Scheduled Sampling Based on Decoding Steps for Neural Machine Translation

Yijin Liu[1][2], Fandong Meng[2], Yufeng Chen[1] Jinan Xu[1] and Jie Zhou[2]
[1]Beijing Jiaotong University, China
[2]Pattern Recognition Center, WeChat AI, Tencent Inc, China
{yijinliu, fandongmeng, withtomzhou}@tencent.com
{jaxu,chenyf}@bjtu.edu.cn

Abstract

Scheduled sampling is widely used to mitigate the exposure bias problem for neural machine translation. Its core motivation is to simulate the inference scene during training by replacing ground-truth tokens with predicted tokens, thus bridging the gap between training and inference. However, vanilla scheduled sampling is merely based on training steps and equally treats all decoding steps. Namely, it simulates an inference scene with uniform error rates, which disobeys the real inference scene, where larger decoding steps usually have higher error rates due to error accumulations. To alleviate the above discrepancy, we propose scheduled sampling methods based on decoding steps, increasing the selection chance of predicted tokens with the growth of decoding steps. Consequently, we can more realistically simulate the inference scene during training, thus better bridging the gap between training and inference. Moreover, we investigate scheduled sampling based on both training steps and decoding steps for further improvements. Experimentally, our approaches significantly outperform the Transformer baseline and vanilla scheduled sampling on three large-scale WMT tasks. Additionally, our approaches also generalize well to the text summarization task on two popular benchmarks.

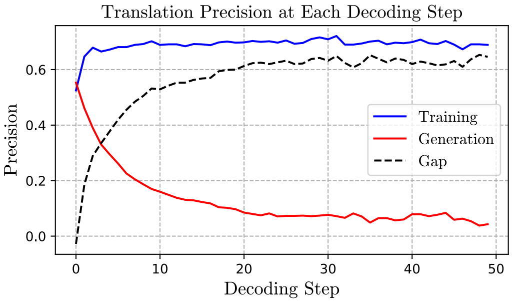

Figure 1: The translation precision for training (blue line) and inference (red line) at each decoding step. The gap between training and inference (black line) increases rapidly with the growth of decoding steps. We randomly sample 100k training data from WMT 2014 EN-DE and report the average precision of 1k tokens for each decoding step[1].

1 Introduction

Neural Machine Translation (NMT) has made promising progress in recent years (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017). Generally, NMT models are trained to maximize the likelihood of next token given previous golden tokens as inputs, *i.e.,* teacher forcing (Salakhutdinov, 2014). However, at the inference stage, golden tokens are unavailable. The model is exposed to an unseen data distribution generated by itself. This discrepancy between training and inference is named as the *exposure bias* problem (Ranzato et al., 2016). With the growth of decoding steps, such discrepancy becomes more problematic due to error accumulations (Zhou et al., 2019; Zhang et al., 2020a) (shown in Figure 1).

Many techniques have been proposed to alleviate the exposure bias problem. To our knowledge, they mainly fall into two categories. The one is sentence-level training, which treats the sentence-level metric (*e.g.,* BLEU) as a reward, and directly maximizes the expected rewards of generated sequences (Ranzato et al., 2016; Shen et al., 2016; Rennie et al., 2017; Pang and He, 2021). Although intuitive, they generally suffer from slow and unstable training due to the high variance of policy gradients and the credit assignment problem (Sutton, 1984; Wiseman and Rush, 2016; Liu et al., 2018; Wang et al., 2018). Another category is sampling-based approaches, aiming to simulate the data distribution

[1] To calculate the precision for training, we strictly match predicted tokens with ground-truth tokens word by word. When inference, we relax the strict matching to the fuzzy matching within a local window of size 3, and truncate or pad hypotheses to the same length of golden references. We also explore n-gram matching in preliminary experiments and observe analogical results with different n. For simplicity, we use the above unigram matching to calculate the translation precision (similarly for the error rate) in all experiments.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3285–3296
November 7–11, 2021. ©2021 Association for Computational Linguistics

of the inference scene during training. Scheduled sampling (Bengio et al., 2015) is a representative method, which samples tokens between golden references and model predictions with a scheduled probability. Zhang et al. (2019) further refine the sampling candidates by beam search. Mihaylova and Martins (2019) and Duckworth et al. (2019) extend scheduled sampling to the Transformer with a novel two-pass decoder architecture. Liu et al. (2021) develop a more fine-grained sampling strategy according to the model confidence.

Although these sampling-based approaches have been shown effective and training efficient, there still exists an essential issue in their sampling strategies. In the real inference scene, the nature of sequential predictions quickly accumulates errors along with decoding steps, which yields higher error rates for larger decoding steps (Zhou et al., 2019; Zhang et al., 2020a) (Figure 1). However, most sampling-based approaches are merely based on training steps and equally treat all decoding steps[2]. Namely, they simulate an inference scene with uniform error rates along with decoding steps, which is inconsistent with the real inference scene.

To alleviate this inconsistent issue, we propose scheduled sampling methods based on decoding steps, which increases the selection chance of predicted tokens with the growth of decoding steps. In this way, we can more realistically simulate the inference scene during training, thus better bridging the gap between training and inference. Furthermore, we investigate scheduled sampling based on both training steps and decoding steps, which yields further improvements. It indicates that our proposals are complementary with existing studies. Additionally, we provide in-depth analyses on the necessity of our proposals from the perspective of translation error rates and accumulated errors. Experimentally, our approaches significantly outperform the Transformer baseline by 1.08, 1.08, and 1.27 BLEU points on WMT 2014 English-German, WMT 2014 English-French, and WMT 2019 Chinese-English, respectively. When comparing with the stronger vanilla scheduled sampling method, our approaches bring further improvements by 0.58, 0.62, and 0.55 BLEU points on these WMT tasks, respectively. Moreover, our approaches generalize well to the text summarization task and achieve consistently better performance on two popular benchmarks, *i.e.,* CNN/DailyMail (See et al., 2017) and Gigaword (Rush et al., 2015).

