Modeling Fluency and Faithfulness for Diverse Neural Machine Translation

Yang Feng[1,2] **Wanying Xie**[1,3] **Shuhao Gu**[1,2] **Chenze Shao**[1,2]
Wen Zhang[4] **Zhengxin Yang**[1,2] **Dong Yu**[3*]
[1] Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Beijing Language and Culture University, China
[4] Smart Platform Product Department of Tencent Inc., China
{fengyang, gushuhao19b, shaochenze18z, yangzhengxin17z}@ ict.ac.cn
xiewanying07@gmail.com, kevinwzhang@tencent.com, yudong@blcu.edu.cn

Abstract

Neural machine translation models usually adopt the teacher forcing strategy for training which requires the predicted sequence matches ground truth word by word and forces the probability of each prediction to approach a 0-1 distribution. However, the strategy casts all the portion of the distribution to the ground truth word and ignores other words in the target vocabulary even when the ground truth word cannot dominate the distribution. To address the problem of teacher forcing, we propose a method to introduce an evaluation module to guide the distribution of the prediction. The evaluation module accesses each prediction from the perspectives of fluency and faithfulness to encourage the model to generate the word which has a fluent connection with its past and future translation and meanwhile tends to form a translation equivalent in meaning to the source. The experiments on multiple translation tasks show that our method can achieve significant improvements over strong baselines.

Introduction

Neural machine translation (NMT) (Kalchbrenner and Blunsom 2013; Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2014; Gehring et al. 2017; Vaswani et al. 2017; Zhang et al. 2019b) has shown its superiority and drawn much attention recently. Most NMT models can fit in the attention-based encoder-decoder framework where the encoder projects the source sentence into representations in a common concept space and the decoder first retrieves related information from these representations and then decodes it into target translation word by word. The training scenario is that each sentence is provided with a ground truth sequence and the teacher forcing strategy (Williams and Zipser 1989) is employed to force the generated translation to approach ground truth word by word via a cross-entropy loss. In this way, the probability distribution of each prediction is expected to reach a 0-1 distribution with the ground truth word approaching to 1 and other words in the target vocabulary close to 0.

However, in practice the translation model cannot always generate the ground truth translation even with teacher forcing during training. One reason is that a source sentence can have multiple gold translations and hence in the training data there may be several different expressions for the same meaning segment. Another reason is that the model cannot fit to the training data perfectly due to noise and the expression ability of the model. But unfortunately teacher forcing ignores unreachable optimization situations where the ground truth word can only struggle to account for a small portion of the whole distribution, but still propagates the supervision only through the path from the ground truth word, having the major portion of the distribution excluded.

In this sense, it is beneficial to introduce an evaluation mechanism to guide the distribution when the prediction cannot converge to the ground truth word. If the prediction gives an alternative gold expression, the evaluation mechanism can give a proper evaluation to this word, otherwise, it can offer another distribution, rather than the 0-1 distribution, which can be seen as a lower optimization bound, to guide the training of the model. For evaluation, a translation is a good translation only when it can form a fluent sentence in the target and as well express the meaning of the source faithfully. Therefore, in the scenario of teacher forcing, each prediction should be evaluated from the perspectives of fluency and faithfulness. For the fluency, the predicted word should be estimated whether to form a fluent sentence together with its past and future translation. For the faithfulness, the predicted word should be accessed whether to translate proper source information so that the whole source and target sequences have equivalent meanings.

In this paper, we follow the above analysis to solve the problem of teacher forcing. Whenever generating a target word, we evaluate the fluency by connecting it with the self-generated past translation and the future ground truth translation and calculating the co-occurrence probability of the three parts. And we meanwhile estimate the faithfulness by first figuring out the cross-attention over the source sentence with the past and future translation and then computing the probability of translating the attention into the target word. Furthermore, to trade off the fluency against the faithfulness, we integrate the two evaluation metrics together into

[*]Corresponding author: Dong Yu.

Our code can be got at https://github.com/ictnlp/DiverseNMT a unified evaluation module and adjust their weights automatically. Finally the evaluation module is introduced to the NMT model to guide the distribution of predictions. The experiments on multiple translation tasks show that our method can outperform strong baselines significantly without any additional load for decoding. And the analysis experiments indicate that our method can have a better parameter fitting and produce more reasonable translation.

Background

In this paper, we will introduce how to apply our method under the framework of *Transformer* (Vaswani et al. 2017) which has an encoder-decoder structure, so before diving into details, we will first introduce Transformer briefly. We denote the input sequence of symbols as $\mathbf{x} = (x_1, ..., x_J)$, the ground truth sequence as $\mathbf{y}^* = (y_1^*, ..., y_I^*)$ and the generated translation as $\mathbf{y} = (y_1, ..., y_I)$.

