Improving Text Generation with Dynamic Masking and Recovering

Zhidong Liu[1] , **Junhui Li**[1*] , **Muhua Zhu**[2]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]Tencent News, Beijing, China
{zdliu0122, zhumuhua}@gmail.com, lijunhui@suda.edu.cn

Abstract

Due to different types of inputs, diverse text generation tasks may adopt different encoder-decoder frameworks. Thus most existing approaches that aim to improve the robustness of certain generation tasks are input-relevant, and may not work well for other generation tasks. Alternatively, in this paper we present a universal approach to enhance the language representation for text generation on the base of generic encoder-decoder frameworks. This is done from two levels. First, we introduce randomness by randomly masking some percentage of tokens on the decoder side when training the models. In this way, instead of using ground truth history context, we use its corrupted version to predict the next token. Then we propose an auxiliary task to properly recover those masked tokens. Experimental results on several text generation tasks including machine translation (MT), AMR-to-text generation, and image captioning show that the proposed approach can significantly improve over competitive baselines without using any task-specific techniques. This suggests the effectiveness and generality of our proposed approach.

1 Introduction

Recent years have seen growing interest in various text generation tasks which aim to generate mostly-grammatical natural language text from diverse input forms. Representative text generation tasks include machine translation [Vaswani *et al.*, 2017] which generates text from an input word sequence, AMR-to-text generation [Zhu *et al.*, 2019] which generates text from a semantic graph, concept-to-text [Mei *et al.*, 2016] which generates text from structured data records, and image captioning [Ren *et al.*, 2016] which generates text description from an image.

Most existing approaches to text generation adopt the widely used encoder-decoder framework: the encoder takes structured or unstructured data as input and returns a sequence of distributed representations, from which the decoder generates text as output. Previous studies on a variety of text generation tasks have shown the effectiveness of the framework. However, training a robust text generation model usually requires large-scale training data. To alleviate such requirement, one feasible way is to introduce more uncertainty to overcome overfitting during training [Dhillon *et al.*, 2018], especially when the training data is limited in size. Techniques in this line include drop-out and randomness. Another possible way is to enlarge the training data through techniques like back translation, data augmentation. In this paper we propose a simple yet effective approach that can achieve these two goals simultaneously.

The proposed approach is inspired by the idea of masked language modeling [Devlin *et al.*, 2018]. To train the models of text generation tasks, we introduce randomness by randomly masking some percentage of tokens on the decoder side. In this way, different from a standard decoder which uses ground truth history context, we use its masked (corrupted) version to predict the next token. The advantage of using corrupted history context is two-fold. On the one hand, with dynamic masking strategy which generates a new random masking pattern every time we feed a sequence to the decoder, we enable the decoder to see more training instances during training. On the other hand, the discrepancy of history context between training and inference is a classic issue in text generation [Zhang *et al.*, 2019]. Using non-ground truth words as history context narrows the gap between training and inference. Upon the masked sequence, we further propose an auxiliary task to properly recover those masked tokens, just as masked language model does. Therefore, our approach jointly maximizes both the likelihoods of both sentence generation and prediction of masked tokens.

We verify the effectiveness and generality of our approach on three types of text generation tasks which use various forms of input data including text, graph, and image. For sequence-to-sequence (seq2seq) generation task (specifically, machine translation), our model obtains significant improvement of 1.01 and 0.90 BLEU scores over competitive baseline on IWSLT14 German→English and WMT14 English→German, respectively. For graph-to-text generation task (specifically, AMR-to-text), our approach achieves a significant improvement of 1.39 BLEU score over the state-of-the-art on AMR 2.0. Finally, for image-to-text generation task (specifically, image captioning), our results on the MSCOCO test set show improvement of 1.5, 0.5, and 0.9 in

