GLM: General Language Model Pretraining with Autoregressive Blank Infilling

Zhengxiao Du[*1,2] **Yujie Qian**[*3] **Xiao Liu**[1,2] **Ming Ding**[1,2] **Jiezhong Qiu**[1,2]
Zhilin Yang[†1,4] **Jie Tang**[†1,2]
[1]Tsinghua University [2]Beijing Academy of Artificial Intelligence (BAAI)
[3]MIT CSAIL [4]Shanghai Qi Zhi Institute
zx-du20@mails.tsinghua.edu.cn yujieq@csail.mit.edu
{zhiliny,jietang}@tsinghua.edu.cn

Abstract

There have been various types of pretraining architectures including autoencoding models (e.g., BERT), autoregressive models (e.g., GPT), and encoder-decoder models (e.g., T5). However, none of the pretraining frameworks performs the best for all tasks of three main categories including natural language understanding (NLU), unconditional generation, and conditional generation. We propose a General Language Model (GLM) based on autoregressive blank infilling to address this challenge. GLM improves blank filling pretraining by adding 2D positional encodings and allowing an arbitrary order to predict spans, which results in performance gains over BERT and T5 on NLU tasks. Meanwhile, GLM can be pretrained for different types of tasks by varying the number and lengths of blanks. On a wide range of tasks across NLU, conditional and unconditional generation, GLM outperforms BERT, T5, and GPT given the same model sizes and data, and achieves the best performance from a single pretrained model with 1.25× parameters of $\text{BERT}_{\text{Large}}$, demonstrating its generalizability to different downstream tasks.[1]

1 Introduction

Language models pretrained on unlabeled texts have substantially advanced the state of the art in various NLP tasks, ranging from natural language understanding (NLU) to text generation (Radford et al., 2018a; Devlin et al., 2019; Yang et al., 2019; Radford et al., 2018b; Raffel et al., 2020; Lewis et al., 2019; Brown et al., 2020). Downstream task performance as well as the scale of the parameters have also constantly increased in the past few years.

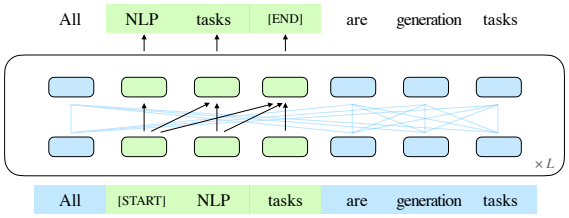

Figure 1: Illustration of GLM. We blank out text spans (green part) and generate them autoregressively. (Some attention edges are omitted; cf. Figure 2.)

In general, existing pretraining frameworks can be categorized into three families: *autoregressive*, *autoencoding*, and *encoder-decoder* models. Autoregressive models, such as GPT (Radford et al., 2018a), learn left-to-right language models. While they succeed in long-text generation and show few-shot learning ability when scaled to billions of parameters (Radford et al., 2018b; Brown et al., 2020), the inherent disadvantage is the unidirectional attention mechanism, which cannot fully capture the dependencies between the context words in NLU tasks. Autoencoding models, such as BERT (Devlin et al., 2019), learn bidirectional context encoders via denoising objectives, e.g. Masked Language Model (MLM). The encoders produce contextualized representations that suit natural language understanding tasks, but could not be directly applied for text generation. Encoder-decoder models adopt bidirectional attention for the encoder, unidirectional attention for the decoder, and cross attention between them (Song et al., 2019; Bi et al., 2020; Lewis et al., 2019). They are typically deployed in conditional generation tasks, such as text summarization and response generation. [2]. T5 (Raffel et al., 2020) unifies NLU and conditional generation via encoder-decoder models but requires more parameters to match the performance

[*]The first two authors contributed equally.
[†]Corresponding authors.
[1]The code and pre-trained models are available at https://github.com/THUDM/GLM

[2]Unconditional generation refers to generating text as a language model without finetuning, while conditional generation refers to sequence-to-sequence tasks.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 320 - 335

of BRET-based models such as RoBERTa (Liu et al., 2019) and DeBERTa (He et al., 2021).

None of these pretraining frameworks is flexible enough to perform competitively across all NLP tasks. Previous works have tried to unify different frameworks by combining their objectives via multi-task learning (Dong et al., 2019; Bao et al., 2020). However, since the autoencoding and autoregressive objectives differ by nature, a simple unification cannot fully inherit the advantages of both frameworks.

In this paper, we propose a pretraining framework named GLM (General Language Model), based on autoregressive blank infilling. We randomly blank out continuous spans of tokens from the input text, following the idea of autoencoding, and train the model to sequentially reconstruct the spans, following the idea of autoregressive pretraining (see Figure 1). While blanking filling has been used in T5 (Raffel et al., 2020) for text-to-text pretraining, we propose two improvements, namely span shuffling and 2D positional encoding. Empirically, we show that with the same amount of parameters and computational cost, GLM significantly outperforms BERT on the SuperGLUE benchmark by a large margin of 4.6% – 5.0% and outperforms RoBERTa and BART when pretrained on a corpus of similar size (158GB). GLM also significantly outperforms T5 on NLU and generation tasks with fewer parameters and data.

Inspired by Pattern-Exploiting Training (PET) (Schick and Schütze, 2020a), we reformulate NLU tasks as manually-crafted cloze questions that mimic human language. Different from the BERT-based models used by PET, GLM can naturally handle multi-token answers to the cloze question via autoregressive blank filling.

Furthermore, we show that by varying the number and lengths of missing spans, the autoregressive blank filling objective can pretrain language models for conditional and unconditional generation. Through multi-task learning of different pretraining objectives, a single GLM can excel in both NLU and (conditional and unconditional) text generation. Empirically, compared with standalone baselines, GLM with multi-task pretraining achieves improvements in NLU, conditional text generation, and language modeling tasks altogether by sharing the parameters.

2 GLM Pretraining Framework

We propose a general pretraining framework GLM based on a novel autoregressive blank infilling objective. GLM formulates NLU tasks as cloze questions that contain task descriptions, which can be answered by autoregressive generation.

2.1 Pretraining Objective

2.1.1 Autoregressive Blank Infilling

GLM is trained by optimizing an *autoregressive blank infilling* objective. Given an input text $\boldsymbol{x} = [x_1, \cdots, x_n]$, multiple text spans $\{\boldsymbol{s}_1, \cdots, \boldsymbol{s}_m\}$ are sampled, where each span $\boldsymbol{s}_i$ corresponds to a series of consecutive tokens $[s_{i,1}, \cdots, s_{i,l_i}]$ in $\boldsymbol{x}$. Each span is replaced with a single [MASK] token, forming a corrupted text $\boldsymbol{x}_{\text{corrupt}}$. The model predicts the missing tokens in the spans from the corrupted text in an autoregressive manner, which means when predicting the missing tokens in a span, the model has access to the corrupted text *and* the previously predicted spans. To fully capture the interdependencies between different spans, we randomly permute the order of the spans, similar to the permutation language model (Yang et al., 2019). Formally, let Z_m be the set of all possible permutations of the length-m index sequence $[1, 2, \cdots, m]$, and $\boldsymbol{s}_{\boldsymbol{z}_{<i}}$ be $[\boldsymbol{s}_{z_1}, \cdots, \boldsymbol{s}_{z_{i-1}}]$, we define the pretraining objective as

$$\max_{\theta} \mathbb{E}_{\boldsymbol{z} \sim Z_m} \left[\sum_{i=1}^{m} \log p_\theta(\boldsymbol{s}_{z_i} | \boldsymbol{x}_{\text{corrupt}}, \boldsymbol{s}_{\boldsymbol{z}_{<i}}) \right] \quad (1)$$

