UniLMv2: Pseudo-Masked Language Models for Unified Language Model Pre-Training

Hangbo Bao[*] [1] **Li Dong**[2] **Furu Wei**[2] **Wenhui Wang**[2] **Nan Yang**[2] **Xiaodong Liu**[2] **Yu Wang**[2] **Songhao Piao**[1]
Jianfeng Gao[2] **Ming Zhou**[2] **Hsiao-Wuen Hon**[2]

Abstract

We propose to pre-train a unified language model for both autoencoding and partially autoregressive language modeling tasks using a novel training procedure, referred to as a pseudo-masked language model (PMLM). Given an input text with masked tokens, we rely on conventional masks to learn inter-relations between corrupted tokens and context via autoencoding, and pseudo masks to learn intra-relations between masked spans via partially autoregressive modeling. With well-designed position embeddings and self-attention masks, the context encodings are reused to avoid redundant computation. Moreover, conventional masks used for autoencoding provide global masking information, so that all the position embeddings are accessible in partially autoregressive language modeling. In addition, the two tasks pre-train a unified language model as a bidirectional encoder and a sequence-to-sequence decoder, respectively. Our experiments show that the unified language models pre-trained using PMLM achieve new state-of-the-art results on a wide range of language understanding and generation tasks across several widely used benchmarks. The code and pre-trained models are available at https://github.com/microsoft/unilm.

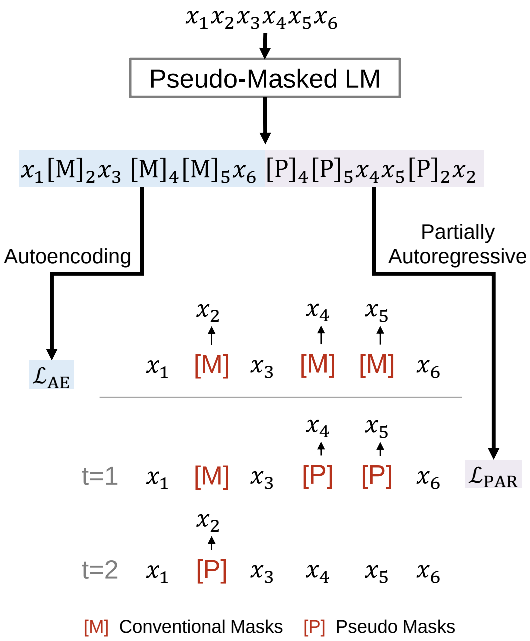

Figure 1. Given input $x_1 \cdots x_6$, the tokens x_2, x_4, x_5 are masked by the special tokens `[M]` and `[P]`. For each example, we jointly train two types of LMs, namely, autoencoding (AE), and partially autoregressive (PAR) masked LMs.

1. Introduction

Language model (LM) pre-training on large-scale text corpora has substantially advanced the state of the art across a variety of natural language processing tasks (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018; Dong et al., 2019; Liu et al., 2019; Yang et al., 2019; Lewis et al., 2019; Lan et al., 2019; Raffel et al., 2019; Chi et al., 2020). After LM pre-training, the obtained model can be fine-tuned to various downstream tasks.

Two types of language model pre-training objectives are commonly employed to learn contextualized text representations by predicting words conditioned on their context. The first strand of work relies on autoencoding LMs (Devlin et al., 2018; Liu et al., 2019). For example, the masked language modeling task used by BERT (Devlin et al., 2018) randomly masks some tokens in a text sequence, and then independently recovers the masked tokens by conditioning on the encoding vectors obtained by a bidirectional Transformer (Vaswani et al., 2017). The second type of pre-training uses autoregressive modeling (Radford et al., 2018; Lewis et al., 2019; Yang et al., 2019; Raffel et al., 2019).

[*] Contribution during internship at Microsoft. [1]Harbin Institute of Technology [2]Microsoft Research. Correspondence to: Li Dong <lidong1@microsoft.com>, Furu Wei <fuwei@microsoft.com>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

Rather than independently predicting words, the probability of a word is dependent on previous predictions.

Inspired by (Dong et al., 2019), we propose a pseudo-masked language model (PMLM) to jointly pre-train a bidirectional LM for language understanding (e.g., text classification, and question answering) and a sequence-to-sequence LM for language generation (e.g., document summarization, and response generation). Specifically, the bidirectional model is pre-trained by autoencoding (AE) LMs, and the sequence-to-sequence model is pre-trained by partially autoregressive (PAR) LMs. As shown in Figure 1, the model parameters are shared in two language modeling tasks, and the encoding results of the given context tokens are reused. We use the conventional mask `[MASK]` (or `[M]` for short) to represent the corrupted tokens for AE pre-training. In order to handle factorization steps of PAR language modeling, we append pseudo masks `[Pseudo]` (or `[P]` for short) to the input sequence without discarding the original tokens. With well-designed self-attention masks and position embeddings, the PMLM can perform the two language modeling tasks in one forward pass without redundant computation of context.

The proposed method has the following advantages. First, the PMLM pre-trains different LMs in a unified manner, which learns both inter-relations between masked tokens and given context (via AE), and intra-relations between masked spans (via PAR). Moreover, conventional masks used for AE provide global masking information, so that every factorization step of PAR pre-training can access all the position embeddings as in fine-tuning. Second, the unified pre-training framework learns models for both natural language understanding and generation (Dong et al., 2019). Specifically, the AE-based modeling learns a bidirectional Transformer encoder, and the PAR objective pre-trains a sequence-to-sequence decoder. Third, the proposed model is computationally efficient in that the AE and PAR modeling can be computed in one forward pass. Because the encoding results of given context are reused for two language modeling tasks, redundant computation is avoided. Fourth, PAR language modeling learns token-to-token, token-to-span, and span-to-span relations during pre-training. By taking spans (i.e., continuous tokens) into consideration, PMLM is encouraged to learn long-distance dependencies by preventing local shortcuts.