The Encoder

The encoder is composed of a stack of N identical layers with each layer having two sublayers. The first sublayer is a multi-head attention unit used to compute the self-attention of the input, named *multi-head sublayer*, and the second is a fully connected feed-forward network, named *FNN sublayer*. Both of the sublayers are followed by a residual connection operation and a layer normalization operation. The multi-head attention unit adopts dot-product attention which processes a set of queries ($\mathbf{Q}$), keys ($\mathbf{K}$) and values($\mathbf{V}$) simultaneously, denoted as $\mathrm{MutiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$.

For the n-th layer of the encoder, the multi-head sublayer can be formalized as

$$\mathbf{Z}^n = \mathrm{AddNorm}(\mathrm{MutiHead}(\mathbf{H}^{n-1}, \mathbf{H}^{n-1}, \mathbf{H}^{n-1}))$$

where $\mathbf{H}^{n-1} \in \mathbb{R}^{J \times d_{\mathrm{model}}}$ is the matrix of the packed output of the $n-1$-th layer. A special case is that the queries, keys and values for the first layer of the encoder are all the matrix of the packed input embeddings $\mathbf{E}_x = [\mathrm{E}_x[x_1]; ...; \mathrm{E}_x[x_J]]^T$ where $\mathrm{E}_x[x_j]$ is the sum of the embedding and position embedding of the source word x_j. Then the FFN sublayer of the n-th layer is formalized as

$$\mathbf{H}^n = \mathrm{AddNorm}(\mathrm{FFN}(\mathbf{Z}^n)). \tag{1}$$

Then the output of the N-th layer is taken as source hidden states and we denote its packed matrix as $\mathbf{H}$.

The Decoder

The decoder is also composed of a stack of N identical layers. For each layer, besides the multi-head sublayer and the FFN sublayer, a third sublayer is inserted, called *cross-attention sublayer*. The cross-attention sublayer performs multi-head attention over source hidden states with the output of the multi-head sublayer in the same layer as query. Residual connection and layer normalization are also applied after each sublayer. In addition, as we do not know the future translation, a mask matrix is applied to prevent the subsequent target words from being involved.

Formally, for the n-th layer of the decoder, the multi-head sublayer is denoted as

$$\mathbf{A}^n = \mathrm{AddNorm}(\mathrm{MutiHead}(\mathbf{S}^{n-1}, \mathbf{S}^{n-1}, \mathbf{S}^{n-1})$$

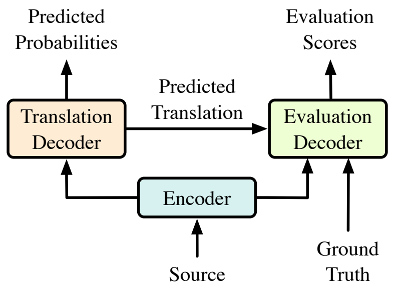

Figure 1: The architecture of the proposed method.

where $\mathbf{S}^{n-1} \in \mathbb{R}^{I \times d_{\mathrm{model}}}$ is the output of the $n-1$-th layer and specifically the queries, keys and values for the first layer are all the target embedding matrix $\mathbf{E}_y$. The cross-attention sublayer is written as

$$\mathbf{C}^n = \mathrm{AddNorm}(\mathrm{MutiHead}(\mathbf{A}^n, \mathbf{H}, \mathbf{H}) \tag{2}$$

and the FFN sublayer is formalized as

$$\mathbf{S}^n = \mathrm{AddNorm}(\mathrm{FFN}(\mathbf{C}^n)) \; . \tag{3}$$

After these operations, the final output of the N-th layer gives the target hidden states, denoted as $\mathbf{S} = [\mathbf{s}_1; ...; \mathbf{s}_I]^T$, where $\mathbf{s}_i$ is the hidden state of y_i.

By performing a linear transformation and a softmax operation to the target hidden states, we can get the translation probability as

$$p(y_i|\mathbf{y}_{<i}, \mathbf{x}) \propto \exp\left(\mathbf{s}_i \mathbf{W}_o\right) \tag{4}$$

where $\mathbf{W}_o \in \mathbb{R}^{d_{\mathrm{model}} \times |V_t|}$ and $|V_t|$ is the size of the target vocabulary.

Transformer is trained by minimizing a cross-entropy loss which maximizes the probability of the ground truth sequence:

$$\mathcal{L} = -\sum_{i=1}^{I} \log p(y_i^*|\mathbf{y}_{<i}, \mathbf{x}) \tag{5}$$

Model

Our work aims to introduce an evaluation module into the NMT model to provide a more reachable distribution to fit, as a complement to the 0-1 distribution adopted by the cross-entropy loss, so our model consists of two modules. One is *the translation module*, composed of the encoder and the translation decoder, which is used to generate candidate translations in the same way as Transformer. The other is *the evaluation module*, containing the encoder and the evaluation decoder, which is used to evaluate the translation produced by the translation module word by word. The whole architecture is shown in Figure 1. Please note that the encoder is shared by the two decoders. Then the distribution drawn by the evaluation module is used in an additional loss to guide the distribution drawn by the translation module. Then at test our model leaves out the evaluation module and performs inference only with the translation module.

