Improved Tree-to-string Transducer for Machine Translation

Ding Liu and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

We propose three enhancements to the tree-to-string (TTS) transducer for machine translation: first-level expansion-based normalization for TTS templates, a syntactic alignment framework integrating the insertion of unaligned target words, and subtree-based n-gram model addressing the tree decomposition probability. Empirical results show that these methods improve the performance of a TTS transducer based on the standard BLEU-4 metric. We also experiment with semantic labels in a TTS transducer, and achieve improvement over our baseline system.

1 Introduction

Syntax-based statistical machine translation (SSMT) has achieved significant progress during recent years, with two threads developing simultaneously: the synchronous parsing-based SSMT (Galley et al., 2006; May and Knight, 2007) and the tree-to-string (TTS) transducer (Liu et al., 2006; Huang et al., 2006). Synchronous SSMT here denotes the systems which accept a source sentence as the input and generate the translation and the syntactic structure for both the source and the translation simultaneously. Such systems are sometimes also called TTS transducers, but in this paper, TTS transducer refers to the system which starts with the syntax tree of a source sentence and recursively transforms the tree to the target language based on TTS templates.

In synchronous SSMT, TTS templates are used similar to the context free grammar used in the standard CYK parser, thus the syntax is part of the output and can be thought of as a constraint on the translation process. In the TTS transducer, since the parse tree is given, syntax can be thought of as an additional feature of the input to be used in the translation. The idea of synchronous SSMT can be traced back to Wu (1997)'s Stochastic Inversion Transduction Grammars. A systematic method for extracting TTS templates from parallel corpora was proposed by Galley et al. (2004), and later binarized by Zhang et al. (2006) for high efficiency and accuracy. In the other track, the TTS transducer originated from the tree transducer proposed by Rounds (1970) and Thatcher (1970) independently. Graehl and Knight (2004) generalized the tree transducer to the TTS transducer and introduced an EM algorithm to estimate the probability of TTS templates based on a bilingual corpus with one side parsed. Liu et al. (2006) and Huang et al. (2006) then used the TTS transducer on the task of Chinese-to-English and English-to-Chinese translation, respectively, and achieved decent performance.

Despite the progress SSMT has achieved, it is still a developing field with many problems unsolved. For example, the word alignment computed by GIZA++ and used as a basis to extract the TTS templates in most SSMT systems has been observed to be a problem for SSMT (DeNero and Klein, 2007; May and Knight, 2007), due to the fact that the word-based alignment models are not aware of the syntactic structure of the sentences and could produce many syntax-violating word alignments. Approaches have been proposed recently towards getting better word alignment and thus better TTS templates, such as encoding syntactic structure information into the HMM-based word alignment model DeNero and Klein (2007), and build-

Proceedings of the Third Workshop on Statistical Machine Translation, pages 62–69,
Columbus, June 2008. ©2008 Association for Computational Linguistics

ing a syntax-based word alignment model May and Knight (2007) with TTS templates. Unfortunately, neither approach reports end-to-end MT performance based on the syntactic alignment. DeNero and Klein (2007) focus on alignment and do not present MT results, while May and Knight (2007) takes the syntactic re-alignment as an input to an EM algorithm where the unaligned target words are inserted into the templates and minimum templates are combined into bigger templates (Galley et al., 2006). Thus the improvement they reported is rather indirect, leading us to wonder how much improvement the syntactic alignment model can directly bring to a SSMT system. Some other issues of SSMT not fully addressed before are highlighted below:

1. Normalization of TTS templates. Galley et al. (2006) mentioned that with only the minimum templates extracted from GHKM (Galley et al., 2004), normalizing the template probability based on its tree pattern "can become extremely biased", due to the fact that bigger templates easily get high probabilities. They instead use a joint model where the templates are normalized based on the root of their tree patterns and show empirical results for that. There is no systematic comparison of different normalization methods.

2. Decomposition model of a TTS transducer (or syntactic language model in synchronous SSMT). There is no explicit modeling for the decomposition of a syntax tree in the TTS transducer (or the probability of the syntactic tree in a synchronous SSMT). Most systems simply use a uniform model (Liu et al., 2006; Huang et al., 2006) or implicitly consider it with a joint model producing both syntax trees and the translations (Galley et al., 2006).

3. Use of semantics. Using semantic features in a SSMT is a natural step along the way towards generating more refined models across languages. The statistical approach to semantic role labeling has been well studied (Xue and Palmer, 2004; Ward et al., 2004; Toutanova et al., 2005), but there is no work attempting to use such information in SSMT, to our limited knowledge.