2. Preliminary

2.1. Backbone Network: Transformer

First, we pack the embeddings of input tokens $\{\mathbf{x}_i\}_{i=1}^{|x|}$ together into $\mathbf{H}^0 = [\mathbf{x}_1, \cdots, \mathbf{x}_{|x|}] \in \mathbb{R}^{|x| \times d_h}$. Then L stacked Transformer (Vaswani et al., 2017) blocks compute the encoding vectors via:

$$\mathbf{H}^l = \mathrm{Transformer}_l(\mathbf{H}^{l-1}),\ l \in [1, L] \tag{1}$$

where L is the number of layers. The hidden vectors of the final layer $\mathbf{H}^L = [\mathbf{h}_1^L, \cdots, \mathbf{h}_{|x|}^L]$ are the contextualized representations of input. Within each Transformer block, multiple self-attention heads aggregate the output vectors of the previous layer, followed by a fully-connected feed-forward network.

Self-Attention Masks The output $\mathbf{A}_l$ of a self-attention head in the l-th Transformer layer is:

$$\begin{aligned} &\mathbf{Q} = \mathbf{H}^{l-1}\mathbf{W}_l^Q,\ \mathbf{K} = \mathbf{H}^{l-1}\mathbf{W}_l^K \\ &\mathbf{M}_{ij} = \begin{cases} 0, & \text{allow to attend} \\ -\infty, & \text{prevent from attending} \end{cases} \\ &\mathbf{A}_l = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}} + \mathbf{M}\right)(\mathbf{H}^{l-1}\mathbf{W}_l^V) \end{aligned} \tag{2}$$

where parameters $\mathbf{W}_l^Q, \mathbf{W}_l^K, \mathbf{W}_l^V \in \mathbb{R}^{d_h \times d_k}$ project the previous layer's output $\mathbf{H}^{l-1}$ to queries, keys, and values, respectively. It is worth noting that the mask matrix $\mathbf{M} \in \mathbb{R}^{|x| \times |x|}$ controls whether two tokens can attend each other.

2.2. Input Representation

The inputs of language model pre-training are sequences sampled from large-scale text corpora. We follow the format used by BERT (Devlin et al., 2018). We add a special start-of-sequence token `[SOS]` at the beginning to get the representation of the whole input. Besides, each text is split into two segments appended with a special end-of-sequence token `[EOS]`. The final input format is "`[SOS] S1 [EOS] S2 [EOS]`", where the segments `S1` and `S2` are contiguous texts. The vector of an input token is represented by the summation of its token embedding, absolute position embedding, and segment embedding. All the embedding vectors are obtained by lookup in learnable matrices.

3. Unified Language Model Pre-Training

We propose a pseudo-masked language model (PMLM) to jointly pre-train both autoencoding (Section 3.1.1) and partially autoregressive (Section 3.1.2) LMs. As shown in Figure 2, PMLM reuses the encoding results of the same example to jointly pre-train both modeling methods by pseudo masking (Section 3.2).

3.1. Pre-Training Tasks

We use the masked language modeling (MLM; Devlin et al. 2018) task to pre-train a Transformer network, which is also known as the cloze task (Taylor, 1953). For a given input, we randomly substitute tokens with a special token `[MASK]`

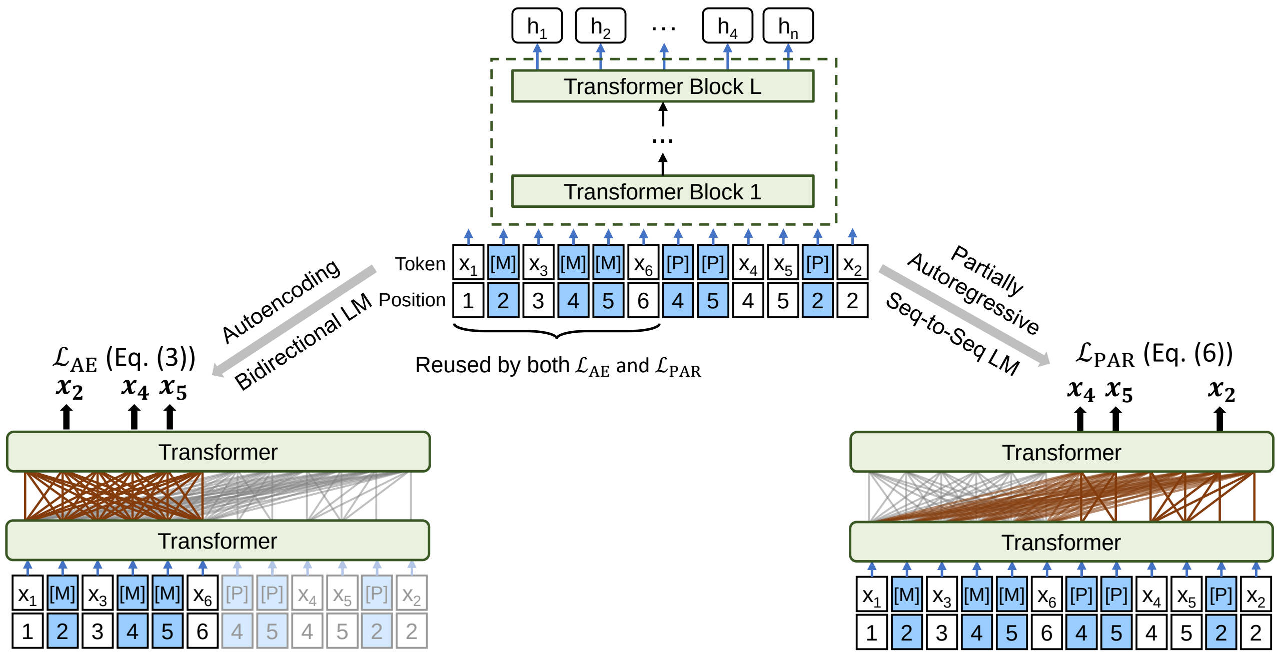

Figure 2. Overview of PMLM pre-training. The model parameters are shared across the LM objectives. The bidirectional LM is trained by autoencoding MLM, and the sequence-to-sequence (Seq-to-Seq) LM is trained by partially autoregressive MLM. We use different self-attention masks to control the access to context for each word token.

