Tree-to-String Alignment Template for Statistical Machine Translation

Yang Liu , Qun Liu , and **Shouxun Lin**
Institute of Computing Technology
Chinese Academy of Sciences
No.6 Kexueyuan South Road, Haidian District
P. O. Box 2704, Beijing, 100080, China
{yliu,liuqun,sxlin}@ict.ac.cn

Abstract

We present a novel translation model based on *tree-to-string alignment template* (TAT) which describes the alignment between a source parse tree and a target string. A TAT is capable of generating both terminals and non-terminals and performing reordering at both low and high levels. The model is linguistically syntax-based because TATs are extracted automatically from word-aligned, source side parsed parallel texts. To translate a source sentence, we first employ a parser to produce a source parse tree and then apply TATs to transform the tree into a target string. Our experiments show that the TAT-based model significantly outperforms Pharaoh, a state-of-the-art decoder for phrase-based models.

1 Introduction

Phrase-based translation models (Marcu and Wong, 2002; Koehn et al., 2003; Och and Ney, 2004), which go beyond the original IBM translation models (Brown et al., 1993) [1] by modeling translations of phrases rather than individual words, have been suggested to be the state-of-the-art in statistical machine translation by empirical evaluations.

In phrase-based models, phrases are usually strings of adjacent words instead of syntactic constituents, excelling at capturing local reordering and performing translations that are localized to substrings that are common enough to be observed on training data. However, a key limitation of phrase-based models is that they fail to model reordering at the phrase level robustly. Typically, phrase reordering is modeled in terms of offset positions at the word level (Koehn, 2004; Och and Ney, 2004), making little or no direct use of syntactic information.

Recent research on statistical machine translation has lead to the development of syntax-based models. Wu (1997) proposes Inversion Transduction Grammars, treating translation as a process of parallel parsing of the source and target language via a synchronized grammar. Alshawi et al. (2000) represent each production in parallel dependency tree as a finite transducer. Melamed (2004) formalizes machine translation problem as synchronous parsing based on multitext grammars. Graehl and Knight (2004) describe training and decoding algorithms for both generalized tree-to-tree and tree-to-string transducers. Chiang (2005) presents a hierarchical phrase-based model that uses hierarchical phrase pairs, which are formally productions of a synchronous context-free grammar. Ding and Palmer (2005) propose a syntax-based translation model based on a probabilistic synchronous dependency insert grammar, a version of synchronous grammars defined on dependency trees. All these approaches, though different in formalism, make use of synchronous grammars or tree-based transduction rules to model both source and target languages.

Another class of approaches make use of syntactic information in the target language alone, treating the translation problem as a parsing problem. Yamada and Knight (2001) use a parser in the target language to train probabilities on a set of

[1] The mathematical notation we use in this paper is taken from that paper: a source string $f_1^J = f_1, \ldots, f_j, \ldots, f_J$ is to be translated into a target string $e_1^I = e_1, \ldots, e_i, \ldots, e_I$. Here, I is the length of the target string, and J is the length of the source string.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 609–616,
Sydney, July 2006.

operations that transform a target parse tree into a source string.

Paying more attention to source language analysis, Quirk et al. (2005) employ a source language dependency parser, a target language word segmentation component, and an unsupervised word alignment component to learn treelet translations from parallel corpus.

In this paper, we propose a statistical translation model based on tree-to-string alignment template which describes the alignment between a source parse tree and a target string. A TAT is capable of generating both terminals and non-terminals and performing reordering at both low and high levels. The model is linguistically syntax-based because TATs are extracted automatically from word-aligned, source side parsed parallel texts. To translate a source sentence, we first employ a parser to produce a source parse tree and then apply TATs to transform the tree into a target string.

One advantage of our model is that TATs can be automatically acquired to capture linguistically motivated reordering at both low (word) and high (phrase, clause) levels. In addition, the training of TAT-based model is less computationally expensive than tree-to-tree models. Similarly to (Galley et al., 2004), the tree-to-string alignment templates discussed in this paper are actually transformation rules. The major difference is that we model the syntax of the source language instead of the target side. As a result, the task of our decoder is to find the best target string while Galley's is to seek the most likely target tree.

