Bayesian Learning of Phrasal Tree-to-String Templates

Ding Liu and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627
{dliu, gildea}@cs.rochester.edu

Abstract

We examine the problem of overcoming noisy word-level alignments when learning tree-to-string translation rules. Our approach introduces new rules, and re-estimates rule probabilities using EM. The major obstacles to this approach are the very reasons that word-alignments are used for rule extraction: the huge space of possible rules, as well as controlling overfitting. By carefully controlling which portions of the original alignments are re-analyzed, and by using Bayesian inference during re-analysis, we show significant improvement over the baseline rules extracted from word-level alignments.

1 Introduction

Non-parametric Bayesian methods have been successfully applied to directly learn phrase pairs from a bilingual corpus with little or no dependence on word alignments (Blunsom et al., 2008; DeNero et al., 2008). Because such approaches directly learn a generative model over phrase pairs, they are theoretically preferable to the standard heuristics for extracting the phrase pairs from the many-to-one word-level alignments produced by the IBM series models (Brown et al., 1993) or the Hidden Markov Model (HMM) (Vogel et al., 1996). We wish to apply this direct, Bayesian approach to learn better translation rules for syntax-based statistical MT (SSMT), by which we specifically refer to MT systems using Tree-to-String (TTS) translation templates derived from syntax trees (Liu et al., 2006; Huang et al., 2006; Galley et al., 2006; May and Knight, 2007), as opposed to *formally syntactic* systems such as Hiero (Chiang, 2007). The stumbling block preventing us from taking this approach is the extremely large space of possible TTS templates when no word alignments are given. Given a sentence pair and syntax tree over one side, there are an exponential number of potential TTS templates and a polynomial number of phrase pairs. In this paper, we explore methods for restricting the space of possible TTS templates under consideration, while still allowing good templates to emerge directly from the data as much as possible. We find an improvement in translation accuracy through, first, using constraints to limit the number of new templates, second, using Bayesian methods to limit which of these new templates are favored when re-analyzing the training data with EM, and, third, experimenting with different renormalization techniques for the EM re-analysis.

We introduce two constraints to limit the number of TTS templates that we extract directly from tree/string pairs without using word alignments. The first constraint is to limit direct TTS template extraction to the part of the corpus where word alignment tools such as GIZA++ do poorly. There is no reason not to re-use the good alignments from GIZA++, which holds a very competitive baseline performance. As already mentioned, the noisy alignments from GIZA++ are likely to cross the boundaries of the tree constituents, which leads to comparatively big TTS templates. We use this fact as a heuristic to roughly distinguish noisy from good word alignments.[1] Here we define *big templates* as those with more than 8 symbols in their right hand sides (RHSs). The word alignments in big templates are considered to be noisy and will be recomposed by extracting smaller TTS templates. Another reason to do extraction on big templates is that the applicability of big templates to new sentences is very limited due to their size, and the portion of the training data from which they are extracted is effectively wasted. The second constraint, after choosing the

[1] Precisely differentiating the noisy/good word alignments is as hard as correctly aligning the words.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1308–1317,
Singapore, 6-7 August 2009.

extraction site, is to extract the TTS templates all the way down to the leaves of the hosting templates. This constraint limits the number of possible left hand sides (LHSs) to be equal to the number of tree nodes in the hosting templates. The entire extraction process can be summarized in 3 steps:

1. Compute word alignments using GIZA++, and generate the basic TTS templates.

2. Select big templates from the basic TTS templates in step 1, and extract smaller TTS templates all the way down to the bottom from big templates, without considering the pre-computed word alignments.

3. Combine TTS templates from step 1 and step 2 and estimate their probabilities using Variational Bayes with a Dirichlet Process prior.

In step 2, since there are no constraints from the pre-computed word alignments, we have complete freedom in generating all possible TTS templates to overcome noisy word alignments. We use variational EM to approximate the inference of our Bayesian model and explore different normalization methods for the TTS templates. A two-stage normalization is proposed by combining LHS-based normalization with normalization based on the root of the LHS, and is shown to be the best model when used with variational EM.