We always generate the tokens in each blank following a left-to-right order, i.e. the probability of generating the span $\boldsymbol{s}_i$ is factorized as:

$$\begin{aligned} & p_\theta(\boldsymbol{s}_i | \boldsymbol{x}_{\text{corrupt}}, \boldsymbol{s}_{\boldsymbol{z}_{<i}}) \\ = & \prod_{j=1}^{l_i} p(s_{i,j} | \boldsymbol{x}_{\text{corrupt}}, \boldsymbol{s}_{\boldsymbol{z}_{<i}}, \boldsymbol{s}_{i,<j}) \end{aligned} \quad (2)$$

We implement the autoregressive blank infilling objective with the following techniques. The input $\boldsymbol{x}$ is divided into two parts: Part A is the corrupted text $\boldsymbol{x}_{\text{corrupt}}$, and Part B consists of the masked spans. Part A tokens can attend to each other, but cannot attend to any tokens in B. Part B tokens can attend to Part A and antecedents in B, but cannot attend to any subsequent tokens in B. To enable autoregressive generation, each span is padded with special tokens [START] and [END], for input and

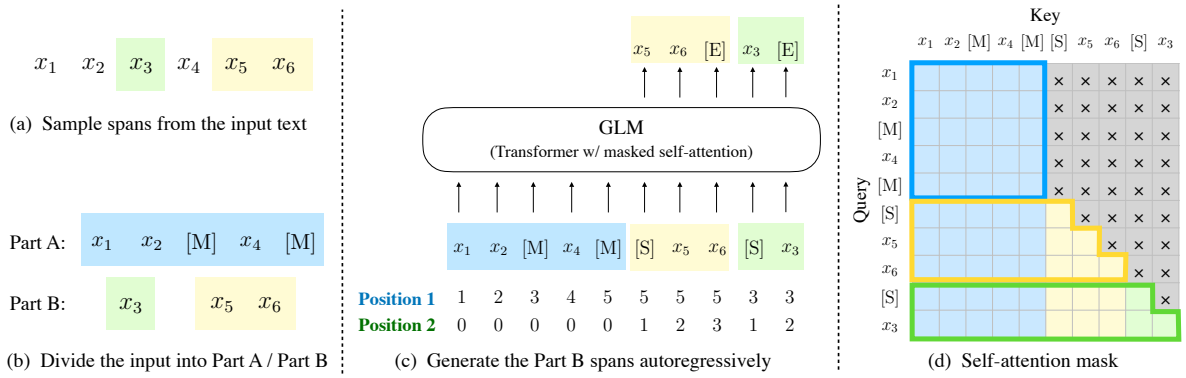

Figure 2: GLM pretraining. (a) The original text is $[x_1, x_2, x_3, x_4, x_5, x_6]$. Two spans $[x_3]$ and $[x_5, x_6]$ are sampled. (b) Replace the sampled spans with [M] in Part A, and shuffle the spans in Part B. (c) GLM autoregressively generates Part B. Each span is prepended with [S] as input and appended with [E] as output. 2D positional encoding represents inter- and intra-span positions. (d) Self-attention mask. Grey areas are masked out. Part A tokens can attend to themselves (blue frame) but not B. Part B tokens can attend to A and their antecedents in B (yellow and green frames correspond to the two spans). [M] := [MASK], [S] := [START], and [E] := [END].

output respectively. In this way, our model automatically learns a bidirectional encoder (for Part A) and a unidirectional decoder (for Part B) in a unified model. The implementation of GLM is illustrated in Figure 2.

We randomly sample spans of length drawn from a Poisson distribution with $\lambda = 3$. We repeatedly sample new spans until at least 15% of the original tokens are masked. Empirically, we have found that the 15% ratio is critical for good performance on downstream NLU tasks.

2.1.2 Multi-Task Pretraining

In the previous section, GLM masks short spans and is suited for NLU tasks. However, we are interested in pretraining a single model that can handle both NLU and text generation. We then study a *multi-task pretraining* setup, in which a second objective of generating longer text is jointly optimized with the blank infilling objective. We consider the following two objectives:

- Document-level. We sample a single span whose length is sampled from a uniform distribution over 50%–100% of the original length. The objective aims for long text generation.
- Sentence-level. We restrict that the masked spans must be full sentences. Multiple spans (sentences) are sampled to cover 15% of the original tokens. This objective aims for seq2seq tasks whose predictions are often complete sentences or paragraphs.

Both new objectives are defined in the same way as the original objective, i.e. Eq. 1. The only difference is the number of spans and the span lengths.

2.2 Model Architecture

GLM uses a single Transformer with several modifications to the architecture: (1) we rearrange the order of layer normalization and the residual connection, which has been shown critical for large-scale language models to avoid numerical errors (Shoeybi et al., 2019); (2) we use a single linear layer for the output token prediction; (3) we replace ReLU activation functions with GeLUs (Hendrycks and Gimpel, 2016).

2.2.1 2D Positional Encoding

One of the challenges of the autoregressive blank infilling task is how to encode the positional information. Transformers rely on positional encodings to inject the absolute and relative positions of the tokens. We propose 2D positional encodings to address the challenge. Specifically, each token is encoded with two positional ids. The first positional id represents the position in the corrupted text $\boldsymbol{x}_{\text{corrupt}}$. For the masked spans, it is the position of the corresponding [MASK] token. The second positional id represents the intra-span position. For tokens in Part A, their second positional ids are 0. For tokens in Part B, they range from 1 to the length of the span. The two positional ids are projected into two vectors via learnable embedding tables, which are both added to the input token embeddings.

Our encoding method ensures that the model is not aware of the length of the masked span when

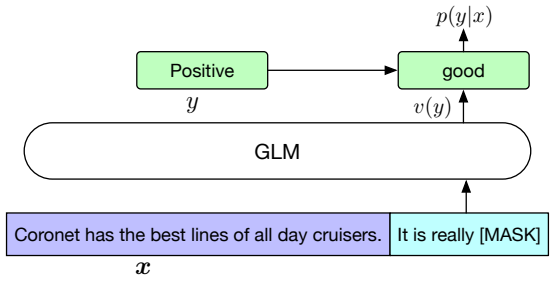

Figure 3: Formulation of the sentiment classification task as blank infilling with GLM.

reconstructing them. It is an important difference as compared to other models. For example, XLNet (Yang et al., 2019) encodes the original position so that it can perceive the number of missing tokens, and SpanBERT (Joshi et al., 2020) replaces the span with multiple [MASK] tokens and keeps the length unchanged. Our design fits downstream tasks as usually the length of the generated text is unknown beforehand.

2.3 Finetuning GLM

Typically, for downstream NLU tasks, a linear classifier takes the representations of sequences or tokens produced by pretrained models as input and predicts the correct labels. The practices are different from the generative pretraining task, leading to inconsistency between pretraining and finetuning.