This paper proposes novel methods towards solving these problems. Specifically, we compare three ways of normalizing the TTS templates based on the tree pattern, the root of the tree pattern, and the first-level expansion of the tree pattern respectively, in the context of hard counting and EM estimation; we present a syntactic alignment framework integrating both the template re-estimation and insertion of unaligned target words; we use a subtree-based n-gram model to address the decomposition of the syntax trees in TTS transducer (or the syntactic language model for synchronous SSMT); we use a statistical classifier to label the semantic roles defined by PropBank (Palmer et al., 2005) and try different ways of using the semantic features in a TTS transducer.

We chose the TTS transducer instead of synchronous SSMT for two reasons. First, the decoding algorithm for the TTS transducer has lower computational complexity, which makes it easier to integrate a complex decomposition model. Second, the TTS Transducer can be easily integrated with semantic role features since the syntax tree is present, and it's not clear how to do this in a synchronous SSMT system. The remainder of the paper will focus on introducing the improved TTS transducer and is organized as follows: Section 2 describes the implementation of a basic TTS transducer; Section 3 describes the components of the improved TTS transducer; Section 4 presents the empirical results and Section 5 gives the conclusion.

2 A Basic Tree-to-string Transducer for Machine Translation

The TTS transducer, as a generalization to the finite state transducer, receives a tree structure as its input and recursively applies TTS templates to generate the target string. For simplicity, usually only one state is used in the TTS transducer, i.e., a TTS template will always lead to the same outcome wherever it is used. A TTS template is composed of a left-hand side (LHS) and a right-hand side (RHS), where LHS is a subtree pattern and RHS is a sequence of the variables and translated words. The variables in the RHS of a template correspond to the bottom level non-terminals in the LHS's subtree pattern, and their relative order indicates the permutation desired at the point where the template is ap-

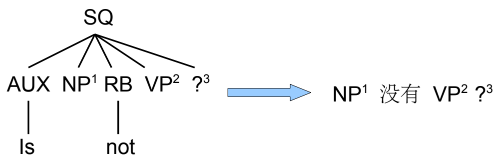

Figure 1: A TTS Template Example

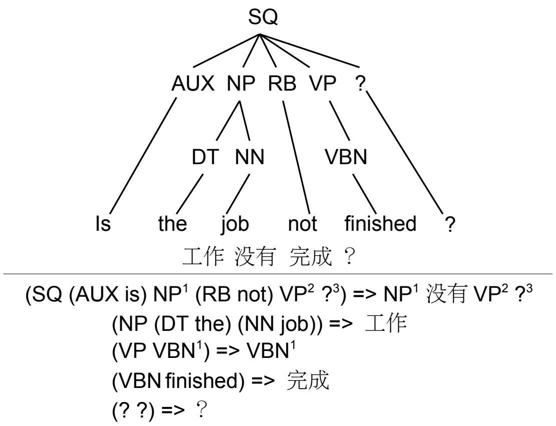

Figure 2: Derivation Example

plied to translate one language to another. The variables are further transformed and the recursive process goes on until there are no variables left. The formal description of a TTS transducer is described in Graehl and Knight (2004), and our baseline approach follows the *Extended Tree-to-String Transducer* defined in (Huang et al., 2006). Figure 1 gives an example of the English-to-Chinese TTS template, which shows how to translate a skeleton YES/NO question from English to Chinese. NP^1 and VP^2 are the variables whose relative position in the translation are determined by the template while their actual translations are still unknown and dependent on the subtrees rooted at them; and the English words *Is* and *not* are translated into the Chinese word *MeiYou* in the context of the template. The superscripts attached on the variables are used to distinguish the non-terminals with identical names (if there is any). Figure 2 shows the steps of transforming the English sentence *"Is the job not finished ?"* to the corresponding Chinese.