2 Tree-to-String Alignment Template

A tree-to-string alignment template z is a triple $\langle \tilde{T}, \tilde{S}, \tilde{A} \rangle$, which describes the alignment $\tilde{A}$ between a source parse tree $\tilde{T} = T(F_1^{J'})$ [2] and a target string $\tilde{S} = E_1^{I'}$. A source string $F_1^{J'}$, which is the sequence of leaf nodes of $T(F_1^{J'})$, consists of both terminals (source words) and non-terminals (phrasal categories). A target string $E_1^{I'}$ is also composed of both terminals (target words) and non-terminals (placeholders). An alignment $\tilde{A}$ is defined as a subset of the Cartesian product of source and target symbol positions:

$$\tilde{A} \subseteq \{(j,i) : j = 1, \ldots, J'; i = 1, \ldots, I'\} \quad (1)$$

[2]We use $T(\cdot)$ to denote a parse tree. To reduce notational overhead, we use $T(z)$ to represent the parse tree in z. Similarly, $S(z)$ denotes the string in z.

Figure 1 shows three TATs automatically learned from training data. Note that when demonstrating a TAT graphically, we represent non-terminals in the target strings by blanks.

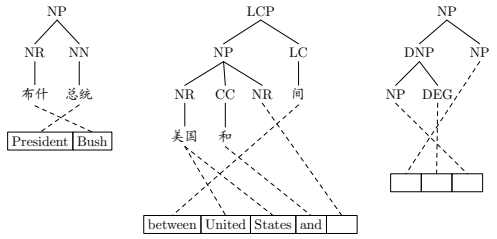

Figure 1: Examples of tree-to-string alignment templates obtained in training

In the following, we formally describe how to introduce tree-to-string alignment templates into probabilistic dependencies to model $Pr(e_1^I|f_1^J)$ [3].

In a first step, we introduce the hidden variable $T(f_1^J)$ that denotes a parse tree of the source sentence f_1^J:

$$Pr(e_1^I|f_1^J) = \sum_{T(f_1^J)} Pr(e_1^I, T(f_1^J)|f_1^J) \quad (2)$$

$$= \sum_{T(f_1^J)} Pr(T(f_1^J)|f_1^J) Pr(e_1^I|T(f_1^J), f_1^J) \quad (3)$$

Next, another hidden variable D is introduced to detach the source parse tree $T(f_1^J)$ into a sequence of K subtrees $\tilde{T}_1^K$ with a preorder transversal. We assume that each subtree $\tilde{T}_k$ produces a target string $\tilde{S}_k$. As a result, the sequence of subtrees $\tilde{T}_1^K$ produces a sequence of target strings $\tilde{S}_1^K$, which can be combined serially to generate the target sentence e_1^I. We assume that $Pr(e_1^I|D, T(f_1^J), f_1^J) \equiv Pr(\tilde{S}_1^K|\tilde{T}_1^K)$ because e_1^I is actually generated by the derivation of $\tilde{S}_1^K$. Note that we omit an explicit dependence on the detachment D to avoid notational overhead.

$$Pr(e_1^I|T(f_1^J), f_1^J) = \sum_D Pr(e_1^I, D|T(f_1^J), f_1^J) \quad (4)$$

$$= \sum_D Pr(D|T(f_1^J), f_1^J) Pr(e_1^I|D, T(f_1^J), f_1^J) \quad (5)$$

$$= \sum_D Pr(D|T(f_1^J), f_1^J) Pr(\tilde{S}_1^K|\tilde{T}_1^K) \quad (6)$$

$$= \sum_D Pr(D|T(f_1^J), f_1^J) \prod_{k=1}^{K} Pr(\tilde{S}_k|\tilde{T}_k) \quad (7)$$

[3]The notational convention will be as follows. We use the symbol $Pr(\cdot)$ to denote general probability distribution with no specific assumptions. In contrast, for model-based probability distributions, we use generic symbol $p(\cdot)$.

