ERNIE-DOC: The Retrospective Long-Document Modeling Transformer

Siyu Ding*, **Junyuan Shang***, **Shuohuan Wang**, **Yu Sun**, **Hao Tian**,
Hua Wu and **Haifeng Wang**
Baidu Inc., China
{dingsiyu, shangjunyuan, wangshuohuan, sunyu02,
tianhao, wu_hua, wanghaifeng}@baidu.com

Abstract

Transformers are not suited for processing long document input due to its quadratically increasing memory and time consumption. Simply truncating a long document or applying the sparse attention mechanism will incur the context fragmentation problem or inferior modeling capability with comparable model size. In this paper, we propose ERNIE-DOC, a document-level language pretraining model based on Recurrence Transformers (Dai et al., 2019). Two well-designed techniques, namely the retrospective feed mechanism and the enhanced recurrence mechanism enable ERNIE-DOC with much longer effective context length to capture the contextual information of a whole document. We pretrain ERNIE-DOC to explicitly learn the relationship among segments with an additional document-aware segment reordering objective. Various experiments on both English and Chinese document-level tasks are conducted. ERNIE-DOC achieves SOTA language modeling result of 16.8 ppl on WikiText-103 and outperforms competitive pretraining models on most language understanding tasks such as text classification, question answering by a large margin.

1 Introduction

Transformers (Vaswani et al., 2017) have achieved remarkable improvements in a wide range of natural language tasks including language modeling (Dai et al., 2019), text classification (Yang et al., 2019), question answering (Devlin et al., 2018; Radford et al., 2019). This success is due in large part to the self-attention mechanism which enables the network to capture contextual information from the whole input sequence. Nevertheless, the memory usage and computation complexity caused by the self-attention mechanism grows quadratically

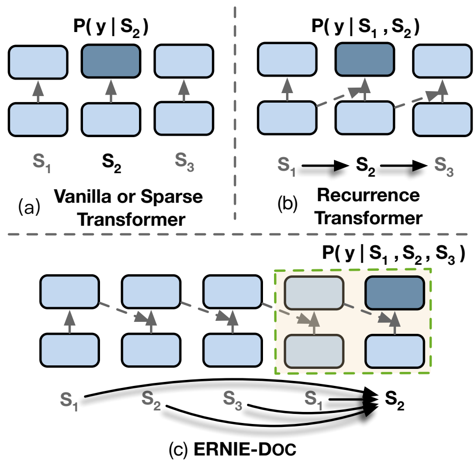

Figure 1: An illustration of available contextual information utilized by transformers variants, where a long document $\mathcal{D}$ is partitioned into three segments $\mathbf{S}_i (i \in [1, 2, 3])$. When training on $\mathbf{S}_2$, (a) and (b) optimize the pretraining objective merely depending on the contextual information from the current segment or the segment in the forward pass, while ERNIE-DOC can utilize the contextual information of the whole document for each segment.

with the sequence length, causing the expensive cost for it to process a long document on current hardware.

Nowadays, most prominent pretrained models like BERT (Devlin et al., 2018) are performed on fixed-length input segments of maximum 512 tokens due to the above-mentioned limitation. Thus, a long document input has to be partitioned into smaller segments with manageable size, leading to the loss of important cross-segments information, i.e., the *context fragmentation* problem (Dai et al., 2019), as shown in Fig. 1(a). To mitigate such problem that lack of interactions among the partitioned segments of a long document, *Recurrence Transformers* (Dai et al., 2019; Rae et al., 2019) permits the contextual information from previous segments to help compute the hidden states for a new segment via maintaining a memory component

*indicates equal contribution.

from the previous activation, enabling the ability to model long document. Another effort, *Sparse Attention Transformers* (Child et al., 2019; Tay et al., 2020; Beltagy et al., 2020; Zaheer et al., 2020) focus on reducing the complexity of self-attention operation to explicitly promotes the modeling length, but still up to a restricted context length, i.e., 4096, on account of the limited resources.

We argue that existing strategies are less effective and reliable since ***the contextual information of a whole document*** *is still not available for each relevant segment during training phase*. As depicted in Fig. 1, when training on the segment S_2, the model is ideally optimized by maximizing $P(y \mid (S_1, S_2, S_3))$ conditioned on the contextual information of the whole document $\mathcal{D} = \{S_1, S_2, S_3\}$ in contrast to the following sub-optimal solutions: $P(y \mid S_2)$ for Vanilla/Sparse Transformers[1] and $P(y \mid (S_1, S_2))$ for Recurrence Transformers.

