bilingual training data consists of 140K Chinese-English sentence pairs in the FBIS data set. For rule extraction, the minimal GHKM rules (Galley *et al.* 2004) were extracted from the bitext, and the composed rules were generated by combining two or three minimal GHKM rules. A 5-gram language model was trained on the target-side of the bilingual data and the Xinhua portion of English Gigaword corpus. The beam size for beam search was set to 20. The base feature set used for all systems is similar to that used in (Marcu *et al.* 2006), including 14 base features in total such as 5-gram language model, bidirectional lexical and phrase-based translation probabilities. All features were linearly combined and their weights are optimized by using MERT. The development data set used for weight training in our approaches comes from NIST MT03 evaluation set. To speed up MERT, sentences with more than 20 words were removed from the development set (Dev set). The test sets are the NIST MT04 and MT05 evaluation sets. The translation quality was evaluated in terms of case-insensitive NIST version BLEU metric. Statistical significance test was conducted by using the bootstrap re-sampling method (Koehn 2004).

4.2 Results

	DEV MT03	MT04		MT05	
		<=20	ALL	<=20	ALL
Baseline	32.99	36.54	32.70	34.61	30.60
This work	34.67^* (+1.68)	36.99^+ (+0.45)	35.03^* (+2.33)	35.16^+ (+0.55)	33.12^* (+2.52)

Table 1. BLEU4 (%) scores of various methods on Dev set (MT03) and two test sets (MT04 and MT05). Each small test set (<=20) was built by removing the sentences with more than 20 words from the full set (ALL). + and * indicate significantly better on performance comparison at $p < .05$ and $p < .01$, respectively.

Table 1 depicts the BLEU scores of various methods on the Dev set and four test sets. Compared to typical tree-to-string decoding (the baseline), our method can achieve significant improvements on all datasets. It is noteworthy that the improvement achieved by our approach on full test sets is bigger than that on small test sets. For example, our method results in an improvement of 2.52 BLEU points over the baseline on the MT05 full test set, but only 0.55 points on the MT05 small test set. As mentioned before, tree-to-string approaches are more vulnerable to parsing errors. In practice, the Berkeley parser (Petrov *et al.* 2006) we used yields unsatisfactory parsing performance on some long sentences in the full test sets. In such a case, it would result in negative effects on the performance of the baseline method on the full test sets. Experimental results show that our SDG approach can effectively alleviate this problem, and significantly improve tree-to-string translation.

Another issue we are interested in is the decoding speed of our method in practice. To investigate this issue, we evaluate the average decoding speed of our SDG method and the baseline on the Dev set and all test sets.

	Decoding Time (seconds per sentence)	
	<=20	ALL
Baseline	0.43s	1.1s
This work	0.50s	1.3s

Table 2. Average decoding speed of various methods on small (<=20) and full (ALL) datasets in terms of *seconds per sentence*. The parsing time of each sentence is not included. The decoders were implemented in C++ codes on an X86-based PC with two processors of 2.4GHZ and 4GB physical memory.

Table 2 shows that our approach only has little impact on decoding speed in practice, compared to the typical tree-to-string decoding (baseline). Notice that in these comparisons our method did not adopt any optimization techniques mentioned in Section 3.1, e.g., to limit the maximum number of similar rules matched at each node. It is obviously that the use of such an optimization technique can effectively increase the decoding speed of our method, but might hurt the performance in practice.

Besides, to speed up decoding long sentences, it seems a feasible solution to first divide a long sentence into multiple short sub-sentences for decoding, e.g., based on comma. In other words, we can segment a complex source-language parse tree into multiple smaller subtrees for decoding, and combine the translations of these small subtrees to form the final translation. This practical solution can speed up the decoding on long sentences in real-world MT applications, but might hurt the translation performance.

For convenience, here we call the rule τ_3 in Figure 2(b) *similar-rules*. It is worth investigating how many similar-rules and pseudo-rules are used to form the best derivations in our similarity-based scheme. To do it, we count the number of similar-rules and pseudo-rules used to form the best derivations when decoding on the MT05 full set. Experimental results show that on average 13.97% of rules used to form the best derivations are similar-rules, and one pseudo-rule per sentence is used. Roughly speaking, average five similar-rules per sentence are utilized for decoding generalization.

5 Related Work

String-to-tree SMT approaches also utilize the similarity-based matching constraint on target side to generate target translation. This paper applies it on source side to reconstruct new similar source parse trees for decoding at the decoding time, which aims to increase the tree-to-string search space for decoding, and improve decoding generalization for tree-to-string translation.