*Corresponding Author

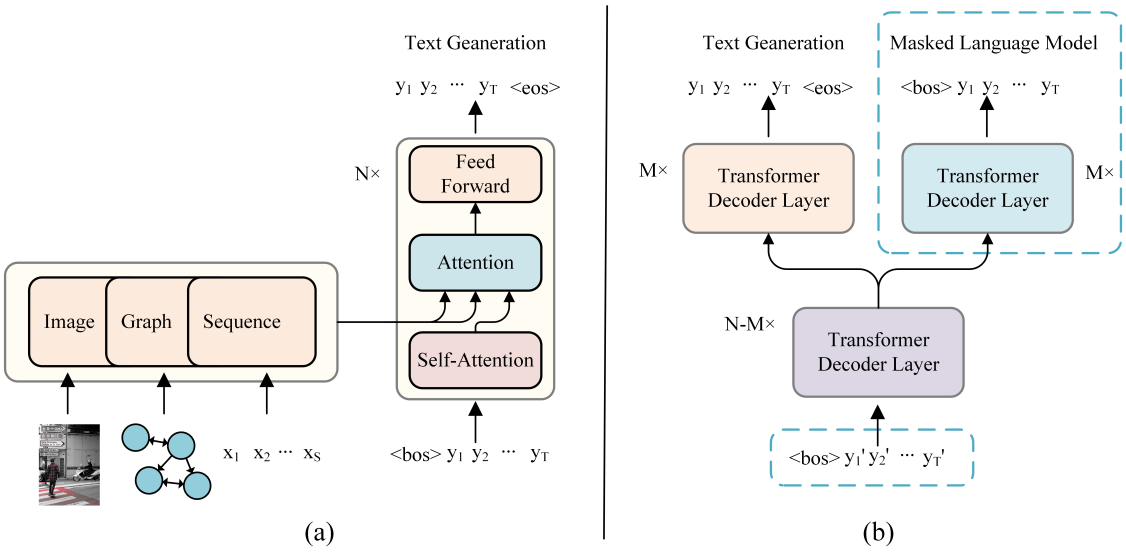

Figure 1: (a) An overview of the encoder-decoder architecture for text generation; (b) Illustration of our proposed approach, where the dotted boxes highlight the difference from the standard decoder for text generation tasks.

CIDEr, SPICE, and BLEU-4 scores, respectively.

Our main contributions can be summarized as follows:

- We add a novel objective to force the models of text generation not only predict the target sentence, but also resume the original tokens, which could enhance the language representation for text generation.
- Our approach is applicable to arbitrary types of input and easily transparent to other encoder-decoder architectures. We do not manipulate the encoder and only mask a number of tokens from target sentence.
- Experimental results show that our method outperforms all baselines on a variety of text generation tasks.

2 Methodology

Since our goal is to improve robustness of encoder-decoder models by focusing on the decoder part, we keep the encoders unchanged.

2.1 Model Architecture

A text generation model basically consists of an encoder and an decoder, as shown in Figure 1(a). Depending on the format of input, the encoders of text generation tasks may be very different. We use X to indicate the input.

We use $Y = (y_1, \cdots, y_T)$ to indicate the output, while T is the length of sequence Y. For simplification, we uniform their decoders as Transformer-based, which consist of a stack of identical decoder layers. Each decoder layer has three sublayers. The first is a masked multi-head self-attention mechanism, the second performs multi-head attention over the output of the encoder stack while the third is a simple, position-wise fully connected feed-forward network. As shown in Figure 1(a), at each decoding time t, the predictions for position y_t can depend on source-side input X and target-side **ground truth history context** $y_{<t}$.

$$P(Y|X) = \prod_{t=1}^{T} P(y_t|y_{<t}, X; \Theta) \tag{1}$$

where Θ is model parameters. The model is trained by maximum likelihood estimate, i.e., minimizing the negative log likelihood loss:

$$L_{\text{TG}}(Y|X) = -\sum_{t=1}^{T} \log P(y_t|y_{<t}, X; \Theta) \tag{2}$$

Figure 1(b) illustrates our proposed decoder, which falls into a multi-task learning framework. Compared to the baseline decoder, our proposed decoder have two differences: 1) rather than using the gold target-side history context to predict Y, we use a corrupted version; 2) we introduce a masked language model on the target side. Next, we present the proposed decoder in details.