For a given derivation (decomposition) of a syntax tree, the translation probability is computed as the product of the templates which generate both the source syntax trees and the target translations. In theory, the translation model should sum over all possible derivations generating the target translation, but in practice, usually only the best derivation is considered:

$$Pr(S|T, D^*) = \prod_{t \in D^*} Weight(t)$$

Here, S denotes the target translation, T denotes the source syntax tree, and D^* denotes the best derivation of T. The implementation of a TTS transducer can be done either top down with memoization to the visited subtrees (Huang et al., 2006), or with a bottom-up dynamic programming (DP) algorithm (Liu et al., 2006). This paper uses the latter approach, and the algorithm is sketched in Figure 3. For the baseline approach, only the translation model and n-gram model for the target language are used:

$$S^* = \operatorname*{argmax}_S Pr(T|S) = \operatorname*{argmax}_S Pr(S)Pr(S|T)$$

Since the n-gram model tends to favor short translations, a penalty is added to the translation templates with fewer RHS symbols than LHS leaf symbols:

$$Penalty(t) = \exp(|t.RHS| - |t.LHSLeaf|)$$

where $|t.RHS|$ denotes the number of symbols in the RHS of t, and $|t.LHSLeaf|$ denotes the number of leaves in the LHS of t. The length penalty is analogous to the length feature widely used in log-linear models for MT (Huang et al., 2006; Liu et al., 2006; Och and Ney, 2004). Here we distribute the penalty into TTS templates for the convenience of DP, so that we don't have to generate the N-best list and do re-ranking. To speed up the decoding, standard beam search is used.

In Figure 3, $BinaryCombine$ denotes the target-size binarization (Huang et al., 2006) combination. The translation candidates of the template's variables, as well as its terminals, are combined pairwise in the order they appear in the RHS of the template. f_i denotes a combined translation, whose probability is equal to the product of the probabilities of the component translations, the probability of the rule, the n-gram probability of connecting the component translations, and the length penalty of

Match(v, t): the descendant tree nodes of *v*, which match the variables in template *t*
$v.sk$: the stack associated with tree node v
In(c_j, f_i): the translation candidate of c_j which is chosen to combine f_i

---

for all tree node *v* in bottom-up order **do**
  for all template *t* applicable at *v* **do**
    $\{c_1, c_2, ..., c_l\}$=Match(v, t);
    $\{f_1, f_2, ..., f_m\}$ = BinaryCombine($c_1.sk, c_2.sk, ..., c_n.sk, t$);
    for i=1:m **do**
      Pr(f_i) = $\prod_{j=1}^{l}$Pr(In(c_j, f_i)) $\cdot$ Weight(t)$^\beta$$\cdot$ Lang(v, t, f_i)$^\gamma$$\cdot$ Penalty(t)$^\alpha$;
      Add $(f_i, Pr(f_i))$ to $v.sk$;
  Prune $v.sk$;

Figure 3: Decoding Algorithm

the template. α, β and γ are the weights of the length penalty, the translation model, and the n-gram language model, respectively. Each state in the DP chart denotes the best translation of a tree node with a certain *prefix* and *suffix*. The length of the *prefix* and the *suffix* is equal to the length of the n-gram model minus one. Without the beam pruning, the decoding algorithm runs in $O(N^{4(n-1)}RPQ)$, where *N* is the vocabulary size of the target language, *n* is the length of the n-gram model, R is the maximum number of templates applicable to one tree node, P is the maximum number of variables in a template, and Q is the number of tree nodes in the syntax tree. The DP algorithm works for most systems in the paper, and only needs to be slightly modified to encode the subtree-based n-gram model described in Section 3.3.

3 Improved Tree-to-string Transducer for Machine Translation

3.1 Normalization of TTS Templates

Given the story that translations are generated based on the source syntax trees, the weight of the template is computed as the probability of the target strings given the source subtree:

$$Weight(t) = \frac{\#(t)}{\#(t' : LHS(t') = LHS(t))}$$

Such normalization, denoted here as *TREE*, is used in most tree-to-string template-based MT systems (Liu et al., 2007; Liu et al., 2006; Huang et al., 2006). Galley et al. (2006) proposed an alteration in synchronous SSMT which addresses the probability of both the source subtree and the target string given the root of the source subtree:

$$Weight(t) = \frac{\#(t)}{\#(t' : root(t') = root(t))}$$

This method is denoted as *ROOT*. Here, we propose another modification:

$$Weight(t) = \frac{\#(t)}{\#(t' : cfg(t') = cfg(t))} \quad (1)$$

cfg in Equation 1 denotes the first level expansion of the source subtree and the method is denoted as *CFG*. *CFG* can be thought of as generating both the source subtree and the target string given the first level expansion of the source subtree. *TREE* focuses on the conditional probability of the target string given the source subtree, *ROOT* focuses on the joint probability of both the source subtree and the target string, while *CFG*, as something of a compromise between *TREE* and *ROOT*, hopefully can achieve a combined effect of both of them. Compared with *TREE*, *CFG* favors the one-level context-free grammar like templates and gives penalty to the templates bigger (in terms of the depth of the source subtree) than that. It makes sense considering that the big templates, due to their sparseness in the corpus, are often assigned unduly large probabilities by *TREE*.