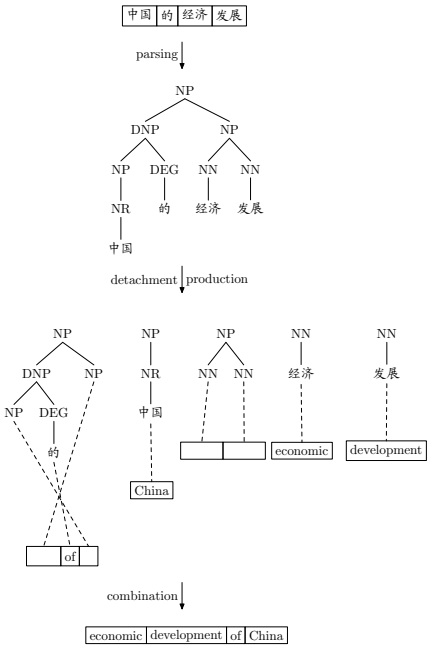

Figure 2: Graphic illustration for translation process

To further decompose $Pr(\tilde{S}|\tilde{T})$, the tree-to-string alignment template, denoted by the variable z, is introduced as a hidden variable.

$$Pr(\tilde{S}|\tilde{T}) = \sum_{z} Pr(\tilde{S}, z|\tilde{T}) \quad (8)$$

$$= \sum_{z} Pr(z|\tilde{T}) Pr(\tilde{S}|z, \tilde{T}) \quad (9)$$

Therefore, the TAT-based translation model can be decomposed into four sub-models:

1. parse model: $Pr(T(f_1^J)|f_1^J)$
2. detachment model: $Pr(D|T(f_1^J), f_1^J)$
3. TAT selection model: $Pr(z|\tilde{T})$
4. TAT application model: $Pr(\tilde{S}|z, \tilde{T})$

Figure 2 shows how TATs work to perform translation. First, the input source sentence is parsed. Next, the parse tree is detached into five subtrees with a preorder transversal. For each subtree, a TAT is selected and applied to produce a string. Finally, these strings are combined serially to generate the translation (we use X to denote the non-terminal):

$X_1 \Rightarrow X_2$ of X_3
$\Rightarrow X_2$ of China
$\Rightarrow X_3\ X_4$ of China
$\Rightarrow$ economic X_4 of China
$\Rightarrow$ economic development of China

Following Och and Ney (2002), we base our model on log-linear framework. Hence, all knowledge sources are described as feature functions that include the given source string f_1^J, the target string e_1^I, and hidden variables. The hidden variable $T(f_1^J)$ is omitted because we usually make use of only single best output of a parser. As we assume that all detachment have the same probability, the hidden variable D is also omitted. As a result, the model we actually adopt for experiments is limited because the parse, detachment, and TAT application sub-models are simplified.

$$Pr(e_1^I, z_1^K|f_1^J) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J, z_1^K)]}{\sum_{e'^I_1, z'^K_1} \exp[\sum_{m=1}^{M} \lambda_m h_m(e'^I_1, f_1^J, z'^K_1)]}$$

$$\hat{e}_1^I = \underset{e_1^I, z_1^K}{\operatorname{argmax}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J, z_1^K) \right\}$$

For our experiments we use the following seven feature functions [4] that are analogous to default feature set of Pharaoh (Koehn, 2004). To simplify the notation, we omit the dependence on the hidden variables of the model.

$$h_1(e_1^I, f_1^J) = \log \prod_{k=1}^{K} \frac{N(z) \cdot \delta(T(z), \tilde{T}_k)}{N(T(z))}$$

$$h_2(e_1^I, f_1^J) = \log \prod_{k=1}^{K} \frac{N(z) \cdot \delta(T(z), \tilde{T}_k)}{N(S(z))}$$

$$h_3(e_1^I, f_1^J) = \log \prod_{k=1}^{K} lex(T(z)|S(z)) \cdot \delta(T(z), \tilde{T}_k)$$

$$h_4(e_1^I, f_1^J) = \log \prod_{k=1}^{K} lex(S(z)|T(z)) \cdot \delta(T(z), \tilde{T}_k)$$

$$h_5(e_1^I, f_1^J) = K$$

$$h_6(e_1^I, f_1^J) = \log \prod_{i=1}^{I} p(e_i|e_{i-2}, e_{i-1})$$

$$h_7(e_1^I, f_1^J) = I$$

[4] When computing lexical weighting features (Koehn et al., 2003), we take only terminals into account. If there are no terminals, we set the feature value to 1. We use $lex(\cdot)$ to denote lexical weighting. We denote the number of TATs used for decoding by K and the length of target string by I.