	Factorization Order	**Probability of Masked Tokens**
Autoencoding (e.g., BERT, and our work)	–	$p(x_2\|x_{\setminus\{2,4,5\}})p(x_3\|x_{\setminus\{2,4,5\}})p(x_5\|x_{\setminus\{2,4,5\}})$
Autoregressive (e.g., GPT, and XLNet)	$2 \rightarrow 4 \rightarrow 5$	$p(x_2\|x_{\setminus\{2,4,5\}})p(x_4\|x_{\setminus\{4,5\}})p(x_5\|x_{\setminus\{5\}})$
	$5 \rightarrow 4 \rightarrow 2$	$p(x_5\|x_{\setminus\{2,4,5\}})p(x_4\|x_{\setminus\{2,4\}})p(x_2\|x_{\setminus\{2\}})$
Partially Autoregressive (our work)	$2 \rightarrow 4, 5$	$p(x_2\|x_{\setminus\{2,4,5\}})p(x_4\|x_{\setminus\{4,5\}})p(x_5\|x_{\setminus\{4,5\}})$
	$4, 5 \rightarrow 2$	$p(x_4\|x_{\setminus\{2,4,5\}})p(x_5\|x_{\setminus\{2,4,5\}})p(x_2\|x_{\setminus\{2\}})$

Table 1. Given input $x = x_1 \cdots x_6$, the tokens x_2, x_4, x_5 are masked. We compare how to compute $p(x_2, x_4, x_5|x_{\setminus\{2,4,5\}})$ with different factorization orders for autoencoding, autoregressive, and partially autoregressive masked language models.

(or `[M]` for short). The training objective is to recover them by conditioning on the output hidden states of Transformer.

As shown in Table 1, we categorize MLMs into autoencoding, autoregressive, and partially autoregressive. Their main difference is how the probability of masked tokens is factorized. In our work, we leverage autoencoding (AE) and partially autoregressive (PAR) modeling for pre-training, which is formally described as follows. It is worth noting that the masked positions are the same for both AE and PAR modeling, but the probability factorization is different.

3.1.1. Autoencoding Modeling

The autoencoding method independently predicts the tokens by conditioning on context, which is the same as BERT. Given original input $x = x_1 \cdots x_{|x|}$ and the positions of masks $M = \{m_1, \cdots, m_{|M|}\}$, the probability of masked tokens is computed by $\prod_{m \in M} p(x_m|x_{\setminus M})$, where $x_M = \{x_m\}_{m \in M}$, $\setminus$ is set minus, $x_{\setminus M}$ means all input tokens except the ones that are in M. The autoencoding pre-training loss is defined as:

$$\mathcal{L}_{\text{AE}} = -\sum_{x \in \mathcal{D}} \log \prod_{m \in M} p(x_m|x_{\setminus M}) \tag{3}$$

where $\mathcal{D}$ is the training corpus.

3.1.2. Partially Autoregressive Modeling

We propose to pre-train partially autoregressive MLMs. In each factorization step, the model can predict one or multiple tokens. Let $M = \langle M_1, \cdots, M_{|M|} \rangle$ denote factorization order, where $M_i = \{m_1^i, \cdots, m_{|M_i|}^i\}$ is the set of mask positions in the i-th factorization step. If all factorization steps only contain one masked token (i.e., $|M_i| = 1$), the modeling becomes autoregressive. In our work, we enable a

Algorithm 1 Blockwise Masking

Input $x = x_1 \cdots x_{|x|}$: Input sequence
Output $M = \langle M_1, \cdots, M_{|M|} \rangle$: Masked positions
$M \leftarrow \langle\rangle$
repeat
  $p \leftarrow \text{rand\_int}(1, |x|)$ ▷ *Randomly sample an index*
  $l \leftarrow \text{rand\_int}(2, 6)$ **if** rand() < 0.4 **else** 1
  if $x_p, \cdots, x_{p+l-1}$ has not been masked **then**
    $M.\text{append}(\{m\}_{m=p}^{p+l-1})$
until $\sum_{j=1}^{|M|} |M_j| \geq 0.15|x|$ ▷ *Masking ratio is 15%*
return M

factorization step to be a span, which makes the LM partially autoregressive. The probability of masked tokens is decomposed as:

$$p(x_M | x_{\setminus M}) = \prod_{i=1}^{|M|} p(x_{M_i} | x_{\setminus M_{\geq i}}) \tag{4}$$

$$= \prod_{i=1}^{|M|} \prod_{m \in M_i} p(x_m | x_{\setminus M_{\geq i}}) \tag{5}$$

where $x_{M_i} = \{x_m\}_{m \in M_i}$, and $M_{\geq i} = \bigcup_{j \geq i} M_j$. The partially autoregressive pre-training loss is defined as:

$$\mathcal{L}_{\text{PAR}} = -\sum_{x \in \mathcal{D}} \mathbb{E}_M \log p\left(x_M | x_{\setminus M}\right) \tag{6}$$

where $\mathbb{E}_M$ is the expectation over the factorization distribution. During pre-training, we randomly sample one factorization order M for each input text (Yang et al., 2019), rather than computing the exact expectation.

