GLM: General Language Model Pretraining with Autoregressive Blank Infilling

Zhengxiao Du[1], **Yujie Qian**[2], **Xiao Liu**[1], **Ming Ding**[1], **Jiezhong Qiu**[1], **Zhilin Yang**[3], **Jie Tang**[1]
[1]Tsinghua University [2]Massachusetts Institute of Technology [3]Recurrent AI
{zx-du20,liuxiao17,dm18,qiujz16}mails.tsinghua.edu.cn;
yujieq@csail.mit.edu;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). On the other hand, NLP tasks differ in nature, with three main categories being natural language understanding (NLU), unconditional generation, and conditional generation, while none of the pretraining frameworks performs the best for all tasks. We propose a General Language Model (GLM) based on autoregressive blank infilling to address this challenge. The proposed architecture has two major benefits: (1) It improves pretrain-finetune consistency via cloze-style finetuning and naturally handles variable-length blank infilling which is crucial for many downstream tasks. Empirically, GLM substantially outperforms BERT on the SuperGLUE natural language understanding benchmark with the same amount of pretraining data and steps. (2) It is flexible enough to handle various NLP tasks with a single pretrained model. GLM with $1.25\times$ parameters of $\text{BERT}_{\text{Large}}$ achieves the best performance in NLU, conditional and unconditional generation at the same time, demonstrating its generalizability to different downstream tasks.

1 Introduction

Large-scale language models pretrained on web texts have substantially advanced the state of the art in various NLP tasks, ranging from natural language understanding (NLU) to text generation [26, 8, 40, 27, 28, 19, 4]. Downstream task performance as well as the scale of the parameters have also constantly increased in the past few years.

In general, existing pretraining frameworks can be categorized into three families: *autoregressive*, *autoencoding*, and *encoder-decoder* models. Autoregressive models, such as GPT [26], learn left-to-right language models. While they have succeeded in long-text generation and shown strong few-shot learning ability when scaled to billions of parameters [27, 4], the inherent disadvantage is that the unidirectional attention mechanism cannot fully capture the dependencies between the context words. Autoencoding models, such as BERT [8], learn bidirectional Transformers as context encoders via denoising objectives like Masked Language Model (MLM). These encoders generate contextualized representations that excel at 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 to connect them [36, 3, 19]. They are typically deployed in tasks of conditional text generation, also called seq2seq, such as text summarization and response generation. T5 [28] unifies NLU and conditional generation via encoder-decoder models but requires more parameters to outperform BRET-based models such as RoBERTa [20]. Table 1 compares different pretraining frameworks.

None of these pretraining frameworks is flexible enough to handle all NLP tasks. Previous works have tried to unify different frameworks by combining their objectives via multi-task learning [10, 2].

Preprint. Under review.

Table 1: Summary of the pretraining frameworks and downstream tasks[1]. "✓" means "is good at", "–" means "can be adapted to", and "×" means "cannot be directly applied to".

Downstream Task	Autoreg.	Autoenc.	Enc.-Dec.	GLM	Example
NLU	—	✓	—	✓	Sentiment Classification
Conditional Generation	—	×	✓	✓	Text Summarization
Unconditional Generation	✓	×	—	✓	Language Modeling

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 general 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. To learn both bidirectional and unidirectional attention mechanisms in a unified framework, we divide the input text into two parts, where the unmasked tokens can attend to each other, but the masked tokens cannot attend to subsequent masked tokens. We also propose a 2D positional encoding technique to indicate the inter- and intra- span position information. Figure 1 illustrates our pretraining framework. As a result, GLM learns both contextual representation and autoregressive generation during pretraining.

When finetuning on downstream tasks, we reformulate them as blank infilling generation, inspired by [32, 33]. Each task is associated with a human-crafted cloze question, and the model predicts the answer to the cloze. For example, a sentiment classification task is reformulated as a cloze question "[SENTENCE]. It's really __". The choice of words "good" or "bad" in the blank indicates the sentiment being positive or negative. With such formulation, GLM benefits from the consistency between pretraining and finetuning, because they both train the model to generate masked spans given their contexts. As a result, GLM is more suitable for downstream NLU tasks compared to autoencoding models such as BERT. To make our pretraining method better suited for text generation, we also study a multi-task pretraining setup, where the model is jointly trained to reconstruct masked spans and generate longer text.