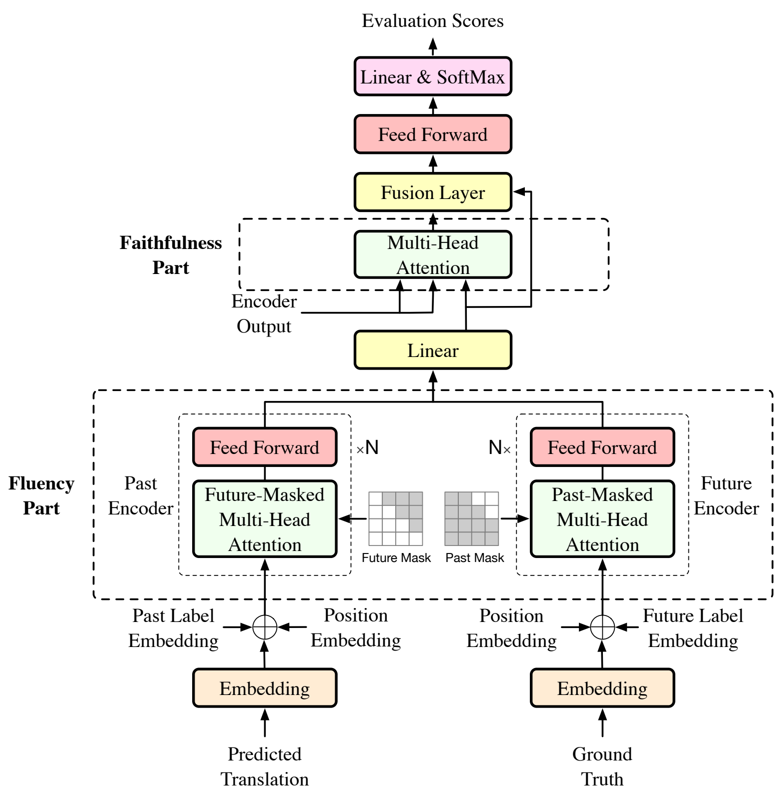

Figure 2: The architecture of the evaluation module.

The Evaluation Module

In our method, whenever the translation module generates a target word, the evaluation module will assess it from the perspectives of fluency and faithfulness. The two metrics are integrated together to come out with a final score so as to trade off against each other. Thus the evaluation module can be logically divided into three parts as the fluency part, the faithfulness part and the fusion layer. The architecture is shown in Figure 2.

The Fluency Part

To ensure the fluency of the whole translation, each generated word should be a good conjunction with its past and future translation. At the time step i, we use as *the past translation* the translation predicted by the translation module, denoted as $y_1, ..., y_{i-1}$. Under the training framework of teacher forcing, in the following steps the model will be taught to generate the rest of the ground truth sequence $y^*_{i+1}, ..., y^*_I$, so we use this as *the future translation*. Please note that the past translation is predicted during training which means the context rolled in the decoder is ground truth words, not self-generated words.

Then to get the representation for the past translation and the future translation, we employ two encoders, called *past encoder* and *future encoder*. Then the fluency of the word y_i can be assessed conditioned on the representations of the past and future information, just like the evaluation of language models.

The past encoder consists of a stack of N identical layers with each layer having two sublayers of a multi-head sublayer and an FFN sublayer, just like the encoder of Transformer. In order to support parallel training, we first collect the whole sequence generated by the translation module with ground truth words as context, denoted as $y_1, ..., y_I$, and then feed the whole sequence to the past encoder. Then to pick the past translation at each time step i, we mask out the current and future translation $y_i, ..., y_I$ in the multi-head sublayer, corresponding to *future-masked multi-head attention* in Figure 2.

The future encoder has the same structure except that its input is the whole ground truth sequence and at the time step i, the masked part is the past and current ground truth words $y^*_1, ..., y^*_i$. Its multi-layer sublayer corresponds to *past-masked multi-head attention* in Figure 2.

In addition, besides the word embedding and the position embedding, we introduce a new label embedding with two labels to indicate whether a word comes from a past translation or a future translation. Then the final embeddings fed to the two encoders are the sum of the three embeddings.