Instead, we reformulate NLU classification tasks as generation tasks of blank infilling, following PET (Schick and Schütze, 2020a). Specifically, given a labeled example $(\boldsymbol{x}, y)$, we convert the input text $\boldsymbol{x}$ to a cloze question $c(\boldsymbol{x})$ via a pattern containing a single mask token. The pattern is written in natural language to represent the semantics of the task. For example, a sentiment classification task can be formulated as "{SENTENCE}. It's really [MASK]". The candidate labels $y \in \mathcal{Y}$ are also mapped to answers to the cloze, called verbalizer $v(y)$. In sentiment classification, the labels "positive" and "negative" are mapped to the words "good" and "bad". The conditional probability of predicting y given $\boldsymbol{x}$ is

$$p(y|\boldsymbol{x}) = \frac{p(v(y)|c(\boldsymbol{x}))}{\sum_{y' \in \mathcal{Y}} p(v(y')|c(\boldsymbol{x}))} \tag{3}$$

where $\mathcal{Y}$ is the label set. Therefore the probability of the sentence being positive or negative is proportional to predicting "good" or "bad" in the blank. Then we finetune GLM with a cross-entropy loss (see Figure 3).

For text generation tasks, the given context constitutes the Part A of the input, with a mask token appended at the end. The model generates the text of Part B autoregressively. We can directly apply the pretrained GLM for unconditional generation, or finetune it on downstream conditional generation tasks.

2.4 Discussion and Analysis

In this section, we discuss the differences between GLM and other pretraining models. We are mainly concerned with how they can be adapted to downstream blank infilling tasks.

Comparison with BERT (Devlin et al., 2019). As pointed out by (Yang et al., 2019), BERT fails to capture the interdependencies of masked tokens due to the independence assumption of MLM. Another disadvantage of BERT is that it cannot fill in the blanks of multiple tokens properly. To infer the probability of an answer of length l, BERT needs to perform l consecutive predictions. If the length l is unknown, we may need to enumerate all possible lengths, since BERT needs to change the number of [MASK] tokens according to the length.

Comparison with XLNet (Yang et al., 2019). Both GLM and XLNet are pretrained with autoregressive objectives, but there are two differences between them. First, XLNet uses the original position encodings before corruption. During inference, we need to either know or enumerate the length of the answer, the same problem as BERT. Second, XLNet uses a two-stream self-attention mechanism, instead of the right-shift, to avoid the information leak within Transformer. It doubles the time cost of pretraining.

Comparison with T5 (Raffel et al., 2020). T5 proposes a similar blank infilling objective to pretrain an encoder-decoder Transformer. T5 uses independent positional encodings for the encoder and decoder, and relies on multiple sentinel tokens to differentiate the masked spans. In downstream tasks, only one of the sentinel tokens is used, leading to a waste of model capacity and inconsistency between pretraining and finetuning. Moreover, T5 always predicts spans in a fixed left-to-right order. As a result, GLM can significantly outperform T5 on NLU and seq2seq tasks with fewer parameters and data, as stated in Sections 3.2 and 3.3.

Comparison with UniLM (Dong et al., 2019). UniLM combines different pretraining objectives under the autoencoding framework by changing the

attention mask among bidirectional, unidirectional, and cross attention. However, UniLM always replaces masked spans with [MASK] tokens, which limits its ability to model the dependencies between the masked spans and their context. GLM feeds in the previous token and autoregressively generates the next token. Finetuning UniLM on downstream generation tasks also relies on masked language modeling, which is less efficient. UniLMv2 (Bao et al., 2020) adopts partially autoregressive modeling for generation tasks, along with the autoencoding objective for NLU tasks. Instead, GLM unifies NLU and generation tasks with autoregressive pretraining.

3 Experiments

We now describe our pretraining setup and the evaluation of downstream tasks.

3.1 Pretraining Setup

For a fair comparison with BERT (Devlin et al., 2019), we use BooksCorpus (Zhu et al., 2015) and English Wikipedia as our pretraining data. We use the uncased wordpiece tokenizer of BERT with 30k vocabulary. We train GLM_{Base} and $\text{GLM}_{\text{Large}}$ with the same architectures as $\text{BERT}_{\text{Base}}$ and $\text{BERT}_{\text{Large}}$, containing 110M and 340M parameters respectively.

For multi-task pretraining, we train two Large-sized models with a mixture of the blank infilling objective and the document-level or sentence-level objective, denoted as GLM_{Doc} and GLM_{Sent}. Additionally, we train two larger GLM models of 410M (30 layers, hidden size 1024, and 16 attention heads) and 515M (30 layers, hidden size 1152, and 18 attention heads) parameters with document-level multi-task pretraining, denoted as GLM_{410M} and GLM_{515M}.

To compare with SOTA models, we also train a Large-sized model with the same data, tokenization, and hyperparameters as RoBERTa (Liu et al., 2019), denoted as $\text{GLM}_{\text{RoBERTa}}$. Due to resource limitations, we only pretrain the model for 250,000 steps, which are half of RoBERTa and BART's training steps and close to T5 in the number of trained tokens. More experiment details can be found in Appendix A.

3.2 SuperGLUE

To evaluate our pretrained GLM models, we conduct experiments on the SuperGLUE benchmark (Wang et al., 2019) and report the standard metrics. SuperGLUE consists of 8 challenging NLU tasks. We reformulate the classification tasks as blank infilling with human-crafted cloze questions, following PET (Schick and Schütze, 2020b). Then we finetune the pretrained GLM models on each task as described in Section 2.3. The cloze questions and other details can be found in Appendix B.1.

For a fair comparison with GLM_{Base} and $\text{GLM}_{\text{Large}}$, we choose $\text{BERT}_{\text{Base}}$ and $\text{BERT}_{\text{Large}}$ as our baselines, which are pretrained on the same corpus and for a similar amount of time. We report the performance of standard finetuning (i.e. classification on the [CLS] token representation). The performance of BERT with cloze questions is reported in Section 3.4. To compare with $\text{GLM}_{\text{RoBERTa}}$, we choose T5, $\text{BART}_{\text{Large}}$, and $\text{RoBERTa}_{\text{Large}}$ as our baselines. T5 has no direct match in the number of parameters for $\text{BERT}_{\text{Large}}$, so we present the results of both T5_{Base} (220M parameters) and T5_{Large} (770M parameters). All the other baselines are of similar size to $\text{BERT}_{\text{Large}}$.

Table 1 shows the results. With the same amount of training data, GLM consistently outperforms BERT on most tasks with either base or large architecture. The only exception is WiC (word sense disambiguation). On average, GLM_{Base} scores 4.6% higher than $\text{BERT}_{\text{Base}}$, and $\text{GLM}_{\text{Large}}$ scores 5.0% higher than $\text{BERT}_{\text{Large}}$. It clearly demonstrates the advantage of our method in NLU tasks. In the setting of $\text{RoBERTa}_{\text{Large}}$, $\text{GLM}_{\text{RoBERTa}}$ can still achieve improvements over the baselines, but with a smaller margin. Specifically, $\text{GLM}_{\text{RoBERTa}}$ outperforms T5_{Large} but is only half its size. We also find that BART does not perform well on the challenging SuperGLUE benchmark. We conjecture this can be attributed to the low parameter efficiency of the encoder-decoder architecture and the denoising sequence-to-sequence objective.

3.3 Multi-Task Pretraining

Then we evaluate the GLM's performance in a multi-task setting (Section 2.1). Within one training batch, we sample short spans and longer spans (document-level or sentence-level) with equal chances. We evaluate the multi-task model for NLU, seq2seq, blank infilling, and zero-shot language modeling.

SuperGLUE. For NLU tasks, we evaluate models on the SuperGLUE benchmark. The results

Table 1: Results on the SuperGLUE dev set.

