PERT: Pre-training BERT with Permuted Language Model

Yiming Cui[1,2,†], **Ziqing Yang**[2], **Shijin Wang**[2,3], **Ting Liu**[1]
[1]Research Center for SCIR, Harbin Institute of Technology, Harbin, China
[2]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, Beijing, China
[3]iFLYTEK AI Research (Central China), Wuhan, China
[†]ymcui@ieee.org

Abstract

Pre-trained Language Models (PLMs) have been widely used in various natural language processing tasks, owing to their powerful text representations, which are trained on large-scale corpora. In this paper, we propose a new PLM called PERT to address the importance of the local context and resolve the discrepancy of pretraining-finetuning in BERT. PERT is an auto-encoding model (like BERT) trained with Permuted Language Model (PerLM). The formulation of the proposed PerLM is straightforward. We permute a proportion of the input sequence, and the training objective is to predict the position of the original token. Moreover, we also apply whole word masking and n-gram masking to improve the performance of PERT. We carried out extensive experiments on both Chinese and English benchmarks. The experimental results show that PERT could bring improvements over various comparable baselines on several tasks. Several quantitative studies are carried out to better understand PERT. Resources are available: dummy-url

1 Introduction

Pre-trained Language Models (PLMs) have shown excellent performance on various natural language processing (NLP) tasks. Usually, PLMs are classified into two categories based on their training protocols: auto-encoding and auto-regressive. The representative work of auto-encoding PLM is the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), which models the input text through deep transformer layers (Vaswani et al., 2017), creating deep contextualized representations. The representative work of auto-regressive PLM is the Generative Pre-training (GPT) model (Radford et al., 2018).

The dominant pre-training task for auto-encoding PLM is the masked language model (MLM). The MLM pre-training task replaces a few input tokens with masking tokens (i.e., [MASK]), and the objective is to recover these tokens in the vocabulary space. The formulation of MLM is pretty simple, but it can model the contextual features around the masking token, which is quite similar to the continuous bag-of-words (CBOW) in word2vec (Mikolov et al., 2013). Based on MLM pre-training task, there are also a few variants proposed to further enhance its performance, such as whole word masking (Devlin et al., 2019; Sun et al., 2019; Cui et al., 2021), N-gram masking (Devlin et al., 2019; Joshi et al., 2019; Cui et al., 2020), etc. Following MLM pre-training scheme, various PLMs are proposed, such as ERNIE (Sun et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020), ELECTRA (Clark et al., 2020), MacBERT (Cui et al., 2021), etc.

	Input	Output
Original	研究表明这句话的顺序并不影响阅读。	-
WordPiece	研 究 表 明 这 句 话 的 顺 序 并 不 影 响 阅 读 。	-
BERT	研 究 表 明 这 句 **[M]** 的 顺 **[M]** 并 不 **[M]** 响 阅 读 。	$\text{Pos}_7 \rightarrow$ 话 $\text{Pos}_{10} \rightarrow$ 序 $\text{Pos}_{13} \rightarrow$ 影
PERT	研 究 明 表 这 句 话 的 顺 序 并 不 响 影 阅 读 。	$\text{Pos}_2 \rightarrow \text{Pos}_3$ $\text{Pos}_3 \rightarrow \text{Pos}_2$ $\text{Pos}_{13} \rightarrow \text{Pos}_{14}$ $\text{Pos}_{14} \rightarrow \text{Pos}_{13}$

Table 1: Input and output for BERT and PERT.

However, there comes with a natural question: ***Can we use pre-training task other than MLM?*** To deal with this question, in this paper, we aim to explore a pre-training task that is not derived from MLM. The original motivation behind our approach is quite interesting. There are many sayings like "Permuting several Chinese characters does not affect reading that much". A vivid illustration is depicted in Figure 1, where we provide both Chinese and English examples. As we can

see, with a first glimpse, we might not notice that some words in the sentence are disordered, but we can still grasp the central meaning of the sentence. This phenomenon makes us curious whether we can model the contextual representation via permuted sentences. To investigate this question, we propose a new pre-training task called permuted language model (PerLM). The proposed PerLM tries to recover the word orders from a disordered sentence, and the objective is to predict the position of the original word. We pre-train both Chinese and English PERT to examine their effectiveness. Extensive experiments are conducted on both Chinese and English NLP datasets, ranging from sentence-level to document-level, such as machine reading comprehension, text classification, etc. The results show that the proposed PERT can bring improvements on a few tasks. While in the meantime, we also discover their deficiencies in others. The contributions of this paper are listed as follows.

- We propose a non-MLM-like pre-training task, called permuted language model, which tries to recover the shuffled input text into the right order.
- Experimental results show both positive and negative results, and further analyses might be helpful in creating non-MLM-like pre-training tasks.

2 Related Work

In this section, we revisit the techniques of the representative pre-trained language models in the recent natural language processing field. Text representations have made significant progress in recent years after the emergence of the pre-trained language models. The pre-trained language model utilizes large-scale text corpora and unsupervised (or self-supervised) learning algorithms to extract text semantics in a continuous space. This paper mainly focuses on the pre-trained language model for natural language understanding (NLU). A list of such PLMs are listed in Table 2, where we also list the new model PERT in it. Next, we will briefly introduce these models and elaborate the difference from PERT.

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) has proven to be successful in various NLU tasks, which is a representative auto-encoding PLM. There are two pre-training tasks for BERT. **Masked Language Model (MLM)**: The MLM task randomly masks several tokens in the input sequence and requires to recover these tokens in the output. To correctly predict the original tokens, the model should utilize bidirectional context around the masking position. **Next Sentence Prediction (NSP)**: The NSP task mainly focuses on the relationship in a larger context. It discriminates whether a sentence is the next sentence of another one.