Tree	String	Alignment
(NR 布什)	Bush	1:1
(NN 总统)	President	1:1
(VV 发表)	made	1:1
(NN 演讲)	speech	1:1
(NP (NR) (NN))	$X_1 \mid X_2$	1:2 2:1
(NP (NR 布什) (NN))	$X \mid$ Bush	1:2 2:1
(NP (NR) (NN 总统))	President $\mid X$	1:2 2:1
(NP (NR 布什) (NN 总统))	President | Bush	1:2 2:1
(VP (VV) (NN))	$X_1 \mid$ a $\mid X_2$	1:1 2:3
(VP (VV 发表) (NN))	made | a $\mid X$	1:1 2:3
(VP (VV) (NN 演讲))	$X \mid$ a | speech	1:1 2:3
(VP (VV 发表) (NN 演讲))	made | a | speech	1:1 2:3
(IP (NP) (VP))	$X_1 \mid X_2$	1:1 2:2

Table 1: Examples of TATs extracted from the TSA in Figure 3 with $h = 2$ and $c = 2$

3 Training

To extract tree-to-string alignment templates from a word-aligned, source side parsed sentence pair $\langle T(f_1^J), e_1^I, A\rangle$, we need first identify *TSA*s (Tree-String-Alignment) using similar criterion as suggested in (Och and Ney, 2004). A TSA is a triple $\langle T(f_{j_1}^{j_2}), e_{i_1}^{i_2}, \bar{A})\rangle$ that is in accordance with the following constraints:

1. $\forall(i,j) \in A : i_1 \le i \le i_2 \leftrightarrow j_1 \le j \le j_2$
2. $T(f_{j_1}^{j_2})$ is a subtree of $T(f_1^J)$

Given a TSA $\langle T(f_{j_1}^{j_2}), e_{i_1}^{i_2}, \bar{A}\rangle$, a triple $\langle T(f_{j_3}^{j_4}), e_{i_3}^{i_4}, \hat{A}\rangle$ is its *sub TSA* if and only if:

1. $T(f_{j_3}^{j_4}), e_{i_3}^{i_4}, \hat{A}\rangle$ is a TSA
2. $T(f_{j_3}^{j_4})$ is rooted at the direct descendant of the root node of $T(f_{j_2}^{j_1})$
3. $i_1 \le i_3 \le i_4 \le i_2$
4. $\forall(i,j) \in \bar{A} : i_3 \le i \le i_4 \leftrightarrow j_3 \le j \le j_4$

Basically, we extract TATs from a TSA $\langle T(f_{j_1}^{j_2}), e_{i_1}^{i_2}, \bar{A}\rangle$ using the following two rules:

1. If $T(f_{j_1}^{j_2})$ contains only one node, then $\langle T(f_{j_1}^{j_2}), e_{i_1}^{i_2}, \bar{A}\rangle$ is a TAT
2. If the height of $T(f_{j_1}^{j_2})$ is greater than one, then build TATs using those extracted from sub TSAs of $\langle T(f_{j_1}^{j_2}), e_{i_1}^{i_2}, \bar{A}\rangle$.

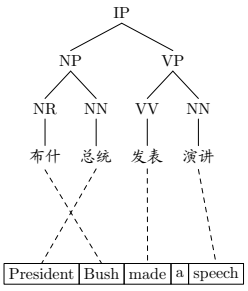

Figure 3: An example of TSA

Usually, we can extract a very large amount of TATs from training data using the above rules, making both training and decoding very slow. Therefore, we impose three restrictions to reduce the magnitude of extracted TATs:

1. A third constraint is added to the definition of TSA:
   $\exists j', j'' : j_1 \le j' \le j_2$ and $j_1 \le j'' \le j_2$ and $(i_1, j') \in \bar{A}$ and $(i_2, j'') \in \bar{A}$
   This constraint requires that both the first and last symbols in the target string must be aligned to some source symbols.
2. The height of $T(z)$ is limited to no greater than h.
3. The number of direct descendants of a node of $T(z)$ is limited to no greater than c.

Table 1 shows the TATs extracted from the TSA in Figure 3 with $h = 2$ and $c = 2$.

As we restrict that $T(f_{j_1}^{j_2})$ must be a subtree of $T(f_1^J)$, TATs may be treated as syntactic hierar-

chical phrase pairs (Chiang, 2005) with tree structure on the source side. At the same time, we face the risk of losing some useful non-syntactic phrase pairs. For example, the phrase pair

布什 总统 发表 ⟷ President Bush made

can never be obtained in form of TAT from the TSA in Figure 3 because there is no subtree for that source string.