To resolve the aforementioned limitation, we propose ERNIE-DOC (the Retrospective Long-Document Modeling Transformer) based on the Recurrence Transformers paradigm. Inspired by the human reading behaviour of skimming a document first, and then looking back upon it attentively, we design a **retrospective feed mechanism** in which segments from a document are fed as input twice. As a result, each segment in the retrospective phase could explicitly fuse the semantic information of the entire document learned in the skimming phase which relieves the problem of context fragmentation.

However, simply incorporating the retrospective feed mechanism into recurrence transformers is infeasible as the maximum effective context is limited by the number of layers (Dai et al., 2019) as shown in Fig. 1 (b). Thus, we present an **enhanced recurrence mechanism**, a drop-in replacement for recurrence transformer by changing the shifting-one-layer-downwards recurrence to the same-layer recurrence. In this way, the past higher-level representations could be exploited to enrich the future lower-level representations and the maximum effective context is expanded.

Moreover, we introduce a **segment reordering objective** to pretrain a document-level model. Specifically, it is a document-aware task of predicting the correct order of the permuted set of segments of a document, to model the relationship among segments directly, which allows ERNIE-DOC to build full document representations for prediction. It is analogue to the sentence reordering task raised in ERNIE 2.0 (Sun et al., 2020), but at a segment-level of granularity, spanning (commonly) multiple training steps.

We first evaluate ERNIE-DOC on autoregressive word-level language modeling using an enhanced recurrence mechanism, which allows the model to process a document of up to an infinite number of words in theory. ERNIE-DOC achieves state-of-the-art results on the WiKiText-103 benchmark dataset, demonstrating its effectiveness in long document modeling. Then, to evaluate the potential of ERNIE-DOC on document-level natural language understanding (NLU) tasks, we pretrain English ERNIE-DOC on the text corpora utilized in BigBird (Zaheer et al., 2020) continuing from the RoBERTa released checkpoint and Chinese ERNIE-DOC on the text corpora utilized in ERNIE 2.0 (Sun et al., 2020) from scratch. After pretraining, we finetune ERNIE-DOC on a wide range of downstream language tasks including text classification and QA in English and Chinese. Empirically, ERNIE-DOC consistently outperforms RoBERTa on various benchmarks and shows significant improvements over other well-performed long-text pretraining models on most tasks.

2 Related Work

Sparse Attention Transformers have been largely explored (Child et al., 2019; Tay et al., 2020; Beltagy et al., 2020; Zaheer et al., 2020). The key idea is to sparsify the self-attention operation which scales quadratically with the sequence length. To name a few, the Sparse Transformer (Child et al., 2019) used a dilated sliding window that reduces the complexity to $\mathcal{O}(L\sqrt{L})$ where L is the sequence length. Reformer (Kitaev et al., 2020) further reduced the complexity to $\mathcal{O}(L\sqrt{L})$ by using locality-sensitive hashing attention to compute nearest neighbors. BP-transformer (Ye et al., 2019) proposed to binary partition the input sequence. Recently, Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020) are proposed and both achieve state-of-the-art performance on a variety of long document tasks. They reduce the complexity of self-attention to $\mathcal{O}(L)$ theoretically by combining random attention, window attention, and global attention. However, it has been proved in Zaheer

[1]For Sparse Transformers, the length of the segment S_2 could be longer up to 4096 in Beltagy et al. (2020); Zaheer et al. (2020)

et al. (2020) that sparse attention mechanisms can not universally replace dense attention mechanisms, and demonstrated that solving a simple problem of finding the furthest vector requires $\Omega(n)$-layers sparse attention but only $\mathcal{O}(1)$-layer of dense attention mechanism. In addition, the above methods require customized CUDA kernel or TVM programming to implement the sparse attention which is not maintainable and hard to use. In this work, we take a different approach to adapting recurrence transformers in a pretraining-then-finetuning setting for modeling a long document.