	Type	Emb	Mask	LM	Pair
GPT (Radford et al., 2018)	AR	T/S/P	-	LM	-
BERT (Devlin et al., 2019)	AE	T/S/P	T	MLM	NSP
ERNIE (Sun et al., 2019)	AE	T/S/P	T/E/Ph	MLM	NSP
XLNet (Yang et al., 2019)	AR	T/S/P	-	PLM	-
RoBERTa (Liu et al., 2019)	AE	T/S/P	T	MLM	-
ALBERT (Lan et al., 2020)	AE	T/S/P	T	MLM	SOP
ELECTRA (Clark et al., 2020)	AE	T/S/P	T	Gen-Dis	-
MacBERT (Cui et al., 2021)	AE	T/S/P	WWM/NM	Mac	SOP
PERT	AE	T/S/P	WWM/NM	PerLM	-

Table 2: Comparisons of the pre-trained language models. (AE: Auto-Encoding, AR: Auto-Regressive, T: Token, S: Segment, P: Position, W: Word, E: Entity, Ph: Phrase, WWM: Whole Word Masking, NM: N-gram Masking, NSP: Next Sentence Prediction, SOP: Sentence Order Prediction, MLM: Masked LM, PLM: Permutation LM, Mac: MLM as correction)

To further improve the difficulties in the MLM task, Devlin et al. (2019) further proposed a technique called whole word masking (wwm). In this setting, instead of randomly selecting WordPiece (Wu et al., 2016) tokens to mask, we always mask all of the tokens corresponding to a whole word at once. The whole word masking alleviates the "input information leaking" issue and has proven to be more effective than original MLM in various tasks (Cui et al., 2021). Furthermore, we can mask a consecutive N-gram to make MLM more difficult. The N-gram masking has also proven to be effective in various PLMs, such SpanBERT (Joshi et al., 2019), MacBERT (Cui et al., 2020), etc.

While MLM and its variants have been dominant in the design of various PLMs, it is intriguing to investigate other pre-training approaches other than MLM. ELECTRA (Clark et al., 2020) employs a new generator-discriminator framework that is similar to GAN (Goodfellow et al., 2014). ELECTRA is largely different from BERT-variant, but it still utilizes MLM in training the generator. StructBERT (Wang et al., 2019) proposes to incorporate language structures for pre-training. Though it uses shuffled input, its motivation and task design are fairly different from ours, such as we do not use MLM-style prediction, etc.

In this paper, we take a step further on designing pre-training tasks. We design a pre-training

task that does not adopt the MLM task, called permuted language model (PerLM). PerLM utilizes shuffled input text, and the objective is to predict the position of the original token.

3 PERT

3.1 Overview

An overview of PERT is depicted in Figure 1. As we can see that the proposed PERT shares identical neural architecture with BERT, while there is a slight difference in the input and the training objective. The proposed PERT uses a shuffled sentence[1] as the input, and the training objective is to predict the position of the original token.

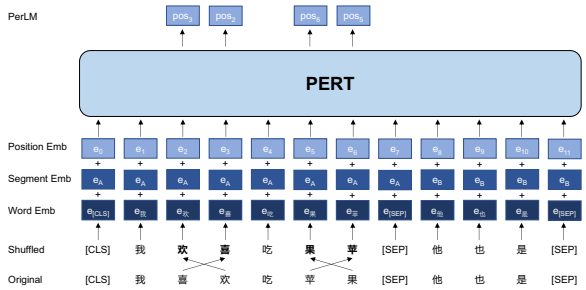


Figure 1: Neural architecture of PERT.

3.2 Permuted Language Model

As stated before, following various well-known PLMs, such as RoBERTa (Liu et al., 2019), we do not employ NSP-like pre-training tasks in PERT. The only pre-training task used in our PERT is the proposed Permuted Language Model (PerLM).[2] The formulation of PerLM is as follows.

- We use whole word masking as well as N-gram masking for selecting candidate tokens for masking, with a percentage of 40%, 30%, 20%, 10% for word-level unigram to 4-gram. This is identical to MacBERT (Cui et al., 2021) setting.
- Following previous works, we use a percentage of 15% input words for masking purposes. Among which,
  - We randomly select a set of 90% tokens and shuffle their orders. Note that the shuffle process only takes place for these 90% tokens, not the whole input sequence.
  - For the rest of 10% tokens, we keep them unchanged, treating them as negatives.

As we can see that PerLM is as simple as the original MLM, while PerLM features the following characters. 1) PerLM does not employ the artificial token `[MASK]` for masking, which alleviates the pretraining-finetuning discrepancy issue (but could still suffer from unnatural word orders). 2) The prediction space for PerLM is the input sequence rather than the whole vocabulary, making it computationally efficient than MLM.

3.3 Pre-training Stage

Formally, given a pair of sequences $A = \{A_1, \ldots, A_n\}$ and $B = \{B_1, \ldots, B_m\}$, we first use the method described in Section 3.2 to create new input sequence pairs $A' = \{A'_1, \ldots, A'_n\}$ and $B' = \{B'_1, \ldots, B'_m\}$, where some of the word positions are switched. Then we concatenate two sequences to form the input sequence X of PERT.

$$X = [\texttt{CLS}]\ A'_1 \ldots A'_n\ [\texttt{SEP}]\ B'_1 \ldots B'_m\ [\texttt{SEP}] \quad (1)$$

Then, PERT converts X into a contextualized representation $\boldsymbol{H} \in \mathbb{R}^{N \times d}$ through an embedding layer, consisting of word embedding, positional embedding, and token type (segment) embedding, and a consecutive L-layer transformer, where N is the maximum sequence length, and d is the dimension of hidden layers.

$$\boldsymbol{H}^{(0)} = \mathbf{Embedding}(X) \quad (2)$$

$$\boldsymbol{H}^{(i)} = \mathbf{Transformer}(\boldsymbol{H}^{(i-1)}), \quad (3)$$

Similar to MLM and Mac (MacBERT objective), we only need to predict the chosen positions in PerLM. We gather a subset with respect to these positions, forming the candidate representation $\boldsymbol{H}^{\mathrm{m}} \in \mathbb{R}^{k \times d}$, where k is the number of the chosen tokens. According to the definition of PerLM, we adopt a masking ratio of 15%, and thus $k = \lfloor N \times 15\% \rfloor$. Then we use a feed-forward dense layer (FFN), followed by a dropout and layer normalization layer.