Blockwise Masking and Factorization Given input sequence x, the masking policy uniformly produces a factorization order $M = \langle M_1, \cdots, M_{|M|} \rangle$ for Equation (6). For the i-th factorization step, the masked position set M_i contains one token, or a continuous text span (Joshi et al., 2019). As described in Algorithm 1, we randomly sample 15% of the original tokens as masked tokens. Among them, 40% of the time we mask a n-gram block, and 60% of the time we mask a token. We then construct a factorization step with the set of masked positions. We repeat the above process until enough masked tokens are sampled. The randomly sampled factorization orders are similar to permutation-based language modeling used by XLNet (Yang et al., 2019). However, XLNet only emits predictions one by one (i.e., autoregressive). In contrast, we can generate one token, or a text span at each factorization step (i.e., partially autoregressive).

3.2. Pseudo-Masked LM

Equation (5) indicates that factorization steps of partially autoregressive language modeling are conditioned on different context. So if masked language models (Devlin et al., 2018) are directly used, we have to construct a new cloze instance (as shown in Figure 3) for each factorization step, which renders partially autoregressive pre-training infeasible. We propose a new training procedure, named as pseudo-masked language model (PMLM), to overcome the issue.

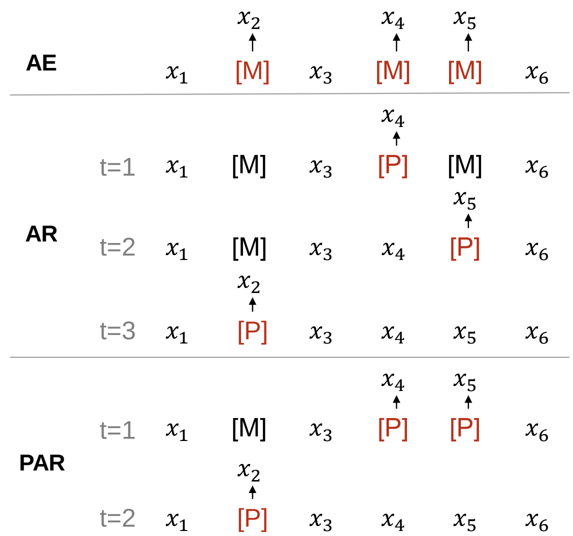

Figure 3. Comparisons between autoencoding (AE), autoregressive (AR), and partially autoregressive (PAR) masked language models. In the example $x = x_1 \cdots x_6$, the tokens x_2, x_4, x_5 are masked by the special tokens `[M]` and `[P]`.

For the last example in Table 1, Figure 4 shows how the PMLM conducts partially autoregressive predictions. Rather than replacing the tokens with masks as in vanilla MLMs, we keep all original input tokens unchanged and append pseudo masks to the input sequence. For each masked token, we insert a `[Pseudo]` (or `[P]` for short) token with the same position embedding of the corresponding token. The top-layer hidden states of `[P]` tokens are fed into a softmax classifier for MLM predictions. Notice that positional information in Transformer is encoded by (absolute) position embeddings, while the model components are order-agnostic. In other words, no matter where a token appears in the input sequence, the position of the token is only determined by its position embedding. So we can assign the same position embedding to two tokens, and Transformer treats both of the tokens as if they have the same position.

Vanilla MLMs allow all tokens to attend to each other, while PMLM controls accessible context for each token according to the factorization order. As shown in Figure 4, the example's factorization order is $4, 5 \rightarrow 2$. When we compute $p(x_4, x_5 | x_{\setminus\{2,4,5\}})$, only x_1, x_3, x_6 and the pseudo masks of x_4, x_5 are conditioned on. The original tokens of x_4, x_5 are masked to avoid information leakage, while their pseudo tokens `[P]` are used as placeholders for MLM predictions. In the second step, the tokens x_1, x_3, x_4, x_5, x_6

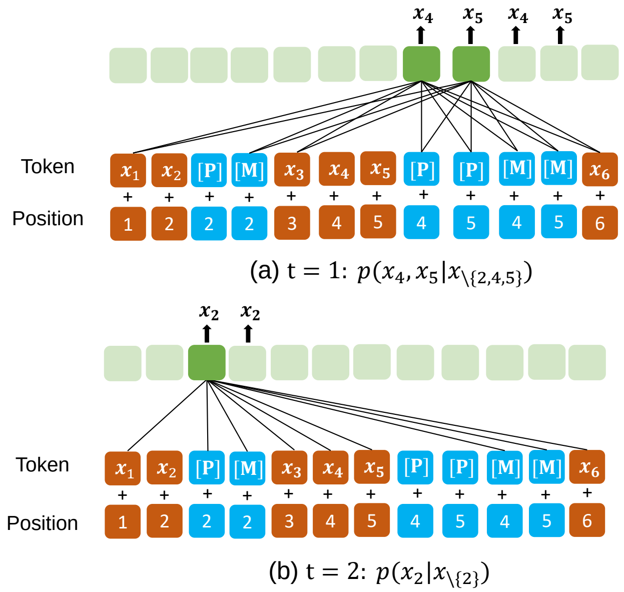

Figure 4. Example of the factorization steps $4, 5 \to 2$. The masks `[P]` and `[M]` are assigned with the same position embeddings as the corresponding tokens. Different context is used to compute the hidden states for the pseudo masks of x_4, x_5 and x_2.

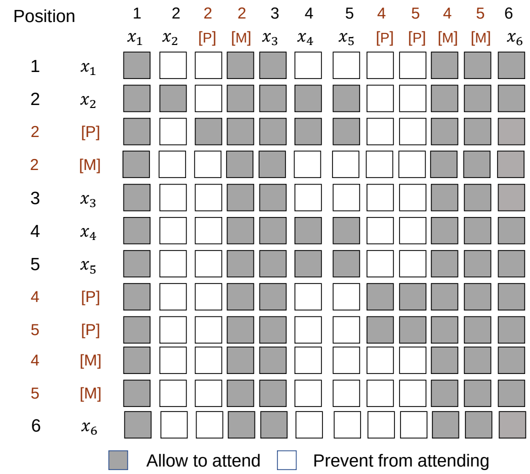

Figure 5. Self-attention mask of the factorization steps $4, 5 \to 2$. Both conventional masks `[M]` and given context (x_1, x_3, x_6) can be attended by all the tokens.

and the pseudo mask of x_2 are conditioned on to compute $p(x_2|x_{\setminus\{2\}})$. Unlike in the first step, the original tokens of x_4, x_5 are used for the prediction.