Recurrence Transformers (Dai et al., 2019; Rae et al., 2019) have been successful in generative language modeling. They apply the transformer decoder as a parametric model for each conditional distribution in $p(\boldsymbol{x}) = \prod_{t=1}^{L} p(x_t|\boldsymbol{x}_{<t})$ ($\boldsymbol{x}$ denotes a text sequence). To capture long dependency in a long text sequence, they process the text in segments from left to right based on segment recurrence mechanism (Dai et al., 2019) The segment recurrence mechanism maintains a memory bank of past activations at each layer to preserve a history of context. Compressive Transformer (Rae et al., 2019) showed adding a compressive memory bank to store sufficiently old activations instead of just discarding them would benefit long-range sequence learning. However, these methods perform in a left-to-right manner which limits their capacity for discriminative language understanding tasks that require bidirectional information. XLNet (Yang et al., 2019) proposed a permutation language modeling objective to construct bidirectional information and achieve superior performance in multiple NLP tasks while its application to long document modeling tasks is still largely unexplored. Our method ERNIE-Doc builds on the ideas of the Recurrence Transformers to 1) tackle the limitation of recurrence transformers to utilize bidirectional contextual information, and 2) enhance the behavior of segment recurrence mechanism to capture longer dependency.

Hierarchical Transformers (Zhang et al., 2019; Lin et al., 2020) have made significant progress on numerous document-level tasks, such as document summarization (Zhang et al., 2019), document ranking (Lin et al., 2020) and etc. Similar to Vanilla Transformers, Hierarchical Transformers also split the long document into shorter segments with manageable length, and then feed them independently to produce the corresponding segment-level semantic representations. Differently, there are separate transformer layers to process the concatenation of these representations to produce the document-level semantic representation. Hierarchical Transformers ignore the contextual information from remaining segments when processing each segment of a long document, thus leading to the *context fragmentation* problem.

3 Proposed Method

In this section, we firstly describe the background which ERNIE-Doc builds on (Sec. 3.1), then we present the detailed implementation of ERNIE-Doc, including retrospective feed mechanism in Sec. 3.2, enhanced recurrence mechanism in Sec. 3.3 and segment reordering objective in Sec. 3.4.

3.1 Background

Formally, a long document $\mathcal{D}$ is sliced into T sequential segments denoted as $\{S_1, S_2, ..., S_T\}$ where $S_\tau = \{x_{\tau,1}, x_{\tau,2}, ..., x_{\tau,L}\}$ is the τ-th segment with length L. Vanilla, Sparse and Recurrence Transformers conduct different strategies to produce the hidden state $\mathbf{h}_\tau^n \in \mathbb{R}^{L \times d}$ for segment S_τ at n-th layer as follows:

$$\begin{aligned}
\widetilde{\mathbf{h}}_{\tau+1}^{n-1} &= \begin{cases} \mathbf{h}_{\tau+1}^{n-1}, & \text{Vanilla or Sparse Transformers} \\ [\mathbf{SG}(\mathbf{h}_\tau^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}], & \text{Recurrence Transformers} \end{cases} \\
\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n &= \mathbf{h}_{\tau+1}^{n-1}\mathbf{W}_q^\top, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_k^\top, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_v^\top \\
\mathbf{h}_{\tau+1}^n &= \text{Transformer-Block}\,(\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n)
\end{aligned} \tag{1}$$

where $\mathbf{q}, \mathbf{k}, \mathbf{v} \in \mathbb{R}^{L \times d}$ are query, key and value vectors separately with dimension d, $\mathbf{W}_* \in \mathbb{R}^{d_* \times d}$ are learnable linear projection parameters, and the function SG() stands for stop-gradient, the notation $[\circ]$ denotes the concatenation of two hidden states along the length dimension. In contrast to vanilla or sparse transforms where $\mathbf{h}_\tau^n$ is produced only employing itself, recurrence transformers introduce a segment-level recurrence mechanism to promote interaction across segments. The hidden state computed for the previous segment $\mathbf{h}_\tau^{n-1}$ is maintained to be utilized as an auxiliary context to help process the next new segment. However, from the concatenation part in Eq. 3.1, i.e. $[\mathbf{h}_u \circ \mathbf{h}_v]$, there is apparently a constraint on u, v where $u = v - 1$ meaning that the current hidden state could only fuse information from the previous hidden state. Thus, the contextual information of a whole document is not available for each relevant segment.