Empirically, we show that with the same pretraining data and a close amount of computational cost, GLM significantly outperforms BERT on the SuperGLUE benchmark by a large margin of 4.6% – 5.0%. GLM also outperforms RoBERTa, T5, and BART when pretrained on a corpus of similar size (158GB). Moreover, compared with standalone baselines, GLM with multi-task pretraining achieves improvements in natural language understanding, conditional text generation, and language modeling tasks all together 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

[1]We define unconditional generation as the task of generating text without further training as in a standard language model, while conditional generation refers to sequence-to-sequence tasks.

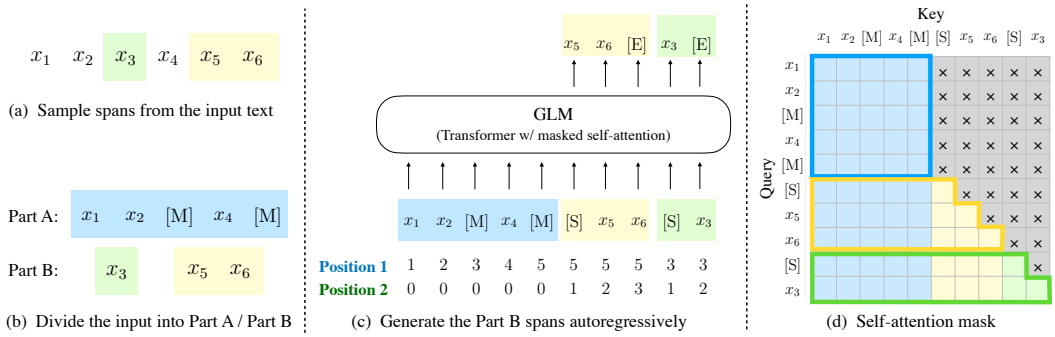

Figure 1: GLM pretraining framework. (a) The original text is $[x_1, x_2, x_3, x_4, x_5, x_6]$, and two spans $[x_3]$ and $[x_5, x_6]$ are sampled. (b) Replace the sampled spans with [M] tokens to form Part A, and shuffle the sampled spans to form Part B. (c) GLM is trained to generate Part B autoregressively. Each span is prepended with a [S] token as input and appended with a [E] token as output. 2D positional encoding represents inter- and intra-span positions. (d) Self-attention mask, where the grey areas are masked out. Part A tokens can attend to themselves (the blue frame) but not B. Part B tokens can attend to A and their antecedent tokens in B (the yellow and green frames denote the tokens to which the two spans in Part B can attend). [M] := [MASK], [S] := [START], and [E] := [END].

to the permutation language model [40]. 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

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

It differs from SpanBERT [15] in the sense that the number of missing tokens in a span is unknown to the model. GLM predicts the missing tokens in an autoregressive manner. 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:

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

Specifically, we implement the autoregressive blank infilling objective with the following techniques. The input $\boldsymbol{x}$ is divided into two parts: Part A consists of the corrupted text $\boldsymbol{x}_{\text{corrupt}}$, and Part B consists of the masked spans. Tokens in Part A can attend to all the tokens in A, but cannot attend to any tokens in B. Tokens in Part B can attend to tokens in A and its antecedents in B, but cannot attend to any subsequent tokens in B. To enable autoregressive generation, each span is padded with two special tokens [START] and [END] at the beginning and the end, for model input and 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 1.

Previous pretraining works have shown optimal performance when masking 15% of the original tokens [8, 40, 28]. In this work, 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 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 that covers 50%–100% of the original tokens, whose length is sampled from a uniform distribution. The span is masked with a [MASK] token and the model autoregressively generates it.

