ERNIE-GEN: An Enhanced Multi-Flow Pre-training and Fine-tuning Framework for Natural Language Generation

Dongling Xiao[*] , **Han Zhang**[*] , **Yukun Li** , **Yu Sun** , **Hao Tian** ,
Hua Wu and **Haifeng Wang**
Baidu Inc., China
{xiaodongling, zhanghan17, liyukun01, sunyu02, tianhao, wu_hua, wanghaifeng}@baidu.com

Abstract

Current pre-training works in natural language generation pay little attention to the problem of exposure bias on downstream tasks. To address this issue, we propose an enhanced multi-flow sequence to sequence pre-training and fine-tuning framework named ERNIE-GEN, which bridges the discrepancy between training and inference with an infilling generation mechanism and a noise-aware generation method. To make generation closer to human writing patterns, this framework introduces a span-by-span generation flow that trains the model to predict semantically-complete spans consecutively rather than predicting word by word. Unlike existing pre-training methods, ERNIE-GEN incorporates multi-granularity target sampling to construct pre-training data, which enhances the correlation between encoder and decoder. Experimental results demonstrate that ERNIE-GEN achieves state-of-the-art results with a much smaller amount of pre-training data and parameters on a range of language generation tasks, including abstractive summarization (Gigaword and CNN/DailyMail), question generation (SQuAD), dialogue generation (Persona-Chat) and generative question answering (CoQA).

1 Introduction

Pre-trained on large-scale unlabeled text corpora and fine-tuned on downstream tasks, self-supervised representation models such as GPT [Radford *et al.*, 2018], BERT [Devlin *et al.*, 2019] and XLNet [Yang *et al.*, 2019b] have achieved remarkable improvements in natural language understanding (NLU). Different from encoder-only pre-training like BERT or decoder-only pre-training like GPT, natural language generation (NLG) relies on the sequence to sequence generation framework (seq2seq) which consists of a bidirectional encoder and a unidirectional decoder. Current pre-training works in NLG such as MASS [Song *et al.*, 2019] and UniLM [Dong *et al.*, 2019] mainly focus on jointly pre-training encoder and decoder on different self-supervised tasks. However, these works pay little attention to the exposure bias issue [Ranzato *et al.*, 2016], a major drawback of teacher-forcing training. This issue is due to the fact that groundtruth words are used during training, while generated words, whether predicted correctly or not, are used for inference where mistakes tend to accumulate.

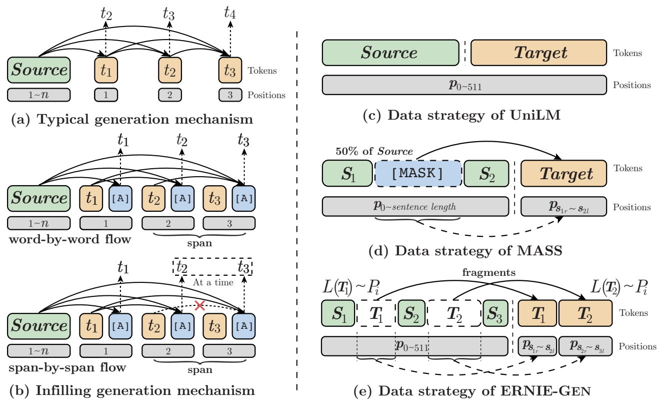

Source
Tokens
Positions
1∼n
(a) Typical generation mechanism
word-by-word flow
span
span-by-span flow
At a time
(b) Infilling generation mechanism
Target
(c) Data strategy of UniLM
50% of Source
[MASK]
(d) Data strategy of MASS
fragments
(e) Data strategy of ERNIE-GEN

Figure 1: Schematic of diverse generation mechanisms (left) and data strategies for pre-training (right). Blocks in green, orange and blue denote source texts, target texts and artificial symbols.

To alleviate this issue, we present ERNIE-GEN, an enhanced multi-flow seq2seq training framework characterized by a carefully-designed Multi-Flow Attention architecture based on Transformer [Vaswani et al., 2017], as illustrated in Figure 2. ERNIE-GEN incorporates a novel infilling generation mechanism and a noise-aware generation method into pre-training and fine-tuning, which is proved to be quite effective through experiments in §4.3.

- **Infilling generation.** Instead of using last groundtruth word in training or last generated word in inference, we adopt an inserted artificial symbol [ATTN] along with its position to gather history contextual representations at each step in both training and inference, which diverts model's attention away from last word and coerces it into focusing on all former representations, thus alleviating negative influence of previous mistakes to subsequent generation, as shown in Figure 1(b).
- **Noise-Aware generation.** We corrupt the input target sequence by randomly replacing words to arbitrary words in the vocabulary. This setup, despite its simplicity, proves to

[*]indicates equal contribution.

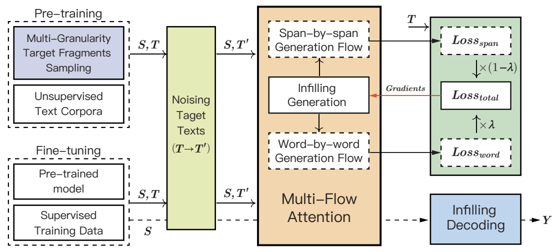


Figure 2: Overview of ERNIE-GEN. $\boldsymbol{S}, \boldsymbol{T}$ and $\boldsymbol{Y}$ donate source, target, and generated texts, $\boldsymbol{T}'$ is the noised version of $\boldsymbol{T}$.

be an effective way to make the model be aware of mistakes in training, so that the model is able to detect mistakes and ingore them during inference.