$$\tilde{\boldsymbol{H}}^{m} = \mathbf{LayerNorm}(\mathbf{DO}(\mathbf{FFN}(\boldsymbol{H}^{m}))) \quad (4)$$

To calculate the positions of the original tokens, we simply make a dot product between the $\tilde{\boldsymbol{H}}^{m}$ and $\boldsymbol{H}$. Then we add a bias term $\boldsymbol{b} \in \mathbb{R}^{L}$ and use the softmax function to get normalized probabilities.

$$\boldsymbol{p}_i = \mathbf{softmax}(\tilde{\boldsymbol{H}}^{\mathrm{m}}_i \boldsymbol{H}^{\top} + \boldsymbol{b}), \quad \boldsymbol{p}_i \in \mathbb{R}^{L} \quad (5)$$

[1] The "sentence" here can be a consecutive text sequence and does not exactly represent the meaning in linguistic view.

[2] To distinguish from the pre-training task (permutation LM) in XLNet (Yang et al., 2019), we use the term "permuted language model (PerLM)" for our PERT in this paper.

Finally, we use the standard cross-entropy loss to optimize the pre-training task.

$$\mathcal{L} = -\frac{1}{M}\sum_{i=1}^{M} \boldsymbol{y}_i \log \boldsymbol{p}_i \quad (6)$$

3.4 Fine-tuning Stage

PERT follows the same paradigm as in BERT to perform fine-tuning on various downstream tasks, as they share the same main architecture. That is to say, PERT can directly fit in any fine-tuning script that is used for BERT or similar. **It is worth noting that, unlike the pre-training stage, we use the natural input sequence rather than changing the word orders in the fine-tuning stage.**

4 Experiments on Chinese Tasks

4.1 Pre-training Setups

We largely follow the training recipe of MacBERT, where we illustrate detail as follows. All models are trained from scratch.

Data: We use the training data as in MacBERT. It consists of the Chinese Wikipedia dump[3], encyclopedia, community question answering, news articles, etc. The total training data has 5.4B words and takes about 20G of disk space. **Tokenization**: We use WordPiece tokenizer (Wu et al., 2016) as in BERT. To detect the Chinese word boundaries, we use LTP (Che et al., 2010) for word segmentation. Note that the Chinese word segmentation is only used for selecting the whole word to perform whole word masking, i.e., only affect which tokens are chosen for masking. The input for PERT is still handled by the WordPiece tokenizer. **Vocabulary**: We directly use the vocabulary of Chinese BERT-base and other PLMs with a vocabulary size of 21128. **Hyper-parameters**: We use a maximum sequence length of 512 throughout the whole pre-training process. **Optimization**: We use a batch size of 416 (base-level) or 128 (large-level) with an initial learning rate of 1e-4. We perform a linear warmup schedule with the first 10k steps. The total training step is 2M. We use ADAM (Kingma and Ba, 2014) with weight decay (rate = 0.1) optimizer with beta values (0.9, 0.999) and an epsilon value 1e-6. **Training Device**: The training was done on a single TPU v3-8 (128G HBM).

Following previous works, we train two PERT models: PERT-base (12-layer, 12-heads, 768-dim) and PERT-large (24-layer, 16-heads, 1024-dim), which are the same as BERT settings.

[3]https://dumps.wikimedia.org/zhwiki/latest/

4.2 Fine-tuning Setups

We choose the following ten popular Chinese NLU datasets. **Machine Reading Comprehension (MRC)**: CMRC 2018 (Cui et al., 2019), DRCD (Shao et al., 2018). **Text Classification (TC)**: XNLI (Conneau et al., 2018), LCQMC (Liu et al., 2018), BQ Corpus (Chen et al., 2018), ChnSentiCorp (Tan and Zhang, 2008), TNEWS (Xu et al., 2020), OCNLI (Hu et al., 2020). **Named Entity Recognition (NER)**: MSRA-NER (SIGHAN 2006) (Levow, 2006), People's Daily[4]. The detailed statistics are listed in Appendix A.

For fair comparisons, we use the models trained on the same amount of corpora (20G), including $\text{BERT}_{\text{base}}$ (i.e., BERT-wwm-ext), $\text{RoBERTa}_{\text{base}}$ (i.e., RoBERTa-wwm-ext), $\text{ELECTRA}_{\text{base}}$, $\text{MacBERT}_{\text{base}}$. Also, to ensure robust experimentation, we carry out each experiment 10 times with different random seeds and report both the maximum and average scores, except for the grammar checking tasks. The fine-tuning scripts are based on original BERT implementation, including `run_classifier.py` for classification tasks, and `run_squad.py` for MRC tasks.[5]

4.3 Results on MRC Tasks

The results on MRC tasks are shown in Figure 3. The evaluation metrics for MRC tasks are the exact match (EM) and F1. As we can see that the proposed PERT yields moderate improvements over MacBERT and is consistently outperform the others, setting state-of-the-art performances on several subsets. This demonstrates that by permuting words in the input sequence, PERT learns both short-range and long-range text inference abilities, which is critical in MRC tasks.

4.4 Results on Text Classification Tasks

The results on text classification (TC) tasks are shown in Table 4. Unfortunately, the proposed PERT does not perform well on text classification tasks. We conjecture that the permuted input text in the pre-training stage brings difficulties in understanding short text compared to MRC tasks.