2.2 Dynamic Masking

In order to obtain more training samples efficiently and cheaply, we corrupt a given word sentence $Y = \left(y_1, \cdots, y_{T_y}, <eos>\right)$ and obtain a noise version $Y' = \left(y_1', \cdots, y_{T_y}', <eos>\right)$. To this end, we follow BERT [Devlin *et al.*, 2018] and randomly sample 15% of tokens in Y. The selected tokens are (1) 80% of time replaced by a mask token `[MASK]`, or (2) 10% of time replaced by a random token, or (3) 10% of time unchanged. Consequently, we update Eq. 1 and Eq. 2 as:

$$P(Y|X) = \prod_{t=1}^{T} P\left(y_t|y'_{<t}, X; \Theta\right) \tag{3}$$

$$L_{\text{TG}}(Y|X) = -\sum_{t=1}^{T} \log P\left(y_t|y_{<t}^{'}, X; \Theta\right) \quad (4)$$

That is to say, rather than using ground truth history context $y_{<t}$, we now use its corrupted version $y_{<t}^{'}$ to predict y_t at the t-th time step.

Note that BERT performs random masking and replacement once in data pre-processing. To avoid using the same mask for each training instance in every epoch, we generate the masking pattern every time we feed a sequence to the model. For example, if the training process finishes after K epochs, each training sequence would be seen with the K different corrupted versions.

2.3 Masked Language Model

Now the history context may contain noise tokens. Besides the text generation task, which predicts the next word based on history context, we propose another task, masked language model, which predict those noised tokens. Unlike masked language model in BERT which uses bidirectional context, here we use the left context and allow the model to see itself, i.e., the [MASK] token. Figure 2 gives a concrete example where the first target word (i.e., *the*) is not masked, i.e., $y_1^{'} = y_1$ while the second y_2 (i.e., *boy*) is masked, i.e., $y_2^{'} =$ [MASK]. At time steps $t = 1$ and $t = 2$, the text generation task predicts y_1 and y_2, respectively while the masked language model predicts nothing. At time step $t = 3$, the former task predicts y_3 while the latter task tries to predict y_2 itself since y_2 is masked in the history context. Formally, the loss function for the masked language model task is defined as:

$$L_{\text{MLM}}\left(Y|X, Y^{'}\right) = -\sum_{t=1}^{T} \mathbf{1}\left(y_t^{'} \neq y_t\right) \log P\left(y_t|y_{<t}^{'}, y_t^{'}, X; \Theta\right) \quad (5)$$

where $\mathbf{1}\left(y_t^{'} \neq y_t\right)$ returns 1 if $y_t^{'}$ is different from y_t, otherwise 0.

2.4 Jointly Learning

As shown in Figure 1(b), the tasks of text generation and masked language model are jointly learned via a hard parameter sharing multi-task learning framework. Specifically, they share N-M decoder layers while have M task-specific decoder layers each. The joint loss is shown as Eq. 6.

$$L_{Joint}\left(Y|X, Y^{'}\right) = L_{TG}(Y|X) + \alpha L_{MLM}\left(Y|X, Y^{'}\right) \quad (6)$$

where α is a hyperparameter used in controlling the weight of masked language model task.

Also note that we introduce dynamic masking and masked language model task only in the training stage.

