Improving Neural Machine Translation with Soft Template Prediction

Jian Yang[1] [*], **Shuming Ma**[2], **Dongdong Zhang**[2], **Zhoujun Li**[1] [†], **Ming Zhou**[2]
[1]State Key Lab of Software Development Environment, Beihang University
[2]Microsoft Research Asia
{jiaya, lizj}@buaa.edu.cn; {shumma, dozhang, mingzhou}@microsoft.com;

Abstract

Although neural machine translation (NMT) has achieved significant progress in recent years, most previous NMT models only depend on the source text to generate translation. Inspired by the success of template-based and syntax-based approaches in other fields, we propose to use extracted templates from tree structures as soft target templates to guide the translation procedure. In order to learn the syntactic structure of the target sentences, we adopt the constituency-based parse tree to generate candidate templates. We incorporate the template information into the encoder-decoder framework to jointly utilize the templates and source text. Experiments show that our model significantly outperforms the baseline models on four benchmarks and demonstrate the effectiveness of soft target templates.

1 Introduction

Recently, neural machine translation (NMT) (Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017; Chen et al., 2018) has achieved significant progress. Some advanced models (Chatterjee et al., 2016; Niehues et al., 2016; Junczys-Dowmunt and Grundkiewicz, 2017; Geng et al., 2018; Zhou et al., 2019a) predict the ultimate translation by multi-pass generation conditioned on the previous text such as CMLMs (Ghazvininejad et al., 2019), ABD-NMT (Zhang et al., 2018), SynST (Akoury et al., 2019), and Deliberation Network (Xia et al., 2017).

Inspired by these works and the successful application of templates for other intriguing tasks, including semantic parsing (Dong and Lapata, 2018), summarization (Cao et al., 2018; Wang et al., 2019a), question answering (Duan et al., 2017; Pandey et al., 2018), and other text generation tasks (Wiseman et al., 2018; Guu et al., 2018), we assume the candidate templates of the target sentences can guide the sentence translation process. We denote these templates extracted from the constituency-based parse tree as soft templates, which consist of tags and target words. The templates are soft because no explicit paradigms are inaugurated to build new translation from them, and the target tokens could be modified.

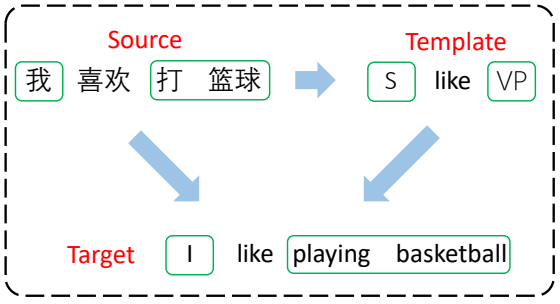

Figure 1: An example of template guided translation results. S denotes subject and VP denotes verb phrase.

In order to effectively use the templates, we introduce soft template-based neural machine translation (ST-NMT), which can use source text and soft templates to predict the final translation. Our approach can be split into two phases. In the first phase, a standard Transformer model is trained to predict soft target templates by using source text and templates extracted from the constituency-based parse tree. Secondly, we use two encoders, including a soft target template encoder and a source language encoder to encode source text and templates and generate the final translation. As shown in Figure 1, given the source text "我喜欢打篮球" and the target template "S like to VP", the final translation "I like to play basketball" is generated by two encoders. In this work, the templates play a part in guiding, and some target tokens in

[*]Contribution during internship at Microsoft Research Asia.
[†]Corresponding author.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5979–5989

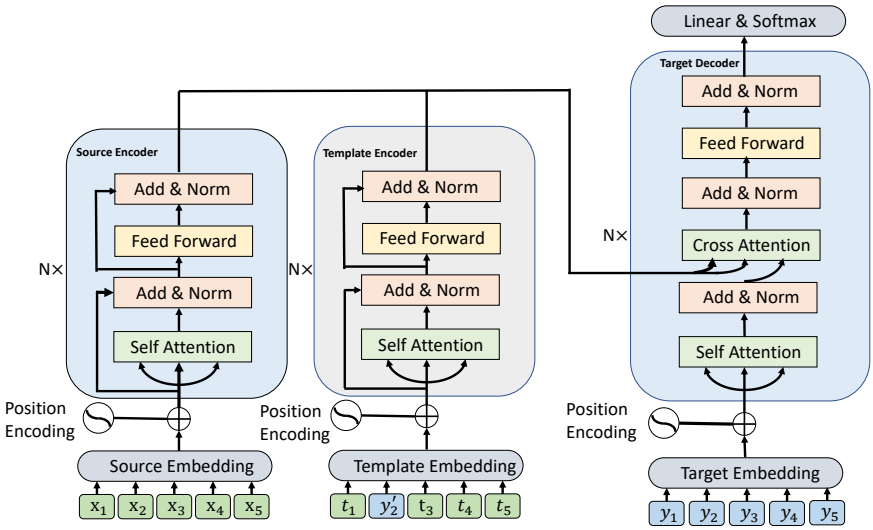

Figure 2: Overview of our ST-NMT. Given the source text and the soft target template predicted by the $P_{\theta_{X\to Y}}$, the source language Transformer encoder and target template Transformer encoder maps two sequences $X = (x_1, x_2, x_3, x_4, x_5)$ and $T = (t_1, y'_2, t_3, t_4, t_5)$ into hidden states Z^X and Z^T. x_i denotes the source word, t_i denotes the template tag and y_i denotes the target word. y'_i also denotes the target word but it can be modified to the other target words. The ultimate translation Y is generated by a Transformer decoder which incorporates the context Z^X and Z^Y in the second phase.

the template could also be modified.