The main contributions of this paper can be summarized as follows[3]:

- To the best of our knowledge, we are the first that propose scheduled sampling methods based on decoding steps from the perspective of simulating the distribution of real translation errors, and provide in-depth analyses on the necessity of our proposals.
- We investigate scheduled sampling based on both training steps and decoding steps, which yields further improvements, suggesting that our proposals complement existing studies.
- Experiments on three large-scale WMT tasks and two popular text summarization tasks confirm the effectiveness and generalizability of our approaches.
- Analyses indicate our approaches can better simulate the inference scene during training and significantly outperform existing studies.

2 Background

2.1 Neural Machine Translation

Given a pair of source language $\mathbf{X} = \{x_1, x_2, \cdots, x_m\}$ and target language $\mathbf{Y} = \{y_1, y_2, \cdots, y_n\}$, neural machine translation aims to model the following translation probability:

$$P(\mathbf{Y}|\mathbf{X}) = \sum_{t=1}^{n} \log P(y_t|\mathbf{y}_{<t}, \mathbf{X}, \theta) \qquad (1)$$

where t is the index of target tokens, $\mathbf{y}_{<t}$ is the partial translation before y_t, and θ is model parameter. In the training stage, $\mathbf{y}_{<t}$ are ground-truth tokens, and this procedure is also known as teacher forcing. The translation model is generally trained with maximum likelihood estimation (MLE).

2.2 Scheduled Sampling for the Transformer

Scheduled sampling is initially designed for Recurrent Neural Networks (Bengio et al., 2015), and further modifications are needed when applied to the Transformer (Mihaylova and Martins, 2019; Duckworth et al., 2019). As shown in Figure 2, we follow the two-pass decoder architecture for the training of Transformers. In the first pass, the

[2] For clarity in this paper, ‘training steps’ refer to the number of parameter updates and ‘decoding steps’ refer to the index of decoded tokens on the decoder side.

[3] Codes are available at https://github.com/Adaxry/ss_on_decoding_steps.

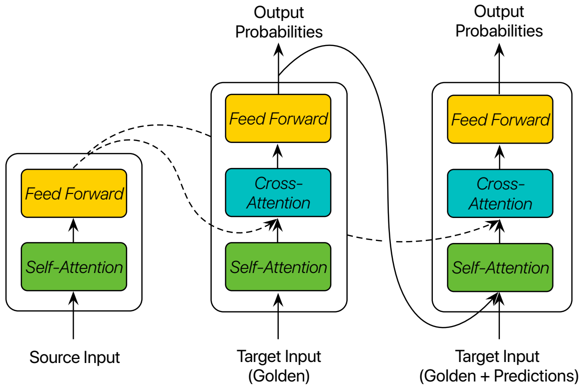

Figure 2: Scheduled sampling for the transformer with a two-pass decoder at training.

model conducts the same as a standard NMT model. Its predictions are used to simulate the inference scene[4]. In the second pass, the decoder's inputs $\widetilde{\mathbf{y}}_{<t}$ are sampled from predictions of the first pass and ground-truth tokens with a certain probability. Finally, predictions of the second pass are used to calculate the cross-entropy loss, and Equation (1) is modified as follow:

$$P(\mathbf{Y}|\mathbf{X}) = \sum_{t=1}^{n} logP(y_t|\widetilde{\mathbf{y}}_{<t}, \mathbf{X}, \theta) \qquad (2)$$

Note that the two decoders are identical and share the same parameters during training. At inference, only the first decoder is used, that is just the standard Transformer. How to schedule the above probability of sampling tokens for training is the key point, which is we aim to improve in this paper.

2.3 Decay Strategies Based on Training Steps

Existing schedule strategies are based on training steps (Bengio et al., 2015; Zhang et al., 2019). At the i-th training step, the probability of sampling golden tokens $f(i)$ is calculated as follow:

- Linear Decay: $f(i) = \max(\epsilon, ki + b)$, where ϵ is the minimum value, and $k < 0$ and b is respectively the slope and offset of the decay.
- Exponential Decay: $f(i) = k^i$, where $k < 1$ is the radix to adjust the decay.
- Sigmoid Decay[5]: $f(i) = \frac{k}{k+e^{\frac{i}{k}}}$, where e is the mathematical constant, and $k \geq 1$ is a hyperparameter to adjust the decay.

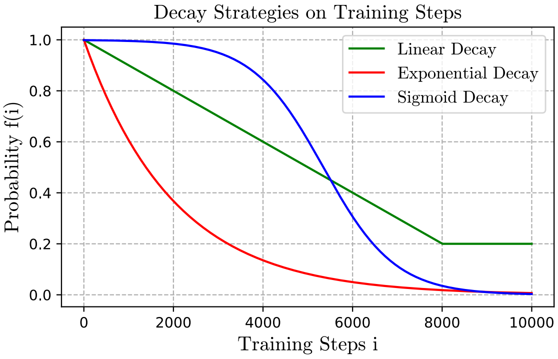

Figure 3: Examples of different decay strategies $f(i)$.

We draw some examples for different decay strategies based on training steps in Figure 3.