Galley et al. (2006) recompose the TTS templates by inserting unaligned target words and combining small templates into bigger ones. The recomposed templates are then re-estimated using the EM algorithm described in Graehl and Knight (2004). This approach also generates TTS templates beyond the precomputed word alignments, but the freedom is only granted over unaligned target words, and most of the pre-computed word alignments remain unchanged. Other prior approaches towards improving TTS templates focus on improving the word alignment performance over the classic models such as IBM series models and Hidden Markov Model (HMM), which do not consider the syntactic structure of the aligning languages and produce syntax-violating alignments. DeNero and Klein (2007) use a syntax-based distance in an HMM word alignment model to favor syntax-friendly alignments. Fossum et al. (2008) start from the GIZA++ alignment and incrementally delete bad links based on a discriminative model with syntactic features. This approach can only find a better subset of the GIZA++ alignment and requires a parallel corpus with gold-standard word alignment for training the discriminative model. May and Knight (2007) factorize the word alignment into a set of re-orderings represented by the TTS templates and build a hierarchical syntax-based word alignment model. The problem is that the TTS templates are generated by the word alignments from GIZA++, which limits the potential of the syntactic re-alignment. As shown by these prior approaches, directly improving the word alignment either falls into the framework of many-to-one alignment, or is substantially confined by the word alignment it builds upon.

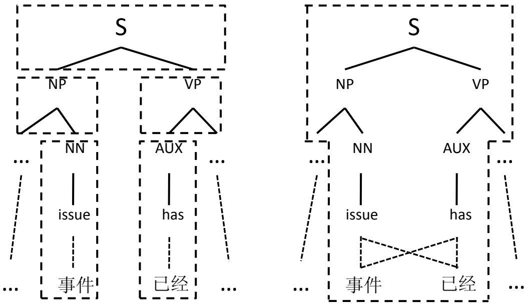

Figure 1: 5 small TTS templates are extracted based on the correct word alignments (left), but only 1 big TTS template (right) can be extracted when the cross-boundary noisy alignments are added in.

The remainder of the paper focuses on the Bayesian approach to learning TTS templates and is organized as follows: Section 2 describes the procedure for generating the candidate TTS templates; Section 3 describes the inference methods used to learn the TTS templates; Section 4 gives the empirical results, Section 5 discusses the characteristics of the learned TTS templates, and Section 6 presents the conclusion.

2 Extracting Phrasal TTS Templates

The Tree-to-String (TTS) template, the most important component of a SSMT system, usually contains three parts: a fragment of a syntax tree in its left hand side (LHS), a sequence of words and variables in its right hand side (RHS), and a probability indicating how likely the template is to be used in translation. The RHS of a TTS template shows one possible translation and reordering of its LHS. The variables in a TTS template are further transformed using other TTS templates, and the recursive process continues until there are no variables left. There are two ways

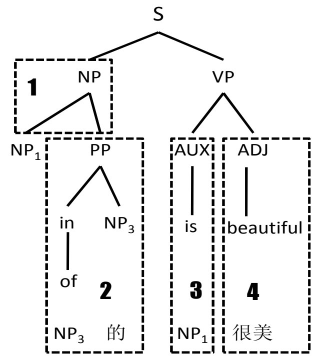

Figure 2: Examples of valid and invalid templates extracted from a big Template. Template 1, invalid, doesn't go all the way down to the bottom. Template 2 is valid. Template 3, invalid, doesn't have the same set of variables in its LHS/RHS. Template 4, invalid, is not a phrasal TTS template.

that TTS templates are commonly used in machine translation. The first is *synchronous parsing* (Galley et al., 2006; May and Knight, 2007), where TTS templates are used to construct synchronous parse trees for an input sentence, and the translations will be generated once the synchronous trees are built up. The other way is the TTS *transducer* (Liu et al., 2006; Huang et al., 2006), where TTS templates are used just as their name indicates: to transform a source parse tree (or forest) into the proper target string. Since synchronous parsing considers all possible synchronous parse trees of the source sentence, it is less constrained than TTS transducers and hence requires more computational power. In this paper, we use a TTS transducer to test the performance of different TTS templates, but our techniques could also be applied to SSMT systems based on synchronous parsing.