Assume the hidden state matrices outputted by the past and future encoders are $\mathbf{A}_p$ and $\mathbf{A}_f$, respectively, then the two outputs are fused together to produce the condition $\mathbf{A}_e$ for fluency estimation as

$$\mathbf{A}_e = \mathbf{W}_p\mathbf{A}_p + \mathbf{W}_f\mathbf{A}_f \quad (6)$$

where $\mathbf{W}_p$ and $\mathbf{W}_f$ are linear transformations.

The Faithfulness Part

In addition to fluency, the generated word should also reflect a proper amount of source meaning so that the whole translation can express the source sentence faithfully. Here faithfulness means being adequate and accurate in meaning. We model the evaluation for faithfulness as a translation task and estimate the translation probability (faithfulness) word by word. The scenario is to first retrieve the related source information and then to assess the probability of translating the retrieved information to the given target word.

To figure out the corresponding source information, we perform the cross-attention over the source hidden states $\mathbf{H}$ generated by the shared encoder, using as the query the fusion of the past and future information $\mathbf{A}_e$. This can be formalized as

$$\mathbf{C}_e = \mathrm{AddNorm}(\mathrm{MutiHead}(\mathbf{A}_e, \mathbf{H}, \mathbf{H})) \tag{7}$$

where $\mathbf{C}_e$ is the generated cross-attention. Then the translation probability is calculated in the same way as Transformer does.

The Fusion Layer

As the generated translation is assessed with two metrics of fluency and faithfulness, the two metrics should be traded off against each other. For the two metrics both aim to get a conditional probability for the current generated word, we fuse the conditions of them, $\mathbf{A}_e$ and $\mathbf{C}_e$, to get a new combined condition which is followed by an FFN layer. This process can be written as

$$\mathbf{B}_e = \mathbf{W}_a\mathbf{A}_e + \mathbf{W}_c\mathbf{C}_e \tag{8}$$

$$\mathbf{S}_e = \mathrm{AddNorm}(\mathrm{FFN}(\mathbf{B}_e)) \tag{9}$$

where $\mathbf{S}_e = [\mathbf{s}_{e1}; ...; \mathbf{s}_{eI}]^T$ and $\mathbf{s}_{ei}$ is the hidden state for the word y_i. Then with a linear transformation $\mathbf{W}_e$ and a softmax operation, we can get the final evaluation score for y_i as

$$p_e(y_i|\mathbf{y}^*_{>i}, \mathbf{y}_{<i}, \mathbf{x}) \propto \exp(\mathbf{s}_{ei}\mathbf{W}_e) \tag{10}$$

Training

During training, our method not only jointly optimizes the translation module and the evaluation module but employs an additional loss to guide the behavior of the translation module. Specifically, for the translation module, a cross-entropy loss is employed as

$$\mathcal{L}_t = -\sum_{k=1}^{K}\sum_{i=1}^{I} \log p(y^*_i|\mathbf{y}_{<i}, \mathbf{x}) \ . \tag{11}$$

The evaluation module is also optimized via a cross-entropy loss as

$$\mathcal{L}_e = -\sum_{k=1}^{K}\sum_{i=1}^{I} \log p_e(y^*_i|\mathbf{y}^*_{>i}, \mathbf{y}_{<i}, \mathbf{x}) \ . \tag{12}$$

For the correlation of the two modules, a common practice is to employ the Kullback-Leibler (KL) divergence as the loss to make sure the distributions drawn by the two modules get close to each other. However, it is not optimal to bind the two distributions over the whole target vocabulary as this will hinder the model to search for a better minimum. Instead, we only pay attention to the word generated by the translation module and use the evaluation module to guide the probability given by the translation module. This loss is given as

$$\mathcal{L}_\mathrm{c} = \sum_{k=1}^{K}\sum_{i=1}^{I} p_e(y_i|\mathbf{y}^*_{>i}, \mathbf{y}_{<i}, \mathbf{x}) \log p(y_i|\mathbf{y}_{<i}, \mathbf{x}) \ . \tag{13}$$

With this loss, if the generated word happens to be the ground truth word, then the distribution drawn by the translation module will be sharper at the ground truth word, otherwise, the translation module tends to reinforce the translation with higher confidence given by the evaluation module. In the experiment section, we will verify that $\mathcal{L}_\mathrm{c}$ can bring about better performance than the KL divergence.

The final loss is

$$\mathcal{L} = \mathcal{L}_t + \mathcal{L}_e + \mathcal{L}_\mathrm{c} \tag{14}$$

In the training, we first pretrain the translation and evaluation modules together with the loss $\mathcal{L}_\mathrm{pretrain} = \mathcal{L}_t + \mathcal{L}_e$. Near convergency, we introduce $\mathcal{L}_\mathrm{c}$ and fine tune the model with the loss in Equation 14.