Moreover, in light of the fact that entities, phrases and sentences in human writing are organized in a coherent manner, we incorporate a **span-by-span generation task** into ERNIE-GEN as a new generation flow to train the model to predict semantically-complete spans consecutively rather than predicting word by word as traditional seq2seq models do. This task is implemented through the infilling generation mechanism in parallel with an infilling-based word-by-word generation flow to facilitate convergence in training, as shown in Figure 1b.

In addition, as shown in Figure 1(c-d), recent pre-training works for NLG like UniLM and MASS only sample a single continuous segment as target sequence. However, this sampling method compromises the correlation between encoder and decoder when it comes to pre-training of long texts (typically 512 words), given that adjacent segments are often relevant semantically. ERNIE-GEN adopts a **multi-granularity target fragments** sampling strategy to force decoder to rely more on the encoder representations other than the previous generated words, thus enhancing the correlation between encoder and decoder, as shown in Figure 1e.

Empirically, ERNIE-GEN is particularly effective and achieves state-of-the-art results on a range of NLG tasks including abstractive summarization (Gigaword and CNN/DailyMail), question generation (SQuAD), dialogue generation (Persona-Chat) and generative question answering (CoQA), utilizing a much smaller amount of pre-training data and parameters.

2 Related Work

Pre-Training for NLP Tasks. Recently, pre-training methods have achieved state-of-the-art results in multiple NLU tasks. ELMo [Peters *et al.*, 2018] pre-trains two unidirectional language models (LMs) with forward and backward direction respectively to feature downstream tasks. GPT utilizes an adjusted Transformer [Vaswani *et al.*, 2017] to learn a forward LM and then fine-tunes the forward LM on supervised datasets. BERT proposes a masked language modeling (MLM) task to learn deep bidirectional representations. Nevertheless, above methods are usually implemented by just one encoder or decoder, which is less effective in encoder-decoder based generation tasks, thus several works have preliminarily explored the pre-training towards NLG by incorporating BERT's MLM into the seq2seq framework and shown excellent performance on a range of generation tasks. MASS masks a consecutive fragment (50%) of the input sentence with artificial `[MASK]` to predict. UniLM masks several words in the input sequence which is a pair of segments for encoder and decoder, and then predicts the masked words in accordance with BERT's MLM. BART [Lewis *et al.*, 2019] corrupts the input sequence and trains the model to generate original sequence as a denoising autoencoder.

Exposure Bias issue. NLG tasks suffer from the exposure bias which is caused by teacher-forcing training. To address such issue, RNN-based variational autoencoders (VAEs) are leveraged in [Yang *et al.*, 2019a; Wang *et al.*, 2019], whereas it requires inference for both posterior and prior distribution. Reinforcement learning is also adopted to text generation against exposure bias issue [Ranzato *et al.*, 2016; Wang *et al.*, 2018], which is, however, inefficient during training because of the word-by-word sampling procedure. These methods are inefficient and less practical for pre-training that relies on large-scale unlabeled text corpora.

Span-level Pre-Training. Works like [Sun *et al.*, 2019; Sun *et al.*, 2020; Joshi *et al.*, 2019] verify that predicting spans reaches substantially better performance on NLU tasks. Meanwhile, inspired by characteristics of human expression, we hope the model have the foresight to generate a semantically-complete span at each step rather than a word. Consequently, a span-by-span generating task is proposed to make the model capable of generating texts more human-like.

3 Proposed Framework

Built on infilling generation mechanism, ERNIE-GEN adopts a Multi-Flow Attention architecture to train the model on word-by-word and span-by-span generation tasks in parallel. In this section, we describe ERNIE-GEN according to the training process shown in Figure 2.