The most related work is the forest-based translation method (Mi *et al.* 2008; Mi and Huang 2008; Zhang *et al.* 2009) in which rule extraction and decoding are implemented over *k*-best parse trees (e.g., in the form of packed forest) instead of one best tree as translation input. Liu and Liu (2010) proposed a joint parsing and translation model by casting tree-based translation as parsing (Eisner 2003), in which the decoder does not respect the source tree. These methods can increase the tree-to-string search space. However, the decoding time complexity of their methods is high, i.e., more than ten or several dozen times slower than typical tree-to-string decoding (Liu and Liu 2010).

Some previous efforts utilized the techniques of soft syntactic constraints to increase the search space in hierarchical phrase-based models (Marton and Resnik 2008; Chiang *et al.* 2009; Huang *et al.* 2010), string-to-tree models (Venugopal *et al.* 2009) or tree-to-tree (Chiang 2010) systems. These methods focus on softening matching constraints on the root label of each rule regardless of its internal tree structure, and often generate many new syntactic categories[3]. It makes them more difficult to satisfy syntactic constraints for the tree-to-string decoding.

6 Conclusion and Future Work

This paper addresses the parse error issue for tree-to-string translation, and proposes a similarity-based decoding generation solution by reconstructing new similar source parse trees for decoding at the decoding time. It is noteworthy that our SDG approach is very easy to implement. In principle, forest-based and tree sequence-based approaches improve rule coverage by changing the rule extraction settings, and use exact tree-to-string matching constraints for decoding. Since our SDG approach is independent of any rule extraction and pruning techniques, it is also applicable to forest-based approaches or other tree-based translation models, e.g., in the case of casting tree-to-tree translation as *tree parsing* (Eisner 2003).

Acknowledgments

We would like to thank Feiliang Ren, Muhua Zhu and Hao Zhang for discussions and the anonymous reviewers for comments. This research was supported in part by the National Science Foundation of China (60873091; 61073140), the Specialized Research Fund for the Doctoral Program of Higher Education (20100042110031) and the Fundamental Research Funds for the Central Universities in China.

[3] Latent syntactic categories were introduced in the method of Huang *et al.* (2010).

References

Chiang David. 2005. A hierarchical phrase-based model for statistical machine translation. In Proc. of ACL2005, pp263-270

Chiang David. 2010. Learning to translate with source and target syntax. In Proc. of ACL2010, pp1443-1452

Chiang David, Kevin Knight and Wei Wang. 2009. 11,001 new features for statistical machine translation. In Proc. of NAACL2009, pp218-226

Eisner Jason. 2003. Learning non-isomorphic tree mappings for machine translation. In Proc. of ACL 2003, pp205-208.

Galley Michel, Mark Hopkins, Kevin Knight and Daniel Marcu. 2004. What's in a translation rule? In Proc. of HLT-NAACL 2004, pp273-280.

Huang Liang. 2007. Binarization, synchronous binarization and target-side binarization. In Proc. of NAACL Workshop on Syntax and Structure in Statistical Translation.

Huang Liang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In Proc. of ACL 2007, pp144-151.

Huang Liang, Kevin Knight and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In Proc. of AMTA 2006, pp66-73.

Huang Zhongqiang, Martin Cmejrek and Bowen Zhou. 2010. Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distribution. In Proc. of EMNLP2010, pp138-147

Koehn Philipp. 2004. Statistical Significance Tests for Machine Translation Evaluation. In Proc. of EMNLP 2004, pp388-395.

Liu Yang and Qun Liu. 2010. Joint parsing and translation. In Proc. of Coling2010, pp707-715

Liu Yang, Qun Liu and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In Proc. of COLING/ACL 2006, pp609-616.

Marcu Daniel, Wei Wang, Abdessamad Echihabi and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In Proc. of EMNLP 2006, pp44-52.

Marton Yuval and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrase-based translation. In Proc. of ACL08, pp1003-1011

Mi Haitao and Liang Huang. 2008. Forest-based Translation Rule Extraction. In Proc. of EMNLP 2008, pp206-214.

Mi Haitao, Liang Huang and Qun Liu. 2008. Forest-based translation. In Proc. of ACL2008.

Och Franz Josef. 2003. Minimum error rate training in statistical machine translation. In Proc. of ACL2003.

Petrov Slav, Leon Barrett, Roman Thibaux and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In Proc. of ACL2006, pp433-440.

Xiao Tong, Jingbo Zhu, Hao Zhang and Muhua Zhu. 2010. An empirical study of translation rule extraction with multiple parsers. In Proc. of Coling2010, pp1345-1353

Venugopal Ashish, Andreas Zollmann, Noah A. Smith and Stephan Vogel. 2009. Preference grammars: softening syntactic constraints to improve statistical machine translation. In Proc. of NAACL2009, pp236-244

Zhang Hui, Min Zhang, Haizhou Li, Aiti Aw and Chew Lim Tan. 2009. Forest-based tree sequence to string translation model. In Proc. of ACL-IJCNLP2009, pp172-180