Related Work

Our work introduces an evaluation module to give an assessment to the predicted word so that it can always have gradient back propagated through. Some researchers also took effort in this direction. Shao, Chen, and Feng (2018) employed a greedy search strategy to generate translation, then used the accuracy of probabilistic n-grams as the training loss. This method calculates the probabilities of n-grams by counting probabilistic occurrences through the entire vocabulary, hence it has all the words involved in the optimization with the sequence-level n-gram loss. Yang et al. (2019) came out with a sentence-level agreement loss to directly model the difference between the representation of the source and target sentences, so that the source representation could be enhanced. Wieting et al. (2019) used a semantic similarity loss as a reward to measure the similarity between the embedding of the generated translation and the reference. EL-BAYAD, Besacier, and Verbeek (2018) extended the training loss with a token-level and sequence-level smoothing loss which smooths the target distribution over similar sentences.

In our method, the self-generated translation other than the ground truth can be involved during gradient back propagation. Some other work also tries to have more translation to take part in parameter update. These work usually adopts the REINFORCE algorithm (Williams 1992) and samples translation to optimize according the probability distribution. This series of work (Wu et al. 2018; Yang et al. 2018; Geng et al. 2018; Kreutzer, Uyheng, and Riezler 2018) samples a translation from all the possible translation and performs gradient descent through the translation with a sequence-level reward under the framework of reinforcement learning .

Our method provides evaluation metrics for translation different from ground truth, allowing for diverse translation. There are also some other work which encourages diverse translation. Ma et al. (2018) presented a bag-of-word loss for the model to generate translation with words in ground truth but in flexible word order. He, Haffari, and Norouzi (2018) developed a sequence-to-sequence mixture model to adopt a committee of translation models. Each translation model selects its own training set via optimization of marginal log-likelihood, leading to a soft clustering of the training data. By this mean, the method can improve the diversity and quality. Shu, Nakayama, and Cho (2019) attempted to generate diverse translation by first extract sentence codes with or without syntax information and then sampling translation based on the sentence codes.

Our method evaluates from the perspectives of fluency and faithfulness and from this point of view, the line of work which first generate future translation also work for better fluency. Hassan et al. (2018) and Zhang et al. (2018) proposed a two-pass decoding algorithm which first generated a translation draft then refined while Zhang et al. (2019a) proposed to decode forwards and backwards simultaneously. The main difference from our work is that they require decoding decoding bi-directionally while our method can decode once just like Transformer. Although Serdyuk et al. (2018) also employ bidirectional decoding during training, they instead use the backward decoder to assistant the forward decoder, so that the forward decoder can generate hidden states close to the backward decoder. In this way, the forward decoder can produce similar translation to the backward decoder in the rest part and hence the backward decoder can be abandoned at test.

Experiments

In the experiment section, we will first report the comparison results with other strong baselines, then analyze the importance of all the factors in the model, then show the upper bound of the performance of the evaluation module, and next verify whether our method can achieve better optimization. Finally, we will indicate whether our method can generate translation of better fluency and faithfulness.

Data Preparation

We conducted experiments on the following three data sets.

CN→EN The training data consists of 1.25M sentence pairs from LDC corpora which has 27.9M Chinese words and 34.5M English words respectively [1]. The data set MT02 is used as validation and MT03, MT04, MT05, MT06, MT08 are used for test. We tokenized and lowercased English sentences using the Moses scripts[2], and segmented the Chinese sentences with the Stanford Segmentor[3]. The two sides were further segmented into subword units using Byte-Pair Encoding(BPE) (Sennrich, Haddow, and Birch 2016) with 30K merge operations.

EN→DE The training data is from WMT2014 which consists about 4.5M sentences pairs with 118M English words and 111M German words. We chose the news test-2013 for validation and news-test 2014 for test. BPE was also employed with 32K merge operations.

EN→RO We used the preprocessed version of WMT16 English-Romanian dataset released by Lee, Mansimov, and Cho (2018) which includes 0.6M sentence pairs. We use news-dev 2016 for validation and news-test 2016 for test. The two languages share the same vocabulary generated with 40K merge operations of BPE.

Systems

We conducted our experiments based on self-attention-based encoder-decoder frame.

TRANSFORMER An open-source toolkit called *Fairseq-py* released by Facebook (Edunov, Ott, and Gross 2017) which was implemented strictly referring to Vaswani et al. (2017).

+RL Transformer trained under the reinforcement learning framework with the BLEU as the rewards, specifically the REINFORCE algorithm (Williams 1992). The implementation details for the RL part is the same as Yang et al. (2018).

+BOW Our implementation of Ma et al. (2018) on the basis of Transformer.