In order to prove the effectiveness of our approach, we conduct main experiments on the popular benchmarks, including IWSLT14 German-English translation task, WMT14 English-German translation task, LDC Chinese-English translation task, and ASPEC Japanese-Chinese translation task. Experiments show that our approach achieves significant improvement compared to the baselines, which demonstrates the soft target templates can provide a positive influence for guiding translation procedure effectively. Our approach can be used for diverse scale data sets, different styles, and multiple language pairs.

2 Our Approach

Our model first reads the source language sequence $X = (x_1, x_2, x_3, \ldots, x_n)$ in the conventional way by a source language Transformer encoder and generates the template sequence $T = (t_1, t_2, t_3, \ldots, t_m)$ by a template Transformer decoder. As shown in Figure 2, our model uses a source language Transformer encoder and a template Transformer encoder, which encodes the source language sequence X and the template sequence T separately. We deploy the target language decoder to generate the final translation. In this section, we present the details of the proposed template-based approach. Our method mainly includes two phases: (1) The training data is constructed by the constituency-based parse tree. Then, we adopt a standard Transformer to convert the source text to the soft target template for the next generation. (2) Based on the source text and the predicted soft target template, we utilize two encoders to encode two sequences into hidden states separately and a target language decoder to generate the ultimate translation.

2.1 Soft Template Prediction

In this procedure, we model the $P_{\theta_{X\to T}}(T|X)$ to predict soft target templates on top of the constructed training data $D_{X,T}$. To construct $D_{X,T}$, we use a constituency-based parser to parse the target sequence and get a tree structure. Then, we prune nodes which are deeper than the specific depth and recover the left leaf nodes to the ordered template sequence. Through these operations, we gain the parallel training data $D_{X,T}$ and train a standard Transformer model $P_{\theta_{X\to T}}(T|X)$ to predict the soft target template.

The constituency-based parse tree could reveal the structure information of the whole sentence which utilizes the constituency grammar to distinguish terminal and non-terminal nodes. More specifically, the interior nodes are labeled by non-terminal categories which belong to the set of non-terminal tokens S, while the leaf nodes are labeled

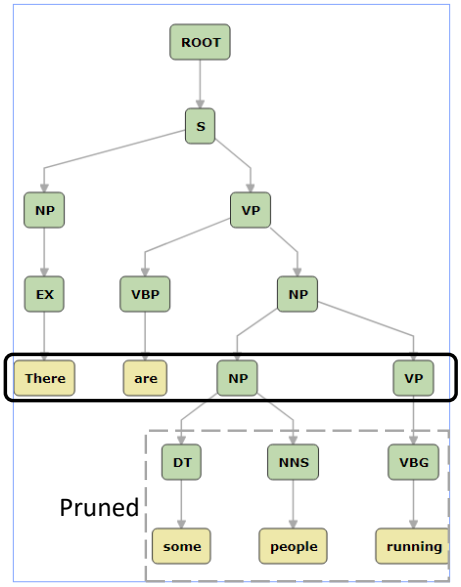

Figure 3: The constituency-based parse tree of the example sentence. Given the target sentence and definite depth of the tree, we gain the sub-tree by pruning the nodes deeper than 4 in this case. Then, the sub-tree can be converted to the soft target template "There are NP VP" from left to right.

by terminal categories V. S = {S, VP, NP, ..., ASBR} and V is the vocabulary set of the target language. For example, the sentence "There are some people running" could be expressed as Figure 3. In this case, the non-terminal tokens consist of S_0 = {S, NP, VP, EX, VBP, NP, DT, NNS, VBG} and the terminal tokens are composed of V_0 = {There, are some, people, running}. Our template $T = \{t_1, t_2, t_3, t_4\}$ is the ordered sequence which is composed of non-terminal tokens and terminal tokens. In this case, t_1=There, t_2=are, t_3=VP and t_4=NP. Our template extraction aims to extract the sub-tree of the specific depth and use these non-terminal and terminal tokens locating at the leaf node of sub-tree.

In order to predict the soft target templates, we train a standard Transformer model given the training data of the source text and extracted templates. The Transformer model reads the source text and predicts the soft target templates using beam search. Then, we select the top-K results of the beam search as templates.