Model	ReCoRD F1/Acc.	COPA Acc.	WSC Acc.	RTE Acc.	BoolQ Acc.	WiC Acc.	CB F1/Acc.	MultiRC F1a/EM	Avg
Pretrained on BookCorpus and Wikipedia									
$\text{BERT}_{\text{Base}}$	65.4 / 64.9	66.0	65.4	70.0	74.9	**68.8**	70.9 / 76.8	68.4 / 21.5	66.1
GLM_{Base}	**73.5 / 72.8**	**71.0**	**72.1**	**71.2**	**77.0**	64.7	**89.5 / 85.7**	**72.1 / 26.1**	**70.7**
$\text{BERT}_{\text{Large}}$	76.3 / 75.6	69.0	64.4	73.6	80.1	**71.0**	94.8 / 92.9	71.9 / 24.1	72.0
$\text{UniLM}_{\text{Large}}$	80.0 / 79.1	72.0	65.4	76.5	80.5	69.7	91.0 / 91.1	**77.2 / 38.2**	74.1
$\text{GLM}_{\text{Large}}$	81.7 / 81.1	76.0	**81.7**	74.0	**82.1**	68.5	96.1 / 94.6	77.1 / 36.3	77.0
GLM_{Doc}	80.2 / 79.6	77.0	78.8	76.2	79.8	63.6	**97.3 / 96.4**	74.6 / 32.1	75.7
GLM_{Sent}	80.7 / 80.2	77.0	79.8	79.1	80.8	70.4	94.6 / 93.7	76.9 / 36.1	76.8
GLM_{410M}	81.5 / 80.9	80.0	**81.7**	**79.4**	81.9	69.0	93.2 / **96.4**	76.2 / 35.5	78.0
GLM_{515M}	**82.3 / 81.7**	**85.0**	**81.7**	79.1	81.3	69.4	95.0 / **96.4**	**77.2** / 35.0	**78.8**
Pretrained on larger corpora									
T5_{Base}	76.2 / 75.4	73.0	79.8	78.3	80.8	67.9	94.8 / 92.9	76.4 / 40.0	76.0
T5_{Large}	85.7 / 85.0	78.0	**84.6**	84.8	84.3	71.6	96.4 / 98.2	80.9 / 46.6	81.2
$\text{BART}_{\text{Large}}$	88.3 / 87.8	60.0	65.4	84.5	84.3	69.0	90.5 / 92.9	81.8 / 48.0	76.0
$\text{RoBERTa}_{\text{Large}}$	89.0 / 88.4	**90.0**	63.5	87.0	**86.1**	**72.6**	96.1 / 94.6	**84.4 / 52.9**	81.5
$\text{GLM}_{\text{RoBERTa}}$	**89.6 / 89.0**	82.0	83.7	**87.7**	84.7	71.2	**98.7 / 98.2**	82.4 / 50.1	**82.9**

Table 2: Results of abstractive summarization on the CNN/DailyMail and XSum test sets.

Model	CNN/DailyMail			XSum		
	RG-1	RG-2	RG-L	RG-1	RG-2	RG-L
BERTSumAbs (Liu and Lapata, 2019)	41.7	19.4	38.8	38.8	16.3	31.2
$\text{UniLMv2}_{\text{Base}}$ (Bao et al., 2020)	43.2	20.4	40.1	44.0	21.1	36.1
T5_{Large} (Raffel et al., 2020)	42.5	20.7	39.8	40.9	17.3	33.0
$\text{BART}_{\text{Large}}$ (Lewis et al., 2019)	**44.2**	**21.3**	**40.9**	45.1	22.3	**37.3**
$\text{GLM}_{\text{RoBERTa}}$	43.8	21.0	40.5	**45.5**	**23.5**	**37.3**

are also shown in Table 1. We observe that with multi-task pretraining, GLM_{Doc} and GLM_{Sent} perform slightly worse than $\text{GLM}_{\text{Large}}$, but still outperform $\text{BERT}_{\text{Large}}$ and $\text{UniLM}_{\text{Large}}$. Among multi-task models, GLM_{Sent} outperforms GLM_{Doc} by 1.1% on average. Increasing GLM_{Doc}'s parameters to 410M ($1.25\times\text{BERT}_{\text{Large}}$) leads to better performance than $\text{GLM}_{\text{Large}}$. GLM with 515M parameters ($1.5\times\text{BERT}_{\text{Large}}$) can perform even better.

Sequence-to-Sequence. Considering the available baseline results, we use the Gigaword dataset (Rush et al., 2015) for abstractive summarization and the SQuAD 1.1 dataset (Rajpurkar et al., 2016) for question generation (Du et al., 2017) as the benchmarks for models pretrained on BookCorpus and Wikipedia. Additionally, we use the CNN/DailyMail (See et al., 2017) and XSum (Narayan et al., 2018) datasets for abstractive summarization as the benchmarks for models pretrained on larger corpora.

The results for models trained on BookCorpus and Wikipedia are shown in Tables 3 and 4. We observe that $\text{GLM}_{\text{Large}}$ can achieve performance matching the other pretraining models on the two generation tasks. GLM_{Sent} can perform better than $\text{GLM}_{\text{Large}}$, while GLM_{Doc} performs slightly worse than $\text{GLM}_{\text{Large}}$. This indicates that the document-level objective, which teaches the model to extend the given contexts, is less helpful to conditional generation, which aims to extract useful information from the context. Increasing GLM_{Doc}'s parameters to 410M leads to the best performance on both tasks. The results for models trained on larger corpora are shown in Table 2. $\text{GLM}_{\text{RoBERTa}}$ can achieve performance matching the seq2seq BART model, and outperform T5 and UniLMv2.

Text Infilling. Text infilling is the task of predicting missing spans of text which are consistent

Table 3: Results on Gigaword summarization.

Model	RG-1	RG-2	RG-L
MASS	37.7	18.5	34.9
$\text{UniLM}_{\text{Large}}$	38.5	19.5	35.8
$\text{GLM}_{\text{Large}}$	38.6	19.7	36.0
GLM_{Doc}	38.5	19.4	35.8
GLM_{Sent}	38.9	20.0	**36.3**
GLM_{410M}	**38.9**	**20.0**	36.2

Table 4: Results on SQuAD question generation.

Model	BLEU-4	MTR	RG-L
SemQG	18.4	22.7	46.7
$\text{UniLM}_{\text{Large}}$	22.1	25.1	**51.1**
$\text{GLM}_{\text{Large}}$	22.4	25.2	50.4
GLM_{Doc}	22.3	25.0	50.2
GLM_{Sent}	22.6	25.4	50.4
GLM_{410M}	**22.9**	**25.6**	50.5

Table 5: BLEU scores on Yahoo text infilling. † indicates the results from (Shen et al., 2020).

Mask ratio	10%	20%	30%	40%	50%
BERT†	82.8	66.3	50.3	37.4	26.2
BLM†	86.5	73.2	59.6	46.8	34.8
$\text{GLM}_{\text{Large}}$	**87.8**	**76.7**	**64.2**	**48.9**	**38.7**
GLM_{Doc}	87.5	76.0	63.2	47.9	37.6

with the surrounding context (Zhu et al., 2019; Donahue et al., 2020; Shen et al., 2020). GLM is trained with an autoregressive blank infilling objective, thus can straightforwardly solve this task. We evaluate GLM on the Yahoo Answers dataset (Yang et al., 2017) and compare it with Blank Language Model (BLM) (Shen et al., 2020), which is a specifically designed model for text infilling. From the results in Table 5, GLM outperforms previous methods by large margins (1.3 to 3.9 BLEU) and achieves the state-of-the-art result on this dataset. We notice that GLM_{Doc} slightly underperforms $\text{GLM}_{\text{Large}}$, which is consistent with our observations in the seq2seq experiments.