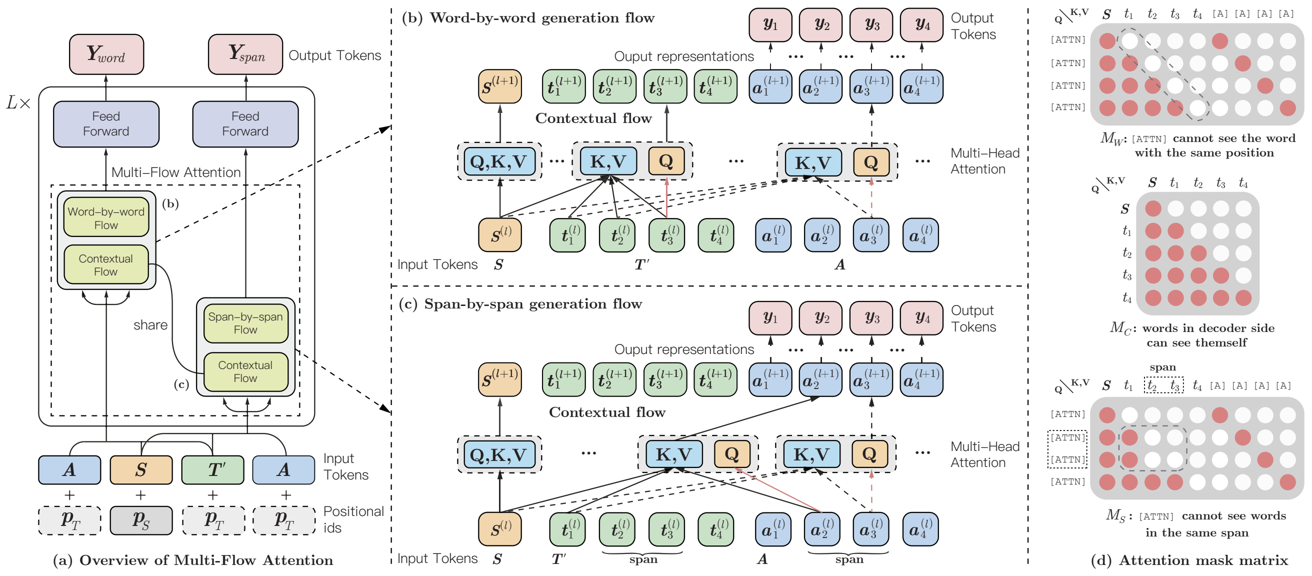
3.1 Multi-Granularity Target Fragments

Given a source sequence $\boldsymbol{S} = [s_1, ..., s_n]$, we first sample a length distribution P_i from a distribution set $\boldsymbol{P} = \{P_1, ..., P_l\}$ with probability p_i, then select a various of fragments according to P_i iteratively until the fragment budget has been spent (e.g. 25% of $\boldsymbol{S}$). We denote S_j^i as the j-th fragment which is sampled in length distribution P_i. Sampled fragments are then removed from $\boldsymbol{S}$ and stitched together to form target sequence $\boldsymbol{T} = \{T_1, ..., T_k\} = \{S_1^i, ..., S_k^i\}$. We denote $\boldsymbol{S}'$ as the left source sequence after removing sampled fragments. ERNIE-GEN performs pre-training by predicting the fragmented target sequence $\boldsymbol{T}$ and minimizing the negative log likelihood:

$$\begin{aligned}\mathcal{L}(\theta; \boldsymbol{S}, P_i) &= -\log P(\boldsymbol{T}|\boldsymbol{S}', P_i; \theta) \\ &= -\log \prod_{j=1}^{k} P(T_j|T_{<j}, \boldsymbol{S}', P_i; \theta).\end{aligned} \tag{1}$$

where the target sequence $\boldsymbol{T}$ is sorted by the position of each fragment. In each fragment T, $P(T) = \prod_{j=1}^{L(T)} P(t_j|t_{<j})$.

Following preliminary trials, we set a hyperparameter $\gamma = 0.25$, which denotes the ratio of length of all fragments to



(a) Overview of Multi-Flow Attention
(b) Word-by-word generation flow
(c) Span-by-span generation flow
(d) Attention mask matrix
Output Tokens
Feed Forward
Multi-Flow Attention
Word-by-word Flow
Contextual Flow
Span-by-span Flow
share
Input Tokens
Positional ids
Ouput representations
Contextual flow
Multi-Head Attention
span
M_W: [ATTN] cannot see the word with the same position
M_C: words in decoder side can see themself
M_S: [ATTN] cannot see words in the same span

Figure 3: Illustration of the Multi-Flow Attention module. (a):Overview of multi-flow attention where $\boldsymbol{p}$ denotes the position ids. The encoder and decoder share the parameters of multi-layer Transformer. (b):Word-by-word generation flow with history contextual representations (*Contextual Flow*). (c):Span-by-span generation flow with shared *Contextual Flow*. (d):The attention mask matrixes of word-by-word generation flow (M_W), contextual flow (M_C) and span-by-span generation flow (M_S).

that of source sequence $\boldsymbol{S}$. Besides, we introduce two uniform distributions $\boldsymbol{P} = \{U(1,4), U(4,32)\}$ with probability of 0.4 and 0.6 respectively to sample fragments, which aims to learn representations from different perspectives. On the one hand, short fragments benefit learning of semantic relation between words; on the other hand, longer fragments help to memorize sentence-level expressions.

3.2 Noise-Aware Generation

To training a generation model which can detect the false prediction and mitigate its impact on subsequent generation, we corrupt the groundtruth sequence $\boldsymbol{T}$ with a procedure where words are being replaced randomly, and the corrupted $\boldsymbol{T}$ is represented as $\boldsymbol{T}'$. There are two hyperparameters, ρ_p and ρ_f, denoting the noising rate in pre-training and fine-tuning respectively.

3.3 Architecture: Multi-Flow Attention

Formally, given a source sequence $\boldsymbol{S} = [s_1, ..., s_n]$, a target sequence $\boldsymbol{T} = [t_1, ..., t_m]$ and an artificial symbol sequence $\boldsymbol{A} = [\texttt{[ATTN]}_1, ..., \texttt{[ATTN]}_m]$ which has the same length as $\boldsymbol{T}$, we denote the inference of seq2seq network based on shared Transformer as follows:

$$\begin{aligned} \boldsymbol{s}_i^{(l+1)} &\leftarrow \texttt{MH-Att}(Q=\boldsymbol{s}_i^{(l)}, KV=\boldsymbol{S}^{(l)}). \\ \boldsymbol{t}_i^{(l+1)} &\leftarrow \texttt{MH-Att}(Q=\boldsymbol{t}_i^{(l)}, KV=\left[\boldsymbol{S}^{(l)}, t_{\leq i}^{(l)}\right]). \end{aligned} \quad (2)$$

where Q, K, V denote the query, key, and value in Multi-Head attention operation [Vaswani *et al.*, 2017]. $\boldsymbol{s}_i^{(l)}$ and $\boldsymbol{t}_i^{(l)}$ indicate the i-th vector representations of the l-th layer of Multi-Head Attention for the encoder and the decoder respectively, $[\cdot]$ denotes the concatenation operation. In this work, we call the above procedure the *Contextual Flow*.