[4]https://github.com/ProHiryu/bert-chinese-ner
[5]https://github.com/google-research/bert

System	CMRC 2018						DRCD			
	D-EM	D-F1	T-EM	T-F1	C-EM	C-F1	D-EM	D-F1	T-EM	T-F1
$\text{BERT}_{\text{base}}$	67.1 (65.6)	85.7 (85.0)	71.4 (70.0)	87.7 (87.0)	24.0 (20.0)	47.3 (44.6)	85.0 (84.5)	91.2 (90.9)	83.6 (83.0)	90.4 (89.9)
$\text{RoBERTa}_{\text{base}}$	67.4 (66.5)	87.2 (86.5)	72.6 (71.4)	89.4 (88.8)	26.2 (24.6)	51.0 (49.1)	86.6 (85.9)	92.5 (92.2)	85.6 (85.2)	92.0 (91.7)
$\text{ELECTRA}_{\text{base}}$	68.4 (68.0)	84.8 (84.6)	73.1 **(72.7)**	87.1 (86.9)	22.6 (21.7)	45.0 (43.8)	87.5 (87.0)	92.5 (92.3)	86.9 (86.6)	91.8 (91.7)
$\text{MacBERT}_{\text{base}}$	**68.5** (67.3)	**87.9 (87.1)**	**73.2** (72.4)	**89.5 (89.2)**	**30.2** (26.4)	54.0 (52.2)	89.4 **(89.2)**	**94.3 (94.1)**	**89.5 (88.7)**	**93.8 (93.5)**
$\text{PERT}_{\text{base}}$	**68.5 (68.1)**	87.2 **(87.1)**	72.8 (72.5)	89.2 (89.0)	28.7 **(28.2)**	**55.4 (53.7)**	**89.5** (88.9)	93.9 (93.6)	89.0 (88.5)	93.5 (93.2)
$\text{RoBERTa}_{\text{large}}$	68.5 (67.6)	88.4 (87.9)	74.2 (72.4)	90.6 (90.0)	31.5 (30.1)	60.1 (57.5)	89.6 (89.1)	94.8 (94.4)	89.6 (88.9)	94.5 (94.1)
$\text{ELECTRA}_{\text{large}}$	69.1 (68.2)	85.2 (84.5)	73.9 (72.8)	87.1 (86.6)	23.0 (21.6)	44.2 (43.2)	88.8 (88.7)	93.3 (93.2)	88.8 (88.2)	93.6 (93.2)
$\text{MacBERT}_{\text{large}}$	70.7 (68.6)	88.9 (88.2)	74.8 (73.2)	**90.7** (90.1)	31.9 (29.6)	**60.2** (57.6)	**91.2 (90.8)**	**95.6 (95.3)**	**91.7 (90.9)**	**95.6 (95.3)**
$\text{PERT}_{\text{large}}$	**72.2 (71.0)**	**89.4 (88.8)**	**76.8 (75.5)**	**90.7 (90.4)**	**32.3 (30.9)**	59.2 **(58.1)**	90.9 **(90.8)**	95.5 (95.2)	91.1 (90.7)	95.2 (95.1)

Table 3: Results on Chinese MRC tasks: CMRC 2018 (Simplified Chinese) and DRCD (Traditional Chinese). We report the maximum and average scores (in brackets) for each set. Overall best performances are depicted in boldface (base-level and large-level are marked individually). D: Dev set, T: Test set, C: Challenge set.

System	XNLI		LCQMC		BQ Corpus		ChnSentiCorp		TNEWS	OCNLI
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Dev
$\text{BERT}_{\text{base}}$	79.4 (78.6)	78.7 (78.3)	89.6 (89.2)	87.1 (86.6)	**86.4** (85.5)	**85.3** (84.8)	**95.4** (94.6)	95.3 (94.8)	57.0 (56.6)	76.0 (75.3)
$\text{RoBERTa}_{\text{base}}$	80.0 (79.2)	78.8 (78.3)	89.0 (88.7)	86.4 (86.1)	86.0 (85.4)	85.0 (84.6)	94.9 (94.6)	**95.6** (94.9)	**57.4** (56.9)	76.5 (76.0)
$\text{ELECTRA}_{\text{base}}$	77.9 (77.0)	78.4 (77.8)	**90.2 (89.8)**	**87.6 (87.3)**	84.8 (84.7)	84.5 (84.0)	93.8 (93.0)	94.5 (93.5)	56.1 (55.7)	76.1 (75.8)
$\text{MacBERT}_{\text{base}}$	**80.3 (79.7)**	**79.3 (78.8)**	89.5 (89.3)	87.0 (86.5)	86.0 (85.5)	85.2 **(84.9)**	95.2 **(94.8)**	**95.6** (94.9)	**57.4 (57.1)**	**77.0 (76.5)**
$\text{PERT}_{\text{base}}$	78.8 (78.1)	78.1 (77.7)	88.8 (88.3)	86.3 (86.0)	84.9 (84.8)	84.3 (84.1)	94.0 (93.7)	94.8 (94.1)	56.7 (56.1)	75.3 (74.8)
$\text{RoBERTa}_{\text{large}}$	82.1 (81.3)	81.2 (80.6)	90.4 (90.0)	87.0 (86.8)	86.3 (85.7)	**85.8** (84.9)	**95.8** (94.9)	95.8 (94.9)	58.8 (58.4)	78.5 (78.2)
$\text{ELECTRA}_{\text{large}}$	81.5 (80.8)	81.0 **(80.9)**	**90.7 (90.4)**	87.3 **(87.2)**	**86.7 (86.2)**	85.1 (84.8)	95.2 (94.6)	95.3 (94.8)	57.2 (56.9)	78.8 (78.4)
$\text{MacBERT}_{\text{large}}$	**82.4 (81.8)**	**81.3** (80.6)	90.6 (90.3)	**87.6** (87.1)	86.2 (85.7)	85.6 **(85.0)**	95.7 **(95.0)**	**95.9 (95.1)**	**59.0 (58.8)**	**79.0 (78.7)**
$\text{PERT}_{\text{large}}$	81.0 (80.4)	80.4 (80.1)	90.0 (89.7)	87.2 (86.9)	86.3 (85.8)	85.0 (84.8)	94.5 (94.0)	95.3 (94.8)	57.4 (57.2)	78.1 (77.8)

Table 4: Results on text classification tasks: XNLI, LCQMC, BQ Corpus, ChnSentiCorp, TNEWS, and OCNLI.