Language Modeling. Most language modeling datasets such as WikiText103 are constructed from Wikipedia documents, which our pretraining dataset already contains. Therefore, we evaluate the language modeling perplexity on a held-out test set of our pretraining dataset, which contains about 20M tokens, denoted as BookWiki. We also evaluate GLM on the LAMBADA dataset (Paperno et al., 2016), which tests the ability of systems to model long-range dependencies in text. The task is to predict the final word of a passage. As the baseline, we train a $\text{GPT}_{\text{Large}}$ model (Radford et al., 2018b; Brown et al., 2020) with the same data and tokenization as $\text{GLM}_{\text{Large}}$.

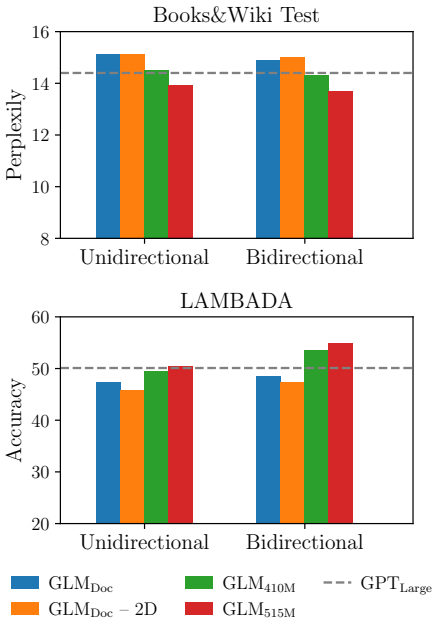

Figure 4: Zero-shot language modeling results.

The results are shown in Figure 4. All the models are evaluated in the zero-shot setting. Since GLM learns the bidirectional attention, we also evaluate GLM under the setting in which the contexts are encoded with bidirectional attention. Without generative objective during pretraining, $\text{GLM}_{\text{Large}}$ cannot complete the language modeling tasks, with perplexity larger than 100. With the same amount of parameters, GLM_{Doc} performs worse than $\text{GPT}_{\text{Large}}$. This is expected since GLM_{Doc} also optimizes the blank infilling objective. Increasing the model's parameters to 410M (1.25× of $\text{GPT}_{\text{Large}}$) leads to a performance close to $\text{GPT}_{\text{Large}}$. GLM_{515M} (1.5× of $\text{GPT}_{\text{Large}}$) can further outperform $\text{GPT}_{\text{Large}}$. With the same amount of parameters, encoding the context with bidirectional attention can improve the performance of language modeling. Under this setting, GLM_{410M} outperforms $\text{GPT}_{\text{Large}}$. This is the advantage of GLM over unidirectional GPT. We also study the contribution of 2D positional encoding to long text generation. We find that removing the 2D positional encoding leads to lower accuracy and higher perplexity in language modeling.

Table 6: Ablation study on the SuperGLUE dev set. (T5 ≈ GLM – shuffle spans + sentinel tokens.)

Model	ReCoRD F1/Acc.	COPA Acc.	WSC Acc.	RTE Acc.	BoolQ Acc.	WiC Acc.	CB F1/Acc.	MultiRC F1a/EM	Avg
$\text{BERT}_{\text{Large}}$	76.3 / 75.6	69.0	64.4	73.6	80.1	**71.0**	94.8 / 92.9	71.9 / 24.1	72.0
$\text{BERT}_{\text{Large}}$ (reproduced)	**82.1 / 81.5**	63.0	63.5	72.2	80.8	68.7	80.9 / 85.7	77.0 / 35.2	71.2
$\text{BERT}_{\text{Large}}$ (cloze)	70.0 / 69.4	**80.0**	76.0	72.6	78.1	70.5	93.5 / 91.1	70.0 / 23.1	73.2
$\text{GLM}_{\text{Large}}$	81.7 / 81.1	76.0	**81.7**	74.0	**82.1**	68.5	**96.1 / 94.6**	77.1 / 36.3	**77.0**
– cloze finetune	81.3 / 80.6	62.0	63.5	66.8	80.5	65.0	89.2 / 91.1	72.3 / 27.9	70.0
– shuffle spans	82.0 / 81.4	61.0	79.8	54.5	65.8	56.3	90.5 / 92.9	76.7 / 37.6	68.5
+ sentinel tokens	81.8 / 81.3	69.0	78.8	**77.3**	81.2	68.0	93.7 / 94.6	**77.5 / 37.7**	76.0

Summary. Above all, we conclude that GLM effectively shares model parameters across natural language understanding and generation tasks, achieving better performance than a standalone BERT, encoder-decoder, or GPT model.

3.4 Ablation Study

Table 6 shows our ablation analysis for GLM. First, to provide an apple-to-apple comparison with BERT, we train a $\text{BERT}_{\text{Large}}$ model with our implementation, data, and hyperparameters (row 2). The performance is slightly worse than the official $\text{BERT}_{\text{Large}}$ and significantly worse than $\text{GLM}_{\text{Large}}$. It confirms the superiority of GLM over Masked LM pretraining on NLU tasks. Second, we show the SuperGLUE performance of GLM finetuned as sequence classifiers (row 5) and BERT with cloze-style finetuning (row 3). Compared to BERT with cloze-style finetuning, GLM benefits from the autoregressive pretraining. Especially on ReCoRD and WSC, where the verbalizer consists of multiple tokens, GLM consistently outperforms BERT. This demonstrates GLM's advantage in handling variable-length blank. Another observation is that the cloze formulation is critical for GLM's performance on NLU tasks. For the large model, cloze-style finetuning can improve the performance by 7 points. Finally, we compare GLM variants with different pretraining designs to understand their importance. Row 6 shows that removing the span shuffling (always predicting the masked spans from left to right) leads to a severe performance drop on SuperGLUE. Row 7 uses different sentinel tokens instead of a single [MASK] token to represent different masked spans. The model performs worse than the standard GLM. We hypothesize that it wastes some modeling capacity to learn the different sentinel tokens which are not used in downstream tasks with only one blank. In Figure 4, we show that removing the second dimension of 2D positional encoding hurts the performance of long text generation.

We note that T5 is pretrained with a similar blank infilling objective. GLM differs in three aspects: (1) GLM consists of a single encoder, (2) GLM shuffles the masked spans, and (3) GLM uses a single [MASK] instead of multiple sentinel tokens. While we cannot directly compare GLM with T5 due to the differences in training data and the number of parameters, the results in Tables 1 and 6 have demonstrated the advantage of GLM.

4 Related Work

Pretrained Language Models. Pretraining large-scale language models significantly improves the performance of downstream tasks. There are three types of pretrained models. First, autoencoding models learn a bidirectional contextualized encoder for natural language understanding via denoising objectives (Devlin et al., 2019; Joshi et al., 2020; Yang et al., 2019; Liu et al., 2019; Lan et al., 2020; Clark et al., 2020). Second, autoregressive models are trained with a left-to-right language modeling objective (Radford et al., 2018a,b; Brown et al., 2020). Third, encoder-decoder models are pretrained for sequence-to-sequence tasks (Song et al., 2019; Lewis et al., 2019; Bi et al., 2020; Zhang et al., 2020).

Among encoder-decoder models, BART (Lewis et al., 2019) conducts NLU tasks by feeding the same input into the encoder and decoder, and taking the final hidden states of the decoder. Instead, T5 (Raffel et al., 2020) formulates most language tasks in the text-to-text framework. However, both models require more parameters to outperform autoencoding models such as RoBERTa (Liu et al., 2019). UniLM (Dong et al., 2019; Bao et al., 2020) unifies three pretraining models under the masked language modeling objective with different attention masks.