Our Method-KL Implemented based on Fairseq-py. For the evaluation module, the fluency part is composed of a stack of N = 6 layers. And the final loss for this system is

$$\mathcal{L} = \mathcal{L}_t + \mathcal{L}_e + \mathcal{L}_{\mathrm{KL}}$$

where

$$\mathcal{L}_{\mathrm{KL}} = \sum_{k=1}^{K} \sum_{\mathbf{y}_i \neq \mathbf{y}_i^*} \mathrm{D}_{\mathrm{KL}}(p_e(y_i|\mathbf{y}_{>i}^*, \mathbf{y}_{<i}, \mathbf{x})||p(y_i|\mathbf{y}_{<i}, \mathbf{x})) \,.$$

The KL loss forces the distributions drawn by the translation module and the evaluation module to approach to each other when the word generated by the translation module is different from the ground truth word.

Our Method Implemented the same as the system *Our Method-KL* except that its final loss is $\mathcal{L} = \mathcal{L}_t + \mathcal{L}_e + \mathcal{L}_{\mathrm{c}}$ as shown in Equation 14.

All the Transformer-based systems have the same configuration as the base model described in Vaswani et al. (2017).

The translation quality was evaluated using the *multi-bleu.pl* scipt (Papineni et al.) based on case-insensitive n-gram matching with n up to 4.

Performance

We compare with methods using different losses, such as Yang et al. (2018) (named +RL) which gives a reward to all the possible translation and perform policy gradient propagation via REINFORCE algorithm, and Ma et al. (2018) (named +BOW) which encourage diverse translation via a bag-of-words loss. We also compared the two different additional losses previously mentioned $\mathcal{L}_{\mathrm{KL}}$ and $\mathcal{L}_{\mathrm{c}}$.

The results are shown in Table 1. We can see that the improvement brought by +RL and +BOW methods are not great. For the +RL method, as we all know, the training is

[1]The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

[2]http://www.statmt.org/moses/

[3]https://nlp.stanford.edu/

	CN→EN							EN→DE		EN→RO	
	MT03	**MT04**	**MT05**	**MT06**	**MT08**	**AVE**	**Δ**	**WMT14**	**Δ**	**WMT16**	**Δ**
TRANSFORMER	44.74	46.27	44.16	43.29	34.72	42.63		27.21		32.85	
+RL	44.50	45.96	44.26	43.92	35.55	42.83	*+0.20*	27.25	*+0.04*	33.00	*+0.15*
+BOW	44.59	46.40	45.03	43.91	35.31	43.04	*+0.41*	27.35	*+0.14*	32.95	*+0.10*
Our Method-KL	45.17	46.86**	45.01**	44.51*	36.03*	43.51	*+0.88*	**27.55**	***+0.34***	33.44	*+0.59*
Our Method	**46.20***	**47.39***	**46.22***	**45.63***	**36.78***	**44.44**	***+1.81***	27.35	*+0.14*	**34.00***	***+1.15***

Table 1: BLEU scores on three translation tasks. * and ** mean the improvements over TRANSFORMER is statistically significant (Collins, Koehn, and Kucerova 2005) ($\rho < 0.01$ and $\rho < 0.05$, respectively).

	MT03	**MT04**	**MT05**	**MT06**	**MT08**	**AVE**
Full	46.20	47.39	46.22	45.63	36.78	44.44
-Faithfulness	44.95	46.85	45.15	44.45	36.18	43.51
-$\mathcal{L}_c$	44.93	45.77	44.61	44.76	35.87	43.18
-Evaluation	44.74	46.27	44.16	43.29	34.72	42.63

Table 2: Ablation study on the CN→EN translation task. Full: our full model. -Faithfulness: deleting cross-attention from the faithfulness part. -$\mathcal{L}_c$: erasing $\mathcal{L}_c$ in Equation 14. -Evaluation: removing the evaluation decoder (degrading to Transformer).

	MT03	**MT04**	**MT05**	**MT06**	**MT08**	**AVE**	**EN→DE**
Tran	31.31	23.87	29.02	30.61	22.26	27.41	31.76
Eval	45.86	39.34	44.18	45.21	36.02	42.12	34.56

Table 3: Performance comparison of the translation and evaluation modules on the CN→EN and EN→DE translation tasks. Ground truth is fed to both the modules and only one reference was used for the CN→EN translation.

not stable and thus it is difficult to converge to good optima. For the +BOW, the reason may be it does not model the word order in the training loss and hence has a loose supervision to the fluency. On the CN→EN and EN→RO translation tasks, the loss $\mathcal{L}_c$ is much more effective than $\mathcal{L}_{KL}$ while on the EN→DE translation the result is reversed. Another finding is that all the methods cannot achieve big improvements on the EN→DE translation. The reason may be the training data is big enough and the two languages EN and DE are closed to each other. As a result, Transformer has already got a good enough optimization and it is difficult for further improvements. Our method with $\mathcal{L}_c$ as an additional loss can outperform all the baselines significantly on the CN→EN and EN→RO translation tasks. Therefore, we can conclude that the evaluation module can help improve translation performance and the loss $\mathcal{L}_c$ is more reasonable.