3.2 Syntactic Word Alignment

The idea of building a syntax-based word alignment model has been explored by May and Knight (2007), with an algorithm working from the root tree node down to the leaves, recursively replacing the variables in the matched tree-to-string templates until there are no such variables left. The TTS templates they use are initially gathered using GHKM

1. Run GIZA++ to get the initial word alignment, use GHKM to gather translation templates, and compute the initial probability as their normalized frequency.
2. Collect all the one-level subtrees in the training corpus containing only non-terminals and create TTS templates addressing all the permutations of the subtrees' leaves if its spanning factor is not greater than four, or only the monotonic translation template if its spanning factor is greater than four. Collect all the terminal rules in the form of $A \rightarrow B$ where A is one source word, B is the consecutive target word sequence up to three words long, and A, B occurs in some sentence pairs. These extra templates are assigned a small probability 10^{-6}.
3. Run the EM algorithm described in (Graehl and Knight, 2004) with templates obtained in step 1 and step 2 to re-estimate their probabilities.
4. Use the templates from step 3 to compute the viterbi word alignment.
5. The templates not occurring in the viterbi derivation are ignored and the probability of the remaining ones are re-normalized based on their frequency in the viterbi derivation.

Figure 4: Steps generating the refined TTS templates

(Galley et al., 2004) with the word alignment computed by GIZA++ and re-estimated using EM, ignoring the alignment from Giza++. The refined word alignment is then fed to the expanded GHKM (Galley et al., 2006), where the TTS templates will be combined with the unaligned target words and re-estimated in another EM framework. The syntactic alignment proposed here shares the essence of May and Knight's approach, but combines the re-estimation of the TTS templates and insertion of the unaligned target words into a single EM framework. The process is described in Figure 4. The insertion of the unaligned target words is done implicitly as we include the extra terminal templates in Figure 4, and the extra non-terminal templates ensure that we can get a complete derivation forest in the EM training. The last viterbi alignment step may seem unnecessary given that we already have the EM-estimated templates, but in experiments we find that it produces better result by cutting off the noisy (usually very big) templates resulting from the poor alignments of GIZA++.

3.3 Tree Decomposition Model

A deficiency of the translation model for tree-to-string transducer is that it cannot fully address the decomposition probability of the source syntax trees. Though we can say that *ROOT/CFG* implicitly includes the decomposition model, a more direct and explicit modeling of the decomposition is still desired. Here we propose a novel n-gram-like model to solve this problem. The probability of a decomposition (derivation) of a syntax tree is computed as the product of the n-gram probability of the decomposed subtrees conditioned on their ascendant subtrees. The formal description of the model is in Equation 2, where D denotes the derivation and $PT(st)$ denotes the direct parent subtree of st.

$$Pr(D|T) = \prod_{\substack{\text{subtrees} \\ st \in D}} Pr(st|PT(st), PT(PT(st)), ...) \tag{2}$$

Now, with the decomposition model added in, the probability of the target string given the source syntax tree is computed as:

$$Pr(S|T) = Pr(D^*|T) \times Pr(S|T, D^*)$$

To encode this n-gram probability of the subtrees in the decoding process, we need to expand the state space of the dynamic programming algorithm in Figure 3, so that each state represents not only the prefix/suffix of the partial translation, but also the decomposition history of a tree node. For example, with a bigram tree model, the states should include the different subtrees in the LHS of the templates used to translate a tree node. With bigger n-grams, more complex history information should be encoded in the states, and this leads to higher computational complexity. In this paper, we only consider the tree n-gram up to size 2. It is not practical to search the full state space; instead, we modify the beam search algorithm in Figure 3 to encode the decomposition history information. The modified algorithm for the tree bigram creates a stack for each tree pattern occurring in the templates applicable to a tree node. This ensures that for each tree node, the decompositions headed with different subtrees have equal number of translation candidates surviving to the upper phase. The function