Word-by-word Generation Flow. Based on infilling generation mechanism, this flow utilizes an inserted [ATTN] symbol to gather history representations word by word (see Figure 1b). To facilitate this process, we place all inserted [ATTN] together as shown in Figure 3b. To be specific, the word-by-word generation flow is updated as follow:

$$\boldsymbol{a}_i^{(l+1)} \leftarrow \texttt{MH-Att}(Q=\boldsymbol{a}_i^{(l)}, KV=\left[\boldsymbol{S}^{(l)}, \boldsymbol{t}_{<i}^{(l)}, \boldsymbol{a}_i^{(l)}\right]). \quad (3)$$

where $\boldsymbol{a}_i^{(l)}$ indicates the i-th vector representation of the l-th layer for the artificial symbol sequence $\boldsymbol{A}$.

Span-by-span Generation Flow. Different from word-by-word generation flow, span-by-span flow uses [ATTN] symbols to predict complete spans consecutively, as shown in Figure 3c. Formally, given a list of span boundaries $B = [b_1, ..., b_k]$, we conduct the span-by-span generation flow as:

$$\boldsymbol{a}_j^{(l+1)} \leftarrow \texttt{MH-Att}(Q=\boldsymbol{a}_j^{(l)}, KV=\left[\boldsymbol{S}^{(l)}, \boldsymbol{t}_{<b_i}^{(l)}, \boldsymbol{a}_j^{(l)}\right]). \quad (4)$$

where $j \in [b_i, b_{i+1})$, and $\boldsymbol{a}_j^{(l)}$ denotes the $(j-b_i)$-th vector representation of the i-th span. Essentially, the model is trained to predict a whole span $\boldsymbol{t}_{b_i:b_{i+1}}$ with the same history context $[\boldsymbol{S}, \boldsymbol{t}_{<b_i}]$.

Instead of randomly sampling spans, we prefer sampling spans with semantical information and knowledge. Specifically, we consider the following two steps to sample spans consecutively in $\boldsymbol{T}$:

- Firstly, we implement a T-test to compute t-statistic scores of all bigrams and trigrams, which is based on an initial hypothesis H_0: a random span of n arbitrary words $\boldsymbol{w} = [w_1, ..., w_n]$ with probability $P'(\boldsymbol{w}) = \prod_{i=1}^{n} P(w_i)$ cannot be a statistical n-gram. The t-statistic score is calculated by $(P(\boldsymbol{w}) - P'(\boldsymbol{w}))/\sqrt{\sigma^2/N}$, where $P(\boldsymbol{w})$ and σ indicate the statistic probability and standard deviation of $\boldsymbol{w}$ respectively, while N denotes the total number of n-grams appearing in the training data. According to the t-statistic scores, we filter out the top 200,000 bigrams, top 50,000 trigrams and all unigrams to construct a specific vocabulary of spans, which is represented as $\boldsymbol{V}_{span}$.

- Secondly, we search the trigram, bigram and unigram in order, starting with current word until a span (n-gram, $n \leq 3$) is retrieved in $\boldsymbol{V}_{span}$.

Multi-Flow Attention. To integrate the word-by-word generation flow and span-by-span generation flow, we apply them in parallel with a shared contextual flow by leveraging the multi-flow attention architecture, as Figure 3a describes. The multi-flow attention is computed as:

$$\begin{cases} \boldsymbol{X}^{(l+1)} \leftarrow \texttt{MH-Att}(Q = \boldsymbol{X}^{(l)}, KV = \boldsymbol{X}^{(l)}, M_C) \\ \boldsymbol{A}_W^{(l+1)} \leftarrow \texttt{MH-Att}(Q = \boldsymbol{A}_W^{(l)}, KV = \left[\boldsymbol{X}^{(l)}, \boldsymbol{A}_W^{(l)}\right], M_W) \\ \boldsymbol{A}_S^{(l+1)} \leftarrow \texttt{MH-Att}(Q = \boldsymbol{A}_S^{(l)}, KV = \left[\boldsymbol{X}^{(l)}, \boldsymbol{A}_S^{(l)}\right], M_S) \end{cases} \tag{5}$$

where $\boldsymbol{X}$ denotes the concatenation of $\boldsymbol{S}$ and $\boldsymbol{T}'$, $\boldsymbol{A}_W$, $\boldsymbol{A}_S$ are the artificial symbol sequences fed into the word-by-word generation flow and span-by-span generation flow respectively. As shown in Figure 3d, attention mask matrix M determines whether query and key can attend to each other by modifying the attention weight $W = \text{softmax}(QK^T/\sqrt{d_k} + M)$ [Vaswani *et al.*, 2017] . Specifically, M is assigned as:

$$M_{ij} = \begin{cases} 0, & \text{allow to see} \\ -\infty, & \text{prevent from seeing} \end{cases} \tag{6}$$

While training, we add the loss of word-by-word generation flow and span-by-span generation flow with an coefficient λ:

$$\begin{aligned} \mathcal{L}(\boldsymbol{T}, \boldsymbol{Y}; \theta) = {} & \lambda \mathcal{L}(\boldsymbol{T}, \boldsymbol{Y}_{word}; \theta) \\ & + (1 - \lambda)\mathcal{L}(\boldsymbol{T}, \boldsymbol{Y}_{span}; \theta). \end{aligned} \tag{7}$$

where $\boldsymbol{Y}_{word}$ and $\boldsymbol{Y}_{span}$ indicate the two generated sequences, and $\mathcal{L}(\cdot)$ denotes the cross entropy loss function. In detail, we set $\lambda = 0.5$ and $\lambda = 1.0$ respectively in pre-training and fine-tuning.