The depth of the tree is a trade-off. In Figure 3, One special case is that when the depth equals 1, the template only has one symbol "S". The template "S" cannot provide any useful information. Another special case is that when depth is greater than 6, the template "There are some people running" only has terminal tokens. The template only contains target words, which cannot provide any additional information. When the depth equals 4, the template is "There are NP VP". The template contains sentence syntactic and structural information, which is suitable for our method.

With the Transformer model $P_{\theta_{X\to T}}(T|X)$, we need to construct the pseudo training data $D_{X,T,Y}$ instead of directly using extracted templates by soft template prediction. Given the source text X, we use $P_{\theta_{X\to T}}(T|X)$ to predict the top-1 soft target template T with beam search. Therefore, we get the triple training data $D_{X,T,Y} = \{X^{(i)}, T^{(i)}, Y^{(i)}\}_{i=1}^{N}$ which is prepared for the next phase.

2.2 Machine Translation via Soft Templates

The triple training data $D_{X,T,Y}$ is used to model the probability $P_{(X,T)\to Y}$ from the two sequences to the ultimate translation. Our approach could generate the target sentence Y, given the source sequence X and template T.

Formulation In formula, we could model the whole procedure on top of the $P_{\theta_{X\to T}}(T|X)$ and $P_{\theta_{(X,T)\to Y}}(Y|X,T)$.

$$P(Y|X) = P_{\theta_{X\to T}}(T|X)P_{\theta_{(X,T)\to Y}}(Y|X,T) \quad (1)$$

where $\theta_{X\to T}$ and $\theta_{(X,T)\to Y}$ are the parameters for the first and the second phase.

The source language Transformer encoder and the soft template Transformer encoder maps the input sequence X and the template T composed of target language words and tags to the hidden states. Then, a Transformer decoder interacting with two encoders generates the final translation Y, described by the Equation 1.

Encoder In the second phase, our template Transformer encoder and the source language Transformer encoder are stacked by blocks which contain self-attention layers with residual connections, layer normalization and fully connected feed-forward network (FFN). Therefore, the hidden states of the source language Transformer encoder and the template Transformer encoder are calculated by:

$$h_l = \text{TransformerBlock}(h_{l-1}) \quad (2)$$

where $h_l = h_l^X$ for the source language Transformer encoder and $h_l = h_l^T$ for the template Transformer encoder. N is the number of layers and $l \in [1, N]$.

Decoder Based on the hidden states h_l^X and h_l^T, the target language Transformer decoder use the encoder-decoder multi-head attention to jointly use the source language and template information to generate the ultimate translation Y. Besides, the target sequence decoder uses multi-head attention to obtain the representations of target language decoder with the parameters (W_X^Q, W_X^K, W_X^V) and (W_T^Q, W_T^K, W_T^V) for different encoders.

In each attention head, the input sequence $X = (x_1, \ldots, x_m)$ and the template $T = (t_1, \ldots, t_n)$ can be mapped into $Z^X = (z_1^X, z_2^X, \ldots, z_m^X)$ and $Z^T = (z_1^T, z_2^T, \ldots, z_n^T)$ using the source language Transformer encoder and the template Transformer encoder.

On top of the Z_X and Z_T, the decoder separately calculate the multi-head attention with source sentence context $X = (x_1, \ldots, x_m)$ and target template sentence $T = (t_1, \ldots, t_n)$, then our model obtain two hidden states $Z^{X,Y}$ and $Z^{T,Y}$ by attention with source context and template context. Here, We incorporate the $Z^{X,Y}$ containing source language information and $Z^{X,Y}$ including template information in a reasonable way:

$$Z = \beta Z^{X,Y} + (1 - \beta) Z^{T,Y} \tag{3}$$

where β is the parameter to control the degree of incorporation between source text and template.

In order to effectively incorporate source and template information, we calculate the parameter β as below:

$$\beta = \sigma(W_Y Z^{X,Y} + U_T Z^{X,T}) \tag{4}$$

where Z_Y is the decoder hidden state and W_Y and U_T are parameter matrices. σ is the sigmoid activation function.

2.3 Training Strategy

Similar to the conventional NMT, in order to make the model predict the target sequence, we use maximum likelihood estimation (MLE) loss function to update the model parameter by maximizing the log likelihood of translation over training set D. When we train the $P_{\theta_{X\to Y}}$ without the template Transformer encoder, we only need to optimize the following loss function:

$$L_{\theta_{X\to Y}}(D) = \sum_{X,Y \in D} \log P_{\theta_{X\to Y}}(Y|X) \tag{5}$$

where $\theta_{X\to Y}$ are the parameters of the source language Transformer encoder and the target language Transformer decoder.

When we train the $P_{\theta_{(X,T)\to Y}}$ with the template Transformer encoder, the loss function could be calculated by:

$$L_{\theta_{(X,T)\to Y}}(D) = \sum_{X,Y \in D} \log P_{\theta_{(X,T)\to Y}}(Y|X,T) \tag{6}$$

where $\theta_{(X,T)\to Y}$ are the parameters of the source language Transformer encoder, template language Transformer encoder and target language Transformer decoder.