4 Decoding

We approach the decoding problem as a bottom-up beam search.

To translate a source sentence, we employ a parser to produce a parse tree. Moving bottom-up through the source parse tree, we compute a list of *candidate translation*s for the input subtree rooted at each node with a postorder transversal. Candidate translations of subtrees are placed in stacks. Figure 4 shows the organization of candidate translation stacks.

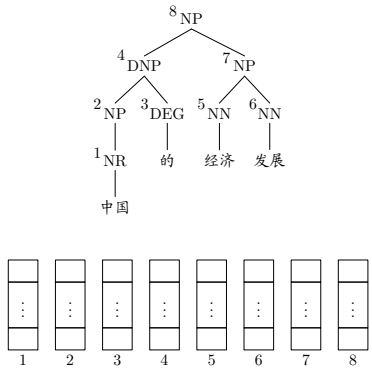

Figure 4: Candidate translations of subtrees are placed in stacks according to the root index set by postorder transversal

A candidate translation contains the following information:

1. the partial translation
2. the accumulated feature values
3. the accumulated probability

A TAT z is $usable$ to a parse tree T if and only if $T(z)$ is rooted at the root of T and *covers* part of nodes of T. Given a parse tree T, we find all usable TATs. Given a usable TAT z, if $T(z)$ is equal to T, then $S(z)$ is a candidate translation of T. If $T(z)$ covers only a portion of T, we have to compute a list of candidate translations for T by replacing the non-terminals of $S(z)$ with candidate translations of the corresponding uncovered subtrees.

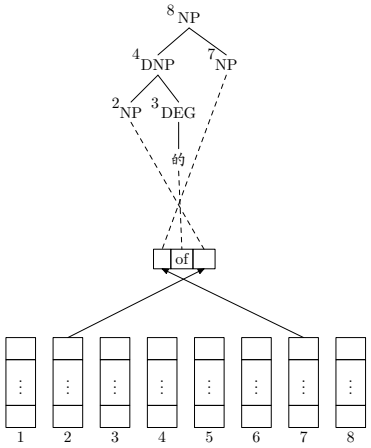

Figure 5: Candidate translation construction

For example, when computing the candidate translations for the tree rooted at node 8, the TAT used in Figure 5 covers only a portion of the parse tree in Figure 4. There are two uncovered subtrees that are rooted at node 2 and node 7 respectively. Hence, we replace the third symbol with the candidate translations in stack 2 and the first symbol with the candidate translations in stack 7. At the same time, the feature values and probabilities are also accumulated for the new candidate translations.

To speed up the decoder, we limit the search space by reducing the number of TATs used for each input node. There are two ways to limit the TAT table size: by a fixed limit (*tatTable-limit*) of how many TATs are retrieved for each input node, and by a probability threshold (*tatTable-threshold*) that specify that the TAT probability has to be above some value. On the other hand, instead of keeping the full list of candidates for a given node, we keep a top-scoring subset of the candidates. This can also be done by a fixed limit (*stack-limit*) or a threshold (*stack-threshold*). To perform recombination, we combine candidate translations that share the same leading and trailing bigrams in each stack.

5 Experiments

Our experiments were on Chinese-to-English translation. The training corpus consists of $31,149$ sentence pairs with $843,256$ Chinese words and

System	Features	BLEU4
Pharaoh	d + $\phi(e\|f)$	0.0573 $\pm$ 0.0033
	d + lm + $\phi(e\|f)$ + wp	0.2019 $\pm$ 0.0083
	d + lm + $\phi(f\|e)$ + lex($f\|e$) + $\phi(e\|f)$ + lex($e\|f$) + pp + wp	0.2089 $\pm$ 0.0089
Lynx	h_1	0.1639 $\pm$ 0.0077
	h_1 + h_6 + h_7	0.2100 $\pm$ 0.0089
	h_1 + h_2 + h_3 + h_4 + h_5 + h_6 + h_7	0.2178 $\pm$ 0.0080

Table 2: Comparison of Pharaoh and Lynx with different feature settings on the test corpus

$949,583$ English words. For the language model, we used SRI Language Modeling Toolkit (Stolcke, 2002) to train a trigram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998) on the $31,149$ English sentences. We selected 571 short sentences from the 2002 NIST MT Evaluation test set as our development corpus, and used the 2005 NIST MT Evaluation test set as our test corpus. We evaluated the translation quality using the BLEU metric (Papineni et al., 2002), as calculated by mteval-v11b.pl with its default setting except that we used case-sensitive matching of n-grams.