3 Approaches

3.1 Definitions and the Overview

At the training stage, in the input of the second-pass decoder, each token is sampled either from the golden token or the predicted token by the first-pass decoder. For clarity, we only define the probability of sampling golden tokens, *e.g.*, $f(i)$, and use $1 - f(i)$ to represent the probability of sampling predicted tokens. Specifically, we define the probability of sampling golden tokens as $f(i)$ when sampling based on the training step i, as $g(t)$ when sampling based on the decoding step t, and as $h(i, t)$ when sampling based on both training steps and decoding steps. In this paper, when we mention a scheduled strategy, it is about the probability of sampling golden tokens at the model training stage. In this section, we firstly point out the drawback of merely sampling based on training steps. Secondly, we describe how to appropriately sample based on decoding steps. Finally, we explore whether sampling based on both training steps and decoding steps can complement each other.

3.2 Sampling Based on Training Steps

As the number of the training step i increases, the model should be exposed to its own predictions more frequently. Thus a decay strategy for sampling golden tokens $f(i)$ (in Section 2.3) is generally used in existing studies (Bengio et al., 2015; Zhang et al., 2019). At a specific training step i, given a target sentence, $f(i)$ is only related to i and equally conducts the same sampling probability for all decoding steps. Therefore, $f(i)$ simulates an inference scene with uniform error rates and still remains a gap with the real inference scene.

[4] Following Goyal et al. (2017), model predictions are the weighted sum of target embeddings over output probabilities. As model predictions cause a mismatch with golden tokens, they can simulate translation errors of the inference scene.

[5] For simplicity, we abbreviate the 'Inverse Sigmoid decay' (Bengio et al., 2015) to 'Sigmoid decay.'

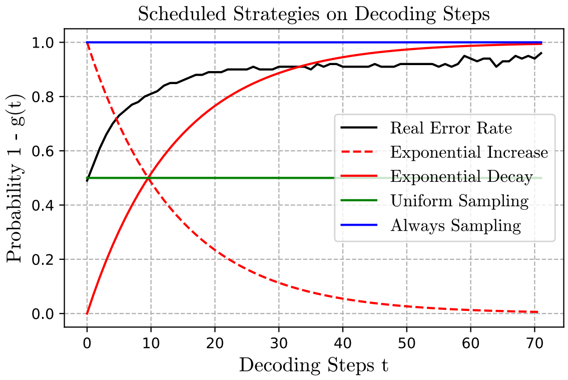

Figure 4: Examples of different strategies for $1 - g(t)$ based on the decoding step t. The black line refers to the real error rate calculated by unigram matching.

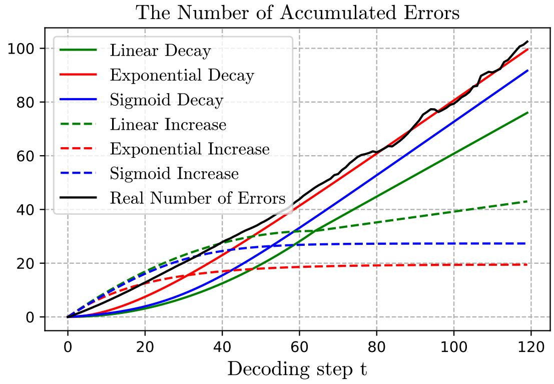

Figure 5: Simulated accumulated errors for different strategies. Solid lines refer to decay strategies, dashed lines refer to increase strategies, and the black line represents the real number of accumulated errors calculated by unigram matching.

3.3 Sampling Based on Decoding Steps

We take a further step to bridge the above gap $f(i)$ left. Specifically, we propose sampling based on decoding steps and schedule the sampling probability $g(t)$ under the guidance of real translation errors. As mentioned earlier (Figure 1), translation error rates are growing rapidly along with decoding steps in the real inference stage. To more realistically simulate such error distributions of the real inference scene during training, we expose more model predictions for larger decoding steps and more golden tokens for smaller decoding steps. Thus it is intuitive to apply a decay strategy for sampling golden tokens based on the number of decoding steps t. Specifically, we directly inherit above decay strategies (Section 2.3) for training steps $f(i)$ to $g(t)$ with a different set of hyperparameters (listed in Table 2).

To rigorously validate the necessity and effectiveness of our proposals, we further conduct the following method variants for comparisons:

- Always Sampling: This model always samples from its own predictions.
- Uniform Sampling: This model randomly samples golden tokens with a uniform probability (0.5 in our experiments).
- Increase Strategies: These models reverse decay strategies to increase strategies, *i.e.,* $g(t) \to 1 - g(t)$.

We draw some representative strategies[6] in Figure 4. Both 'Always Sampling' (blue line) and 'Uniform Sampling' (green line) parallel to the x-axis, namely irrelevant with t. They serve as baseline models to verify whether a scheduled strategy is necessary on the dimension of t. The exponential decay (solid red line) shows a similar trend with the real error rate (black line): the larger decoding steps and the higher error rates. On the other hand, the exponential increase (dashed red line) is entirely contrary to the real error rate. However, we cannot take it for granted that the exponential increase is inappropriate, as it can still simulate the error accumulation phenomenon[7]. Therefore, merely comparing error rates is not enough. We need to step deeper into the dimension of error accumulations for further comparisons.

Error Accumulations. At the decoding step t, the number of accumulated errors $accum(t)$ is the definite integral of the probability of sampling model predictions $1 - g(t)$:

$$accum(t) = \int_0^t (1 - g(x))\, dx \qquad (3)$$

As shown in Figure 5, $accum(t)$ is a monotonically increasing function, which can simulate the error accumulation phenomenon no matter which kind of scheduled strategy $g(t)$ during training. Nevertheless, we observe that different strategies show different speeds and distributions for simulating error accumulations. For instance, decay strategies (solid lines) show a slower speed at the beginning of decoding steps and then rapidly accumulate errors with the growth of decoding steps,

[6] For brevity, we omit linear and sigmoid strategies, which show analogical trends with the exponential strategy.