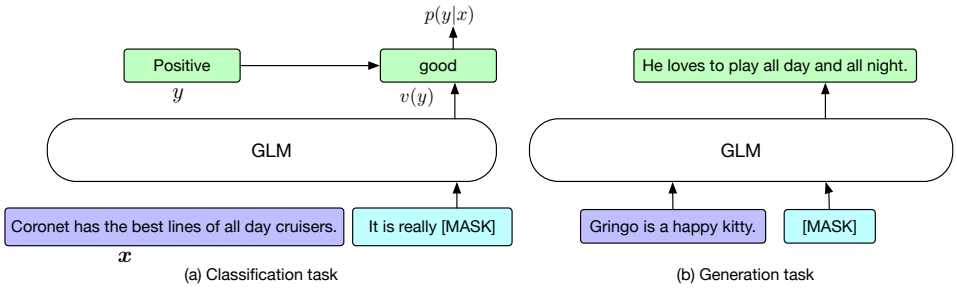

Figure 2: GLM finetuning. (a) Formulation of the sentiment classification task as blank infilling with GLM. (b) GLM for text generation given the context. This can be the language modeling, or sequence-to-sequence tasks with finetuning.

- Sentence-level. We restrict that the masked spans must be full sentences. Multiple spans (sentences) are sampled to cover 15% of the original tokens.

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 Transformer encoder 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 [35]; (2) we use a single linear layer for the output token prediction; (3) we replace ReLU activation functions with GeLUs [14].

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 reconstructing them. It is an important difference as compared to other models, e.g. XLNet [40] encodes the original position so that it can perceive the number of missing tokens, SpanBERT [15] 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 produced by pretrained models as input and predicts the correct answer. The practices are different from the pretraining task of blank filling, leading to inconsistency between pretraining and finetuning.

Instead, we reformulate NLU classification tasks as generation tasks of blank infilling, following PET [33]. 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 [MASK]. 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 label y is also mapped to an answer to the cloze, called verbalizer $v(y)$. In sentiment classification, the labels "positive" or "negative" are mapped to words "good" or "bad" in the blank. The probability of the sentence being positive or negative is proportional

to predicting "good" or "bad" in the blank. Therefore, the conditional probability of 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. Then we finetune GLM with the cross entropy loss.

For text generation tasks, we directly apply GLM as an autoregressive model. 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. The finetuning method is illustrated in Figure 2.

2.4 Discussion and Analysis

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

Comparison with BERT [8]. BERT is pretrained with an autoencoding objective, i.e. masked language model. Since BERT predicts the masked tokens independently, it fails to capture the interdependencies of masked tokens [40]. Another disadvantage of BERT is that it cannot fill in 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 [40]. Both GLM and XLNet are pretrained with autoregressive objectives. There are two differences between GLM and XLNet. First, XLNet uses the original position encoding 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 right-shift, to avoid the information leak within Transformer. It doubles the time cost of pretraining.

Comparison with T5 [28]. T5 proposes a similar blank infilling objective to pretrain an encoder-decoder Transformer. Instead, GLM uses a single Transformer encoder model to learn both bidirectional and unidirectional attention. By sharing parameters for the two types of attention, GLM is more parameter-efficient than the encoder-decoder architecture. Besides, 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. We empirically analyze the difference with T5 in Section 3.4.

Comparison with UniLM [10, 2] UniLM combines different pretraining objectives under the autoencoding framework by changing the attention mask among bidirectional, unidirectional, and cross attention. Compared with autoregressive models, the model cannot fully capture the current token's dependencies with previous tokens due to the independence assumption of autoencoding models. Finetuning UniLM on downstream generation tasks also relies on masked language modeling, which is less efficient than autoregressive models. UniLMv2 [2] 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 conduct experiments in two settings. In Section 3.2, we pretrain GLM with only the short span objective and evaluate it on the NLU tasks. In Section 3.3, we explore multi-task pretraining as discussed in Section 2.1.2.

3.1 Pretraining Setup

For a fair comparison with BERT [8], we use BooksCorpus [46] 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. 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 $\text{GLM}_{\text{Large}}$.

Table 2: 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**

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 [20], denoted as $\text{GLM}_{\text{RoBERTa}}$.[2] 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 the appendix.

3.2 SuperGLUE

To evaluate our pretrained GLM models, we conduct experiments on the SuperGLUE benchmark [39]. SuperGLUE consists of 8 challenging natural language understanding tasks, including question answering [5, 16, 43], textual entailment [7, 5], coreference resolution [18], word sense disambiguation [25], and causal reasoning [30]. We adopt the standard evaluation metrics as [39]. We reformulate each task as a blank infilling task with human-crafted cloze questions, using the patterns constructed by PET [32]. 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 the appendix.