5.1 Pharaoh

The baseline system we used for comparison was Pharaoh (Koehn et al., 2003; Koehn, 2004), a freely available decoder for phrase-based translation models:

$$p(e|f) = p_\phi(f|e)^{\lambda_\phi} \times p_{\mathrm{LM}}(e)^{\lambda \mathrm{LM}} \times p_{\mathrm{D}}(e,f)^{\lambda \mathrm{D}} \times \omega^{\mathrm{length}(e)\lambda \mathrm{W}(e)} \quad (10)$$

We ran GIZA++ (Och and Ney, 2000) on the training corpus in both directions using its default setting, and then applied the refinement rule "diag-and" described in (Koehn et al., 2003) to obtain a single many-to-many word alignment for each sentence pair. After that, we used some heuristics, which including rule-based translation of numbers, dates, and person names, to further improve the alignment accuracy.

Given the word-aligned bilingual corpus, we obtained $1,231,959$ bilingual phrases ($221,453$ used on test corpus) using the training toolkits publicly released by Philipp Koehn with its default setting.

To perform minimum error rate training (Och, 2003) to tune the feature weights to maximize the system's BLEU score on development set, we used optimizeV5IBMBLEU.m (Venugopal and Vogel, 2005). We used default pruning settings for Pharaoh except that we set the distortion limit to 4.

5.2 Lynx

On the same word-aligned training data, it took us about one month to parse all the $31,149$ Chinese sentences using a Chinese parser written by Deyi Xiong (Xiong et al., 2005). The parser was trained on articles $1-270$ of Penn Chinese Treebank version 1.0 and achieved 79.4% (F1 measure) as well as a 4.4% relative decrease in error rate. Then, we performed TAT extraction described in section 3 with $h = 3$ and $c = 5$ and obtained $350,575$ TATs ($88,066$ used on test corpus). To run our decoder Lynx on development and test corpus, we set tatTable-limit $= 20$, tatTable-threshold $= 0$, stack-limit $= 100$, and stack-threshold $= 0.00001$.

5.3 Results

Table 2 shows the results on test set using Pharaoh and Lynx with different feature settings. The 95% confidence intervals were computed using Zhang's significance tester (Zhang et al., 2004). We modified it to conform to NIST's current definition of the BLEU brevity penalty. For Pharaoh, eight features were used: distortion model *d*, a trigram language model *lm*, phrase translation probabilities $\phi(f|e)$ and $\phi(e|f)$, lexical weightings *lex*$(f|e)$ and *lex*$(e|f)$, phrase penalty *pp*, and word penalty *wp*. For Lynx, seven features described in section 2 were used. We find that Lynx outperforms Pharaoh with all feature settings. With full features, Lynx achieves an absolute improvement of 0.006 over Pharaoh (3.1% relative). This difference is statistically significant ($p < 0.01$). Note that Lynx made use of only $88,066$ TATs on test corpus while $221,453$ bilingual phrases were used for Pharaoh.

The feature weights obtained by minimum er-

System	Features							
	d	lm	$\phi(f\|e)$	lex$(f\|e)$	$\phi(e\|f)$	lex$(e\|f)$	pp	wp
Pharaoh	0.0476	0.1386	0.0611	0.0459	0.1723	0.0223	0.3122	-0.2000
Lynx	-	0.3735	0.0061	0.1081	0.1656	0.0022	0.0824	0.2620

Table 3: Feature weights obtained by minimum error rate training on the development corpus

	BLEU4
tat	0.2178 ± 0.0080
tat + bp	0.2240 ± 0.0083

Table 4: Effect of using bilingual phrases for Lynx

ror rate training for both Pharaoh and Lynx are shown in Table 3. We find that $\phi(f|e)$ (i.e. h_2) is not a helpful feature for Lynx. The reason is that we use only a single non-terminal symbol instead of assigning phrasal categories to the target string. In addition, we allow the target string consists of only non-terminals, making translation decisions not always based on lexical evidence.