Self-attention masks (as described in Section 2.1) are used to control what context a token can attend to when computing its contextualized representation. Figure 5 shows the self-attention mask matrix used for the example of Figure 4. The self-attention mask matrix is designed in order to avoid two kinds of information leakage. The first type is *explicit leakage*, i.e., the masked token can be directly accessed by its pseudo token, which renders the LM prediction trivial. So pseudo tokens `[P]` are not allowed to attend to the content of "themselves" in a PMLM. The second type is *implicit leakage*, which implicitly leaks prediction information by multi-step attention propagation. For example, as shown in Figure 5, if the context token x_6 has access to x_4, there is a connected attention flow "x_4's pseudo mask token $\to x_6 \to x_4$", which eases the prediction of x_4. As a result, for each token, we mask the attentions to the tokens that are predicted in the future factorization steps.

The most relevant work for the proposed model is XLNet (Yang et al., 2019). In comparison, XLNet uses an AR-based objective. Our model jointly pre-trains two types of LMs, where AE and PAR modelings are complementary for pre-training. The two objectives pre-train a unified LM as a bidirectional encoder and a sequence-to-sequence decoder, respectively. Moreover, XLNet does not explicitly uses the mask `[M]` in two-stream attention. In contrast, we use the original token, the conventional mask `[M]`, and the pseudo mask `[P]` to represent a word's different roles in terms of context modeling.

3.3. Unified Pre-Training

As shown in Figure 2, we jointly pre-train bidirectional and sequence-to-sequence LMs with the same input text and masked positions. Both the special tokens `[M]` and `[P]` emit predicted tokens. The training objective is to maximize the likelihood of correct tokens, which considers two types of LMs (i.e., autoencoding, and partially autoregressive) in one example. The loss is computed via:

$$\mathcal{L} = \mathcal{L}_{\text{AE}} + \mathcal{L}_{\text{PAR}} \tag{7}$$

where $\mathcal{L}_{\text{AE}}, \mathcal{L}_{\text{PAR}}$ are defined as in Equation (3), and Equation (6) respectively. The proposed method sufficiently reuses the computed hidden states for both LM objectives. In addition, experiments in Section 4.6 show that the pre-training tasks are complementary to each other, as they capture both inter- (i.e., between given context and masked tokens) and intra- (i.e., among masked tokens) relations of the input tokens.

There have been two main strands of research for unified pre-training. The first one jointly pretrains a shared Transformer network for language understanding and generation, such as UniLM (Dong et al., 2019). In comparison, UniLM uses multiple training instances (i.e., forward passes) for different types of LMs. In contrast, the proposed model is more sample efficient in that bidirectional LM (via AE) and sequence-to-sequence LM (via PAR) can be computed in one forward pass. Because the encoding results of given context are reused for two LM tasks, redundant computation is avoided. Moreover, the sequence-to-sequence LM

Model	SQuAD v1.1 F1	SQuAD v1.1 EM	SQuAD v2.0 F1	SQuAD v2.0 EM
BERT	88.5	80.8	76.3	73.7
XLNet	-	-	-	80.2
RoBERTa	91.5	84.6	83.7	80.5
UniLMv2	**93.1**	**87.1**	**86.1**	**83.3**
– rel pos	93.0	86.7	85.2	82.4

Table 2. Results of BASE-size pre-trained models on the SQuAD v1.1/v2.0 development sets. We report F1 and exact match (EM) scores. Results of UniLMv2 are averaged over five runs. "– rel pos" is the model without relative position bias.

Model	MNLI Acc	SST-2 Acc	MRPC Acc	RTE Acc	QNLI Acc	QQP Acc	STS PCC	CoLA MCC
BERT	84.5	93.2	87.3	68.6	91.7	91.3	89.5	58.9
XLNet	86.8	94.7	88.2	74.0	91.7	91.4	89.5	60.2
RoBERTa	87.6	94.8	90.2	78.7	92.8	**91.9**	**91.2**	63.6
UniLMv2	**88.5**	**95.1**	**91.8**	**81.3**	**93.5**	91.7	91.0	**65.2**
– rel pos	88.4	95.0	91.2	78.1	93.4	91.8	**91.2**	63.8

Table 3. Results of BASE-size models on the development set of the GLUE benchmark. We report Matthews correlation coefficient (MCC) for CoLA, Pearson correlation coefficient (PCC) for STS, and accuracy (Acc) for the rest. Metrics of UniLMv2 are averaged over five runs for the tasks. "– rel pos" is the ablation model without relative position bias.

in our work learns to recover the masked tokens, while UniLM's sequence-to-sequence LM predicts the next sentences. Another strand of unified pre-training is using the encoder-decoder architecture as backbone networks, such as BART (Lewis et al., 2019). First, BART contains two parts of Transformers, i.e., encoder, and decoder. We instead share one network for different LMs, which has less parameters and more unified modeling. Second, BART uses an autoregressive-based objective, while we jointly pre-train AE- and PAR-based objectives. Third, to use BART for classification, the input text is fed into the encoder and decoder twice. In contrast, we directly use the model as a bidirectional encoder, which is easier to use in practice.

3.4. Fine-tuning on NLU and NLG Tasks

Following (Dong et al., 2019), we fine-tune the pre-trained PMLM (with additional task-specific layers if necessary) to both natural language understanding (NLU) and natural language generation (NLG) tasks.