For 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 to our models. 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 $\text{BERT}_{\text{Large}}$.

The results are shown in Table 2. With the same amount of training data (BookCorpus + Wikipedia), GLM consistently outperforms BERT on most of the tasks, with either base or large architecture. The only exception is WiC, which is a word sense disambiguation task. 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

[2]The STORIES dataset [37] used by RoBERTa is no longer available. Therefore, we replace OpenWebText (38GB) [13] with OpenWebText2 (66GB) [12]. The whole corpus totals 158GB of uncompressed text, close in size to RoBERTa's 160GB corpus.

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

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. During multi-task pretraining, as described in Section 2.1, within one training batch, 50% of the time we sample the BERT-style spans and 50% of the time we sample the generation spans, which can be document-level or sentence-level depending on the objective. 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 are also shown in Table 2. 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. We use abstractive summarization and question generation as the evaluation tasks. Abstractive summarization aims to produce a concise and fluent summary conveying the key information in the input text. We use the Gigaword dataset [31] for model fine-tuning and evaluation. We use MASS [36] and UniLM [10] as our baselines since BART [19] and PALM [3] are trained with much more data and steps. Question generation aims to generate the corresponding question given an input passage and an answer. This was first proposed by [11] to enhance the reading comprehension models. We use the SQuAD 1.1 dataset [29] and follow the dataset split of [11]. During decoding, we use beam search with beam size 5 and tweak the value of length penalty on the development set. We use SemQG [44] and UniLM as our baselines.

The results 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.

Text Infilling. Text infilling is the task of predicting missing spans of text which are consistent with the surrounding context [45, 9, 34]. GLM is trained with an autoregressive blank infilling objective, thus can be straightforwardly applied to this task. Following [34], we evaluate GLM on the Yahoo Answers dataset [41]. For each document in the dataset, we randomly mask a given ratio $r \in \{10\% \cdots 50\%\}$ of its tokens and contiguous masked tokens are collapsed into a single blank. We finetune GLM to autoregressively generate the masked tokens and evaluate the generation's quality by its BLEU score against the original document. We compare with BERT and the Blank Language Model (BLM) [34] which is specifically designed for the text infilling task. 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 sequence-to-sequence experiments.

Language Modeling. We evaluate GLM's performance on language modeling. Most language modeling datasets such as WikiText103 are constructed from Wikipedia documents, which our

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

Table 5: BLEU scores on Yahoo text infilling. † indicates the results from [34].

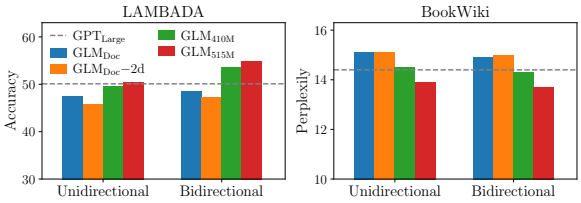

Figure 3: Zero-shot language modeling results.

pretraining dataset already contains. For fair comparison with BERT, we cannot remove Wikipedia from the pretraining dataset. 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 on the LAMBADA dataset [23], 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 [27, 4] with the same data and tokenization as $\text{GLM}_{\text{Large}}$.

The results are shown in Figure 3. 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 ($1.25\times$ of $\text{GPT}_{\text{Large}}$) leads to a performance close to $\text{GPT}_{\text{Large}}$. GLM_{515M} ($1.5\times$ 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.

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 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 and exclude the effect of implementation, data, and hyperparameters, we train a $\text{BERT}_{\text{Large}}$ model using the Masked LM objective with our code (row 2). The performance is slightly worse than Google's $\text{BERT}_{\text{Large}}$ and significantly worse than $\text{GLM}_{\text{Large}}$. It confirms the superiority of GLM over Masked LM pretraining on NLU tasks. Second, we perform an ablation study to understand the importance of autoregressive pretraining, and the formulation of classification tasks as blank infilling. 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 outperform 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 also 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 is not used in downstream tasks with only one blank. In Figure 3, 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 2 and 6 have demonstrated the advantage of GLM.