3 Experiments

To evaluate the effect and generality of our proposed approach, we conduct experiments on several representative text generation tasks, including machine translation, AMR-to-text generation, and image captioning. Note that those tasks have different types of inputs, thus employ different encoders, i.e., sequential, graph-based, and CNN-based, respectively. For each task, we evaluate the effect of dynamic masking (DM), and its combination with the masked language model (MLM) task. For the decoder layers, we set N to 6 and set M to 1 for all experiments, as shown in Figure 1. The M task-specific decoder layers do not share parameters. That is to say, we share the $N - M$ decoder layers in the bottom while the two prediction tasks have their respective M decoder layers in the upper. For the two tasks, we use the same final linear transformation layers to convert output logits to token probabilities. Moreover, we report results of single models that are trained at most 300K iterations and tuned on the respective development sets.

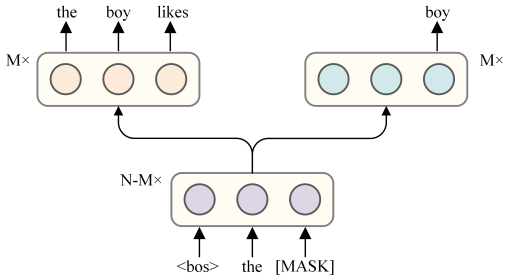

Figure 2: A concrete example of the text generation task and the masked language model task. Here y_1 (i.e., *the*) is not changed while y_2 (i.e., *boy*) is masked.

3.1 Machine Translation

Dataset and evaluation. For machine translation, we evaluate our approach on two widely used benchmarks: WMT14 English→German (WMT14 EN-DE)[1] and IWSLT14 German→English (IWSLT14 DE-EN).[2] WMT14 EN-DE consists of 3.9M training sentence pairs after filtering out long and imbalanced pairs. We use newstest2013 and newstest2014 as the validation and test set respectively. For IWSLT14 DE-EN, we conduct the same data cleanup and train/dev splitting as [Ott *et al.*, 2019], resulting 160K parallel sentence pairs for training and 7,284 sentence pairs for development. We combine tst2010, tst2011, tst2012, dev2010, and dev2012 as our testing data. We tokenize all sentences with Moses scripts.[3] Then we segment words into subwords by using byte-pair encoding (BPE) [Sennrich *et al.*, 2016b]. The sizes of resulting vocabularies shared by the source and target language are 32K and 10K for respectively WMT14 EN-DE and IWSLT14 DE-EN. For evaluation, we utilize multi-bleu.perl to report BLEU scores, as in [Papineni *et al.*, 2002].

Experimental settings. We use OpenNMT [Klein *et al.*, 2017] as the implementation of the Transformer model.[4]

Results. Table 1 shows BLEU scores on WMT14 EN-DE and IWSLT14 DE-EN. From the results we can see that with-

[1]https://wit3.fbk.eu

[2]http://statmt.org/wmt14/translation-task.html

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

[4]https://github.com/OpenNMT/OpenNMT-py

Model	WMT14 EN-DE		IWSLT14 DE-EN	
	#Param	BLEU	#Param	BLEU
Transformer*	65.0M	27.30	-	34.40
Transformer (our)	61.3M	27.51	49.3M	34.87
+ DM	61.3M	28.12†	49.3M	35.53†
+ DM + MLM	**65.6M**	**28.41**†	**53.5M**	**35.88**†

Table 1: Translation results (BLEU score) on WMT14 EN-DE and IWSLT14 EN-DE. #Param indicates the number of model parameters. '*' indicates the results achieved by Transformer are reported in previous studies [Vaswani *et al.*, 2017] and [Wu *et al.*, 2019]. '†/‡' indicates statistically significant difference at 0.01/0.05 from Baseline model, tested by bootstrap resampling [Koehn, 2004].

out introducing any parameters, dynamic masking achieves significant improvement of 0.61 and 0.66 BLEU scores for the two translation tasks, respectively. This suggests that our approach is effective even when the training data is big, e.g., 3.9M for WMT14 EN-DE. One particularly nice property of dynamic masking is that it is incredibly easy to implement. By introducing a decoder layer with 4.3M parameters, the masked language model task further improves translation performance. Our approach achieves final improvement of 0.90 and 1.01 BLEU scores over the Transformer baseline.