Ablation Study

Our method introduces three new factors into Transformer via the evaluation module, including the cross attention for faithfulness, past and future encoders for fluency and the additional loss $\mathcal{L}_c$. Here we conducted experiments to check their influence to our method by leaving them out one by one. The results are given in Table 2.

We first left out the cross-attention unit which means we discard faithfulness and only consider fluency in our method, and find that the translation performance decreases greatly. This is in line with our conjecture that a good translation should be ensured to be faithful with the source meaning. We also tried to only abandon the loss $\mathcal{L}_c$ in Equation 14 and now the evaluation module can only participate in the optimization of the shared encoder (referring to Figure 1). We can find the performance declines most of all. This is not difficult to understand. When the predicted word is different from the ground truth word, the probability of the ground truth word only accounts for a small portion of the distribution and the cross-entropy loss distributing the rest great portion to 0 is not reasonable. Next we excluded the whole evaluation decoder, the performance further declines. This indicates that even we do not employ the distribution generated by the evaluation module to directly guide the behaviors of the translation module, the evaluation decoder can help seek better parameters for the share encoder.

The Performance of the Evaluation Module

As we use the evaluation module to guide the probability distribution of the translation model, to make sure the reasonability of this mechanism, the evaluation module should have an obvious superiority in the performance over the translation module. We conducted experiments to check this. For the evaluation module requires the participation of ground truth, we fed the reference of the test set to the evaluation module, so that the evaluation module can assess the fluency with self-generated translation as past translation and ground truth as future translation. For the CN→EN translation, we only selected one reference to feed and calculated BLEU scores based on this single reference. In order to compare fairly, we also fed reference to the translation module to use it as context.

According to the results in Table 3, we can find that the evaluation module indeed has a great margin in performance over the translation module consistently on all the test sets.

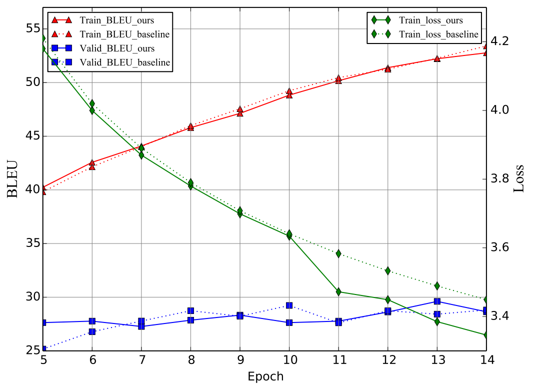

Figure 3: Training losses and BLEU scores on the CN→EN training and valid sets.

The margin is bigger on the CN→EN translation than on the EN→DE translation which also gives an explanation to why the improvement on the CN→EN is much bigger. Then we can conclude that it is reasonable to leverage the evaluation module to help optimize the translation module.

The Reasonability of the Loss

Our method uses the evaluation module to introduce a new distribution, then it adds $\mathcal{L}_c$ to the loss, aiming to achieve a better optimization with the help of the new distribution. We design experiments to check this in two aspects. First is whether the loss in Equation 14 is reasonable, which means lower loss can lead to better translation (greater BLEU scores) on the training data. Second is whether the loss can result in better optimization, that is higher BLEU scores on the valid data.

The experiment details are as follows. We first pretrained our method with the loss $\mathcal{L}_{\text{pretrain}} = \mathcal{L}_t + \mathcal{L}_e$ for the first 10 epochs and then added $\mathcal{L}_c$ to the loss afterwards. Then we sampled 1000 sentences from the training set and tested the BLEU score on sampled training set with ground truth words as context. In this way, we can see whether the translation module behaves as we want. We also tested BLEU scores on the valid set with self-generated translation as context.

We put training losses and BLEU scores all in the Figure 3 with two scalars for y-axis. From the results, we can see that the training loss of our method decreases gradually before $\mathcal{L}_c$ is added (epoch 10) and afterwards declines greatly although there are more loss terms. Meanwhile, the BLEU scores on the training set keep rising after $\mathcal{L}_c$ is added till converged. This shows lower training losses correspond to better translation which can be a proof that the training loss is reasonable.

Our method converged on the 13th epoch and Transformer converged on the 9th epoch. When converged, our method has higher BLEU scores on both the training and valid sets, then We can think our method reaches better optimization than Transformer.

	1-gram	**2-gram**	**3-gram**	**4-gram**	**Cosine**
Transformer	79.10	52.72	35.34	23.82	0.873
Our method	79.82	54.18	36.90	25.28	0.877

Table 4: N-gram accuracy and cosine similarity on CN→EN translation. N-gram accuracy is the average on all the test sets. Cosine similarity is calculated with the average embeddings of the translation against that of the reference.