For NLU tasks, we fine-tune PMLM as a bidirectional Transformer encoder, like BERT. Let us take text classification as an example. Similar to the text format described in Section 2.2, the input is "`[SOS] TEXT [EOS]`". We use the encoding vector of `[SOS]` as the representation of input, and then feed it to a randomly initialized softmax classifier (i.e., the task-specific output layer). We maximize the likelihood of the labeled training data by updating the parameters of the pre-trained PMLM and the added softmax classifier.

For sequence-to-sequence generation tasks, the example is concatenated as "`[SOS] SRC [EOS] TGT [EOS]`", where `SRC` and `TGT` are source and target sequences, respectively. The fine-tuning procedure is similar to pre-training as in Section 3.2. For a source sequence, the dependencies between the tokens are bidirectional, i.e., all the source tokens can attend to each other. In contrast, the target sequence is produced in an autoregressive manner. So we append a pseudo mask `[P]` for each target token, and use self-attention masks to perform autoregressive generation. The fine-tuning objective is to maximize the likelihood of the target sequence given source input. It is worth noting that `[EOS]` is used to mark the end of the target sequence. Once `[EOS]` is emitted, we terminate the generation process of the target sequence. During decoding, we use beam search to generate the target tokens one by one (Dong et al., 2019).

4. Experimental Results

We employ pseudo-masked language model to conduct unified language model pre-training (UniLMv2), and fine-tuned the model on both natural language understanding (i.e., question answering, the GLUE benchmark) and generation (i.e., abstractive summarization, and question generation) tasks. Details about hyperparameters and datasets can be found in the supplementary material. In addition, we conducted ablation studies to compare different choices of pre-training objectives.

4.1. Pre-Training Setup

We followed the same model size as $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2018) for comparison purposes. Specifically, we used a 12-layer Transformer with 12 attention heads. The hidden size was 768, and inner hidden size of feed-forward network was 3072. The weight matrix of the softmax classifier was tied with the token embedding matrix. We also add relative position bias (Raffel et al., 2019) to attention scores. The whole model contains about 110M parameters.

For fair comparisons, we report the major results using similar pre-training datasets and optimization hyperparameters as in $\text{RoBERTa}_{\text{BASE}}$ (Liu et al., 2019). We use 160GB text corpora from English Wikipedia[1], BookCorpus (Zhu et al., 2015), OpenWebText[2], CC-News (Liu et al., 2019), and Stories (Trinh & Le, 2018). We follow the preprocess and

[1] Wikipedia version: enwiki-20181101.
[2] `skylion007.github.io/OpenWebTextCorpus`

Model	#Param	CNN/DailyMail			XSum		
		RG-1	RG-2	RG-L	RG-1	RG-2	RG-L
Without pre-training							
Lead-3		40.42	17.62	36.67	16.30	1.60	11.95
PtrNet (See et al., 2017)		39.53	17.28	36.38	28.10	8.02	21.72
Fine-tuning LARGE-*size pre-trained models*							
UniLM$_{\text{LARGE}}$ (Dong et al., 2019)	340M	43.08	20.43	40.34	-	-	-
BART$_{\text{LARGE}}$ (Lewis et al., 2019)	400M	44.16	21.28	40.90	45.14	22.27	37.25
T5$_{\text{11B}}$ (Raffel et al., 2019)	11B	43.52	21.55	40.69	-	-	-
Fine-tuning BASE-*size pre-trained models*							
MASS$_{\text{BASE}}$ (Song et al., 2019)	123M	42.12	19.50	39.01	39.75	17.24	31.95
BERTSumAbs (Liu & Lapata, 2019)	156M	41.72	19.39	38.76	38.76	16.33	31.15
T5$_{\text{BASE}}$ (Raffel et al., 2019)	220M	42.05	20.34	39.40	-	-	-
UniLMv2$_{\text{BASE}}$	110M	43.16	20.42	40.14	**44.00**	**21.11**	**36.08**
– relative position bias	110M	**43.45**	**20.71**	**40.49**	43.69	20.71	35.73

Table 4. Abstractive summarization results on CNN/DailyMail and XSum. The evaluation metric is the F1 version of ROUGE (RG) scores. We also present the number of parameters (#Param) for the methods using pre-trained models.

	#Param	BLEU-4	MTR	RG-L
(Du & Cardie, 2018)		15.16	19.12	-
(Zhang & Bansal, 2019)		18.37	22.65	46.68
UniLM$_{\text{LARGE}}$	340M	22.78	25.49	51.57
UniLMv2$_{\text{BASE}}$	110M	24.43	**26.34**	51.97
– rel pos	110M	**24.70**	26.33	**52.13**
(Zhao et al., 2018)		16.38	20.25	44.48
(Zhang & Bansal, 2019)		20.76	24.20	48.91
UniLM$_{\text{LARGE}}$	340M	24.32	26.10	52.69
UniLMv2$_{\text{BASE}}$	110M	26.29	**27.16**	**53.22**
– rel pos	110M	**26.30**	27.09	53.19

Table 5. Results on question generation. The first block follows the data split in (Du & Cardie, 2018), while the second block is the same as in (Zhao et al., 2018). MTR is short for METEOR, and RG for ROUGE. "#Param" indicates the size of pre-trained models. "– rel pos" is the model without relative position bias.

the uncased WordPiece (Wu et al., 2016) tokenization used in (Devlin et al., 2018). The vocabulary size was $30,522$. The maximum length of input sequence was 512. The token masking probability was 15%. Among masked positions, 80% of the time we replaced the token with masks, 10% of the time with a random token, and keeping the original token for the rest. The block masking (see Algorithm 1) can mask up to 6-gram for one factorization step in partially autoregressive modeling. The batch size was set to 7680. We used Adam (Kingma & Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon =$ 1e-6 for optimization. The peak learning rate was set to 6e-4, with linear warmup over the first $24,000$ steps and linear decay. The weight decay was 0.01. The dropout rate was set to 0.1. We ran the pre-training procedure for 0.5 million steps, which took about 20 days using 64 Nvidia V100-32GB GPU cards.