4.5 Results on NER Tasks

The results on named entity recognition (NER) tasks are shown in Figure 5. We extract the predicted entities and use `seqeval`[6] to evaluate the NER performance in terms of P/R/F metrics. As we can see that PERT yields relatively consistent improvements over all baseline systems, indicating its good abilities in sequence tagging tasks.

Based on all the experiments above, we make several conclusions as follows. 1) PERT yields better performances on MRC and NER tasks, but it does not perform well on TC tasks; 2) The whole word masking and n-gram masking make PERT more sensitive to the word/phrase boundaries, which is helpful in span-extraction MRC and NER tasks. 3) The input sequence for pre-training PERT is shuffled to some extent. According to the results of TC tasks, PERT yields inferior performances. This means text classification tasks are more sensitive to word permutation.

Some of the word permutations will bring a complete meaning change for the input text. This will affect all the fine-tuning tasks presented above. However, TC tasks suffer from this issue more than MRC or NER tasks, as the input text of TC tasks is relatively shorter than the others. MRC and NER task also suffers from word permutation. However, input text for MRC tasks is typically long, and several permutations may not change the whole narrative flow of the passage. For NER tasks, such text permutation may not affect the NER process, as the named entities only take a small proportion of the whole input text.

System	MSRA (Test)	People's Daily (Dev)
$\text{BERT}_{\text{base}}$	95.3 (94.9)	**95.3 (95.1)**
$\text{RoBERTa}_{\text{base}}$	95.5 (95.1)	95.1 (94.9)
$\text{ELECTRA}_{\text{base}}$	95.4 (95.0)	95.1 (94.9)
$\text{MacBERT}_{\text{base}}$	95.3 (95.1)	95.2 (94.9)
$\text{PERT}_{\text{base}}$	**95.6 (95.3)**	**95.3 (95.1)**
$\text{RoBERTa}_{\text{large}}$	95.5 (95.5)	95.7 (95.4)
$\text{ELECTRA}_{\text{large}}$	95.0 (94.8)	94.9 (94.8)
$\text{MacBERT}_{\text{large}}$	**96.2** (95.9)	95.8 (95.7)
$\text{PERT}_{\text{large}}$	**96.2 (96.0)**	**96.1 (95.8)**

Table 5: Results (F-score) on Chinese NER tasks.

4.6 Results on Grammar Checking Tasks

Besides traditional public Chinese NLU tasks, we also test PERT on *Word Order Recovery* (WOR), which is a part of the grammar checking task.

4.6.1 Task Definition and Modeling

The objective of the WOR task is to fix the grammar errors caused by incorrect word orders. To be specific, the examples are the sentences where some words have been moved to incorrect posi-

[6] https://github.com/chakki-works/seqeval

tions. For example, in the sentence "我每天一个吃苹果(I everyday an eat apple)" there is an error span "一个(an)吃(eat)" where the order of "一个(an)" and "吃(eat)" have been swapped. WOR task asks the model to recover the correct sentence "我每天吃一个苹果(I everyday eat an apple)." A sentence can have multiple error spans. To simplify the problem, we only consider the case where only one word has been moved in each span.

We treat the WOR task as a sequence labeling task and take the BIEO tagging scheme. The inputs to the model are incorrect sentences; the label of each word in the sentences stands for whether the word is in the right position (O), or it is the beginning of an error span (B), or it should be moved to the last of the error span (I), or it should be moved to the beginning of the error span (E). For example, the sentence "我每天一个吃苹果" is labeled as "我(O)每(O)天(O)一(B)个(I)吃(E)苹(O)果(O)". With the tagging scheme above, we can easily recover the correct word orders in the sentence.

4.6.2 Results

We test WOR task under four domains (train/dev): Wikipedia (990K/86K), Formal Doc. (1.4M/33K), Customs (682K/34K), Legal (1.8M/13K). We report precision, recall, and F1 scores for the following experiments. The results are shown in Table 6. PERT yields consistent and significant improvements over all baseline systems in terms of all evaluation metrics (P/R/F). This is in accordance with our expectations, as the fine-tuning task is quite similar to the pre-training task of PERT. Though the fine-tuning is performed in a sequence tagging manner (just like NER), it still can benefit from the pre-training of PERT, which focuses on ordering the words in the correct position.

System	Wiki	Formal Doc.	Customs	Legal
$\text{BERT}^{\text{Google}}_{\text{base}}$	79.8	89.6	85.4	92.0
$\text{RoBERTa}_{\text{base}}$	80.4	90.1	86.3	92.2
$\text{ELECTRA}_{\text{base}}$	63.6	84.7	70.3	88.4
$\text{MacBERT}_{\text{base}}$	80.5	90.2	86.4	92.3
$\text{PERT}_{\text{base}}$	**82.9**	**91.2**	**88.2**	**92.9**

Table 6: Results on in-house grammar checking tasks.

5 Experiments on English Tasks

5.1 Pre-training Setups

We largely follow the training recipe of BERT, where we illustrate detail as follows. All models are trained from scratch.

Data: We use English Wikipedia and BookCorpus as the pre-training data, which is widely used in the previous literature. **Tokenization**: We use WordPiece tokenizer (Wu et al., 2016) as in BERT. **Vocabulary**: We directly use the vocabulary of English BERT-base-uncased with a vocabulary size of 30522. **Hyper-parameters**: We use a maximum sequence length of 512 throughout the whole pre-training process. **Optimization**: We use a batch size of 416 (base-level) or 128 (large-level) with an initial learning rate of 1e-4. We perform a linear warmup schedule with the first 10k steps. The total training step is 2M. We use ADAM (Kingma and Ba, 2014) with weight decay (rate = 0.1) optimizer with beta values (0.9, 0.999) and an epsilon value 1e-6. **Training Device**: The training was done on a single TPU v3-8 (128G HBM). Similar to Chinese PERT, we train base and large-level PERT models.