3.4 Infilling Decoding

While decoding, the target sequence $\boldsymbol{T}$ is unknown, we insert `[ATTN]` step by step to gather the representation of history context instead of preparing an artificial symbol sequence $\boldsymbol{A}$ in advance. Meanwhile, for the purpose of efficiency, we need to drop the inserted `[ATTN]` after inference at each step, as detailed in Figure 4.

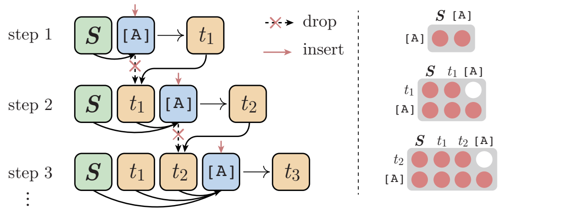


Figure 4: Schematic of infilling decoding: the particular procedures in infilling decoding including dropping and inserting (left) and the attention mask matrixes at each step (right).

4 Experiments

In this section, we compare our ERNIE-GEN with previous works and conduct several ablation experiments to assess the performance of proposed methods in §3.

4.1 Pre-training and Implementation

Analogous to BERT and UniLM, ERNIE-GEN is trained on English Wikipedia[1] and BookCorpus [Zhu *et al.*, 2015], totaling 16GB. The input sequence is lowercased and truncated to a maximum length of 512. We train a base model ERNIE-GEN$_{BASE}$ (L=12, H=768, A=12, Total Parameters=110M)[2] and a large model ERNIE-GEN$_{LARGE}$ (L=24, H=1024, A=16, Total Parameters=340M) with parameters initialized by BERT$_{BASE}$ and BERT$_{LARGE}$ respectively. Specifically, Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-9}$ is employed. The peak learning rate is 5e-5 with warmup over the first 4,000 steps and linear decay scheduling. The noising rate ρ_p for pre-training is 0.05. Batches are organized by limiting the maximum number of tokens to 24,576. Pre-training experiments are carried out on PaddlePaddle platforms[3] and Nvidia Tesla V100 GPU. We use 32 GPU cards and 64 GPU cards for ERNIE-GEN$_{BASE}$ and ERNIE-GEN$_{LARGE}$ respectively. By virtue of float16 mixed precision training, it takes almost 4 days for 400,000 steps to train ERNIE-GEN$_{BASE}$ while almost 7 days for 450,000 steps to train ERNIE-GEN$_{LARGE}$.

4.2 Fine-tuning on Downstream Tasks

Abstractive Summarization aims at generating fluent and concise summaries without being constrained to extracting subsequences from the input articles. We execute experiments on Gigaword dataset [Rush *et al.*, 2015] and CNN/DailyMail dataset [Hermann *et al.*, 2015]. Gigaword dataset contains 3.8M articles extracted from the Gigaword corpus, while CNN/DailyMail dataset consists of 93k articles and 220k articles from the CNN and Daily Mail respectively .

Model	**Data**	**Params**	**RG-1 / RG-2 / RG-L**
10k training samples : Gigaword 10k			
MASS [Song *et al.*, 2019]	18G	160M	25.03 / 9.48 / 23.48
UniLM$_{LARGE}$ [Dong *et al.*, 2019]	16G	340M	32.96 / 14.68 / 30.56
ERNIE-GEN$_{BASE}$	16G	110M	33.75 / 15.23 / 31.35
ERNIE-GEN$_{LARGE}$	16G	340M	**35.05 / 16.10 / 32.50**
Fully 3.8M training samples			
MASS [Song *et al.*, 2019]	18G	160M	37.66 / 18.53 / 34.89
BERTSHARE [Rothe *et al.*, 2019]	16G	110M	38.13 / 19.81 / 35.62
UniLM$_{LARGE}$ [Dong *et al.*, 2019]	16G	340M	38.45 / 19.45 / 35.75
PEGASUS$_{(C4)}$ [Zhang *et al.*, 2019]	750G	568M	38.75 / 19.96 / 36.14
PEGASUS$_{(HugeNews)}$ [Zhang *et al.*, 2019]	3.8T	568M	39.12 / 19.86 / 36.24
ERNIE-GEN$_{BASE}$	16G	110M	38.83 / 20.04 / 36.20
ERNIE-GEN$_{LARGE}$	16G	340M	**39.25 / 20.25 / 36.53**

Table 2: Comparison on Gigaword dataset with state-of-the-art results. Models in the upper block use 10k sample for fine-tuning. We also report the size of pre-training data and parameters utilized for each listed model (columns 2-3). RG is short for ROUGE.

[1]English Wikipedia version: enwiki-20181101.

[2]We donate the number of layers as L, the hidden size as H and the number of self-attention heads as A.

[3]https://github.com/PaddlePaddle/Paddle

Task	Epoch		Learning Rate		Noising Rate ρ_f		Dropout Rate		Batch Size	Label Smooth	Beam Size	Evaluation Metric
	BASE	*LARGE*	*BASE*	*LARGE*	*BASE*	*LARGE*	*BASE*	*LARGE*				
SQuAD QG	10	10	2.5e-5	1.5e-5	0.7	0.7	0.1	0.2	32	0.1	1	BLEU-4, METEOR (MTR), ROUGE-L (RG-L)
CNN/DailyMail	30	20	5e-5	4e-5	0.7	0.7	0.1	0.1	64	0.1	5	ROUGE-F1 scores:
Gigaword	10	5	3e-5	3e-5	0.5	0.6	0.1	0.2	128	0.1	5	ROUGE-1 (RG-1), ROUGE-2 (RG-2), ROUGE-L (RG-L)
Persona-Chat	-	30	-	1e-4	-	0.0	-	0.1	64	0.1	10	BLEU-1, BLEU-1, Distinct-1, Distinct-1
Generative CoQA	-	10	-	1e-5	-	0.5	-	0.1	32	0.1	3	F1-score

Table 1: Hyperparamters of fine-tuning for ERNIE-GEN$_{BASE}$ and ERNIE-GEN$_{LARGE}$.