To balance the two objectives, our model is trained on $L_{\theta_{X\to Y}}(D)$ objective for the $\alpha\%$ iterations, and trained on $L_{\theta_{(X,T)\to Y}}(D)$ objective for the $(1-\alpha)\%$ interations. Therefore, this procedure is equivalent to the following formula:

$$L_\theta(D) = \alpha L_{\theta_{X\to Y}}(D) + (1 - \alpha) L_{\theta_{(X,T)\to Y}}(D) \tag{7}$$

where α is a scaling factor accounting for the difference in magnitude between $L_{\theta_{X\to Y}}(D)$ and $L_{\theta_{(X,T)\to Y}}(D)$.

In practice, we find optimizing these two objectives can make training procedure easier and get a higher BLEU score since there exist a few low-quality templates to influence the translation quality. Through optimizing two objectives simultaneously, we can reduce the effect of some low-quality templates and improve the stability of our model.

3 Experiments

We conducted experiments on four benchmarks, including LDC Chinese-English, WMT14 English-German, IWSLT14 German-English, and ASPEC Japanese-Chinese translation tasks. By conducting experiments on these four benchmarks, these settings prove that our approach is suitable for diverse situations: (1) These four benchmarks provide a wide coverage of both scale and genres. They vary from small scale to large scale (2) We use the different domains, which include news, science, and talk domain. (3) We also conduct the experiments

on different language pairs, including the German-English translation task, the English-German translation task, the Chinese-English translation task, and the Japanese-Chinese translation task.

3.1 Datasets

In order to verify the effectiveness of our method, we conduct experiments on four benchmarks. WMT14 and LDC datasets are from the news domain. IWSLT14 dataset is from TED talk. ASPEC dataset is from a scientific paper excerpt corpus.

LDC Chinese-English We use a subset from LDC corpus[1] which has nearly 1.4M sentences originally. The training set is selected from the LDC corpus that consists of 1.2M sentence pairs after dropping the low-quality sentence pairs of which the length is more than 2. We used the NIST 2006 dataset as the validation set for evaluating performance in the training procedure, and NIST 2003, 2005, 2008 and 2012 as test sets, which all have 4 English references for each Chinese sentence.

IWSLT14 German-English This dataset contains 16K training sequence pairs. We randomly sample 5% of the training data as valid test. Besides, we merge the multiple testsets dev2010, dev2012, tst2010, tst2011, tst2012 for testing.

WMT14 English-German The training data consists of 4.5M sentence pairs. The validation set is devtest2014, and the test set is newstest2014.

ASPEC Japanese-Chinese We use 0.67M sentence pairs from ASPEC Japanese-Chinese corpus (Nakazawa et al., 2016) [2]. We use the devtest as the development data, which contains 2090 sentences, and the test data contains 2107 sentences with a single reference per source sentence.

3.2 Preprocessing and Training Details

LDC Chinese-English The base Transformer model is used for this task, which includes 6 layers, each layer of which has the hidden dimensions of 512, feedforward dimensions of 2048 , and 8 attention heads. We use Moses (Koehn et al., 2007) to tokenize English sentences and our in-house tool to tokenize Chinese sentences. We use Byte Pair Encoding (BPE) (Sennrich et al., 2016) to encode sentences using a shared vocabulary of 40K symbols.

IWSLT14 German-English We adopt the small setup of the Transformer model. The model has 6 layers with the embedding size of 512, a feedforward size of 1024, and 4 attention heads. In order to prevent overfitting, we use a dropout of 0.3, a l_2 weight decay of 10^{-4}, and a label smoothing of 0.1. We use BPE to encode sentences with a shared vocabulary of 10K symbols.

WMT14 English-German We use the big setting of Transformer (Vaswani et al., 2017), in which both the encoder and the decoder have 6 layers, with the embedding size of 1024, feedforward size of 4096, and 16 attention heads. The dropout rate is fixed as 0.3. We adopt Adam (Kingma and Ba, 2015) optimizer with a learning rate 0.1 of the similar learning rate schedule as Transformer (Vaswani et al., 2017). We set the batch size as 6000 and the update frequency as 16 on 8 GPUs for updating parameters (Ott et al., 2018) to imitate 128 GPUs. The datasets are encoded by BPE with a shared vocabulary (Sennrich et al., 2016) of 40K symbols.

ASPEC Japanese-Chinese We use the base setting of Transformer the same to the Chinese-English translation task. Following the similar learning rate schedule (Vaswani et al., 2017), we set the learning rate as 0.1. Chinese and Japanese sentences are tokenized with our in-house tools and encoded by BPE with a shared vocabulary of 10K symbols.

3.3 Evaluation

We evaluate the performance of the translation results. The evaluation metric is BLEU (Papineni et al., 2002). For the Chinese-English and German-English translation tasks, we use case-insensitive tokenized BLEU scores. For the English-German translation task, we use case-sensitive tokenized BLEU scores for evaluation. All the experiments last for 150 epochs and use Stanford parser to generate templates (Manning et al., 2014).