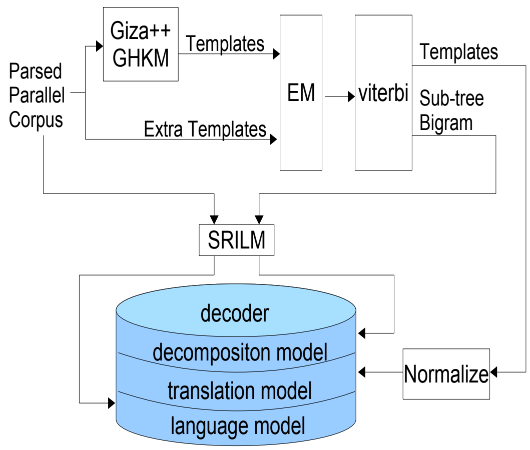

Figure 5: Flow graph of the system with all components integrated

$BinaryCombine$ is almost the same as in Figure 3, except that the translation candidates (states) of each tree node are grouped according to their associated subtrees. The bigram probabilities of the subtrees can be easily computed with the viterbi derivation in last subsection. Also, a weight should be assigned to this component. This tree n-gram model can be easily adapted and used in synchronous SSMT systems such as May and Knight (2007), Galley et al. (2006). The flow graph of the final system with all the components integrated is shown in Figure 5.

3.4 Use of Semantic Roles

Statistical approaches to MT have gone through word-based systems, phrase-based systems, and syntax-based systems. The next generation would seem to be semantic-based systems. We use PropBank (Palmer et al., 2005) as the semantic driver in our TTS transducer because it is built upon the same corpus (the Penn Treebank) used to train the statistical parser, and its shallow semantic roles are more easily integrated into a TTS transducer. A MaxEntropy classifier, with features following Xue and Palmer (2004) and Ward et al. (2004), is used to generate the semantic roles for each verb in the syntax trees. We then replace the syntactic labels with the semantic roles so that we have more general tree labels, or combine the semantic roles with the syntactic labels to generate more refined tree node labels. Though semantic roles are associated with the verbs, it is not feasible to differentiate the roles of different

	NP VP	VP NP
(S NP-agent VP)	0.983	0.017
(S NP-patient VP)	0.857	0.143

Table 1: The *TREE*-based weights of the skeleton templates with *NP* in different roles

verbs due to the data sparseness problem. If some tree nodes are labeled different roles for different verbs, those semantic roles will be ignored.

A simple example demonstrating the need for semantics in the TTS transducer is that in English-Chinese translation, the *NP VP* skeleton phrase is more likely to be inverted when *NP* is in a *patient* role than when it is in an *agent* role. Table 1 shows the *TREE*-based weights of the 4 translation templates, computed based on our training corpus. This shows that the difference caused by the roles of *NP* is significant.

4 Experiment

We used 74,597 pairs of English and Chinese sentences in the FBIS data set as our experimental data, which are further divided into 500 test sentence pairs, 500 development sentence pairs and 73597 training sentence pairs. The test set and development set are selected as those sentences having fewer than 25 words on the Chinese side. The translation is from English to Chinese, and Charniak (2000)'s parser, trained on the Penn Treebank, is used to generate the syntax trees for the English side. The weights of the MT components are optimized based on the development set using a grid-based line search. The Chinese sentence from the selected pair is used as the single reference to tune and evaluate the MT system with word-based BLEU-4 (Papineni et al., 2002). Huang et al. (2006) used character-based BLEU as a way of normalizing inconsistent Chinese word segmentation, but we avoid this problem as the training, development, and test data are from the same source.

4.1 Syntax-Based System

The decoding algorithm described in Figure 3 is used with the different normalization methods described in Section 3.1 and the results are summarized in Table 2. The TTS templates are extracted using GHKM based on the many-to-one alignment

	Baseline		Syntactic Alignment		Subtree bigram	
	dev	test	dev	test	dev	test
TREE	12.29	8.90	13.25	9.65	14.84	10.61
ROOT	12.41	9.66	13.72	10.16	14.24	10.66
CFG	13.27	9.69	14.32	10.29	15.30	10.99
PHARAOH	9.04	7.84				

Table 2: BLEU-4 scores of various systems with the syntactic alignment and subtree bigram improvements added incrementally.

from Chinese to English obtained from GIZA++. We have tried using alignment in the reverse direction and the union of both directions, but neither of them is better than the Chinese-to-English alignment. The reason, based on the empirical result, is simply that the Chinese-to-English alignments lead to the maximum number of templates using GHKM. A modified Kneser-Ney bigram model of the Chinese sentence is trained using SRILM (Stolcke, 2002) using the training set. For comparison, results for Pharaoh (Koehn, 2004), trained and tuned under the same condition, are also shown in Table 2. The phrases used in Pharaoh are extracted as the pair of longest continuous spans in English and Chinese based on the union of the alignments in both direction. We tried using alignments of different directions with Pharaoh, and find that the union gives the maximum number of phrase pairs and the best BLEU scores. The results show that the TTS transducers all outperform Pharaoh, and among them, the one with *CFG* normalization works better than the other two.