Model	**Data**	**Params**	**RG-1 / RG-2 / RG-L**
BERTSHARE [Rothe *et al.*, 2019]	16G	110M	39.25 / 18.09 / 36.45
BERTSUMABS [Liu and Lapata, 2019]	16G	110M	41.72 / 19.39 / 38.76
MASS [Song *et al.*, 2019]	18G	160M	42.12 / 19.50 / 39.01
UniLM$_{LARGE}$ [Dong *et al.*, 2019]	16G	340M	43.33 / 20.21 / 40.51
T5$_{LARGE}$ [Raffel *et al.*, 2019]	750G	340M	42.50 / 20.68 / 39.75
T5$_{XLARGE}$ [Raffel *et al.*, 2019]	750G	11B	43.52 / **21.55** / 40.69
BART$_{LARGE}$ [Lewis *et al.*, 2019]	160G	400M	44.16 / 21.28 / 40.90
PEGASUS(C4) [Zhang *et al.*, 2019]	750G	568M	43.90 / 21.20 / 40.76
PEGASUS(HugeNews) [Zhang *et al.*, 2019]	3.8T	568M	**44.17** / 21.47 / 41.11
ERNIE-GEN$_{BASE}$	16G	110M	42.30 / 19.92 / 39.68
ERNIE-GEN$_{LARGE}$	16G	340M	44.02 / 21.17 / **41.26**

Table 3: Evaluation results on CNN/DailyMail. C4 and HugeNews are two massive datasets of 750G and 3.8T respectively.

The results on Gigaword with two scales (10k and 3.8M) are presented in Table 2, and the fine-tuning settings are shown in Table 1. On low-resource task (Gigaword 10k), ERNIE-GEN$_{BASE}$ outperforms UniLM$_{LARGE}$ by $+0.79$ points in ROUGE-L while ERNIE-GEN$_{LARGE}$ yields a gain of $+1.94$ ROUGE-L compared with UniLM$_{LARGE}$. On full Gigaword dataset, ERNIE-GEN$_{LARGE}$ creates the state-of-the-art results, outperforming various pervious methods. Specifically, ERNIE-GEN$_{BASE}$ outperforms PEGASUS$_{LARGE}$ (568M) trained with C4 (750G) by using only 16G training data and 110M parameters. Meanwhile, it is also interesting to see that with model size scaling up, gains in low-resource tasks appear to be more remarkable.

Table 3 shows the performance on CNN/DailyMail. With a similar amount of pre-training data and parameters, ERNIE-GEN$_{BASE}$ outperforms MASS by $+0.67$ ROUGE-L scores. Fairly compared with UniLM$_{LARGE}$, ERNIE-GEN$_{LARGE}$ obtains substantial gain of $+0.73$ ROUGE-L scores. Meanwhile, in spite of small pre-training data and parameters, our large model ERNIE-GEN$_{LARGE}$ also achieves state-of-the-art result on ROUGE-L and comparable performance on ROUGE-1 and ROUGE-2.

Question Generation is to generate a question according to a given input passage and a corresponding answer. We evaluate on the SQuAD 1.1 dataset [Rajpurkar *et al.*, 2016] for question generation task (called SQuAD QG). Following UniLM, we redistribute the original dataset into a new training set and testing set with the original development set unchanged. We also conduct experiment with the reversed dev↔test split as [Zhao *et al.*, 2018] indicates.

Model	**BLEU-4**	**METEOR**	**ROUGE-L**
SemQG [Zhang and Bansal, 2019]	18.37	22.65	46.68
UniLM$_{LARGE}$ [Dong *et al.*, 2019]	22.12	25.06	51.07
ERNIE-GEN$_{BASE}$	22.28	25.13	50.58
ERNIE-GEN$_{LARGE}$	**24.03**	**26.31**	**52.36**
Reversed test ↔ dev split			
SemQG [Zhang and Bansal, 2019]	20.76	24.20	48.91
UniLM$_{LARGE}$ [Dong *et al.*, 2019]	23.75	25.61	52.04
ERNIE-GEN$_{BASE}$	23.52	25.61	51.45
ERNIE-GEN$_{LARGE}$	**25.57**	**26.89**	**53.31**

Table 4: SQuAD QG results. Models in the upper block and lower block use different *test ↔ dev* split method.

Model	**BLEU-1/2**	**Distinct-1/2**
LIC [Bao *et al.*, 2019]	40.5 / 32.0	0.019 / 0.113
PLATO$_{\text{w/o latent}}$ [Bao *et al.*, 2019]	45.8 / 35.7	0.012 / 0.064
PLATO [Bao *et al.*, 2019]	41.8 / 32.4	0.014 / 0.081
ERNIE-GEN$_{LARGE}$	**46.8 / 36.4**	**0.023 / 0.168**

Table 5: Comparison with state-of-the-art results on Persona-Chat.

Specifically, the input source sequence is the concatenation of the input passage and the answer text, while the target sequence is a given question. We fine-tune ERNIE-GEN with the settings shown in Table 1. In Table 4, we present results of ERNIE-GEN and several previous works. Again, ERNIE-GEN outperforms UniLM and achieves a state-of-the-art result on question generation by giving $+1.82$ BLEU-4 scores.