4.2. Question Answering

Question answering aims at returning answers for the given question and documents. We conduct experiments on the benchmarks SQuAD v1.1 (Rajpurkar et al., 2016) and v2.0 (Rajpurkar et al., 2018). The model learns to extract answer spans within a passage. We formulate the task as a natural language understanding problem. The input is concatenated as "`[SOS] Question [EOS] Passage [EOS]`". We add a classification layer on the pre-trained PMLM, which predicts whether each token is the start or end position of an answer span by conditioning on the final outputs of Transformer. For SQuAD v2.0, we use the output vector of `[SOS]` to predict whether the instance is unanswerable or not.

The fine-tuning results are presented in Table 2, where we report F1 scores and exact match (EM) scores. We compare previous BASE-size models with PMLM. Notice that the publicly available BERT$_{\text{BASE}}$ checkpoint (Devlin et al., 2018) is pre-trained on 13GB corpora with 256 batch size, while XLNet$_{\text{BASE}}$ and RoBERTa$_{\text{BASE}}$ are more directly comparable. The results show that UniLMv2$_{\text{BASE}}$ achieves better performance than the other models on both SQuAD datasets.

4.3. GLUE Benchmark

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) contains various tasks. There are two single-sentence classification tasks, i.e., linguistic

	Model	Objective	SQuAD v1.1 F1	SQuAD v1.1 EM	SQuAD v2.0 F1	SQuAD v2.0 EM	MNLI m	MNLI mm	SST-2 Acc
	$\text{BERT}_{\text{BASE}}$	AE	88.5	80.8	76.3	73.7	84.3	84.7	92.8
	$\text{XLNet}_{\text{BASE}}$	AR	-	-	81.0	78.2	85.6	85.1	**93.4**
	$\text{RoBERTa}_{\text{BASE}}$	AE	90.6	-	79.7	-	84.7	-	92.7
	$\text{BART}_{\text{BASE}}$	AR	90.8	-	-	-	83.8	-	-
[1]	$\text{UniLMv2}_{\text{BASE}}$	AE+PAR	**92.0**	**85.6**	**83.6**	**80.9**	**86.1**	**86.1**	93.2
[2]	[1] – relative position bias	AE+PAR	91.5	85.0	81.8	78.9	85.6	85.5	93.0
[3]	[2] – blockwise factorization	AE+AR	90.8	84.1	80.7	77.8	85.4	85.5	92.6
[4]	[2] – PAR	AE	91.0	84.2	81.3	78.4	84.9	85.0	92.4
[5]	[2] – AE	PAR	90.7	83.9	79.9	77.0	84.9	85.2	92.5
[6]	[5] – blockwise factorization	AR	89.9	82.9	79.3	76.1	84.8	85.0	92.3

Table 6. Comparisons between the pre-training objectives. All models are pre-trained over Wikipedia and BookCorpus for one million steps with a batch size of 256. Results in the second block are average over five runs for each task. We report F1 and exact match (EM) scores for SQuAD, and accuracy (Acc) for MNLI and SST-2.

acceptability (CoLA; Warstadt et al. 2018), and sentiment analysis (SST-2; Socher et al. 2013). The text similarity (STS; Cer et al. 2017) task is formulated as a regression problem. The other tasks are pairwise classification tasks, including natural language inference (RTE, MNLI; Dagan et al. 2006; Bar-Haim et al. 2006; Giampiccolo et al. 2007; Bentivogli et al. 2009; Williams et al. 2018), question answering (QNLI; Rajpurkar et al. 2016), and paraphrase detection (QQP, MRPC; Dolan & Brockett 2005).

Table 3 presents the results on GLUE. We compare PMLM with three strong pre-trained models, i.e., BERT (Devlin et al., 2018), XLNet (Yang et al., 2019), and RoBERTa (Liu et al., 2019), in the single task fine-tuning setting. All the models are in BASE-size for fair comparisons. We observe that the proposed $\text{UniLMv2}_{\text{BASE}}$ outperforms both $\text{BERT}_{\text{BASE}}$ and $\text{XLNet}_{\text{BASE}}$ across 8 tasks. Comparing to state-of-the-art pre-trained $\text{RoBERTa}_{\text{BASE}}$, $\text{UniLMv2}_{\text{BASE}}$ obtains the best performance on 6 out of 8 tasks, e.g., 88.4 vs 87.6 ($\text{RoBERTa}_{\text{BASE}}$) in terms of MNLI accuracy, indicating the effectiveness of our $\text{UniLMv2}_{\text{BASE}}$.

4.4. Abstractive Summarization

We evaluate the pre-trained PMLM on two abstractive summarization datasets, i.e., XSum (Narayan et al., 2018), and the non-anonymized version of CNN/DailyMail (See et al., 2017). This is a language generation task, where the texts (such as news articles) are shortened to readable summaries that preserve salient information of the original texts. The pre-trained PMLM is fine-tuned as a sequence-to-sequence model as described in Section 3.4.

We report ROUGE scores (Lin, 2004) on the datasets. Table 4 shows two baseline methods that do not rely on pre-training. Lead-3 uses the first three input sentences as the summary. PtrNet (See et al., 2017) is a sequence-to-sequence model with pointer networks. Results indicate that pre-training achieves significant improvements over the baselines. We also compare $\text{UniLMv2}_{\text{BASE}}$ with state-of-the-art pre-trained models of both BASE-size and LARGE-size. We focus on the comparisons in the third block because the models contain similar numbers of parameters. BERT-SumAbs (Liu & Lapata, 2019) fine-tunes a BERT encoder that is pre-trained with an autoencoding objective, concatenating with a randomly initialized decoder. MASS (Song et al., 2019) and T5 (Raffel et al., 2019) pre-train encoder-decoder Transformers with masked LM, which relies on the autoregressive pre-training. Although PMLM has the smallest size, we find that $\text{UniLMv2}_{\text{BASE}}$ outperforms the other BASE-size pre-trained models on both datasets.