We tried the three normalization methods in the syntactic alignment process in Figure 4, and found that the initialization (step 1) and viterbi alignment (step 3 and 4) based on the least biased model *ROOT* gave the best performance. Table 2 shows the results with the final template probability re-normalized (step 5) using *TREE*, *ROOT* and *CFG* respectively. We can see that the syntactic alignment brings a reasonable improvement for the TTS transducer no matter what normalization method is used. To test the effect of the subtree-based n-gram model, SRILM is used to compute a modified Kneser-Ney bigram model for the subtree patterns used in the viterbi alignment. The last 3 lines in Table 2 show the improved results by further incorporating the subtree-based bigram model. We can see that the difference of the three normalization methods is lessened and *TREE*, the weakest normalization in terms of addressing the decomposition probability, gets the biggest improvement with the subtree-based bigram model added in.

4.2 Semantic-Based System

Following the standard division, our max-entropy based SRL classifier is trained and tuned using sections 2-21 and section 24 of PropBank, respectively. The F-score we achieved on section 23 is 88.70%. We repeated the experiments in last section with the semantic labels generated by the SRL classifier. Table 3 shows the results, comparing the non-semantic-based systems with similar systems using the refined and general semantic labels, respectively. Unfortunately, semantic based systems do not always outperform the syntactic based systems. We can see that for the baseline systems based on *TREE* and *ROOT*, semantic labels improve the results, while for the other systems, they are not really better than the syntactic labels. Our approach to semantic roles is preliminary; possible improvements include associating role labels with verbs and backing off to the syntactic-label based models from semantic-label based TTS templates. In light of our results, we are optimistic that more sophisticated use of semantic features can further improve a TTS transducer's performance.

5 Conclusion

This paper first proposes three enhancements to the TTS transducer: first-level expansion-based normalization for TTS templates, a syntactic alignment framework integrating the insertion of unaligned target words, and a subtree-based n-gram model addressing the tree decomposition probability. The experiments show that the first-level expansion-based

	No Semantic Labels			Refined Labels			General Labels		
	Baseline	Syntactic Alignment	Subtree Bigram	Baseline	Syntactic Alignment	Subtree Bigram	Baseline	Syntactic Alignment	Subtree Bigram
TREE	8.90	9.65	10.61	9.40	10.25	10.42	9.40	10.02	10.47
ROOT	9.66	10.16	10.66	9.89	10.32	10.43	9.82	10.17	10.42
CFG	9.69	10.29	10.99	9.66	10.16	10.33	9.58	10.25	10.59

Table 3: BLEU-4 scores of semantic-based systems on test data. As in Table 2, the syntactic alignment and subtree bigram improvements are added incrementally within each condition.

normalization for TTS templates is better than the root-based one and the tree-based one; the syntactic alignment framework and the n-gram based tree decomposition model both improve a TTS transducer's performance. Our experiments using PropBank semantic roles in the TTS transducer show that the approach has potential, improving on our baseline system. However, adding semantic roles does not improve our best TTS system.

References

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *The Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *Proceedings of ACL-07*, pages 17–24.

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

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 *Proceedings of COLING/ACL-06*, pages 961–968, July.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proceedings of NAACL-04*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, MA.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *The Sixth Conference of the Association for Machine Translation in the Americas*, pages 115–124.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING/ACL-06*, Sydney, Australia, July.

Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *Proceedings of ACL-07*, Prague.

J. May and K. Knight. 2007. Syntactic re-alignment models for machine translation. In *Proceedings of EMNLP*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4).

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL-02*.

William C. Rounds. 1970. Mappings and grammars on trees. *Mathematical Systems Theory*, 4(3):257–287.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, volume 2, pages 901–904.

J. W. Thatcher. 1970. Generalized^2 sequential machine maps. *J. Comput. System Sci.*, 4:339–367.

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL-05*, pages 589–596.

Wayne Ward, Kadri Hacioglu, James Martin, , and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of EMNLP*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of NAACL-06*, pages 256–263.