3.2 AMR-to-text Generation

Dataset and evaluation. Following previous studies on AMR-to-text, we use the benchmark dataset AMR2.0 (LDC2017T10), which contains 36,521/1,368/1,371 training/development/testing sentences with corresponding AMR graphs. Following Zhu et al. [2019] and Ge et al. [2019], we segment tokens in AMR graphs and words in sentences into subwords by BPE with 10K operations and form a shared vocabulary for the source and target side. For evaluation, we follow related studies and adopt three different metrics: BLEU, Meteor [Banerjee and Lavie, 2005], and chrF++ [Popović, 2017].

Experimental settings. We use self-attention-based graph Transformer [Zhu *et al.*, 2019] as our baseline system,[5] which achieves the state-of-the-art performance for AMR-to-text generation. We follow the parameter settings in [Zhu *et al.*, 2019].

Results. Table 2 presents the detailed results of AMR-to-text generation on AMR 2.0. From the results we can see that both dynamic masking and masked language model task succeed to achieve improvement according to the three evaluation metrics. For example, dynamic masking achieves a significant improvement of 0.87 BLEU scores over the baseline and adding masked language model task gives a further improvement of 0.52 BLEU scores.

3.3 Image Caption Generation

Dataset and evaluation. For image caption generation, we experiment with the widely used dataset MSCOCO 2014 [Lin *et al.*, 2014]. The popular Karpathy splitting [Karpathy and Fei-Fei, 2015] is adopted, which results in 113,287 images for training, 5K images for validation, and 5K images for testing. Each of the images has five corresponding captions. We follow standard practice and perform only minimal text preprocessing that converts all sentences to lower case, tokenizes on white spaces, and filters words that occur less than five times. This way, we obtain a vocabulary of 9,487 words for this task. All sentences are truncated to contain at most 16 words during training. For evaluation, we employ several standard metrics, including SPICE [Anderson *et al.*, 2016], CIDEr [Vedantam *et al.*, 2015], METEOR, ROUGE-L [Lin, 2004], and BLEU.

Experimental settings. Following previous studies, we first train Faster R-CNN [Ren *et al.*, 2016] on Visual Genome dataset [Krishna *et al.*, 2017] to identify and localize instances of objects. For each image, we take the global average pooling of the final convolutional layer output, which results in a vector of 2,048 dimensions. In practice, these spatial features are extracted before the training of captioning models and are fixed during training. For captioning models, the dimension sizes of Transformer hidden states, image feature embeddings, and word embeddings are all set to 512.

Results. Table 3 presents the results of various single models on MSCOCO. From the results we can see that dynamic masking achieves improvement over the baseline and adding the task of masked language model obtains further improvement. The conclusion comes true regarding all the evaluation metrics.

4 Analysis

Next, we provide a series of analyses on a number of key factors that may affect the system performance. Such analyses help to better understand relative importance of the factors.

4.1 Effect of Different Noisy Strategies

We compare the dynamic masking strategy with two other noisy strategies.

Static masking (SM). Following BERT, we perform random masking during data preprocessing. Therefore, each training example is seen with fixed masks no matter how many times it is used in the training stage.

Gaussian noise (GN). Following Cheng et al. [2018], we modify the target sentence at *feature* level. Specifically, given a target sentence, we add Gaussian noises to word embeddings according to the following distribution:

$$E[y_i]' = E[y_i] + \epsilon, \quad \epsilon \sim N\left(0, \sigma^2 I\right), \tag{7}$$

where $E[y_i]$ is the word embedding for y_i, vector ϵ is a sample from a Gaussian distribution with variance σ^2. σ is a hyperparameter which we set to 0.01. We simply inject Gaussian noises to all of word embeddings when we feed a target sentence to decoder. Note that we perform Gaussian noise injection every time we feed a sequence to the model. Thus, each training example would be seen with K different corrupted versions, where K is the number of epochs in training stage.