Table 6: Ablation study on the SuperGLUE dev set. T5 $\approx$ 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

4 Related Work

Pretrained Language Models. In NLP, self-supervised learning has long been used to learn word vectors as inputs to neural networks [21, 24]. Recently, pretraining large-scale language models with self-supervised learning on abundant web texts significantly improves the performance on downstream tasks [8, 17, 20, 36, 4, 28, 6, 42]. There are three types of pretrained language models. The first type is the autoencoding model, which learns a bidirectional contextualized encoder for natural language understanding via denoising objectives. BERT [8] pretrains a large transformer model [38] via masked language modeling to obtain contextualized word representations. SpanBERT [15] masks continuous spans of tokens for improved span representations. The second type is the autoregressive model, which learns a left-to-right language model for text generation. GPT [26] shows that the representations learned by generative pretraining can also improve language understanding. XLNet [40] generalizes the autoregressive model with permutation language modeling to learn bidirectional attention for language understanding tasks. The third type is the encoder-decoder model pretrained for sequence-to-sequence tasks. MASS [36] maps an input text with continuous spans masked to the masked tokens. BART [19] applies various transformations, including masking, deletion, and permutation, and recovers the original text with the decoder. PALM [3] is pretrained for generating coherent text from a given context and adds a BERT-based autoencoding objective to the encoder.

NLU as Generation. Previously, pretrained language models complete classification tasks for NLU with linear classifiers on the learned representations. GPT-2 [27] shows that generative language models can learn to complete understanding tasks such as question answering and reading comprehension by directly predicting the correct answers, even without any explicit supervision. GPT-3 [4] further proves that language models can achieve strong performance on NLU tasks in the few-shot learning setting by adding a few labeled examples in the context. However generative models require much more parameters to work due to the limit of unidirectional attention. T5 [28] formulates most language tasks in the text-to-text framework but requires more parameters to outperform BRET-based models such as RoBERTa [20].

Recently, PET [33, 32] 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 on ALBERT [17], PET can achieve better performance in the few-shot setting than GPT-3 while requiring only 0.1% of its parameters. Athiwaratkun et al. [1] and Paolini et al. [22] convert structured prediction tasks, such as sequence tagging and relation extraction, to sequence generation tasks. Donahue et al. [9] and Shen et al. [34] also study blank infilling language models. Different from their work, we pretrain language models by blank infilling and evaluate their performance in downstream NLU and generation tasks.

5 Conclusion

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 mask 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. In the future, we hope to scale GLM to larger Transformer models and more data, and examine its performance in more settings such as knowledge probing and few-shot learning.

References

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

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

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

[4] 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. Language Models are Few-Shot Learners. In *NeurIPS 2020*, 2020.

[5] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, 2019.

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

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

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

[9] Chris Donahue, Mina Lee, and Percy Liang. Enabling language models to fill in the blanks. *arXiv preprint arXiv:2005.05339*, 2020.

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

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

[12] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[13] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

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

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

[16] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, 2018.

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

[18] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer, 2012.

[19] 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. In *ACL 2020*, pages 7871–7880, 2019.

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

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*, pages 3111–3119, 2013.

[22] Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. Structured prediction as translation between augmented natural languages. *arXiv preprint arXiv:2101.05779*, 2021.

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

[24] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *EMNLP 2014*, pages 1532–1543, 2014.

[25] Mohammad Taher Pilehvar and Jose Camacho-Collados. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, 2019.

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

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

[28] 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. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[29] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *EMNLP 2016*, pages 2383–2392, 2016.

[30] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, pages 90–95, 2011.

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

[32] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. *CoRR*, abs/2009.07118, 2020.

[33] Timo Schick and Hinrich Schütze. Exploiting cloze questions for few-shot text classification and natural language inference. *CoRR*, abs/2001.07676, 2020.

[34] Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi Jaakkola. Blank language models. *arXiv preprint arXiv:2002.03079*, 2020.

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

[36] 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*, volume 97, pages 5926–5936, 2019.

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

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS 2017*, pages 5999–6009, 2017.

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

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

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

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

[43] Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*, 2018.

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

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

[46] Yukun Zhu, Ryan Kiros, Richard S. 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 2015*, pages 19–27, 2015.