2.1 Baseline Approach: TTS Templates Obeying Word Alignment

TTS templates are commonly generated by decomposing a pair of aligned source syntax tree and target string into smaller pairs of tree fragments and target string (i.e., the TTS templates). To keep the number of TTS templates to a manageable scale, only the non-decomposable TTS templates are generated. This algorithm is referred to as GHKM (Galley et al., 2004) and is widely used in SSMT systems (Galley et al., 2006; Liu et al., 2006; Huang et al., 2006). The word alignment used in GHKM is usually computed independent of the syntactic structure, and as DeNero and Klein (2007) and May and Knight (2007) have noted, is not the best for SSMT systems. In fact, noisy word alignments cause more damage to a SSMT system than to a phrase based SMT system, because the TTS templates can only be derived from tree constituents. If some noisy alignments happen to cross over the boundaries of two constituents, as shown in Figure 2, a much bigger tree fragment will be extracted as a TTS template. Even though the big TTS templates still carry the original alignment information, they have much less chance of getting matched beyond the syntax tree where they were extracted, as we show in Section 4. In other words, a few cross-boundary noisy alignments could disable a big portion of a training syntax tree, while for a phrase-based SMT system, their effect is limited to the phrases they align. As a rough measure of how the training corpus is affected by the big templates, we calculated the distribution of target words in big and non-big TTS templates. The word alignment is computed using GIZA++[2] for the selected 73,597 sentence pairs in the FBIS corpus in both directions and then combined using union and heuristic diagonal growing (Koehn et al., 2003). Table 1 shows that big templates consume 20.1% to 45.9% of the training corpus depending on different types of word alignments. The statistics indicate that a significant portion of the training corpus is simply wasted, if the TTS templates are extracted based on word alignments from GIZA++. On the other hand, it shows the potential for improving an SSMT system if we can efficiently re-use the wasted training corpus. By further examining the selected big templates, we find that the most common form of big templates is a big skeleton template starting

Ch-En	En-Ch	Union	Heuristic
28.6%	33.0%	45.9%	20.1%

Table 1: Percentage of corpus used to generate big templates, based on different word alignments

	9-12	13-20	≥21
Ch-En	18.2%	17.4%	64.4%
En-Ch	15.9%	20.7%	63.4%
Union	9.8%	15.1%	75.1%
Heuristic	24.6%	27.9%	47.5%

Table 2: In the selected big templates, the distribution of words in the templates of different sizes, which are measured based on the number of symbols in their RHSs

[2] GIZA++ is available at
http://www.fjoch.com/GIZA++.html

from the root of the source syntax tree, and having many terminals (words) misaligned in the bottom. Table 2 shows, in the selected big templates, the distribution of words in the templates of different sizes (measured based on the number of symbols in their RHS). We can see that based on either type of word alignment, the most common big templates are the TTS templates with more than 20 symbols in their RHSs, which are generally the big skeleton templates. The advantage of such big skeleton templates is that they usually have good marginal accuracy[3] and allow accurate smaller TTS templates to emerge.

2.2 Liberating Phrasal TTS Templates From Noisy Word Alignments

To generate better TTS templates, we use a more direct way than modifying the underlying word alignment: extract smaller phrasal TTS templates from the big templates without looking at their pre-computed word alignments. We define *phrasal* TTS templates as those with more than one symbol (word or non-terminal) in their LHS. The reason to consider only phrasal TTS templates is that they are more robust than the word-level TTS templates in addressing the complicated word alignments involved in big templates, which are usually not the simple type of one-to-many or many-to-one. Abandoning the pre-computed word alignments in big templates, an extracted smaller TTS template can have many possible RHSs, as long as the two sides have the same set of variables. Note that the freedom is only given to the alignments of the words; for the variables in the big templates, we respect the pre-computed word alignments. To keep the extracted smaller TTS templates to a manageable scale, the following two constraints are applied:

1. The LHS of extracted TTS templates should go all the way down to the bottom of the LHS of the big templates. This constraint ensures that at most N LHSs can be extracted from one big Template, where N is the number of tree nodes in the big Template's LHS.

2. The number of leaves (including both words and variables) in an extracted TTS template's LHS should not exceed 6. This constraint limits the size of the extracted TTS templates.

(VP (AUX is) (ADJ beautiful)) → 很美

(PP (IN of) NP$_3$) → NP$_3$ 的

(NP NP$_1$ (PP (IN of) NP$_3$)) → NP$_3$ 的 NP$_1$

(NP NP$_1$ (PP (IN of) NP$_3$)) → NP$_3$ 的 NP$_1$ 很美

Figure 3: All valid templates that can be extracted from the example in Figure 2.1

```
for all template t do
    if size(t.rhs) $>$ 8 then
        for all tree node s in t.lhs do
            $subt$ = subtree(s, t.lhs);
            if leaf_num($subt$) $\leq$ 6 then
                for i=1:size(t.rhs) do
                    for j=i:size(t.rhs) do
                        if valid($subt$, i, j) then
                            create_template($subt$, i, j);
```

Figure 4: Algorithm that liberates smaller TTS Templates from big templates

As we show in Section 4, use of bigger TTS templates brings very limited performance gain.