[7] We will elaborately analyze effects of different schedule strategies in Section 5.1.

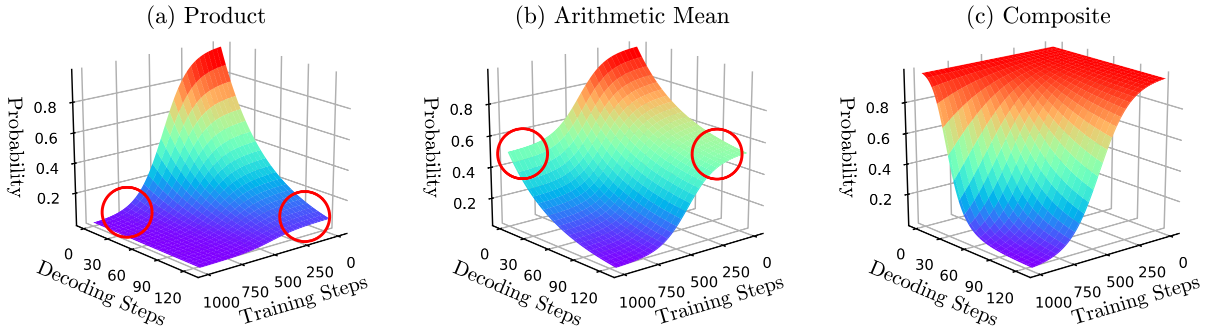

Figure 6: Examples for different $h(i,t)$. The wavelengths of colors represent the probability of sampling golden tokens. Namely, the closer the color to the red, the greater the probability. Red circles are for the sake of highlights.

which is analogous with the real inference scene (black line). However, increase strategies (dashed lines) are just on the contrary. They simulate a distribution with lots of errors at the beginning and an almost fixed number of errors in following decoding steps. Moreover, although different decay strategies show similar trends for simulating error accumulations in the training stage, the degrees of their approximations with real error numbers are still different. We will further validate whether the proximity is closely related to the final performance in Section 5.1.

3.4 Sampling Based on Both Training Steps and Decoding Steps

When comparing above two types of approaches, *i.e.,* $f(i)$ and $g(t)$, our approach $g(t)$ focus on simulating the distribution of real translation errors, and the vanilla $f(i)$ emphasizes the competence of the current model. Thus it is intuitive to verify whether $f(i)$ and $g(t)$ complement each other. How to combine them is the critical point. At the training step i and decoding step t, we define the probability of sampling golden tokens $h(i,t)$ by the following joint distribution function:

- Product: $h(i,t) = f(i) \cdot g(t)$
- Arithmetic Mean: $h(i,t) = \frac{f(i)+g(t)}{2}$
- Composite[8]: $h(i,t) = g(t \cdot (1 - f(i)))$

One simple solution ('Product') is to directly multiply $f(i)$ and $g(t)$. However, both $f(i)$ and $g(t)$ are less than or equal to 1, thus their product quickly shrinks to a tiny value close to 0. Consequently, it exposes too few golden tokens and too many predicted tokens to the model (Figure 6 (a)), which increases the difficulty for training. 'Arithmetic Mean' is another possible solution with a relatively gentle combination. However, it still inappropriately exposes too few golden tokens to the model at the beginning of training steps (Figure 6 (b)). Finally, we propose to apply function compositions on both $f(i)$ and $g(t)$ (*i.e.,* 'Composite'). It guarantees enough golden tokens at the beginning of training steps, and gradually exposes more predicted tokens to the model with the increase of both i and t (Figure 6 (c)). We will analyze effects of different $h(i,t)$ in Section 5.2.

Dataset	**Size (M)**	**Valid / Test set**
WMT14 EN-DE	4.5	newstest 2013 / 2014
WMT14 EN-FR	36	newstest 2013 / 2014
WMT19 ZH-EN	20	newstest 2018 / 2019
CNN/DailyMail	0.3	standard data
Gigaword	3.8	standard data

Table 1: Dataset statistics in our experiments.

4 Experiments

We validate our proposals on two important sequence generation tasks, *i.e.,* machine translation and text summarization.

4.1 Tasks and Datasets

Machine Translation. We use the standard WMT 2014 English-German (EN-DE), WMT 2014 English-French (EN-FR), and WMT 2019 Chinese-English (ZH-EN) datasets. We respectively build a shared source-target vocabulary for EN-DE and EN-FR, and unshared vocabularies for ZH-EN. We apply byte-pair encoding (Sennrich et al., 2016) with 32k merge operations for all datasets.

Text Summarization. We use two popular summarization datasets: (a) the non-anonymized version of the CNN/DailyMail dataset (See et al.,

[8] We also tried $f(i \cdot (1 - g(t))$ in preliminary experiments, but it slightly underperformed the above $g(t \cdot (1 - f(i)))$.

Variable	Task	Maximum Value	Hyperparameter k		
			Linear	Exponential	Sigmoid
Training Steps $f(i)$ (vanilla)	Translation	300,000	-1/150,000	0.99999	20,000
	Summarization	100,000	-1/50,000	0.9999	15,000
Decoding Steps $g(t)$ (ours)	Translation	128	-1/64	0.99	20
	Summarization	512	-1/256	0.999	50

Table 2: Hyperparameters k for different schedule strategies in our experiments.