5.2 Fine-tuning Setups

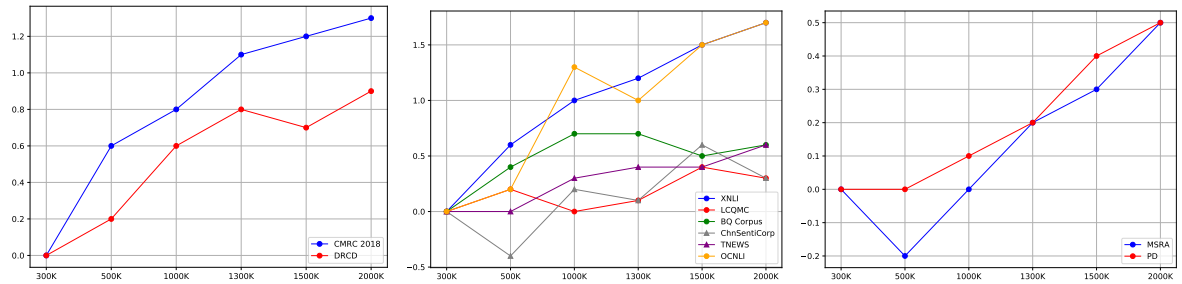
We choose the following six popular English NLU datasets: SQuAD (Rajpurkar et al., 2016), SQuAD 2.0 (Rajpurkar et al., 2018), MNLI (Williams et al., 2018), SST-2 (Socher et al., 2013), CoLA (Warstadt et al., 2019), and MRPC (Dolan and Brockett, 2005). The fine-tuning settings for each task are shown in Appendix A. Note that, for all experiments on English tasks, we do not use additional tricks that were used in other papers, such as data augmentation, fine-tuning from MNLI checkpoints, etc. We report five-run average scores for our experiments.

System	SQuAD EM	SQuAD F1	SQuAD 2.0 EM	SQuAD 2.0 F1	MNLI Acc	SST-2 Acc	CoLA M.C.	MRPC Acc
$\text{BERT}_{\text{base}}$	80.8	88.5	-	-	84.4	**92.7**	60.6	86.7
$\text{BERT}_{\text{base}}{}^{\dagger}$	81.2	88.5	72.4	75.4	84.4	92.6	59.3	86.0
$\text{RoBERTa}_{\text{base}}$	-	90.4	-	79.1	84.7	92.5	-	-
$\text{XLNet}_{\text{base}}$	-	-	**78.4**	**81.3**	**85.8**	92.6	-	-
$\text{ALBERT}_{\text{base}}$	82.1	89.3	76.1	79.1	81.9	89.4	-	-
$\text{PERT}_{\text{base}}$	**84.8**	**91.3**	78.3	81.0	84.5	92.0	**61.2**	**87.5**
$\text{BERT}_{\text{large}}$	84.1	90.9	78.7	81.9	86.6	93.2	60.6	88.0
$\text{BERT}_{\text{large-wwm}}{}^{\dagger}$	87.4	93.4	82.8	85.6	87.3	93.4	63.1	87.2
$\text{RoBERTa}_{\text{large}}$	-	93.6	-	87.3	**89.0**	**95.3**	-	-
$\text{XLNet}_{\text{large}}$	**88.2**	**94.0**	**85.1**	**87.8**	88.4	94.4	65.2	**90.0**
$\text{ALBERT}_{\text{large}}$	84.1	90.9	79.0	82.1	83.8	90.6	-	-
$\text{PERT}_{\text{large}}$	87.4	93.3	83.5	86.3	87.6	93.4	**65.7**	87.3

Table 7: Development set results on English NLU tasks. The system marked with † means the reproduced results (rerun). We only list those model variants that were trained on Wikibooks. M.C.: Matthews correlation.

5.3 Results

We report EM/F1 for SQuAD and SQuAD 2.0, accuracy for MNLI, SST-2, and MRPC, Matthews



(a) MRC (b) TC (c) NER

Figure 2: Performance on MRC, TC, and NER tasks with different pre-training steps.

System	CMRC 2018						XNLI		TNEWS	OCNLI	Average
	D-EM	D-F1	T-EM	T-F1	C-EM	C-F1	Dev	Test	Dev	Dev	
PERT$_{\text{base}}$ (no limit)	65.4	85.0	70.2	87.3	22.4	45.6	74.8	74.4	54.5	70.6	65.02
└Word	59.3	80.6	64.8	83.5	12.6	32.2	73.2	72.1	53.2	69.3	60.08
└N-gram	62.2	82.5	67.3	84.8	17.2	36.1	73.4	73.2	53.8	69.5	62.00
└Sentence	63.7	83.2	69.1	86.2	16.8	38.0	74.3	73.0	54.1	70.0	62.84

Table 8: Permutation within different granularities: word, N-gram, and sentence. We report five-run average scores.

correlation for CoLA. The results are shown in Table 7. Similar to the results of Chinese NLU tasks, PERT$_{\text{base}}$ yields better performance on MRC tasks and moderate improvements on a few TC tasks. However, we noticed that PERT$_{\text{large}}$ performs worse than the others on most of the tasks.

6 Analysis

In this section, we will look deeper into PERT with quantitative analyses. We use Chinese PERT$_{\text{base}}$ for all analyses in the following subsections.

6.1 Performance on Different Training Steps

Just like the performance curve for fine-tuning tasks, the optimum performance for each fine-tuning task may not happen at the same pre-training step. To investigate the performance of different types of the fine-tuning task, we plot their performance on 300K, 500K, 1000K, 1300K, 1500K, and 2000K pre-training steps. We use the performance of 300K as the baseline and calculate the gap to the baseline in terms of different training steps, where a positive value means a performance growth and a negative value for decrease. We report F1 for MRC tasks, accuracy for TC tasks, and F1 for NER tasks. The results are shown in Figure 2.