For all translation tasks, we use the checkpoint, which has the best valid performance on the valid set. For different test sets, we adapt the beam size and the length penalty to get better performance. In order to avoid the difference of the tokenizer for Chinese translation result evaluation, we adopt the character-level BLEU for testing. Checkpoint averaging is not used, except notification.

[1] LDC2002E17, LDC2002E18, LDC2003E07, LDC2003E14, LDC2005E83, LDC2005T06, LDC2005T10, LDC2006E17, LDC2006E26, LDC2006E34, LDC2006E85, LDC2006E92, LDC2006T06, LDC2004T08, LDC2005T10

[2] http://orchid.kuee.kyoto-u.ac.jp/ASPEC/

Zh → En	MT06	MT03	MT05	MT08	MT12	Avg.
ConvS2S (Gehring et al., 2017)	39.98	42.25	41.22	33.43	32.21	37.28
GNMT (Wu et al., 2016)	40.53	42.88	42.73	33.97	32.55	38.03
Transformer (our implementation)	43.60	45.80	44.52	36.62	34.60	40.39
ST-NMT (our proposed)	**44.69**	**46.56**	**46.04**	**37.53**	**35.99**	**41.53**

Table 1: Evaluation results on Zh → En translation task with BLEU% metric. The "Avg." column means the averaged result of all NIST test sets except NIST2006. The result of our model is statistically significant compared to the other baselines ($p < 0.01$).

3.4 Baselines

We compare our approach with two types of baselines including one-pass baselines and multi-pass baselines.

One-pass Baselines: **ConvS2S** (Gehring et al., 2017) is a strong CNN-based baseline. We report the results referring to the paper of convolutional sequence to sequence model (ConvS2S). **RNMT+** (Chen et al., 2018) is a state-of-the-art RNN-based NMT model. **GNMT** (Wu et al., 2016) is the typical encoder-decoder framework. We use the similar setting[3] for all experiments. **Transformer** (Vaswani et al., 2017) is a strong baseline which has the state-of-the-art performance. We reimplement this baseline[4]. **LightConv** and **DynamicConv** (Wu et al., 2019) are simpler but effective baselines. We directly report the results in the paper.

Multi-pass Baselines: **Deliberation network** (Xia et al., 2017) and **SoftPrototype** (Wang et al., 2019b) generates and polishes the raw text by a two-pass manner. **SB-NMT** (Zhou et al., 2019a) is a synchronous bidirectional neural machine translation which predicts its outputs using two direction simultaneously. **ABD-NMT** (Zhang et al., 2018) is an encoder-decoder NMT framework with the forward and backward decoder. By considering the agreement of both directions left-to-right (L2R) and right-to-left (R2L), **Rerank-NMT** (Liu et al., 2016) rescores all candidates. **SBSG** (Zhou et al., 2019b) is a synchronous bidirectional sequence generation model which predicts its translation from both sides to the middle simultaneously. **Insertion Transformer** (Stern et al., 2019) is a non-monotonic method which predicts the translation by inserting method.

[3]https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Translation/GNMT

[4]https://github.com/pytorch/fairseq

De → En	BLEU
GNMT (Wu et al., 2016)	31.44
RNMT+ (Chen et al., 2018)	34.51
ConvS2S (Gehring et al., 2017)	30.41
LightConv (Wu et al., 2019)	34.80
DynamicConv (Wu et al., 2019)	35.20
Rerank-NMT (Liu et al., 2016)	34.82
Transformer (our implementation)	34.43
ST-NMT (our proposed)	**35.24**

Table 2: BLEU-4 scores (%) on IWSLT14 De→En task. The result of our model is statistically significant compared to the other baselines ($p < 0.05$).

3.5 Results

For the IWSLT14 German-English machine translation task, we present the results of the ST-NMT and other strong baselines in Table 2. We compare our method with other various methods, including GNMT, RNMT+, convS2S, LightConv, DynamicConv, and the Transformer model with the small setting. The Rerank-NMT model gets 34.82 BLEU by using the two-pass results, including left-to-right (L2R) and right-to-left (R2L), and selects the best candidates. As shown in Table 2, our model also significantly outperforms others and gains an improvement of 0.81 BLEU points than a strong Transformer baseline model. Moreover, our method outperforms the GNMT by 3.80 BLEU points, ConvS2S by 4.83 BLEU, LightConv by 0.44 BLEU, Dynamic by 0.04 BLEU and Rerank-NMT by 0.42 BLEU.