[5] https://github.com/Amazing-J/structural-transformer

Model	#Param	BLEU	Meteor	chrF++
Graph Transformer (Zhu et al. [2019])	-	31.54	36.02	63.84
Graph Transformer (our)	54.2M	31.38	36.07	63.01
+ DM	54.2M	32.25†	36.51	63.68
+ DM + MLM	**58.4M**	**32.77**†	**36.85**	**64.46**

Table 2: Performance of AMR-to-text generation on AMR 2.0.

Model	CIDEr	BLEU-4	BLEU-1	ROUGE-L	METEOR	SPICE
Transformer (our)	114.1	35.8	76.0	56.3	27.7	20.8
+ DM	115.1	36.4‡	**76.7**	56.6	28.0	21.1
+ DM + MLM	**115.6**	**36.7**‡	**76.7**	**56.7**	**28.1**	**21.3**

Table 3: Performance of image caption generation on MSCOCO.

Model	WMT14 EN-DE	IWSLT14 DE-EN	AMR-to-text	Image Caption
Baseline	27.51	34.87	31.38	35.8
+ SM	27.66	35.47	31.10	35.8
+ SM + MLM	27.62	35.50	31.57	35.9
+ GN	27.69	35.10	31.41	36.1
+ DM	28.12	35.53	32.25	36.4
+ DM + MLM	**28.41**	**35.88**	**32.77**	**36.7**

Table 4: Performance comparison in BLEU score of different noising strategies on the text generation tasks.

Results. Table 4 compares the performance of different noisy strategies. From the results, we observe that:

- Static masking achieves an improvement of 0.6 BLEU score on IWSLT14 DE-EN. However, it has very limited effect in the other three text generation tasks.
- Based on static masking, the masked language model also fails to bring further improvement in the text generation tasks, except AMR-to-text generation.
- Although adding Gaussian noises to word embedding has positive impact across all text generation tasks, the effect is quite limited (improvement of $0.1 \sim 0.3$ BLEU score).
- Finally, our approach outperforms both static masking and Gaussian noise across all the text generation tasks.

4.2 Effect of Hyperparameter α

We explore the effect of hyperparameter α in Eq. 6, which controls the weight of the proposed task of masked language model. Table 5 presents the BLEU scores on the respective development set of the text generation tasks when α ranges from 0.0 to 1.0. From the results in Table 5, we find that for different text generation tasks, best performance is achieved with different values of α smaller than 0.7, which suggests that the optimal value of α is irrelevant to the task or the size of training data. Moreover, it shows that the task of masked language model brings more improvement on AMR-to-text than others. This is probably due to the smaller training data of AMR-to-text generation.

4.3 Effect of Sampled Token Ratio

An important hyperparameter in dynamic masking is the sampled token ratio. A low ratio makes the history context less different from gold one while sampling tokens at a high ratio makes the history context have more noise. Taking AMR-to-text generation as example, we compare four variants of sampled token ratio (10%, 15%, 20%, and 25%). As shown in Table 6, we see that the best performance of different metrics always appears at the ratio of 15%.

α	EN-DE	DE-EN	AMR-to-text	Image Caption
0.0	26.43	37.24	31.56	34.2
0.1	26.48	37.26	32.20	34.5
0.2	26.43	**37.32**	32.35	34.6
0.3	26.56	37.15	32.24	**34.8**
0.4	26.45	37.29	32.39	34.4
0.5	26.42	37.24	32.27	34.3
0.6	**26.62**	37.21	**32.64**	34.4
0.7	26.49	37.13	32.59	33.9
0.8	26.37	37.21	32.30	33.9
0.9	26.46	37.15	32.19	34.1
1.0	26.43	37.13	32.05	34.1

Table 5: Performance comparison in BLEU scores of different α values on the development data sets of the text generation tasks.