2017), and (b) Gigaword corpus (Rush et al., 2015). We list dataset statistics for all datasets in Table 1.

4.2 Implementation Details

Training Setup. For the translation task, we follow the default setup of the Transformer$_{base}$ and Transformer$_{big}$ models (Vaswani et al., 2017), and provide detailed setups in Appendix A (Table 7). All Transformer models are first trained by teacher forcing with 100k steps, and then trained with different training objects or scheduled sampling approaches for 300k steps. All experiments are conducted on 8 NVIDIA V100 GPUs, where each is allocated with a batch size of approximately 4096 tokens. For the text summarization task, we base on the ProphetNet (Qi et al., 2020) and follow its training setups. We set hyperparameters involved in various scheduled sampling strategies (*i.e.,* $f(i)$ and $g(t)$) according to the performance on validation sets of each tasks and list k in Table 2. For the linear decay, we set ϵ and b to 0.2 and 1, respectively. Please note that scheduled sampling is only used during training instead of the inference stage.

Evaluation. For the machine translation task, we set the beam size to 4 and the length penalty to 0.6 during inference. We use *multibleu.perl* to calculate cased sensitive BLEU scores for EN-DE and EN-FR, and use *mteval-v13a.pl* script to calculate cased sensitive BLEU scores for ZH-EN. We use the paired bootstrap resampling methods (Koehn, 2004) to compute the statistical significance of translation results. We report mean and standard-error variation of BLEU scores over three runs. For the text summarization task, we respectively set the beam size to 4/5 and length penalty to 1.0/1.2 for Gigaword and CNN/DailyMail dataset following previous studies (Song et al., 2019; Qi et al., 2020). We report the F1 scores of ROUGE-1, ROUGE-2, and ROUGE-L for both datasets.

4.3 Systems

Mixer. A sequence-level training algorithm for text generations by combining both REINFORCE and cross-entropy (Ranzato et al., 2016).

Minimal Risk Training. Minimal Risk Training (MRT) (Shen et al., 2016) introduces evaluation metrics (*e.g.,* BLEU) as loss functions and aims to minimize expected loss on the training data.

Target denoising. Meng et al. (2020a) and Meng et al. (2020b) propose to add noisy perturbations into decoder inputs for a more robust translation model against prediction errors.

TeaForN. Teacher forcing with n-grams (Goodman et al., 2020) enables the standard teacher forcing with a broader view by a n-grams optimization.

Sampling based on training steps. For distinction, we name vanilla scheduled sampling as *Sampling based on training steps*. We defaultly adopt the sigmoid decay following Zhang et al. (2019).

Sampling with sentence oracles. Zhang et al. (2019) refine the sampling candidates of scheduled sampling with sentence oracles, *i.e.,* predictions from beam search. Note that its sampling strategy is based on training steps with the sigmoid decay.

Sampling based on decoding steps. Sampling based on decoding steps with exponential decay.

Sampling based on training and decoding steps. Our sampling based on both training steps and decoding steps with the 'Composite' method.

4.4 Main Results

Machine Translation. We list translation qualities on three WMT tasks in Table 3. The sentence-level training based approaches (*e.g.,* Mixer) bring limited improvements due to the high variance of policy gradients and the credit assignment problem. On the contrary, sampling-based approaches

Model	BLEU		
	EN-DE	ZH-EN	EN-FR
Transformer$_{base}$ (Vaswani et al., 2017)	27.30	–	38.10
Transformer$_{base}$ (Vaswani et al., 2017) †	27.90 ± .02	24.97 ± .01	39.90 ± .02
+ Mixer (Ranzato et al., 2016) †	28.54 ± .02	25.28 ± .03	40.17 ± .01
+ Minimal Risk Training (Shen et al., 2016) †	28.55 ± .01	25.33 ± .05	40.10 ± .02
+ TeaForN (Goodman et al., 2020)	27.90 ± .03	–	40.84 ± .07
+ TeaForN (Goodman et al., 2020) †	28.60 ± .02	25.45 ± .02	40.34 ± .01
+ Target denoising (Meng et al., 2020a) †	28.45 ± .02	25.78 ± .03	40.79 ± .02
+ Sampling based on training steps (Bengio et al., 2015) †	28.40 ± .01	25.43 ± .04	40.62 ± .03
+ Sampling with sentence oracles (Zhang et al., 2019)	28.65	–	–
+ Sampling with sentence oracles (Zhang et al., 2019) †	28.65 ± .03	25.50 ± .04	40.65 ± .02
+ Sampling based on decoding steps (ours) †	28.83 ± .05$\ast\ast$	25.96 ± .07$\ast$	41.05 ± .04$\ast\ast$
+ Sampling based on training and decoding steps (ours) †	**28.98** ± **.03**$\ast\ast$	**26.05** ± **.04**$\ast\ast$	**41.17** ± **.03**$\ast\ast$
Transformer$_{big}$ (Vaswani et al., 2017)	28.40	–	41.80
Transformer$_{big}$ (Vaswani et al., 2017) †	28.90 ± .03	25.22 ± .04	41.89 ± .03
+ Mixer (Ranzato et al., 2016) †	29.27 ± .01	25.58 ± .02	42.37 ± .01
+ Minimal Risk Training (Shen et al., 2016) †	29.35 ± .02	25.65 ± .01	42.46 ± .01
+ TeaForN (Goodman et al., 2020)	29.30 ± .01	–	42.73 ± .01
+ TeaForN (Goodman et al., 2020) †	29.32 ± .01	25.48 ± .02	42.62 ± .01
+ Error correction (Song et al., 2020)	29.20	–	–
+ Target denoising (Meng et al., 2020a) †	29.68 ± .02	25.56 ± .03	42.62 ± .03
+ Sampling based on training steps (Bengio et al., 2015) †	29.62 ± .01	25.60 ± .02	42.55 ± .01
+ Sampling with sentence oracles (Zhang et al., 2019) †	29.57 ± .03	25.78 ± .02	42.65 ± .01
+ Sampling based on decoding steps (ours) †	29.85 ± .02$\ast$	**26.23** ± **.01**$\ast\ast$	42.87 ± .01$\ast\ast$
+ Sampling based on training and decoding steps (ours) †	**30.16** ± **.01**$\ast\ast$	26.10 ± .01$\ast\ast$	**43.13** ± **.01**$\ast\ast$