We secondly evaluate our method on the LDC Chinese-English translation task. The evaluation results on all NIST test sets against baselines are listed in Table 1. Our ST-NMT beats the other

En → De	BLEU
GNMT (Wu et al., 2016)	24.61
ConvS2S (Gehring et al., 2017)	25.16
Transformer (Vaswani et al., 2017)	28.40
RNMT+ (Chen et al., 2018)	28.49
Rerank-NMT (Liu et al., 2016)	27.81
ABD-NMT (Liu et al., 2016)	28.22
Deliberation Network (Xia et al., 2017)	29.11
SoftPrototype (Wang et al., 2019b)	29.46
SB-NMT (Zhou et al., 2019a)	29.21
SBSG (Zhou et al., 2019b)	27.45
Insertion Transformer (Stern et al., 2019)	27.41
Transformer (our implementation)	29.25
ST-NMT (our proposed)	**29.68**

Table 3: BLEU-4 scores (%) on WMT14 En→De task. The result of our model is statistically significant compared to the other baselines ($p < 0.05$).

Ja → Zh	BLEU
GNMT (Wu et al., 2016)	49.12
ConvS2S (Gehring et al., 2017)	50.32
Transformer (our implementation)	52.02
ST-NMT (our proposed)	**52.84**

Table 4: Character-level BLEU-4 scores (%) on ASPEC Ja→Zh task. The result of our model is statistically significant compared to the other baselines ($p < 0.01$).

baselines and outperforms the Transformer baseline by 1.14 BLEU point on average, which shows that the template could effectively improve the performance. More specifically, our model outperforms the Transformer model by 0.76 BLEU on NIST2003, 1.52 BLEU on NIST 2005, 0.91 BLEU on NIST 2008, and 1.39 BLEU on NIST 2012.

We further demonstrate the effectiveness of our model on WMT14 English-German translation tasks, and we also compare our model with other competitive models, including ABD-NMT (Zhang et al., 2018), Deliberation Network (Xia et al., 2017), SoftPrototype (Wang et al., 2019b), SB-NMT (Zhou et al., 2019a) and SBSG (Zhou et al., 2019b). As shown in Table 3, our model also significantly outperforms others and gets an improvement of 0.43 BLEU points than a strong Transformer model.

To investigate the effect of our approach on the different language pairs, we also evaluate

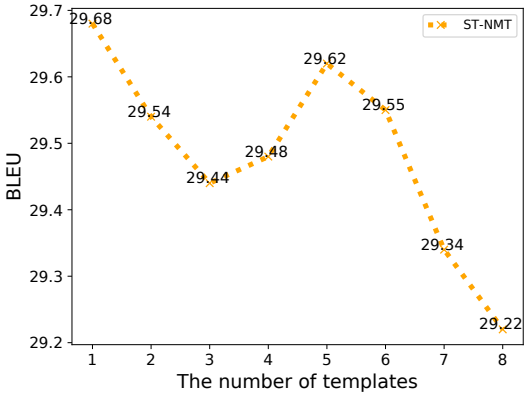

Figure 4: The effect of the multiple templates. We feed the the top-K results of the beam search as multiple templates and source sentence to generate the target translation.

our model on the Japanese-Chinese translation task. According to Table 4, ST-NMT outperforms GNMT by 3.72 BLEU points, ConvS2S by 2.52 BLEU points, and the Transformer model by 0.82 BLEU points, which demonstrates that the soft template extracted by constituency-based parse tree can also bring strong positive effects.

3.6 Multiple Templates

Because of the diversity of the templates, we investigate the performance with the different numbers of the templates. On top of the original parallel training data $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, we construct the training data from the source text to the soft target template $D_{X \to T} = \{(x^{(i)}, t^{(i)})\}_{i=1}^{N}$, by the model $P_{\theta_{X \to T}}$. Through this construction procedure, we could use the top-K results of the beam search as multiple templates by model $P_{\theta_{X \to T}}$. We could expand the training data of the source text to the target template as $D_{X \to T} = \{(x^{(1)}, t^{(1)}_{top_1}), \ldots, (x^{(1)}, t^{(1)}_{top_K}), \ldots, (x^{(N)}, t^{(N)}_{top_1}), \ldots, (x^{(N)}, t^{(N)}_{top_K})\}$. As shown in Figure 4, our model gains the best performance only using the single template. When the number of templates is 8, our model gains the worst BLEU score of 29.22. We can summarize that our model can be more robust but maybe get worse performance with the number of templates rising. Besides, in order to further improve the stability of our model, we expand the dataset by selecting random templates for the source sentence. The different templates confuse our model, although it can make our model more robust.