Generative Question Answering / Dialogue Response in multi-turn conversations are challenging because of complex background knowledge and diverse utterances. We conduct an experiment on Persona-Chat dataset [Zhang *et al.*, 2018] to generate responses according to given multi-turn conversations and persona profile. Table 5 shows that ERNIE-GEN outperforms current task-specific pre-training model on dialogue generation. Beside, we also execute an experiment on CoQA dataset [Reddy *et al.*, 2019] to generate free-form answers for input questions and conversations. Compared with early models listed in Table 6, our generative question answering model works considerably better than early works by $+2.0$ F1-scores.

Model	**F1-score**
Seq2Seq [Reddy *et al.*, 2019]	27.5
PGNet [Reddy *et al.*, 2019]	45.4
UniLM$_{LARGE}$ [Dong *et al.*, 2019]	82.5
ERNIE-GEN$_{LARGE}$	**84.5**

Table 6: Generative question answering results on the development set of CoQA.

4.3 Ablation Studies

To better understand the importance of each proposed generation methods, we conduct experiments concerning the following two aspects:

- The robustness of infilling generation mechanism and noise-aware generation method against the exposure bias.
- The effectiveness of span-by-span generation task and the complete ERNIE-GEN model.

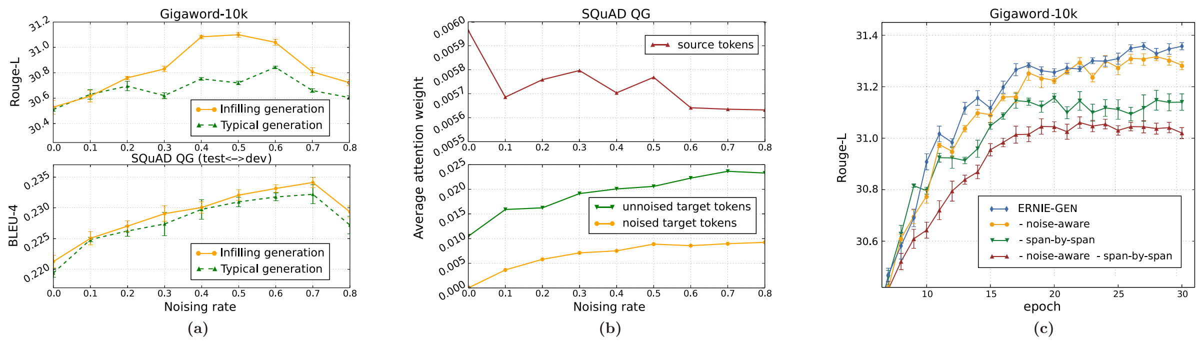


Figure 5: Results of ablation studies. (a):Comparisons between typical generation and infilling generation on Gigaword 10k and SQuAD QG for different fine-tuning noising rate ρ_f. (b):Noising Analysis, average attention weights of source words, unnoised target words and noised target words for diverse fine-tuning noising rate ρ_f. (c):Ablation study on Gigaword 10k, the x-axis shows fine-tuning epochs.

# Fine-tuning method	1 *Noising fine-tuning*: *Fine-tuning with noise-aware generation*			2 *Masking fine-tuning*: *Only updating the gradients of masked words*		
# Model	Gigaword 10k RG-1 / RG-2 / RG-L	CNN/DailyMail 10k RG-1 / RG-2 / RG-L	SQuAD QG Bleu-4 / MTR / RG-L	Gigaword 10k RG-1 / RG-2 / RG-L	CNN/DailyMail 10k RG-1 / RG-2 / RG-L	SQuAD QG Bleu-4 / MTR / RG-L
1 ERNIE-GEN	**33.75 / 15.23 / 31.35**	**39.92** / 17.46 / **37.40**	**23.52 / 25.61 / 51.45**	**33.30 / 15.04 / 31.22**	**39.54 / 17.61 / 37.00**	22.99 / 25.14 / 51.31
2 - noise-aware	33.57 / 15.15 / 31.28	39.78 / **17.63** / 37.23	23.40 / 25.50 / 51.36	33.01 / 14.94 / 31.00	39.53 / **17.61** / 36.97	23.09 / **25.15** / 51.41
3 - span-by-span	33.43 / 15.04 / 31.14	39.75 / 17.62 / 37.21	23.37 / 25.56 / 51.32	32.97 / 14.92 / 30.94	**39.54** / 17.57 / 36.95	**23.10** / 25.14 / **51.42**
4 - 2 and 3	33.23 / 14.77 / 31.00	39.71 / 17.55 / 37.18	23.34 / 25.54 / 51.30	32.57 / 14.68 / 30.60	39.49 / 17.66 / 36.96	22.89 / 25.08 / 51.28

Table 7: Ablation study for ERNIE-GEN$_{BASE}$ and its variants. Particularly, ERNIE-GEN sets $\rho_p = 0.05$ in pre-training (row 1), while removing the span-by-span generation task (row 3), we set $\rho_p = 0.2$ because the training becomes easier.

# Task (Metrics)	Typical generation	Infilling generation
Fine-tuning without noise-aware generation		
1 Gigaword 10k (RG-1 / RG-2 / RG-L)	**32.98 / 14.67** / 30.51	32.93 /14.46/ **30.53**
2 CNN/DM 10k (RG-1 / RG-2 / RG-L)	39.25 / 16.70 / 36.65	**39.56 / 16.93 / 36.94**
3 SQuAD QG (Bleu-4 / MTR / RG-L)	21.95 / 24.53 / 50.34	**22.13 / 24.66 / 50.51**
Fine-tuning with noise-aware generation		
4 Gigaword 10k (RG-1 / RG-2 / RG-L)	32.99 / **14.83** / 30.84	**33.23** / 14.77 / **31.00**
5 CNN/DM 10k (RG-1 / RG-2 / RG-L)	39.34 / 17.30 / 36.75	**39.71 / 17.55 / 37.18**
6 SQuAD QG (Bleu-4 / MTR / RG-L)	23.23 / 25.47 / 51.25	**23.34 / 25.54 / 51.30**

Table 8: Results of models pre-trained with typical generation and infilling generation. Tasks in the upper block are fine-tuned without noising, while the others are fine-tuned with noise-aware generation.