The Fluency and Faithfulness of the Translation

As the evaluation module is designed to consider the fluency and faithfulness of the translation, we assess whether this brings the improvement on the two metrics. For fluency, we test the n-gram accuracy on all the test sets where the n-gram accuracy is the ratio of the number of matched n-gram between translation and reference against the total number of n-gram in the translation. For faithfulness, the cosine similarity between translation and reference is calculated using the average embeddings of all words.

According to the results in Table 4, our method can generate translation with higher n-gram accuracy for order 1 to order 4 and the difference becomes wider as the order increases. Higher 1-gram accuracy indicates that our method can reach more ground truth in optimization and this is another proof that the loss we use can lead to better optima. Besides, the greater accuracy on n-gram, especially on 3-gram and 4-gram, implies better fluency. Furthermore, our method can have a bigger cosine similarity to the reference and this means the generate translation is more faithful in meaning to the source sentence. In conclusion, our method can produce translation with better fluency and faithfulness.

Conclusions

Teacher forcing employs a cross-entropy loss to supervise the training with a 0-1 distribution and back-propagates gradients only through the ground truth words . When the ground truth word cannot dominate the probability distribution, a major portion of the distribution is discarded, leading to poor optimization. To solve this problem, we introduce an evaluation module to draw a new distribution over all the words and further use the new distribution to guide the training. To make a proper evaluation, we estimate the translation from the perspectives of fluency and faithfulness to appreciate the translation which is fluent in the target and faithful in meaning to the source. The experiments prove that our method can get better performance on multiple data sets with better optimization and meanwhile the generated translation is more fluent in the target and more faithfulness to the source.

Acknowledgements

We thank all the anonymous reviewers for their insightful and valuable comments. This work was supported by National Natural Science Foundation of China (NO61876174) and National Key R&D Program of China (NO2017YFE9132900).

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Collins, M.; Koehn, P.; and Kucerova, I. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL 2015*, 531–540.

Edunov, S.; Ott, M.; and Gross, S. 2017. https://github.com/pytorch/ fairseq.

ELBAYAD, M.; Besacier, L.; and Verbeek, J. 2018. Token-level and sequence-level loss smoothing for rnn language models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2094–2103.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1243–1252. JMLR. org.

Geng, X.; Feng, X.; Qin, B.; and Liu, T. 2018. Adaptive multi-pass decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 523–532.

Hassan, H.; Aue, A.; Chen, C.; Chowdhary, V.; Clark, J.; Federmann, C.; Huang, X.; Junczys-Dowmunt, M.; Lewis, W.; Li, M.; et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

He, X.; Haffari, G.; and Norouzi, M. 2018. Sequence to sequence mixture model for diverse machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, 583–592.

Kalchbrenner, N., and Blunsom, P. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1700–1709.

Kreutzer, J.; Uyheng, J.; and Riezler, S. 2018. Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1777–1788.

Lee, J.; Mansimov, E.; and Cho, K. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1173–1182.

Ma, S.; Xu, S.; Wang, Y.; and Lin, J. 2018. Bag-of-words as target for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 332–338.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1715–1725.

Serdyuk, D.; Ke, N. R.; Sordoni, A.; Trischler, A.; Pal, C.; and Bengio, Y. 2018. Twin networks: Matching the future for sequence generation.

Shao, C.; Chen, X.; and Feng, Y. 2018. Greedy search with probabilistic n-gram matching for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4778–4784.

Shu, R.; Nakayama, H.; and Cho, K. 2019. Generating diverse translations with sentence codes. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 1823–1827.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wieting, J.; Berg-Kirkpatrick, T.; Gimpel, K.; and Neubig, G. 2019. Beyond bleu: Training neural machine translation with semantic similarity. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 4344–4355.

Williams, R. J., and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Wu, L.; Tian, F.; Qin, T.; Lai, J.; and Liu, T.-Y. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3612–3621.

Yang, Z.; Chen, W.; Wang, F.; and Xu, B. 2018. Improving neural machine translation with conditional sequence generative adversarial nets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1346–1355.

Yang, M.; Wang, R.; Chen, K.; Utiyama, M.; Sumita, E.; Zhang, M.; and Zhao, T. 2019. Sentence-level agreement for neural machine translation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 3076–3082.

Zhang, X.; Su, J.; Qin, Y.; Liu, Y.; Ji, R.; and Wang, H. 2018. Asynchronous bidirectional decoding for neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhang, J.; Zhou, L.; Zhao, Y.; and Zong, C. 2019a. Synchronous bidirectional inference for neural sequence generation. *arXiv preprint arXiv:1902.08955*.

Zhang, W.; Feng, Y.; Meng, F.; You, D.; and Liu, Q. 2019b. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.