Table 3: Translation performance of each dataset. '†' is our implementations under unified settings. The original TeaForN (Goodman et al., 2020) reports SacreBLEU scores. For fair comparison, we re-implement it and report BLEU scores. '$\ast$ / $\ast\ast$': significantly better than vanilla 'Sampling based on training steps' ($p < 0.05$ / $p < 0.01$).

show better translation qualities while preserving efficient training. TeaForN also yields competitive translation qualities due to its long-term optimization. Among all existing methods, our 'Sampling based on decoding steps' shows consistent improvements on various datasets. Moreover, 'Sampling based on training and decoding steps' combines the advantages of both existing methods and our proposals, and achieves better performance. Specifically for the Transformers$_{base}$, it brings significant improvements by 1.08, 1.08, and 1.27 BLEU points on EN-DE, ZH-EN, and EN-FR, respectively. Moreover, it significantly outperforms vanilla scheduled sampling by 0.58, 0.62, and 0.55 BLEU points on these tasks, respectively. For the more powerful Transformers$_{big}$, we observe similar experimental conclusions as above. Specifically, 'Sampling based on training and decoding steps' significantly outperforms the Transformers$_{big}$ by 1.26, 0.88 and 1.24 BLEU points on EN-DE, ZH-EN, and EN-FR, respectively.

Text Summarization. In Table 4, we list F1 scores of ROUGE-1 / ROUGE-2 / ROUGE-L on test sets of both text summarization datasets. We take the powerful ProphetNet$_{large}$ as our primary baseline[9] and apply different sampling-based approaches. For vanilla scheduled sampling (second last row of Table 4), we observe marginal improvements on Gigaword and even degenerations on CNN/DailyMail. We speculate that poor performance comes from their uniform sampling rate along with decoding steps, which violates the distribution of the real inference scene. Namely, the model is overexposed to golden tokens and underexposed to predicted tokens at larger decoding steps. Especially for CNN/DailyMail, its averaged target sequence length exceeds 64, and more than 90% of sentences are longer than 50, which exacerbates the above issue in existing sampling-based approaches. We further analyze the effects of different sampling approaches on different sequence lengths in Section 5.3. Nevertheless, our approaches are not affected by the above issue and show consistent improvements in all criteria of both

[9] The codes of previous SOTA (Aghajanyan et al., 2020) are not publicly available. Thus we base our approach on the second-best ProphetNet (Qi et al., 2020).

Model	RG-1 / RG-2 / RG-L	
	CNN/DailyMail	**Gigaword**
RoBERTSHARE$_{large}$ (Rothe et al., 2020)	40.31 / 18.91 / 37.62	38.62 / 19.78 / 35.94
MASS (Song et al., 2019)	42.12 / 19.50 / 39.01	38.73 / 19.71 / 35.96
UniLM (Dong et al., 2019)	43.33 / 20.21 / 40.51	38.45 / 19.45 / 35.75
PEGASUS$_{large}$ (Zhang et al., 2020b)	44.17 / 21.47 / 41.11	39.12 / 19.86 / 36.24
PEGASUS$_{large}$ + TeaForN (Goodman et al., 2020)	44.20 / 21.70 / 41.32	39.16 / 20.16 / 36.54
ERNIE-GEN$_{large}$ (Xiao et al., 2020)	44.31 / 21.35 / 41.60	39.46 / 20.34 / 36.74
BART+R3F (Aghajanyan et al., 2020) (previous SOTA)	44.38 / **21.53** / 41.17	**40.45** / 20.69 / 36.56
ProphetNet$_{large}$ (Qi et al., 2020) (primary baseline) †	44.08 / 21.14 / 41/19	39.59 / 20.33 / 36.62
+ Target denoising (Meng et al., 2020a) †	43.98 / 21.09 / 41.08	39.68 / 20.18 / 36.78
+ Sampling based on training steps (Bengio et al., 2015) †	43.47 / 20.76 / 40.59	39.77 / 20.44 / 36.79
+ Sampling based on decoding steps (ours) †	44.20 / 21.33 / 41.41	40.11 / 20.39 / 37.15
+ Sampling based on training and decoding steps (ours) †	**44.40** / 21.44 / **41.61**	40.01 / **20.70** / **37.24**

Table 4: F1 scores of ROUGE-1 / ROUGE-2 / ROUGE-L on test sets of both datasets. 'RG' is short for 'ROUGE'. '†' is our implementations under a unified framework. Our approaches achieve consistently better performance.

datasets. Specifically, our approaches achieve consistently better performance than the baseline system on both datasets, and significantly improve the previous SOTA on ROUGE-L score of Gigaword to 37.24 (+0.5). In conclusion, the strong performance on the text summarization task indicates that our approaches have a good generalization ability across different tasks.