In Table 8, we compare two ERNIE-GEN$_{BASE}$ variants that are pre-trained with typical generation mechanism and infilling generation mechanism and that generates word by word. Row 1-3 shows that without noising groundtruth texts, infilling generation outperforms typical generation across tasks. Furthermore, both variants achieve remarkable improvements by fine-tuning with noise-aware generation method (row 4-6). Meanwhile it is interesting that infilling generation benefits more from the noising procedure, suggesting the robustness of infilling generation against mistakes while decoding. Specifically, Figure 5a shows the results with diverse choices of noising rate ρ_f on Gigaword 10k and SQuAD QG, indicating that appropriate noising substantially benefits the training and alleviates the training-inference discrepancy.

To further analyze the excellence of infilling generation mechanism with noising, we compute the average attention weights of source tokens, unnoised target tokens and noised target tokens in the last self-attention layer respectively on 1,000 samples. Average attention weights with diverse noising rate ρ_f are shown in Figure 5b, which tells us that the model pays more attention on the decoder side to figure out noised points and assign them less attention weights as the noising rate ρ_f increased in fine-tuning. Thereby, the model is able to detect and ignore the false predictions properly to alleviate accumulating mistakes while inference.

In column 1 of Table 7, we compare 4 ERNIE-GEN$_{BASE}$ variants on three tasks. We see that noise-aware generation method and span-by-span generation task (rows 2-3 of Table 7) play an important role in ERNIE-GEN pre-training and significantly outperform the baseline model which is only pre-trained with word-by-word infilling generation flow (row 4 of Table 7). After integrating noise-aware generation method and span-by-span generation task, ERNIE-GEN boosts the performance across all three tasks, as shown in row 1 of Table 7. In addition, to verify the idea that fine-tuning with unidirectional MLM like UniLM is inefficient due to the coupling of masking (noising) and predicting, we also list the fine-tuning results obtained by a unidirectional MLM with masking probability of 0.7, as shown in column 2 of Table 7. We observe that our noise-aware generation method significantly outperforms the unidirectional MLM in fine-tuning.

5 Conclusions

We present an enhanced multi-flow seq2seq pre-training and fine-tuning framework (ERNIE-GEN) for language generation, which incorporates a infilling generation mechanism and a noise-aware generation method to alleviate the exposure bias. Besides, ERNIE-GEN integrates a new span-by-span generation task to train the model to generate texts like human writing, which further improves the performance on downstream tasks. Through extensive experiments, ERNIE-GEN achieves state-of-the-art results on a range of NLG tasks.

References

[Bao *et al.*, 2019] Siqi Bao, Huang He, Fan Wang, and Hua Wu. Plato: Pre-trained dialogue generation model with discrete latent variable. *arXiv preprint arXiv:1910.07931*, 2019.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, 2019.

[Dong *et al.*, 2019] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiaowuen Hon. Unified language model pre-training for natural language understanding and generation. In *NIPS*, pages 13042–13054, 2019.

[Hermann *et al.*, 2015] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701, 2015.

[Joshi *et al.*, 2019] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*, 2019.

[Lewis *et al.*, 2019] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[Liu and Lapata, 2019] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *EMNLP-IJCNLP*, pages 3721–3731, 2019.

[Peters *et al.*, 2018] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237, 2018.

[Radford *et al.*, 2018] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[Raffel *et al.*, 2019] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392, 2016.

[Ranzato *et al.*, 2016] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *ICLR*, 2016.

[Reddy *et al.*, 2019] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. In *ACL*, pages 249–266, 2019.

[Rothe *et al.*, 2019] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. Leveraging pre-trained checkpoints for sequence generation tasks. *arXiv preprint arXiv:1907.12461*, 2019.

[Rush *et al.*, 2015] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015.

[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.

[Sun *et al.*, 2019] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

[Sun *et al.*, 2020] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. In *AAAI*, 2020.

[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*, pages 5998–6008, 2017.

[Wang *et al.*, 2018] Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. In *IJCAI*, 2018.

[Wang *et al.*, 2019] Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. Topic-guided variational autoencoder for text generation. In *NAACL-HLT*, 2019.

[Yang *et al.*, 2019a] Qian Yang, Dinghan Shen, Yong Cheng, Wenlin Wang, Guoyin Wang, Lawrence Carin, et al. An end-to-end generative architecture for paraphrase generation. In *EMNLP-IJCNLP*, pages 3123–3133, 2019.

[Yang *et al.*, 2019b] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NIPS*, pages 5754–5764, 2019.

[Zhang and Bansal, 2019] Shiyue Zhang and Mohit Bansal. Addressing semantic drift in question generation for semi-supervised question answering. In *EMNLP-IJCNLP*, pages 2495–2509, 2019.

[Zhang *et al.*, 2018] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*, pages 2204–2213, 2018.

[Zhang *et al.*, 2019] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777*, 2019.

[Zhao *et al.*, 2018] Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *EMNLP*, pages 3901–3910, 2018.

[Zhu *et al.*, 2015] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and

Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, pages 19–27, 2015.