For MRC and NER tasks, we see relatively consistent performance improvements with the growth of pre-training steps. However, for TC tasks, the performance for some tasks reaches its optimum at 1000K to 1300K pre-training steps, such as BQ Corpus and TNEWS. These results indicate that it is necessary to "harvest" the pre-trained model in an earlier pre-training step if the performance of a specific task is considered.

6.2 Different Permutation Granularities

In the formulation of PERT, we did not make restrictions on the permuting granularity, i.e., we directly permute input text in token-level, and any word can be replaced by another word in the passage. However, *what if we permute **within** a complete word/N-gram/sentence?* Such restrictions will make the input text more readable as the permutation is done within a linguistic unit rather than the whole sequence. We compare the performance by permuting the input text within different granularities. We pre-train PERT$_{\text{base}}$ with 300k steps and observe their performances on CMRC 2018, XNLI, TNEWS, and OCNLI tasks. The results are shown in Table 8.

Among various permutation granularities, PERT with no permutation limit yields the best performances on both tasks. We noticed that if we choose a smaller granularity (such as word), the system performance goes the worst. Though using a smaller granularity will make the input text more readable, it is less challenging for the pre-training task and thus cannot extract useful semantics for text representation.

System	CMRC 2018						XNLI		TNEWS	OCNLI	Average
	D-EM	D-F1	T-EM	T-F1	C-EM	C-F1	Dev	Test	Dev	Dev	
PERT$_{\text{base}}$ (local)	64.1	84.0	69.1	86.5	21.0	43.3	74.1	74.4	54.5	70.6	64.16
└Global	61.1	81.4	65.8	84.3	15.8	36.2	73.6	74.0	55.4	69.4	61.70
└Local + Global	63.2	83.6	67.7	85.8	19.3	42.0	74.6	74.6	55.1	70.0	63.59

Table 9: Comparison of local prediction (default) and global prediction.

System	CMRC 2018						XNLI		TNEWS	OCNLI	Average
	D-EM	D-F1	T-EM	T-F1	C-EM	C-F1	Dev	Test	Dev	Dev	
PERT$_{\text{base}}$ (partial)	64.1	84.0	69.1	86.5	21.0	43.3	74.1	74.4	54.5	70.6	64.16
└Full Prediction	63.7	84.1	68.0	86.1	18.7	40.9	74.3	73.9	54.2	71.0	63.49

Table 10: Comparison of partial prediction (default) and full prediction.

6.3 Global v.s. Local Prediction

The output space of PERT is the whole input sequence (the length of input), which is different from the MLM that predicts in the vocabulary space. In this section, we modify the output of PERT to directly predict the original word in the vocabulary space instead of its position in the sequence. We pre-train PERT$_{\text{base}}$ with 300K training steps.[7] The results are shown in Table 9.

The experimental results show that predicting in the vocabulary space is not necessary for PerLM, which is significantly worse than predicting on the local input sequence, where an average of 2.46 performance gap is observed. Unfortunately, combining both local and glocal prediction (i.e., both predict the original token's position and the exact word in the vocabulary space) does not yield better performance than the local prediction only. This reminds us that predicting the missing word in the global (vocabulary) space is not always necessary for pre-training tasks.

6.4 Partial v.s. Full Prediction

Most of the pre-trained language model that adopts MLM-like pre-training task uses partial prediction. The partial prediction only makes predictions on the masked tokens rather than the whole input sequence (full prediction). Through the experiments in ELECTRA (Clark et al., 2020), the authors present that the full prediction yields better performance than the partial prediction, and thus ELECTRA adopts full prediction for the discriminator using replaced token detection (RTD). However, does it apply to other pre-training tasks other than RTD? In this experiment, we compare the results on pre-training with full prediction and partial prediction. We pre-train PERT$_{\text{base}}$ with 300K training steps. The results are shown in Table 10.

As we can see that full prediction does not yield better performance in PERT, where its performance is significantly worse in MRC tasks and similar performance in text classification tasks. This demonstrates that it is not always effective to use full prediction in pre-training tasks and should be adjusted to the nature of the designed pre-training task.

7 Conclusion

In this paper, we propose a new pre-trained language model, called PERT, which uses Permuted Language Model (PerLM) as the pre-training task. The objective of PerLM is to predict the position of the original token in a shuffled input text, which is different from the MLM-like pre-training task. To evaluate the performance of PERT, we carried out extensive experiments on both Chinese and English NLU tasks. The experimental results show that PERT yields improvements on MRC and NER tasks but does not perform that well on TC tasks. Additional quantitative analyses on PERT are also performed to better understand our model, and we make several observations that are different from the previous works. We hope the trial of PERT can inspire our community to design non-MLM-like pre-training tasks for text representation learning.

Limitations

In this paper, we propose PERT to explore whether it is possible to learn text representations from a

[7] Note that the hyperparameter for this PERT (as well as the one in Section 6.4) is slightly different from PERT$_{\text{base}}$ in Section 6.2, and thus their baseline results are different.

non-MLM-like pre-training task. The presented results (on both Chinese and English) show that PERT achieves better performance on MRC and NER tasks but not on classification tasks. The inferior results on classification tasks remind us that a pre-training task might not always be friendly to all NLU tasks and sometimes may pose another issue in creating pre-trained language models. In this study, the proposed method does resolve one of the most famous issues in the auto-encoding pre-trained model: "pre-training and fine-tuning discrepancy", that PERT does not employ any masking tokens. However, the permuted word order makes the pre-training sentence unnatural to the original sentence, which creates another type of discrepancy. Through our experiments and analysis, we think that though it is necessary to study those existing issues in the current pre-training tasks, we might seek another path – try to explore how to place the right model on the right task, i.e., embrace both positive and negative effect and use them in a positive way.

References

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13–16. Association for Computational Linguistics.

Jing Chen, Qingcai Chen, Xin Liu, Haijun Yang, Daohe Lu, and Buzhou Tang. 2018. The BQ corpus: A large-scale domain-specific Chinese corpus for sentence semantic equivalence identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4946–4951, Brussels, Belgium. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 657–668, Online. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. 2021. Pre-training with whole word masking for chinese bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3504–3514.

Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2019. A Span-Extraction Dataset for Chinese Machine Reading Comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5886–5891, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kuebler, and Larry Moss. 2020. Ocnli: Original chinese natural language inference. In *Findings of EMNLP*.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, pages 1–17.

Gina-Anne Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of*

the Fifth SIGHAN Workshop on Chinese Language Processing, pages 108–117, Sydney, Australia. Association for Computational Linguistics.

Xin Liu, Qingcai Chen, Chong Deng, Huajun Zeng, Jing Chen, Dongfang Li, and Buzhou Tang. 2018. Lcqmc: A large-scale chinese question matching corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1952–1962.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Chih Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. 2018. Drcd: a chinese machine reading comprehension dataset. *arXiv preprint arXiv:1806.00920*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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

Songbo Tan and Jin Zhang. 2008. An empirical study of sentiment analysis for chinese documents. *Expert Systems with applications*, 34(4):2622–2629.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. 2019. Structbert: Incorporating language structures into pretraining for deep language understanding. *arXiv preprint arXiv:1908.04577*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. 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, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

A Hyperparameters of Fine-tuning

B Full Results on NER

C Full Results on Grammar Checking Tasks

Task	Dataset	MaxLen	Batch	Epoch	InitLR	Train #	Dev #	Test #
MRC	CMRC 2018	512	64	2	3e-5	10K	3.2K	4.9K
	DRCD	512	64	2	3e-5	27K	3.5K	3.5K
TC	XNLI	128	64	2	3e-5	392K	2.5K	5K
	LCQMC	128	64	3	2e-5	240K	8.8K	12.5K
	BQ Corpus	128	64	3	3e-5	100K	10K	10K
	ChnSentiCorp	256	64	3	2e-5	9.6K	1.2K	1.2K
	TNEWS	128	64	3	2e-5	53.3K	10K	10K
	OCNLI	128	64	3	2e-5	56K	3K	3K
NER	MSRA-NER	256	64	5	3e-5	45K	-	3.4K
	People's Daily	256	64	5	3e-5	51K	4.6K	-

Table 11: Hyper-parameter settings and data statistics for Chinese tasks.

Dataset	MaxLen	Batch	Epoch	InitLR	Train #	Dev #
SQuAD 1.1	512	64	2	3e-5	87.6K	10.6K
SQuAD 2.0	512	64	2	3e-5	130.3K	11.9K
MNLI	256	64	3	3e-5	392.7K	9.8K
SST-2	128	64	3	3e-5	67.3K	0.9K
CoLA	128	64	10	2e-5	8.6K	1.0K
MRPC	512	64	5	3e-5	3.7K	0.4K

Table 12: Data statistics and hyper-parameter settings for English tasks.

System	MSRA-NER (Test)			People's Daily (Dev)		
	P	R	F	P	R	F
$\text{BERT}_{\text{base}}$	95.2 (94.8)	95.4 (95.1)	95.3 (94.9)	95.3 **(95.1)**	95.3 (95.1)	**95.3 (95.1)**
$\text{RoBERTa}_{\text{base}}$	95.3 (94.9)	95.6 (95.4)	95.5 (95.1)	94.9 (94.8)	95.3 (95.1)	95.1 (94.9)
$\text{ELECTRA}_{\text{base}}$	95.0 (94.5)	**95.9** (95.4)	95.4 (95.0)	94.8 (94.7)	95.3 **(95.2)**	95.1 (94.9)
$\text{MacBERT}_{\text{base}}$	95.2 (94.9)	95.4 (95.4)	95.3 (95.1)	94.9 (94.6)	95.6 (95.1)	95.2 (94.9)
$\text{PERT}_{\text{base}}$	**95.4 (95.2)**	95.5 **(95.5)**	**95.6 (95.3)**	**95.4 (95.1)**	95.2 (95.0)	**95.3 (95.1)**
$\text{RoBERTa}_{\text{large}}$	95.4 (95.3)	95.7 (95.7)	95.5 (95.5)	95.7 (95.4)	95.7 (95.4)	95.7 (95.4)
$\text{ELECTRA}_{\text{large}}$	94.9 (94.8)	95.5 (95.0)	95.0 (94.8)	94.8 (94.6)	95.3 (95.3)	94.9 (94.8)
$\text{MacBERT}_{\text{large}}$	96.3 (95.8)	96.3 (95.9)	**96.2** (95.9)	95.8 (95.6)	95.8 **(95.7)**	95.8 (95.7)
$\text{PERT}_{\text{large}}$	**96.4 (95.9)**	**96.4 (96.1)**	**96.2 (96.0)**	**96.3 (96.0)**	96.0 **(95.7)**	**96.1 (95.8)**

Table 13: Results on Chinese named entity recognition (NER) tasks.

System	Wikipedia			Formal Doc.			Customs			Legal		
	P	R	F	P	R	F	P	R	F	P	R	F
$\text{BERT}^{\text{Google}}_{\text{base}}$	83.6	76.3	79.8	92.1	87.1	89.6	85.7	85.1	85.4	94.3	89.8	92.0
$\text{RoBERTa}_{\text{base}}$	84.2	76.9	80.4	92.6	87.7	90.1	86.8	85.9	86.3	94.6	90.0	92.2
$\text{ELECTRA}_{\text{base}}$	69.9	57.8	63.6	88.1	81.6	84.7	69.6	71.0	70.3	91.7	85.4	88.4
$\text{MacBERT}_{\text{base}}$	84.3	77.1	80.5	92.7	87.8	90.2	86.4	86.5	86.4	94.6	90.1	92.3
$\text{PERT}_{\text{base}}$	**86.5**	**79.5**	**82.9**	**93.6**	**89.0**	**91.2**	**88.3**	**88.0**	**88.2**	**95.2**	**90.7**	**92.9**

Table 14: Results on in-house grammar checking tasks.