Figure 2.2 describes the template liberating algorithm running in $O(NM^2)$, where N denotes the number of tree nodes in the LHS of the input big Template and M denotes the length of the RHS. In the algorithm, function *valid* returns *true* if there are the same set of variables in the left/right hand side of an extracted TTS template; *subtree(x, y)* denotes the sub-tree in *y* which is rooted at *x* and goes all the way down to *y*'s bottom. Figure 2.1 shows valid and invalid TTS templates which can be extracted from an example hosting TTS template. Note that, in order to keep the example simple, the hosting TTS template only has 4 symbols in its RHS, which does not qualify as a big template according to our definition. Figure 2.2 shows the complete set of valid TTS templates which can be extracted from the example TTS template. The subscripts of the non-terminals are used to differentiate identical non-terminals in different positions. The extraction process blindly releases smaller TTS templates from the big templates, among which only a small fraction are correct TTS templates. Therefore, we need an inference method to raise the weight of the correct templates and decrease the weight of the noisy templates.

[3] Here, marginal accuracy means the correctness of the TTS template's RHS corresponding to its LHS.

3 Estimating TTS Template Probability

The Expectation-Maximization (EM) algorithm (Dempster et al., 1977) can be used to estimate the TTS templates' probabilities, given a generative model addressing how a pair of source syntax tree and target string is generated. There are two commonly used generative models for syntax-based MT systems, each of which corresponds to a normalization method for the TTS templates. The LHS-based normalization (LHSN) (Liu et al., 2006; Huang et al., 2006), corresponds to the generative process where the source syntax subtree is first generated, and then the target string is generated given the source syntax subtree. The other one is normalization based on the root of the LHS (ROOTN) (Galley et al., 2006), corresponding to the generative process where, given the root of the syntax subtree, the LHS syntax subtree and the RHS string are generated simultaneously. By omitting the decomposition probability in the LHS-based generative model, the two generative models share the same formula for computing the probability of a training instance:

$$Pr(T,S) = \sum_{R} Pr(T,S,R) = \sum_{R} \left(\prod_{t \in R} Pr(t) \right)$$

where T and S denote the source syntax tree and target string respectively, R denotes the decomposition of (T,S), and t denotes the TTS template. The expected counts of the TTS templates can then be efficiently computed using an inside-outside-like dynamic programming algorithm (May and Knight, 2007).

LHSN, as shown by Galley et al. (2006), cannot accurately restore the true conditional probabilities of the target sentences given the source sentences in the training corpus. This indicates that LHSN is not good at predicting unseen sentences or at translating new sentences. But this deficiency does not affect its ability to estimate the expected counts of the TTS templates, because the posteriors of the TTS templates only depend on the comparative probabilities of the different derivations of a training instance (a pair of tree and string). In fact, as we show in Section 4, LHSN is better than ROOTN in liberating smaller TTS templates out of the big templates, since it is less biased to the big templates in the EM training.[4] Because the two normalization methods have their own strength and weakness, both of them are used in our EM algorithm: LHSN is used in all EM iterations except the last one to compute the expected counts of the TTS templates, and ROOTN is used in the last EM iteration to compute the final probabilities of the TTS templates. This two-stage normalization method is denoted as MIXN in this paper.

```
E-step:
for all pair of syntax tree $T$ and target string $S$ do
    for all TTS Template $t$ do
        $EC(t) += \frac{\sum_{R:t \in R} Pr(T,S,R)^{\beta}}{\sum_{R'} Pr(T,S,R')^{\beta}}$;
Increase $\beta$;

M-step:
for all TTS Template $t$ do
    if it is the last iteration then
        $Pr(t) = \frac{EC(t)}{\sum_{t':t'.root=t.root} EC(t')}$;
    else
        $Pr(t) = \frac{EC(t)}{\sum_{t':t'.lhs=t.lhs} EC(t')}$;
```

Figure 5: EM Algorithm For Estimating TTS Templates

Deterministic Annealing (Rose et al., 1992) is is used in our system to speed up the training process, similar to Goldwater et al. (2006). We start from a high temperature and gradually decrease the temperature to 1; we find that the initial high temperature can also help small templates to survive the initial iterations. The complete EM framework is sketched in Figure 3, where β is the inverse of the specified temperature, and *EC* denotes the expected count.