Ratio (%)	BLEU	Meteor	ChrF++
10	31.66	36.30	63.47
15	**32.77**	**36.85**	**64.46**
20	31.73	36.40	63.53
25	31.98	36.51	63.87

Table 6: Performance comparison on the test set of AMR-to-text generation when use different sampled token ratios.

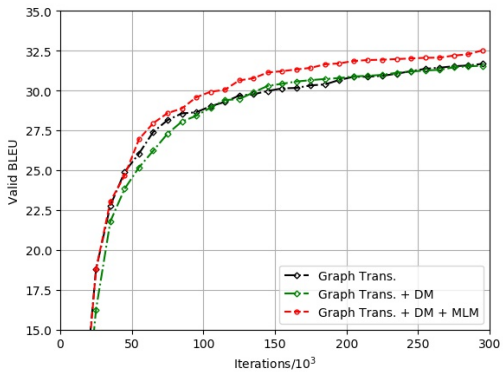

Figure 3: Learning curve of BLEU score over different training iterations on the development set of AMR 2.0.

4.4 Convergence Analysis

Introducing the task of masking language model to predict the masked token can not only achieve better performance, but also help the model converge faster. One reason is that compared to standard text generation tasks, our approach roughly makes prediction on additional 15% tokens in each batch.

Figure 3 shows the BLEU scores on the development set of AMR-to-text generation over different training iterations. Comparing the baseline (e.g., Graph Transformer) to the system with dynamic masking (e.g., + *DM*), we observe that the performance of + *DM* is lower than that of the baseline in the first 100K iterations. Then it starts to surpass the baseline after 100K iterations. With the task of masking language model, the system of + *DM* + *MLM* consistently outperforms both the baseline and + *DM* from the beginning. Moreover, it converges at about 200K iterations while the other two at about 300K iterations, suggesting fewer training iterations may be required for our approach to converge.

5 Related Work

We describe related work from the following two perspectives: robust text generation and masked language modeling.

5.1 Robust Text Generation

Using neural networks to generate natural language texts has been widely studied, ranging from machine translation to dialogue generation, AMR-to-text and image captioning. However, neural networks often suffer from vulnerability in the sense that small perturbations in various parameters or inputs can nevertheless result in different and often incorrect output.

To alleviate the effect of noisy perturbations and enhance the robustness of machine translation systems, Belinkov and Bisk [2017] design structure-invariant representations and resort to robust adversarial training. Cheng et al. [2018] introduce adversarial stability training to improve the robustness on arbitrary noise type. Sennrich et al. [2016a] and He et al. [2016] improve the robustness of machine translation models with monolingual corpora with the aid of an inverse model. Different from prior works, we do not resort to methods like adversarial training or monolingual data to improve robustness. Instead, we introduce randomness by masking tokens in the target side of training data. The idea is similar to Zhong et al. [2020] in spirit which improves the robustness of image processing by randomly erasing a rectangle region in an image. Such a method is simple and universal enough to be applied to arbitrary encoder-decoder architectures.

5.2 Masked Language Modeling

Our approach is inspired by masked language modeling (MLM), which is first adopted in a novel pre-training model to learn deep bidirectional language representations [Devlin *et al.*, 2018]. Subsequently, the same technique has become an essential component in many state-of-the-art pre-training methods [Lample and Conneau, 2019; Chen *et al.*, 2021a; Chen *et al.*, 2021b]. Briefly speaking, MLM randomly masks out a subset of tokens in the input sentences and is optimized to predict the masked tokens according to residual tokens.