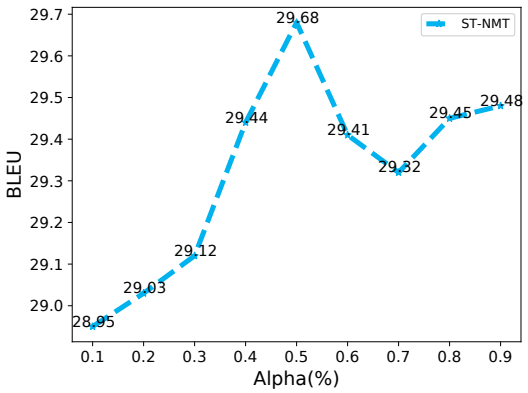

Figure 5: We use two objective to update our model parameters. One objective only use source sentence and another utilizes source and soft template sentences together to generate the final translation. The hyper-parameter α is used to balance the two objectives,

3.7 Balance of Two Objectives

To further control how much our model leverages templates for translation, we tune the hyper-parameter α. With the value rising, the contribution of template information gradually decreases. We study the influence of the ratio α. To investigate the effect of this hyper-parameter, we set the discrete value $\alpha = \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%, 100\%\}$. According to Figure 5, when the α switches from 0.4 to 0.9, our model can get the better performance which is greater than or equal to 29.3 BLEU. The results show that we can set the hyper-parameter α in a reasonable interval ($0.4 \leq \alpha \leq 0.9$) to keep the balance between source text and template.

3.8 Depth of Parsing Tree

Considering that the template derived from the specific depth can lead to the divergent performance, our model is examined with the different depth. The effect of the template extraction which is described as Section 3 is decided by the sub-tree which is controlled by the depth of sub-tree. For the same constituency-based parse tree, the different sub-tree can be obtained based on the different chosen depth d. When we get the sub-tree, the template could be derived from it. The depth of the constituency-based parse tree is decided by a simple but effective strategy as formula:

$$d = \min(\max(L \times \lambda, \gamma_1), \gamma_2) \quad (8)$$

where L is the length of the input sentence, γ_1 is the lower bound, γ_2 is the upper bound depth of the sub-tree and λ is the ratio of the length of source sentence. When the λ approximates 1.0, the template contains more target tokens and less tags. In addition, we tune the depth on the LDC training data and list the results. According to the Table 5, the soft templates of the specific depth provide helpful information to the translation procedure when the $\lambda = 0.15$ in the LDC dataset.

λ	MT03	MT05	MT08	MT12
0.10	45.92	45.01	36.55	35.34
0.15	**46.56**	**46.04**	**37.53**	**35.99**
0.20	46.02	45.20	37.08	35.82
0.25	46.27	44.83	36.88	35.64
0.30	46.08	45.02	36.72	35.54
0.35	46.22	44.92	36.84	35.51
0.40	46.32	45.40	36.94	35.61

Table 5: The results of the different depth on NIST2003, NIST2005, NIST2008 and NIST2012.

λ	MT03	MT05	MT08	MT12
0.15	79.4	81.6	78.6	77.6

Table 6: The ratio(%) of overlapping words between the predicted soft target template and the translation on NIST2003, NIST2005, NIST2008 and NIST2012.

3.9 Ratio of Overlapping Words

To measure contribution of the predicted soft target template for final translation, we calculate the overlapping words between the template and the translation. Table 6 gives the specific overlapping words ratio on the different test sets including NIST2003, NIST2005, NIST2008 and NIST2012. The overlapping ratio is calculated by the following formula:

$$ratio = \frac{\sum_{w \in T} \min\left(Count_y(w), Count_t(w)\right)}{\sum_{w \in T} Count_t(w)} \quad (9)$$

where $Count_y(\cdot)$ and $Count_t(\cdot)$ denote the number of w in the target translation Y and the template T, and w is the words in the target language. The overlapping ratio represents the correlation between the predicted template T and the target translation Y. According to Table 6, the correlation between the template T and the translation Y is highly relevant which demonstrates the contribution of our template to the final translation.

Source	另一方面 , 如果 我们 反应 过度 , 将 会 被 他们 欺骗 .
Reference	on the other hand , if we overreact , we will be deceived by their trick .
Template	on the other hand , if NP VP , we will VP .
Ours	on the other hand , if we react too much , we will be hit by them .

Table 7: A Chinese-English translation example of our proposed method. VP and NP represent non-terminal nodes in the constituency-based parse tree.

3.10 Example Study

To further illustrate which aspects of NMT are improved by the target soft template, we provide a Chinese-English translation example shown in 7. Templates provide the structural and grammatical information of the target sentence. For instance, Chinese source sentence "另一方面 , 如果 我们 反应 过度 , 将 会 被 他们 欺骗 ", our model first predicts the target template "on the other hand , if NP VP , we will VP ", and then generate the final translation "on the other hand , if we react too much, we will be hit by them". Our target template provides the sentence pattern "If sb. do sth, sb. will be done". Our method introduces the constituency-based parse tree and utilizes the constituency grammar to distinguish terminal and non-terminal nodes. Therefore, our model can automatically learn sentence patterns, including grammatical and structural information.

4 Related Work

Many types of encoder-decoder architecture (Bahdanau et al., 2015; Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017; Chen et al., 2018) have been proposed in the past few years. Furthermore, Transformer enhances the capability of NMT in capturing long-distance dependencies based on these backbone models, including CNN-based, RNN-based, and Transformer based architecture.

To improve the quality of the translation, many authors have endeavored to adopt multi-pass generation decoding method, their models first predict the rough translation and then generate the final translation based on the previous draft (Niehues et al., 2016; Chatterjee et al., 2016; Junczys-Dowmunt and Grundkiewicz, 2017; Xia et al., 2017; Geng et al., 2018; Wang et al., 2019b).