3.1 Bayesian Inference with the Dirichlet Process Prior

Bayesian inference plus the Dirichlet Process (DP) have been shown to effectively prevent MT models from overfitting the training data (DeNero et al., 2008; Blunsom et al., 2008). A similar approach can be applied here for SSMT by considering each TTS template as a cluster, and using DP to adjust the number of TTS templates according to the training data. Note that even though there is a size limitation on the liberated phrasal TTS templates, standard EM will still tend to overfit the training data by pushing up the probabilities of the big templates from the noisy word alignments. The complete generative process, integrating the DP prior and the generative models described in

[4] Based on LHSN, the difference between the probability of a big Template and the product of the probabilities of its decomposing TTS templates is much less than the one based on ROOTN, thus LHSN gives comparably more expected counts to the smaller TTS templates than ROOTN.

```
for all TTS Template $t$ do
    if it is the last iteration then
        $Pr(t) = \frac{\exp(\psi(EC(t)+\alpha G_0(t)))}{\exp\big(\psi\big(\big(\sum_{t':t'.root=t.root} EC(t')\big)+\alpha\big)\big)}$;
    else
        $Pr(t) = \frac{\exp(\psi(EC(t)+\alpha G_0(t)))}{\exp\big(\psi\big(\big(\sum_{t':t'.lhs=t.lhs} EC(t')\big)+\alpha\big)\big)}$;
```

Figure 6: M-step of the Variational EM

Section 3.1, is given below:

$$\begin{aligned} \theta^r \mid \{\alpha_r, G_0^r\} &\sim DP(\alpha_r, G_0^r) \\ t \mid \theta^{t.root} &\sim \theta^{t.root} \\ (T,S) \mid \{SG, \{t\}, \theta\} &\sim SG(\{t\}, \theta) \end{aligned}$$

where G_0 is a base distribution of the TTS templates, t denotes a TTS template, $\theta^{t.root}$ denotes the multinomial distribution over TTS templates with the same root as t, SG denotes the generative model for a pair of tree and string in Section 3.1, and α is a free parameter which adjusts the rate at which new TTS templates are generated.

It is intractable to do exact inference under the Bayesian framework, even with a conjugate prior such as DP. Two methods are commonly used for approximate inference: Markov chain Monte Carlo (MCMC) (DeNero et al., 2008), and Variational Bayesian (VB) inference (Blunsom et al., 2008). In this paper, the latter approach is used because it requires less running time. The E-step of VB is exactly the same as standard EM, and in the M-step the digamma function ψ and the base distribution G_0 are used to increase the uncertainty of the model. Similar to standard EM, both LHS- and root-based normalizations are used in the M-step, as shown in Figure 3.1. For the TTS templates, which are also pairs of subtrees and strings, a natural choice of G_0 is the generative models described in Section 3.1. Because G_0 estimates the probability of the new TTS templates, the root-based generative model is superior to the LHS-based generative model and used in our approach.

3.2 Initialization

Since the EM algorithm only converges to a local minimum, proper initializations are needed to achieve good performance for both standard EM and variational EM. For the baseline templates derived from word alignments, the initial counts are set to the raw counts in the training corpus. For the templates blindly extracted from big templates, the raw count of a LHS tree fragment is distributed among their RHSs based on the likelihood of the template, computed by combining the word-based inside/outside scores. The algorithm is sketched in Figure 3.2, where the inside score *w_in(g)* is the product of the IBM Model 1 scores in both directions, computed based on the words in g's LHS and RHS. The outside score *w_out(g, t)* is computed similarly, except that the IBM Model 1 scores are computed based on the words in the hosting template t's LHS/RHS excluding the words in g's LHS/RHS. The initial probabilities of the TTS templates are then computed by normalizing their initial counts using LHSN or ROOTN.

```
for all big template $t$ do
    for all template $g$ extracted from $t$ do
        $g.count$ = $g.lhs.count$ = 0;
    for all template $g$ extracted from $t$ do
        $g.count$ += w_in($g$)×w_out($g$, $t$);
        $g.lhs.count$ += w_in($g$)×w_out($g$, $t$);
    for all template $g$ extracted from $t$ do
        $g.init$ += $\frac{g.count}{g.lhs.count}$;
```