To adapt MLM to the seq2seq framework, Song et al. [2019] (MASS) and Wang et al. [2019] (PoDA) propose to feed the encoder with a masked sequence and the decoder sequentially generates the masked tokens word-by-word. This seq2seq MLM can benefit the seq2seq-style downstream tasks, such as dialogue generation, machine translation, and summarization. However, for text generation tasks, the input to the encoder may be beyond token sequence. Considering the diversity of text generation tasks, the input may be table, graph, or even image. As a result, it is difficult to mask the different types of input. This is the reason that we propose to mask the output sentence of decoder, based on which we add a simple network to predict the masked tokens.

6 Conclusion

In this paper, we have proposed a general approach to improve the performance of text generation tasks. This is done from two levels. To train the models, we first have introduced randomness by randomly masking some percentage of tokens on the decoder side. This strengths the decoder by asking it to predict next tokens with rather than ground truth history context, but corrupted one. Meanwhile, we have proposed an auxiliary task to recover those masked tokens via a multi-task learning framework. Experimental results on various text generation tasks have demonstrated the effectiveness and generality of our proposed approach.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant No. 2020AAA0108600 and by the National Natural Science Foundation of China under Grant No. 61876120.

References

[Anderson *et al.*, 2016] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *ECCV*, 2016.

[Banerjee and Lavie, 2005] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL*, 2005.

[Belinkov and Bisk, 2017] Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*, 2017.

[Chen *et al.*, 2021a] Linqing Chen, Junhui Li, Zhengxian Gong, Boxing Chen, Weihua Luo, Min Zhang, and Guodong Zhou. Breaking the corpus bottleneck for context-aware neural machine translation with a novel joint pre-training approach. In *ACL-IJCNLP*, 2021.

[Chen *et al.*, 2021b] Linqing Chen, Junhui Li, Zhengxian Gong, Xiangyu Duan, Boxing Chen, Weihua Luo, Min Zhang, and Guodong Zhou. Improving context-aware neural machine translation with source-side monolingual documents. In *IJCAI*, 2021.

[Cheng *et al.*, 2018] Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. Towards robust neural machine translation. In *ACL*, 2018.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Dhillon *et al.*, 2018] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. In *arXiv Preprint arXiv:1803.01442*, 2018.

[Ge *et al.*, 2019] Donglai Ge, Junhui Li, Muhua Zhu, and Shoushan Li. Modeling source syntax and semantics for neural amr parsing. In *IJCAI*, 2019.

[He *et al.*, 2016] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *NIPS*, 2016.

[Karpathy and Fei-Fei, 2015] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

[Klein *et al.*, 2017] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. OpenNMT: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*, 2017.

[Koehn, 2004] Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, 2004.

[Krishna *et al.*, 2017] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 2017.

[Lample and Conneau, 2019] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[Lin, 2004] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *ACL*, 2004.

[Mei *et al.*, 2016] Hongyuan Mei, Mohit Bansal, and Mattew R Walter. What to talk about and how? Selective generation using LSTMs with coarse-to-fine alignment. In *NAACL-HLT*, 2016.

[Ott *et al.*, 2019] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT*, 2019.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[Popović, 2017] Maja Popović. chrF++: words helping character n-grams. In *WMT*, 2017.

[Ren *et al.*, 2016] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2016.

[Sennrich *et al.*, 2016a] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *ACL*, 2016.

[Sennrich *et al.*, 2016b] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL*, 2016.

[Song *et al.*, 2019] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. In *ICML*, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[Vedantam *et al.*, 2015] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015.

[Wang *et al.*, 2019] Liang Wang, Wei Zhao, Ruoyu Jia, Sujian Li, and Jingming Liu. Denoising based sequence-to-sequence pre-training for text generation. In *EMNLP-IJCNLP*, 2019.

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

[Zhang *et al.*, 2019] Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. In *ACL*, 2019.

[Zhong *et al.*, 2020] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *ICLR*, 2020.

[Zhu *et al.*, 2019] Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. Modeling graph structure in transformer for better AMR-to-text generation. In *EMNLP-IJCNLP*, 2019.