Besides, some works (Liu et al., 2016; Zhang et al., 2018; Zhou et al., 2019b,a) use the right-to-left (R2L) and left-to-right (L2R) to improve the quality of machine translation. Non-Autoregressive decoding (Ghazvininejad et al., 2019) first predicts the target tokens and masked tokens, which will be filled in the next iterations. Then, the model predicts the unmasked tokens on top of the source text and a mixed translation consisting of the masked and unmasked tokens. Semi-autoregressive also (Akoury et al., 2019) predicts chunked fragments or the unmasked tokens based on the tree structure before the final translation. In addition, there are many existing works (Eriguchi et al., 2016; Aharoni and Goldberg, 2017; Wu et al., 2017; Wang et al., 2018; Dong and Lapata, 2018; Wang et al., 2018; Gu et al., 2018) which incorporate syntax information or the tree structure into NMT to improve the quality of translation results.

5 Conclusion

In this work, we propose a novel approach that utilizes source text and additional soft templates. More specifically, our approach can extract the templates from the sub-tree, which derives from the specific depth of the constituency-based parse tree. Then, we use a Transformer model to predict the soft target templates conditioned on the source text. On top of soft templates and source text, we incorporate the template information to guide the translation procedure. We compare our soft-template neural machine translation (ST-NMT) with other baselines on four benchmarks and multiple language pairs. Experimental results show that our ST-NMT significantly improves performance on these datasets.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos.U1636211, 61672081,61370126), the Beijing Advanced Innovation Center for Imaging Technology (Grant No.BAICIT2016001), and the Fund of the State Key Laboratory of Software Development Environment (Grant No.SKLSDE2019ZX-17).

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *ACL 2017*, pages 132–140.

Nader Akoury, Kalpesh Krishna, and Mohit Iyyer. 2019. Syntactically supervised transformers for faster neural machine translation. In *ACL 2019*, pages 1269–1281.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *ACL 2018*, pages 152–161.

Rajen Chatterjee, José G. C. de Souza, Matteo Negri, and Marco Turchi. 2016. The FBK participation in the WMT 2016 automatic post-editing shared task. In *WMT 2016*, pages 745–750.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *ACL 2018*, pages 76–86.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *ACL 2018*, pages 731–742.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *EMNLP 2017*, pages 866–874.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *ACL 2016*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML 2017*, pages 1243–1252.

Xinwei Geng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2018. Adaptive multi-pass decoder for neural machine translation. In *EMNLP 2018*, pages 523–532.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP 2019*, pages 6111–6120.

Jetic Gu, Hassan S. Shavarani, and Anoop Sarkar. 2018. Top-down tree structured decoding with syntactic connections for neural machine translation and parsing. In *EMNLP 2018*, pages 401–413.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *TACL*, 6:437–450.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2017. An exploration of neural sequence-to-sequence architectures for automatic post-editing. In *IJCNLP 2017*, pages 120–129.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR 2015*.

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

Lemao Liu, Masao Utiyama, Andrew M. Finch, and Eiichiro Sumita. 2016. Agreement on target-bidirectional neural machine translation. In *NAACL 2016*, pages 411–416.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL 2014*, pages 55–60.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In *LREC 2016*.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2016. Pre-translation for neural machine translation. In *COLING 2016*, pages 1828–1836.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *WMT 2018*, pages 1–9.

Gaurav Pandey, Danish Contractor, Vineet Kumar, and Sachindra Joshi. 2018. Exemplar encoder-decoder for neural conversation generation. In *ACL 2018*, pages 1329–1338.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL 2002*, pages 311–318.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL 2016*, pages 1715–1725.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *ICML 2019*, pages 5976–5985.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS 2017*, pages 5998–6008.

Kai Wang, Xiaojun Quan, and Rui Wang. 2019a. Biset: Bi-directional selective encoding with template for abstractive summarization. In *ACL 2019*, pages 2153–2162.

Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. 2018. A tree-based decoder for neural machine translation. In *EMNLP 2018*, pages 4772–4777.

Yiren Wang, Yingce Xia, Fei Tian, Fei Gao, Tao Qin, Cheng Xiang Zhai, and Tie-Yan Liu. 2019b. Neural machine translation with soft prototype. In *NIPS 2019*, pages 6313–6322.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2018. Learning neural templates for text generation. In *EMNLP 2018*, pages 3174–3187.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *ICLR 2019*.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *ACL 2017*, pages 698–707.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *NIPS 2017*, pages 1784–1794.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. In *AAAI 2018*, pages 5698–5705.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019a. Synchronous bidirectional neural machine translation. *TACL*, 7:91–105.

Long Zhou, Jiajun Zhang, Chengqing Zong, and Heng Yu. 2019b. Sequence generation: From both sides to the middle. In *IJCAI 2019*, pages 5471–5477.