Figure 7: Compute the initial counts of the liberated TTS templates

4 Experiments

We train an English-to-Chinese translation system using the FBIS corpus, where 73,597 sentence pairs are selected as the training data, and 500 sentence pairs with no more than 25 words on the Chinese side are selected for both the development and test data.[5] Charniak (2000)'s parser, trained on the Penn Treebank, is used to generate the English syntax trees. Modified Kneser-Ney trigram models are trained using SRILM (Stolcke, 2002) upon the Chinese portion of the training data. The trigram language model, as well as the TTS templates generated based on different methods, are used in the TTS transducer. The model weights of the transducer are tuned based on the development set using a grid-based line search, and the translation results are evaluated based on a single Chinese reference[6] using 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.

[5] The total 74,597 sentence pairs used in experiments are those in the FBIS corpus whose English part can be parsed using Charniak (2000)'s parser.

[6] BLEU-4 scores based on a single reference are much lower than the ones based on multiple references.

	E2C	C2E	Union	Heuristic
w/ Big	13.37	12.66	**14.55**	14.28
w/o Big	13.20	12.62	14.53	14.21

Table 3: BLEU-4 scores (test set) of systems based on GIZA++ word alignments

	≤ 5	≤ 6	≤ 7	≤ 8	$\leq \infty$
BLEU-4	14.27	14.42	14.43	14.45	14.55

Table 4: BLEU-4 scores (test set) of the *union* alignment, using TTS templates up to a certain size, in terms of the number of leaves in their LHSs

4.1 Baseline Systems

GHKM (Galley et al., 2004) is used to generate the baseline TTS templates based on the word alignments computed using GIZA++ and different combination methods, including *union* and the diagonal growing heuristic (Koehn et al., 2003). We also tried combining alignments from GIZA++ based on intersection, but it is worse than both single-direction alignments, due to its low coverage of training corpus and the incomplete translations it generates. The baseline translation results based on ROOTN are shown in Table 4.1. The first two columns in the table show the results of the two single direction alignments. *e2c* and *c2e* denote the many English words to one Chinese word alignment and the many Chinese words to one English word alignment, respectively. The two rows show the results with and without the big templates, from which we can see that removing the big templates does not affect performance much; this verifies our postulate that the big templates have very little chance of being used in the translation. Table 4.1, using the *union* alignments as the representative and measuring a template's size by the number of leaves in its LHS, also demonstrates that using big TTS templates brings very limited performance gain.

The result that the union-based combination outperforms either single direction alignments and even the heuristic-based combination, combined with the statistics of the disabled corpus in Section 2.2, shows that more disabled training corpus actually leads to better performance. This can be explained by the fact that the *union* alignments have the largest number of noisy alignments gathered together in the big templates, and thus have the least amount of noisy alignments which lead to small and low-quality TTS templates.

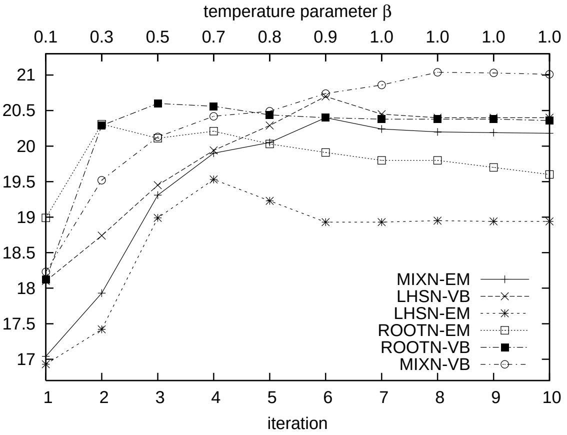

Figure 8: BLEU-4 scores (development set) of annealing EM and annealing VB in each iteration.

4.2 Learning Phrasal TTS Templates

To test our learning methods, we start with the TTS templates generated based on *e2c*, *c2e*, and *union* alignments using GHKM. This gives us 0.98M baseline templates. We use the big templates from the *union* alignments as the basis and extract 10.92M new phrasal TTS templates, which, for convenience, are denoted by NEW-PHR. Because based on Table 1 and Table 2 the *union* alignment has the greatest number of alignment links and therefore produces the largest rules, this gives us the greatest flexibility in realigning the input sentences. The baseline TTS templates as well as NEW-PHR are initialized using the method in Section 3.3 for both annealing EM and annealing VB. To simplify the experiments, the same Dirichlet Process prior is used for all multinomial distributions of the TTS templates with different roots. G_0 in the Dirichlet prior is computed based on the 1-level TTS templates selected from the baseline TTS templates, so that the big templates are efficiently penalized. The training algorithms follow the same annealing schedule, where the temperature parameter β is initialized to 0.1, and gradually increased to 1.