4.5. Question Generation

We perform evaluations on question generation (Du & Cardie, 2018), the task of automatically producing relevant questions that ask for the given answer and context. The input of the sequence-to-sequence problem is defined as the concatenation of a paragraph and an answer. We fine-tune the pre-trained PMLM to predict output questions.

As shown in Table 5, we report BLEU (Papineni et al., 2002), METEOR(Banerjee & Lavie, 2005), and ROUGE (Lin, 2004) scores on two different data splits. Among the compared results, UniLM (Dong et al., 2019) is based on pre-trained models, while the other three methods are sequence-to-sequence models enhanced with manual features (Du & Cardie, 2018), gated self-attention (Zhao et al., 2018), and reinforcement learning (Zhang & Bansal, 2019). Results show that $\text{UniLMv2}_{\text{BASE}}$ achieves better evaluation metrics compared with $\text{UniLM}_{\text{LARGE}}$ and several baselines. It is worth noting that $\text{UniLMv2}_{\text{BASE}}$ consists of three times fewer parameters than $\text{UniLM}_{\text{LARGE}}$.

4.6. Effect of Pre-Training Objectives

We conduct ablation experiments on using PMLM to implement different pre-training objectives, i.e., autoencoding (AE), autoregressive (AR), partially autoregressive (PAR), and jointly training (AE+AR, and AE+PAR). The variants use the same masking strategy. The evaluations follow the settings[3] as in BERT (Devlin et al., 2018), so that the results in Table 6 can be directly compared with each other. Notice that XLNet (Yang et al., 2019) is an autoregressive MLM augmented with more advanced relative position embeddings, and long-context memory.

As shown in Table 6, we compare the PMLM-based variants against previous models on question answering (SQuAD; Rajpurkar et al. 2016; 2018), natural language inference (MNLI; Williams et al. 2018), and sentiment classification (SST-2; Socher et al. 2013). First, we ablate relative position bias to better compare with BERT, RoBERTa, and BART. On text classification (MNLI and SST-2), the PAR-only objective compares favorably with both AE-only and AR-only objectives, which indicates the effectiveness of the proposed PAR modeling. In comparison, the SQuAD tasks require more precise modeling of spans in order to extract correct answer spans from the input passage, where both AE-only and PAR-only objectives outperform the AR-only objective. The results indicate that blockwise masking and factorization are important for LM pre-training. Besides, the settings of jointly training (AE+AR, and AE+PAR) tend to improve the results over using single LM task. Among the five objectives, AE+PAR performs the best with the help of PMLM, which shows that autoencoding and partially autoregressive modelings are complementary for pre-training.

5. Conclusion

We pre-train a unified language model for language understanding and generation by joint learning bidirectional LM (via AE) and sequence-to-sequence LM (via PAR). We introduce a pseudo-masked language model (PMLM) to efficiently realize the unified pre-training procedure. PMLM is computationally efficient in that AE and PAR can be computed in one forward pass without redundant computation. Besides, the two modeling tasks are complementary to each other. Because conventional masks of AE provide global masking information to PAR, and PAR can learn intra-relations between masked spans. In addition, the proposed PAR pre-training encourages to learn long-distance dependencies by preventing local shortcuts. Experimental results show that PMLM improves the end-task results on several language understanding and generation benchmarks.

[3]Models were trained for 1M steps with batch size of 256 over English Wikipedia and BookCorpus (Zhu et al., 2015). The learning rate of Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$) was set to 1e-4, with linear schedule and warmup over the first 10K steps.

References

Banerjee, S. and Lavie, A. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72, 2005.

Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., and Giampiccolo, D. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Bentivogli, L., Dagan, I., Dang, H. T., Giampiccolo, D., and Magnini, B. The fifth pascal recognizing textual entailment challenge. In *In Proc Text Analysis Conference*, 2009.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

Chi, Z., Dong, L., Wei, F., Wang, W., Mao, X., and Huang, H. Cross-lingual natural language generation via pre-training. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7570–7577. AAAI Press, 2020.

Dagan, I., Glickman, O., and Magnini, B. The PASCAL recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges*, pp. 177–190, 2006.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

Dolan, W. B. and Brockett, C. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*, 2005.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. Unified language model pre-training for natural language understanding and generation. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

Du, X. and Cardie, C. Harvesting paragraph-level question-answer pairs from wikipedia. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 1907–1917, 2018.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 1–9, 2007.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. SpanBERT: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, San Diego, CA, 2015.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2019.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pp. 74–81, Barcelona, Spain, 2004.

Liu, Y. and Lapata, M. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3730–3740, Hong Kong, China, 2019.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1797–1807, Brussels, Belgium, 2018.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, 2002.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2227–2237, New Orleans, Louisiana, 2018.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. 2018.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, 2016.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pp. 784–789, 2018.

See, A., Liu, P. J., and Manning, C. D. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 1073–1083, Vancouver, Canada, 2017.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

Taylor, W. L. Cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433, 1953.

Trinh, T. H. and Le, Q. V. A simple method for commonsense reasoning. *ArXiv*, abs/1806.02847, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pp. 5998–6008, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

Williams, A., Nangia, N., and Bowman, S. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1112–1122, New Orleans, Louisiana, 2018.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. XLNet: Generalized autoregressive pretraining for language understanding. In *33rd Conference on Neural Information Processing Systems*, 2019.

Zhang, S. and Bansal, M. Addressing semantic drift in question generation for semi-supervised question answering. *CoRR*, abs/1909.06356, 2019.

Zhao, Y., Ni, X., Ding, Y., and Ke, Q. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3901–3910, Brussels, Belgium, 2018.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.