5 Analysis and Discussion

In this section, we provide in-depth analyses on the necessity of our proposals and conducts experiments on the validation set of WMT14 EN-DE with the Transformer$_{base}$ model.

5.1 Effects of Scheduled Strategies

In this section, we focus on the effects of different scheduled strategies based on the decoding step t, and aim to answer the following two questions:

(a) Is a Scheduled Strategy is Necessary? We take the Transformer without sampling as the baseline, then respectively apply 'Always Sampling', 'Uniform Sampling', and our 'Exponential Decay'. Results are listed in the part (a) of Table 5. We observe a noticeable drop when conducting 'Always Sampling', as the model is entirely exposed to its predictions and fails to converge fully. As to 'Uniform Sampling', it is essentially a simulation of the vanilla 'Sampling based on Training Steps'. Although 'Uniform Sampling' conducts an inappropriate sampling strategy, it still can simulate the data distribution of the inference scene to some extent and bring BLEU improvements modestly. In contrast, our 'Exponential Decay' conducts a sampling strategy following real translation errors. It significantly outperforms both 'No Sampling' and 'Uniform Sampling' by 1.64 and 0.68 BLEU scores. In short, we conclude that an appropriate scheduled strategy based on decoding steps is necessary.

ID	Scheduled Strategies	BLEU	Δ
(a)	No Sampling Baseline	27.10	ref.
	+ Always Sampling	26.52	-0.58
	+ Uniform Sampling	27.48	+0.38
	+ Exponential Decay	**28.16**	**+1.06**
(b)	Uniform Sampling Baseline	27.48	ref.
	+ Linear Increase	27.33	-0.15
	+ Exponential Increase	27.25	-0.23
	+ Sigmoid Increase	27.17	-0.31
(c)	+ Linear Decay	27.98	+0.50
	+ Exponential Decay	**28.16**	**+0.68**
	+ Sigmoid Decay	28.05	+0.57

Table 5: BLEU scores (%) on the validation set of WMT14 EN-DE for different schedule strategies $g(t)$. 'ref.' indicates the reference baseline.

(b) Why Decay Instead of Increase Strategies? Considering errors naturally accumulate along with decoding steps, both decay strategies and increase strategies can simulate error accumulations. We respectively apply both kinds of sampling strategies upon the 'Uniform Sampling' baseline model, and list results in the part(b) and part(c) of Table 5. Surprisingly, all increase strategies consistently decrease performance by considerable margins. We conjecture that these increase strategies simulate an unreasonably high error rate at the beginning of decoding steps. Too many translation errors are

Combination Methods	BLEU	Δ
Sampling based on decoding steps	28.16	reference
+ Product	27.65	-0.51
+ Arithmetic Mean	28.06	-0.10
+ Composite	**28.37**	**+0.21**

Table 6: BLEU scores (%) on the WMT14 EN-DE validation set with different combination methods. 'Product' and 'Arithmetic Mean' lead to performance degradation in different degrees. While 'Composite' can further improve the strong baseline to a certain extent.

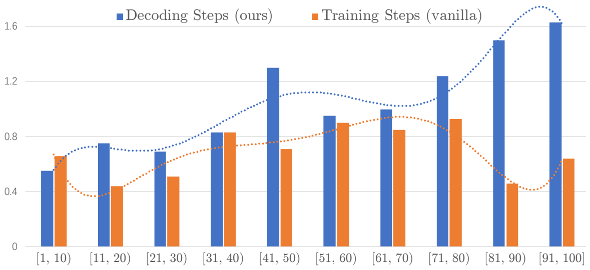

Figure 7: Absolute BLEU scores (%) gains over the Transformer baseline on different sequence lengths, where dashed lines are polynomial trendlines.

propagated to subsequent decoding steps, which hinders the final performance. On the contrary, all decay strategies bring consistent improvements with different degrees. Moreover, we observe that the more a decay strategy approximates real error numbers (Figure 5), the more performance improvements. In summary, we need to apply decay strategies instead of increase strategies based on decoding steps in the perspective of simulating real error accumulations.

5.2 Effects of Different $h(i, t)$ Strategies

We take our strong 'Sampling based on decoding steps' as the baseline and then apply different combination methods $h(i, t)$. As shown in Table 6, the performance drop of 'Product' and 'Arithmetic Mean' confirms our speculation in Section 3.4. Namely, the model is overexposed to its predictions at the beginning of training steps and decoding steps, thus fails to converge well. In contrast, 'Composite' brings certain improvements over the strong baseline model. Since it stabilizes the model training and successfully combines the advantages of both dimensions of training steps and decoding steps. In summary, a well-designed strategy is necessary when combining both $f(i)$ and $g(t)$, and we provide an effective alternative (*i.e.,* 'Composite').

5.3 Effects on Different Sequence Lengths

According to our early findings, the exposure bias problem gets worse as the sentence length grows. Thus it is intuitive to verify whether our approaches improve translations of long sentences. Since the size of WMT14 EN-De validation set (3k) is too small to cover scenarios with various sentence lengths, we randomly select training data with different sequence lengths. Specifically, we divide WMT14 EN-DE training data into ten bins according to the source side's sentence length. The maximal length is 100, and the interval size is 10. Then we randomly select 1000 sentence pairs from each bin and calculate BLEU scores for different approaches. Specifically, we take the Transformer as the baseline, and draw absolute BLEU gains of scheduled sampling on training steps and decoding steps. As shown in Figure 7, BLEU gains of the vanilla scheduled sampling are relatively uniform over different sentence lengths. In contrast, BLEU gains of our scheduled sampling on decoding steps gradually increase with sentence lengths. Moreover, our approach consistently outperforms the vanilla one at most sentence length intervals. Specifically, we observe more than 1.0 BLEU improvements when sentence lengths in $[80; 100]$.