We experiment with the two training algorithms, annealing EM and annealing VB, with different normalization methods. The experimental results based on the development data are shown in Figure 4.2, where the free parameter α of annealing VB is set to 1, 100, and 100 respectively for ROOTN, LHSN, and MIXN. The results verify that LHSN is worse than ROOTN in predicting the translations, since MIXN outperforms LHSN with both annealing EM and VB. ROOTN is on par with MIXN and much better

	Max Likelihood		Annealing EM		Annealing VB	
	w/o new-phr	with new-phr	w/o new-phr	with new-phr	w/o new-phr	with new-phr
LHSN	14.05	13.16	14.31	15.33	14.82	16.15
ROOTN	14.50	13.49	14.90	16.06	14.76	16.12
MIXN	NA	NA	14.82	16.37	14.93	**16.84**

Table 5: BLEU-4 scores (test set) of different systems.

	Initial Template		Final Template	
	number	new-phr%	number	new-phr%
ROOTN	11.9M	91.8%	408.0K	21.9%
LHSN	11.9M	91.8%	557.2K	29.8%
MIXN	11.9M	91.8%	500.5K	27.6%

Table 6: The total number of templates and the percentage of NEW-PHR, in the beginning and end of annealing VB

than LHSN when annealing EM is used; but with annealing VB, it is outperformed by MIXN by a large margin and is even slightly worse than LHSN. This indicates that ROOTN is not giving large expected counts to NEW-PHR and leaves very little space for VB to further improve the results. For all the normalization methods, annealing VB outperforms annealing EM and maintains a longer ascending path, showing better control of overfitting for the Bayesian models. Figure 4.2 shows the optimized results of the development set based on annealing VB with different α. The best performance is achieved as α approaches 1, 100, and 100 for ROOTN, LHSN and MIXN respectively. The α parameter can be viewed as a weight used to balance the expected counts and the probabilities from G_0. Thus it is reasonable for LHSN and MIXN to have bigger optimal α than ROOTN, since ROOTN gives lower expected counts to NEW-PHR than LHSN and MIXN do.

To see the contribution of the phrasal template extraction in the performance gain, MT experiments are conducted by turning this component on and off. Results on the test set, obtained by using parameters optimized on the development set, are shown in Table 4.2. The template counts used in the Max-Likelihood training are the same as the ones used in the initialization of annealing EM and VB. Results show that for annealing EM and VB, use of NEW-PHR greatly improves performance, while for the Max-Likelihood training, use of NEW-PHR hurts performance. This is not surprising, because Max-Likelihood training cannot efficiently filter out the noisy phrasal templates introduced in the initial NEW-PHR. Another observation is that annealing VB does not always outperform annealing EM. With NEW-PHR turned on, annealing VB shows consistent superiority over annealing EM; while without NEW-PHR, it only outperforms annealing EM based on LHSN and MIXN, and the improvement is not as big as when NEW-PHR is turned on. This indicates that without NEW-PHR, there is less need to use VB to shrink down the size of the template set. Table 4.2 shows the statistics of the initial template set including NEW-PHR and the final TTS template set after annealing VB is conducted, where we can see annealing VB efficiently reduces NEW-PHR to a relatively small size and results in much more compact systems than the system based on the baseline templates from GIZA++ alignments. Comparing with the best GIZA++-based system *union*, our best system, utilizing NEW-PHR and the two-stage template normalization, demonstrates the strength of annealing VB by an absolute improvement of 2.29% in BLEU-4 score, from 14.55 to 16.84. This improvement is significant at $p < 0.005$ based on 2000 iterations of paired bootstrap re-sampling of the test set (Koehn, 2004).