5.4 Using bilingual phrases

It is interesting to use bilingual phrases to strengthen the TAT-based model. As we mentioned before, some useful non-syntactic phrase pairs can never be obtained in form of TAT because we restrict that there must be a corresponding parse tree for the source phrase. Moreover, it takes more time to obtain TATs than bilingual phrases on the same training data because parsing is usually very time-consuming.

Given an input subtree $T(F_{j_1}^{j_2})$, if $F_{j_1}^{j_2}$ is a string of terminals, we find all bilingual phrases that the source phrase is equal to $F_{j_1}^{j_2}$. Then we build a TAT for each bilingual phrase $\langle f_1^{J'}, e_1^{I'}, \hat{A}\rangle$: the tree of the TAT is $T(F_{j_1}^{j_2})$, the string is $e_1^{I'}$, and the alignment is $\hat{A}$. If a TAT built from a bilingual phrase is the same with a TAT in the TAT table, we prefer to the greater translation probabilities.

Table 4 shows the effect of using bilingual phrases for Lynx. Note that these bilingual phrases are the same with those used for Pharaoh.

5.5 Results on large data

We also conducted an experiment on large data to further examine our design philosophy. The training corpus contains 2.6 million sentence pairs. We used all the data to extract bilingual phrases and a portion of 800K pairs to obtain TATs. Two trigram language models were used for Lynx. One was trained on the 2.6 million English sentences and another was trained on the first 1/3 of the Xinhua portion of Gigaword corpus. We also included rule-based translations of named entities, dates, and numbers. By making use of these data, Lynx achieves a BLEU score of 0.2830 on the 2005 NIST Chinese-to-English MT evaluation test set, which is a very promising result for linguistically syntax-based models.

6 Conclusion

In this paper, we introduce tree-to-string alignment templates, which can be automatically learned from syntactically-annotated training data. The TAT-based translation model improves translation quality significantly compared with a state-of-the-art phrase-based decoder. Treated as special TATs without tree on the source side, bilingual phrases can be utilized for the TAT-based model to get further improvement.

It should be emphasized that the restrictions we impose on TAT extraction limit the expressive power of TAT. Preliminary experiments reveal that removing these restrictions does improve translation quality, but leads to large memory requirements. We feel that both parsing and word alignment qualities have important effects on the TAT-based model. We will retrain the Chinese parser on Penn Chinese Treebank version 5.0 and try to improve word alignment quality using log-linear models as suggested in (Liu et al., 2005).

Acknowledgement

This work is supported by National High Technology Research and Development Program contract "Generally Technical Research and Basic Database Establishment of Chinese Platform"(Subject No. 2004AA114010). We are grateful to Deyi Xiong for providing the parser and Haitao Mi for making the parser more efficient and robust. Thanks to Dr. Yajuan Lv for many helpful comments on an earlier draft of this paper.

References

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26(1):45-60.

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.

Stanley F. Chen and Joshua Goodman. 1998. Am empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of 43rd Annual Meeting of the ACL*, pages 263-270.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insert grammars. In *Proceedings of 43rd Annual Meeting of the ACL*, pages 541-548.

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

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proceedings of NAACL-HLT 2004*, pages 105-112.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127-133.

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

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of 43rd Annual Meeting of the ACL*, pages 459-466.

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

Dan Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of 42nd Annual Meeting of the ACL*, pages 653-660.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of 38th Annual Meeting of the ACL*, pages 440-447.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of 40th Annual Meeting of the ACL*, pages 295-302.

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417-449.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of 41st Annual Meeting of the ACL*, pages 160-167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the ACL*, pages 311-318.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of 43rd Annual Meeting of the ACL*, pages 271-279.

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

Ashish Venugopal and Stephan Vogel. 2005. Considerations in maximum mutual information and minimum classification error training for statistical machine translation. In *Proceedings of the Tenth Conference of the European Association for Machine Translation (EAMT-05)*.

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

Deyi Xiong, Shuanglong Li, Qun Liu, Shouxun Lin, and Yueliang Qian. 2005. Parsing the Penn Chinese treebank with semantic knowledge. In *Proceedings of IJCNLP 2005*, pages 70-81.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of 39th Annual Meeting of the ACL*, pages 523-530.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC)*, pages 2051-2054.