NLU as Generation. Previously, pretrained language models complete classification tasks for

NLU with linear classifiers on the learned representations. GPT-2 (Radford et al., 2018b) and GPT-3 (Brown et al., 2020) show that generative language models can complete NLU tasks such as question answering by directly predicting the correct answers without finetuning, given task instructions or a few labeled examples. However, generative models require much more parameters to work due to the limit of unidirectional attention. Recently, PET (Schick and Schütze, 2020a,b) proposes to reformulate input examples as cloze questions with patterns similar to the pretraining corpus in the few-shot setting. It has been shown that combined with gradient-based finetuning, PET can achieve better performance in the few-shot setting than GPT-3 while requiring only 0.1% of its parameters. Similarly, Athiwaratkun et al. (2020) and Paolini et al. (2020) convert structured prediction tasks, such as sequence tagging and relation extraction, to sequence generation tasks.

Blank Language Modeling. Donahue et al. (2020) and Shen et al. (2020) also study blanking infilling models. Different from their work, we pre-train language models with blank infilling objectives and evaluate their performance in downstream NLU and generation tasks.

5 Conclusions

GLM is a general pretraining framework for natural language understanding and generation. We show that the NLU tasks can be formulated as conditional generation tasks, and therefore solvable by autoregressive models. GLM unifies the pretraining objectives for different tasks as autoregressive blank infilling, with mixed attention masks and the novel 2D position encodings. Empirically we show that GLM outperforms previous methods for NLU tasks and can effectively share parameters for different tasks.

Acknowledgements

The work is supported by the NSFC for Distinguished Young Scholar(61825602), and Beijing Academy of Artificial Intelligence (BAAI).

References

Ben Athiwaratkun, Cicero dos Santos, Jason Krone, and Bing Xiang. 2020. Augmented natural language for generative sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 375–385.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, and Hsiao-Wuen Hon. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *ICML 2020*, volume 119, pages 642–652.

Bin Bi, Chenliang Li, Chen Wu, Ming Yan, Wei Wang, Songfang Huang, Fei Huang, and Luo Si. 2020. PALM: Pre-training an Autoencoding&Autoregressive Language Model for Context-conditioned Generation. In *EMNLP 2020*, pages 8681–8691.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS 2020*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR 2020*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL 2019*, pages 4171–4186.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. pages 2492–2501.

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 *NeurIPS 2019*, pages 13042–13054.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to Ask: Neural Question Generation for Reading Comprehension. In *ACL 2017*, pages 1342–1352.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. *ArXiv*, abs/2006.03654.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR 2020*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL 2020*, pages 7871–7880.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. pages 74–81.

Yang Liu and Mirella Lapata. 2019. Text Summarization with Pretrained Encoders. In *EMNLP 2019*, pages 3730–3740.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Joel Mackenzie, Rodger Benham, Matthias Petri, Johanne R. Trippas, J. Shane Culpepper, and Alistair Moffat. 2020. CC-News-En: A Large English News Corpus. In *CIKM 2020*, pages 3077–3084.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *EMNLP 2018*, pages 1797–1807.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2020. Structured Prediction as Translation between Augmented Natural Languages.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *ACL 2016*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A Method for Automatic Evaluation of Machine Translation. In *ACL 2002*, pages 311–318.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018a. Improving Language Understanding by Generative Pre-Training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018b. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *ACL 2018*, pages 784–789.

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

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *KDD 2020*, pages 3505–3506.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP 2015*, pages 379–389.

Timo Schick and Hinrich Schütze. 2020a. Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference. pages 255–269.

Timo Schick and Hinrich Schütze. 2020b. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. pages 2339–2352.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL 2017*, pages 1073–1083.

Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi S. Jaakkola. 2020. Blank language models. pages 5186–5198.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *EMNLP 2013*, pages 1631–1642.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked Sequence to Sequence Pre-training for Language Generation. In *ICML 2019*, volume 97, pages 5926–5936.

Trieu H. Trinh and Quoc V. Le. 2019. A Simple Method for Commonsense Reasoning. *arXiv:1806.02847 [cs]*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *NeurIPS 2019*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *ICLR 2019*, pages 353–355.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *NAACL 2018*, pages 1112–1122.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS 2019*, pages 5754–5764.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *ICML 2017*, volume 70, pages 3881–3890.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. In *ICML 2020*, pages 11328–11339.

Wanrong Zhu, Zhiting Hu, and Eric Xing. 2019. Text infilling. *arXiv preprint arXiv:1901.00158*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV 2015*, pages 19–27.

A Pretraining Setting

A.1 Datasets

To train GLM_{Base} and $\text{GLM}_{\text{Large}}$, we use BookCorpus (Zhu et al., 2015) and Wikipedia used by BERT (Devlin et al., 2019).

To train $\text{GLM}_{\text{RoBERTa}}$, we follow the pretraining datasets of RoBERTa (Liu et al., 2019), which consist of BookCorups (Zhu et al., 2015),Wikipedia (16GB), CC-News (the English portion of the CommonCrawl News dataset[3] 76GB), OpenWebText (web content extracted from URLs shared on Reddit with at least three upvotes(Gokaslan and Cohen, 2019), 38GB) and Stories (subset of CommonCrawl data filtered to match the story-like style of Winograd schemas (Trinh and Le, 2019), 31GB). The Stories dataset is no longer publicly available[4]. Therefore, we remove the Stories dataset and replace OpenWebText with OpenWebText2[5] (66GB). The CC-News dataset is not publicly available and we use the CC-News-en published by (Mackenzie et al., 2020). All the datasets used total 158GB of uncompressed texts, close in size to RoBERTa's 160GB datasets.

A.2 Hyperparameters

The hyperparameters for GLM_{Base} and $\text{GLM}_{\text{Large}}$ are similar to those used by BERT. For trade-off of training speed and fair comparison with BERT (batch size 256 and 1,000,000 training steps), we use batch size of 1024 and 200,000 training steps for $\text{GLM}_{\text{Large}}$. Since GLM_{Base} is smaller, we reduce the number of training steps to 120,000 to speed up pre-training. The hyperparameters for GLM_{Doc} and GLM_{Sent} are the same as those of $\text{GLM}_{\text{Large}}$. The hyperparameters except Transformer architecture for GLM_{410M} and GLM_{515M} are the same as those of $\text{GLM}_{\text{Large}}$. The models are trained on 64 V100 GPUs for 200K steps with batch size of 1024 and maximum sequence length of 512, which takes about 2.5 days for $\text{GLM}_{\text{Large}}$.