6 Conclusion

In this paper, we propose scheduled sampling methods based on decoding steps from the perspective of simulating real translation error rates, and provide in-depth analyses on the necessity of our proposals. We also confirm that our proposals are complementary with existing studies (based on training steps). Experiments on three large-scale WMT translation tasks and two text summarization tasks confirm the effectiveness of our approaches. In the future, we will investigate low resource settings which may suffer from a more serious error accumulation problem. In addition, more autoregressive-based tasks would be explored as future work.

Acknowledgements

The research work described in this paper has been supported by the National Key R&D Program of China (2020AAA0108001) and the National Nature Science Foundation of China (No. 61976015, 61976016, 61876198 and 61370130). Jinan Xu is the corresponding author of the paper.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2020. Better fine-tuning by reducing representational collapse. *arXiv preprint arXiv:2008.03156*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1171–1179.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13042–13054.

Daniel Duckworth, Arvind Neelakantan, Ben Goodrich, Lukasz Kaiser, and Samy Bengio. 2019. Parallel scheduled sampling. *arXiv preprint arXiv:1906.04331*.

Sebastian Goodman, Nan Ding, and Radu Soricut. 2020. TeaForN: Teacher-forcing with n-grams. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8704–8717, Online. Association for Computational Linguistics.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017. Differentiable scheduled sampling for credit assignment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 366–371, Vancouver, Canada. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. 2018. Action-dependent control variates for policy optimization via stein identity. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Yijin Liu, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou. 2021. Confidence-aware scheduled sampling for neural machine translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2327–2337, Online. Association for Computational Linguistics.

Fandong Meng, Jianhao Yan, Yijin Liu, Yuan Gao, Xianfeng Zeng, Qinsong Zeng, Peng Li, Ming Chen, Jie Zhou, Sifan Liu, and Hao Zhou. 2020a. WeChat neural machine translation systems for WMT20. In *Proceedings of the Fifth Conference on Machine Translation*, pages 239–247, Online. Association for Computational Linguistics.

Fandong Meng, Jianhao Yan, Yijin Liu, Yuan Gao, Xianfeng Zeng, Qinsong Zeng, Peng Li, Ming Chen, Jie Zhou, Sifan Liu, and Hao Zhou. 2020b. WeChat neural machine translation systems for WMT20. In *Proceedings of the Fifth Conference on Machine Translation*, pages 239–247, Online. Association for Computational Linguistics.

Tsvetomila Mihaylova and André F. T. Martins. 2019. Scheduled sampling for transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 351–356, Florence, Italy. Association for Computational Linguistics.

Richard Yuanzhe Pang and He He. 2021. Text generation by learning from demonstrations. In *International Conference on Learning Representations*.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1179–1195. IEEE Computer Society.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015*

Conference on Empirical Methods in Natural Language Processing, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Ruslan Salakhutdinov. 2014. Deep learning. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, page 1973. ACM.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 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)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

Kaitao Song, Xu Tan, and Jianfeng Lu. 2020. Neural machine translation with error correction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3891–3897. ijcai.org.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5926–5936. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Richard Stuart Sutton. 1984. Temporal credit assignment in reinforcement learning.

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 *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 899–909, Melbourne, Australia. Association for Computational Linguistics.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Dongling Xiao, Han Zhang, Yu-Kun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE-GEN: an enhanced multi-flow pre-training and fine-tuning framework for natural language generation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3997–4003. ijcai.org.

Jiajun Zhang, Long Zhou, Yang Zhao, and Chengqing Zong. 2020a. Synchronous bidirectional inference for neural sequence generation. *Artificial Intelligence*, 281:103234.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2020b. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *ICML*.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343, Florence, Italy. Association for Computational Linguistics.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics*, 7:91–105.

Parameter	**Transformer**$_{base}$	**Transformer**$_{big}$
batch size	4096	4096
number of GPUs	8	8
hidden size	512	1024
filter size	2048	4096
number of heads	8	16
number of encoders	6	6
number of decoders	6	6
dropout	0.1	0.3
label smoothing	0.1	0.1
pre-training steps	100,000	200,000
fine-tuning steps	300,000	300,000
warmup steps	4,000	8,000
learning rate	1.0	1.0
optimizer	Adam	Adam
Adam beta1	0.9	0.9
Adam beta2	0.98	0.98
layer normalization	post-norm	post-norm
position encoding	absolute	absolute
share embeddings	True	True
share softmax weights	True	True

Table 7: Detailed parameters for Transformer$_{base}$ and Transformer$_{big}$ on all WMT datasets. Note that the 'share embedding' is set to 'False' on WMT19 ZH-EN.

A Training Details

We list detailed parameters for training Transformer models in Table 7.

B Real Error Rates as Sampling Priors

In the above contents of this paper, we aim to better simulate the inference scene under the guidance of real error rates. We can not help wondering the effect of directly taking the above error rates as sampling priors. Disappointingly, it fails to outperform our exponential decay strategy within a gap of 0.1 BLEU scores. We conjecture the metric we used to measure translation errors at each decoding step may not be good enough. Considering the optimal metric is currently unknown and unavailable, our unigram matching can yet be regarded as a simple and effective alternative. It succeeds in reflecting the trend of real error rates and brings significant improvements by simulating the error distribution estimated by unigram matching. We believe a better metric would bring further improvements and leave this exploration for future work.