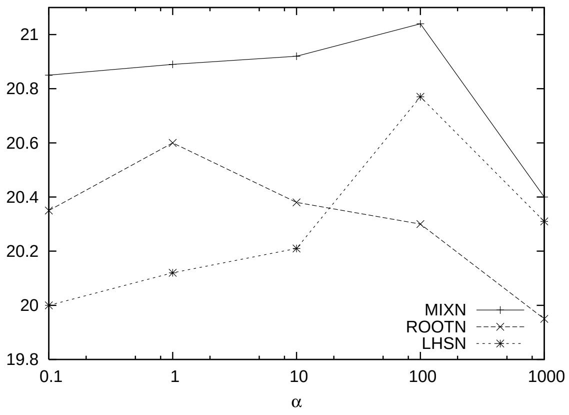

Figure 9: BLEU-4 scores (development set) of annealing VB with different α.

5 Discussion

Our experimental results are obtained based on a relatively small training corpus, the improved performance may be questionable when a larger training corpus is used. Someone may wonder if the performance gain primarily comes from the

Many-to-one Alignment	
(VP (VB make) (NP (DT a) (JJ complete) (NN statement)))	充分陈述
(S (VP VBG (NP (DT the) (NN mass) (NN line)) PP))	PP VBG 群众路线
(PP (TO to) (NP (DT the) (JJS greatest) (NN extent)))	最大限度 地
(PP (IN of) (NP (JJ peaceful) (NNP coexistence)))	和平共处
Many-to-many Alignment	
(VP (VBN based) (PP (IN on) (NP (JJ actual) (NNS needs))))	从 实际 出发
(PP (IN into) (NP (NP (DT the) (NNS hands)) PP))	掌握 在 PP 手上
(VP (VBP exercise) (NP (JJ strict) (NN self-discipline)))	严 以 律 己
(SBAR (S (NP (DT the) (VBG aging) NN) (VP (aux is) NP)))	NN 老 龄 化 是 NP
(NP NP$_1$ PP (, ,) (VP (VBN centered) (PP (IN around) NP$_2$)))	以 NP$_2$ 为 核心 的 NP$_1$ PP
Allowance of Bad Word Segmentation	
(NP (NP (NNP japan) (POS 's)) (NNP sdf) (NNP navy))	日本海 上 自卫队
(NP (PDT all) (NP (NNS people) (POS 's)) (NNS organizations))	各人 民团 体

Figure 10: Examples of the learned TTS templates

reduced out of vocabulary (OOV) ratio. We examined the OOV ratio of the test set with/without the learned TTS templates, and found the difference was very small. In fact, our method is designed to learn the phrasal TTS templates, and explictly avoids lexical pairs. To further understand the characteristics of the learned TTS templates, we list some representative templates in Figure 4.2 classified in 3 groups. The group *Many-to-one Alignment* and *Many-to-many Alignment* show the TTS templates based on complicated word alignments, which are difficult to compute based on the existing word alignment models. These templates do not have rare English words, whose translation cannot be found outside the big templates. The difficulty lies in the non-literal translation of the source words, which are unlikely to learnt by soly increasing the size of the training corpus. One other interesting observation is that our learning method is tolerant to noisy Chinese word segmentation, as shown in group *Allowance of Bad Word Segmentation*.

6 Conclusion

This paper proposes a Bayesian model for extracting the Tree-to-String templates directly from the data. By limiting the extraction to the big templates from the pre-computed word alignments and applying a set of constraints, we restrict the space of possible TTS templates under consideration, while still allowing new and more accurate templates to emerge from the training data. The empirical results demonstrate the strength of our approach, which outperforms the GIZA++-based systems by a large margin. This encourages a move from word-alignment-based systems to systems based on consistent, end-to-end probabilistic modeling. Because our Bayesian model employs a very simple prior, more sophisticated generative models provide a possible direction for further experimentation.

Acknowledgments This work was supported by NSF grants IIS-0546554 and ITR-0428020.

References

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. Bayesian synchronous grammar induction. In *Neural Information Processing Systems (NIPS)*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-01*, pages 132–139.

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

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society*, 39(1):1–21.

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

John DeNero, Alexandre Bouchard-Cote, and Dan Klein. 2008. Sampling alignment structure under a bayesian translation model. In *EMNLP08*.

Victoria Fossum, Kevin Knight, and Steven Abney. 2008. Using syntax to improveword alignment precision for syntax-based machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, Columbus, Ohio. ACL.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of NAACL-04*, 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 *Proceedings of COLING/ACL-06*, pages 961–968, July.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT/NAACL).*

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, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL-03*, Edmonton, Alberta.

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

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.

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

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*.

K. Rose, E. Gurewitz, and G. C. Fox. 1992. Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory*, 38(4):1249–1257, July.

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

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *COLING-96*, pages 836–841.