To train $\text{GLM}_{\text{RoBERTa}}$, we follow most of the hyperparameters of RoBERTa. The main difference

[3] https://commoncrawl.org/2016/10/news-dataset-available

[4] https://github.com/tensorflow/models/tree/archive/research/lm_commonsense#1-download-data-files

[5] https://openwebtext2.readthedocs.io/en/latest

Table 7: Hyperparameters for pretraining

Hyperparameters	GLM $_{\text{Base}}$	GLM $_{\text{Large}}$	GLM $_{\text{RoBERTa}}$
Number of Layers	12	24	24
Hidden size	768	1024	1024
FFN inner hidden size	3072	4096	4096
Attention heads	12	16	16
Attention head size	64	64	64
Dropout	0.1	0.1	0.1
Attention Dropout	0.1	0.1	0.1
Warmup Steps	6k	8k	30K
Peak Learning Rate	4e-4	2e-4	4e-4
Batch Size	1024	1024	8192
Weight Decay	0.1	0.1	0.01
Max Steps	120k	200k	250k
Learning Rate Decay	Cosine	Cosine	Cosine
Adam ϵ	1e-6	1e-6	1e-6
Adam β_1	0.9	0.9	0.9
Adam β_2	0.98	0.98	0.98
Gradient Clipping	1.0	1.0	1.0

includes: (1) Due to resource limit, we only pretrain GLM $_{\text{RoBERTa}}$ for 250,000 steps, which are half of RoBERTa and BART's training steps, and close to T5 in number of trained tokens. (2) We use cosine decay instead of linear decay for learning rate scheduling (3) We additionally apply gradient clipping with value 1.0.

The hyperparameters for all the pre-training settings are summarized in Table 7.

A.3 Implementation

Our pretraining implementation is based on Megatron-LM (Shoeybi et al., 2019) and DeepSpeed (Rasley et al., 2020). We include our code in the supplementary material. Due to the size limit of supplementary material, we cannot include the pretrained models, but will make them public available in the future.

B Downstream Tasks

B.1 SuperGLUE

The SuperGLUE benchmark consists of 8 NLU tasks. We formulate them as blank infilling tasks, following (Schick and Schütze, 2020b). Table 8 shows the cloze questions and verbalizers we used in our experiments. For 3 tasks (ReCoRD, COPA, and WSC), the answer may consist of multiple tokens, and for the other 5 tasks, the answer is always a single token.

When finetuning GLM on the SuperGLUE tasks, we construct the input using the cloze questions in Table 8 and replace the blank with a [MASK] token. Then we compute the score of generating each answer candidate. For the 5 single-token tasks, the score is defined to be the logit of the verbalizer token. For the 3 multi-token tasks, we use the sum of the log-probabilities of the verbalizer tokens. Thanks to the autoregressive blank infilling mechanism we proposed, we can obtain all the log-probabilities in one pass. Then we compute the cross entropy loss using the groundtruth label and update the model parameters.

For the baseline classifiers, we follow the standard practice to concatenate the input parts of each task (such as the premise and hypothesis for textual entailment, or the passage, question and answer for ReCORD and MultiRC) and add a classification layer on top of the [CLS] token representation. We also implemented cloze-style finetuning for the other pre-trained models, but the performance was usually similar to the standard classifier, as we shown in the ablation study. Models with blank-infilling objectives, such as T5 and our GLM, benefits more from converting the NLU tasks into cloze questions. Thus for T5 and GLM, we report the performance after such conversion in our main results.

Table 8: Cloze questions and verbalizers for the 8 SuperGLUE tasks used in our experiments. * denotes the answer contains multiple tokens.

Dataset	Task	Cloze Question	Verbalizers
ReCoRD*	Question answering	[passage p] [cloze question q]	Answer candidates
COPA*	Causal reasoning	"[choice c_1]" or "[choice c_2]"? [premise p], so ___.	c_1 / c_2
WSC*	Coreference resolution	[sentence s] The pronoun '$*p*$' refers to ___.	Noun n
RTE	Textual entailment	"[hypothesis h]"? | ___, "[premise p]"	"yes" (entailment), "no" (not entailment)
BoolQ	Question answering	[passage p]. Question: q? Answer: ___.	"yes" / "no"
WiC	Word sense disambiguation	"[sentence s_1]" / "[sentence s_2]" Similar sense of [word w]? ___.	"yes" / "no"
CB	Textual entailment	"[hypothesis h]"? | ___, "[premise p]"	"yes" (entailment), "no" (contradiction), "maybe" (neutral)
MultiRC	Question answering	[passage p]. Question: q? Is it [answer a]? ___.	"yes" / "no"

B.2 Sequence-to-Sequence

Fot the text summarization task, we use the dataset Gigaword (Rush et al., 2015) for model fine-tuning and evaluation. We finetune $\text{GLM}_{\text{LARGE}}$ on the training set for 4 epochs with AdamW optimizer. The learning rate has a peak value of 3e-5, warm-up over the 6% training steps and a linear decay. We also use label smoothing with rate 0.1 (Pereyra et al., 2017). The maximum document length is 192 and the maximum summary length is 32. During decoding, we use beam search with beam size of 5 and remove repeated trigrams. We tweak the value of length penalty on the development set. The evaluation metrics are the F1 scores of Rouge-1, Rouge-2, and Rouge-L (Lin, 2004) on the test set.

For the question generation task, we use the SQuAD 1.1 dataset (Rajpurkar et al., 2016) and follow the dataset split of (Du et al., 2017). The optimizer hyperparameters are the same as those of abstractive summarization. The maximum passage length is 464 and the maximum question length is 48. During decoding, we use beam search with beam size 5 and tweak the value of length penalty on the development set. The evaluation metrics are the scores of BLEU-1, BLEU-2, BLEU-3, BLEU-4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and Rouge-L (Lin, 2004).

Results of T5_{Large} on XSum are obtained by running the summarization script provided by Huggingface transformers[6]. All the other results of baselines on seq2seq tasks are obtained from the corresponding papers.

[6] https://github.com/huggingface/transformers/tree/master/examples/pytorch/summarization

B.3 Text Infilling

We follow (Shen et al., 2020) and evaluate text infilling performance on the Yahoo Answers dataset (Yang et al., 2017), which contains 100K/10K/10K documents for train/valid/test respectively. The average document length is 78 words. To construct the text infilling task, we randomly mask a given ratio $r \in \{10\% \cdots 50\%\}$ of each document's tokens and the contiguous masked tokens are collapsed into a single blank. We finetune $\text{GLM}_{\text{Large}}$ on the training set for 5 epochs with dynamic masking, i.e. the blanks are randomly generated at training time. Similar to the sequence-to-sequence experiments, we use an AdamW optimizer with a peak learning rate 1e-5 and 6% warm-up linear scheduler.

For comparison with previous work, we use the same test set constructed by (Shen et al., 2020). The evaluation metric is the BLEU score of the infilled text against the original document. We compare with two baselines: (1) BERT, which learns a left-to-right language model to generate the masked tokens on top of the blank representation, and (2) BLM proposed by (Shen et al., 2020), which can fill in the blank with arbitrary trajectories.

B.4 Language Modeling

We evaluate the model's ability of language modeling with perplexity on BookWiki and accuracy on the LAMBDA dataset (Paperno et al., 2016).

Perplexity is an evaluation criterion that has been

well studied for language modeling. Perplexity is the exponentiation of the average cross entropy of a corpus.

$$\text{PPL} = \exp(-\frac{1}{T}\sum_{t=1}^{T} p(x_t|\boldsymbol{x}_{<t})) \quad (4)$$

where $\boldsymbol{x}_{<t} = [x_0, \cdots, x_{t-1}]$. Since transformers can only operate on a window of fixed input size w, we cannot fully calculate $p(x_t|\boldsymbol{x}_{<t})$ and can only calculate $p(x_t|\boldsymbol{x}_{t-w:t-1})$. Even calculating this value for each token is prohibitively expensive, since we need to conduct T evaluations of w-size contexts. To improve evaluation efficiency, we adopt *overlapping evaluation*, where we advance the sliding windows by some overlap o each time and only compute the cross entropy loss for the last o tokens of the window. In our experiments we set $o = 256$ for all the models.

LAMBDA is a cloze-style dataset to test the ability of long-range dependency modeling. Each example is a passage consisting of 4-5 sentences with the last word missing and the model is required to predict the last word of the passage. Since we use WordPiece tokenization, a word can be split into several subword units. We use teacher forcing and consider the prediction correct only when all the predicted tokens are correct.

C Results on Other NLU Benchmarks

GLUE (Wang et al., 2018) is another widely-used NLU benchmark, including single sentence tasks (e.g. sentiment analysis (Socher et al., 2013)) and sentence pair tasks (e.g. text similarity (Cer et al., 2017) and natural language inference (Williams et al., 2018; Dagan et al., 2005)). The benchmark is usually considered as less challenging than SuperGLUE. SQuAD (Rajpurkar et al., 2016, 2018) is an extractive question answering benchmark. We further compare GLM with BERT on the two benchmarks.

The results on GLUE and SQuAD are shown in Tables 9 and 10. On the two benchmarks, GLM can still outperform BERT with the same amount of parameters, but with a smaller margin.

D Text Generation Samples

We show texts generated by GLM_{Doc} given unseen contexts randomly sampled from the test set. We use top-k random sampling with $k = 40$ for generation and set maximum sequence length to 512. Some of the texts are cut short.

Example D.1. The Wyoming State Legislature is the legislative branch of the U.S. State of Wyoming.

GLM: It consists of a state senate and a house of representatives. As of 2019, state senators are Steve Kish, Joe L. Williams, and Linda S. Bollens. Members of the Wyoming State Legislature are elected from single-member districts representing the majority of the state. The current state senate members are: In recent years, there have been four changes to the senate. The most recent is the creation of a six-seat district that includes all or part of the following: In the 2009 elections, the state senate members were elected to six-year terms. The current state house members are: The Wyoming Constitution assigns certain powers to the governor. Most notably, the governor is president of the senate and governor. However, if the governor desires to appoint a member to the Wyoming state senate, a law authorizes the governor to do so. The governor of Wyoming holds no legislative power but has the power to veto lawmakers, which is not limited to the veto of laws. Under the wyoming state constitution, the governor can veto the actions of the other members of the wyoming house of representatives. The governor can also appoint members of the wyoming senate. In addition, the governor can appoint members of the Wyoming house of representatives. Wyoming's constitution provides that the governor can appoint a member of the wyoming state senate to the wyoming supreme court, and the chairman of the wyoming senate.

Example D.2. Jonathan Terry is a television and film actor.

GLM: He was a voice actor for the "X-Men" cartoon series. He was also a voice actor for "the Simpsons" as well as "the marvelous misadventures of superman. He also has voiced characters in "the legend of korra" He has appeared on several television series, including "the simpsons", "the x-files", "heroes" and "the simpsons movie" as the character captain billy higgledy-pig, and the web series "krusty mysteries" as the character Colonel Trungus. He wrote for "the Guardian" newspaper. Jonathan Terry was born in London. Terry grew up in Surrey, England and attended the University of Sussex in the United Kingdom, graduating with a degree in english literature. He was a guest lecturer at King's College London, and then took two years of acting courses at the brit school of acting to prepare for his future career in the entertainment industry. Terry first appeared in the TV series "the

Table 9: Results on the GLUE dev set.

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	Avg
$\text{BERT}_{\text{Large}}$	86.6	92.3	91.3	73.6	93.2	88.0	60.6	90.0	84.4
$\text{GLM}_{\text{Large}}$	**86.7**	**92.8**	**91.5**	**74.0**	**93.5**	**90.0**	**61.4**	**90.7**	**85.1**

Table 10: Results on the SQuAD v1.1/v2.0 dev sets.

Model	SQuAD v1.1 EM/F1	SQuAD v2.0 EM/F1
$\text{BERT}_{\text{Base}}$	80.8 / 88.5	73.7 / 76.3
GLM_{Base}	**81.5 / 88.6**	**74.7 / 77.8**
$\text{BERT}_{\text{Large}}$	84.1 / 90.9	79.0 / 81.8
$\text{GLM}_{\text{Large}}$	**85.4 / 91.6**	**80.3 / 83.3**

Simpsons" as the character captain Billy Higgledy-pig, but his character was only a one-time recurring character in the series' first six seasons. He later appeared as a regular for the show's final six seasons, and has been a frequent guest in the show since. He appeared in the first few episodes of "" as the character major Jack Ryan. He has also appeared as part of the supporting cast of several episodes of "the secret life of pets". He has also worked on "the simpsons" TV show since "the simpsons movie", most notably playing the roles of Captain Skeletor and the ghost of the same name. He plays characters in several films, including "", "", "" and "". He has appeared in music videos for the killers in 1993, the pretenders in 1995, and in the TV shows "the royal" and "the bill".

Example D.3. Corona was a station along the port Washington branch of the long island rail road in the Corona section of queens, New York City. It was one of two stations built by the flushing railroad in Corona, this one having been at Grand Avenue (later called National Avenue, now National Street) and 45th Avenue.

GLM: It had a brick entrance building next to the tracks, and one platform with 2 side platforms. The station burned down and was rebuilt in 1908. The station closed on september 15, 1927, with the train service transferred from Grand Avenue to 45th Avenue. There were plans to build a subway extension to Corona, but it was never built. In the 1980s, two stations were constructed on the line, Corona Road and Corona Park. The new Corona station opened in 1988, and the original Corona station was demolished.

Example D.4. Robert Lee Smith (born July 5, 1938) is a former American football cornerback in the national football league. He played for the Los Angeles Rams (1962–1965) and the Detroit Lions (1965–1966). He was also a kick and punt returner.

GLM: In his four-year NFL career, he played in 33 games and started 14, registering 62 career interceptions. He completed his NFL career with five interceptions. Smith was born in La Canada Flintridge, Michigan, in 1938. He attended Bishop O' Dowd high school in Flintridge. Smith' s father was a pharmacist. He graduated from Michigan State University in 1958 with a degree in business administration. Smith played for the Michigan Wolverines football team from 1959 to 1963. In his two years as a reserve cornerback, he led the conference in interceptions with five. In 1962, he set the Wolverines' all-time interception record with 13, and was second overall in the 1962 season's Heisman Trophy voting. Smith also won the Dick Butkus award as the nation's outstanding linebacker. In 1961, the "Los Angeles Times" wrote that Smith "is an outstanding pass rusher, with an average of almost 100 yards per punt return." Smith was inducted into the university of Michigan athletic hall of honor in 1989 and the national football foundation hall of fame in 1991. He was elected to the Michigan sports hall of fame in 1995. Smith earned the honor because of his accomplishments prior to his NFL career. He was one of four Michigan players honored as first-overall selections in the 1964 NFL draft. The others were Joe Namath, Bill Nelsen, and Jerry Kramer. In 1966, the NFL gave players $300,000 a season to play football. After his rookie season, he was not selected to play in the 1966 pro bowl. On January 13, 1966, the Rams traded smith to the Detroit Lions for Paul Hornung, and later that year he was traded to the Lions for Ray "the Lion" Jones in exchange for Linebacker Jim "the Hawk" Johnson. On September 10, 1968, he was traded back to Los Angeles for a second round pick in the 1970 draft. He was also traded to the St. Louis Cardinals for

a second round pick in the 1970 draft. On June 2, 1970 he was cut by the Cardinals. On November 15, 1970, the Los Angeles Rams acquired Smith from the Lions in exchange for Linebacker Tony Harris. The Rams waived Smith during the September 1, 1972 offseason. Smith's number at Michigan State was # 7 in 1969.