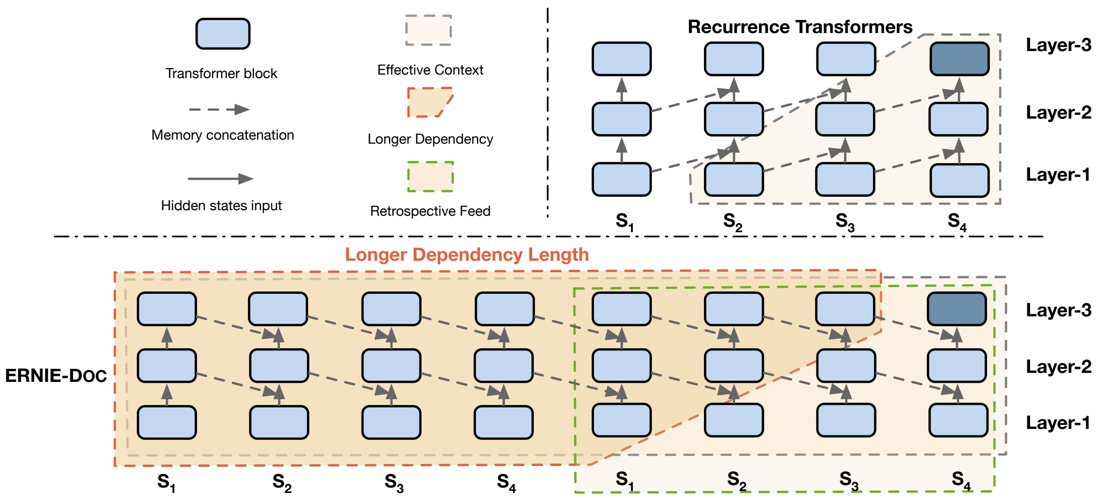

Figure 2: Illustrations of ERNIE-DOC and Recurrence Transformers where models with 3 layers take as input a long document $\mathcal{D}$ which sliced into four segments $\mathbf{S}_i, i \in [1, 2, 3, 4]$. **Recurrence Transformers (Upper-right)**: When training on $\mathbf{S}_4$, it fuses the contextual information of previous two consecutive segments $\mathbf{S}_2$ and $\mathbf{S}_3$ since the largest effective context length grows linearly w.r.t the number of layers. **ERNIE-DOC (Lower)**: In contrast, the effective context length is much longer aided by enhanced recurrence mechanism (Sec 3.3) so that $\mathbf{S}_4$ could fuse the information of $\mathbf{S}_1$ discarded by Recurrence Transformers. Moreover, segments in retrospective feed phase (shown in green dotted box) have bidirectional information of $\mathcal{D}$ enabled by retrospective feed mechanism (Sec 3.2)

3.2 Retrospective Feed Mechanism

ERNIE-DOC develops a retrospective feed mechanism to address the limitation, that is lacking *the contextual information of a whole document for each relevant segment*. In detail, the segments from a long document are fed as input twice. Mimicking the human reading behavior, we refer to the first and second input-taking phase as the skimming phase and the retrospective phase respectively. In the skimming phase, we conduct the recurrence mechanism to cache hidden states for each segment. While in the retrospective phase, we reuse the cached hidden states from the skimming phase to enable the bi-directional information flow. Formally, we rewrite Eq. 3.1 as:

$$\widehat{\mathbf{H}}^{n-1}_{1:L} = [\widehat{\mathbf{h}}^{n-1}_1 \circ \widehat{\mathbf{h}}^{n-1}_2 \cdots \widehat{\mathbf{h}}^{n-1}_L], \text{ (skimming phase)}$$
$$\widetilde{\mathbf{h}}^{n-1}_{\tau+1} = [\mathbf{SG}(\widehat{\mathbf{H}}^{n-1}_{1:L} \circ \mathbf{h}^{n-1}_{\tau}) \circ \mathbf{h}^{n-1}_{\tau+1}], \text{ (retro. phase)} \quad (2)$$

where $\widehat{\mathbf{H}}^{n-1}_{1:L} \in \mathbb{R}^{(T*L)\times d}$ denotes cached hidden states in the skimming phase. In this way, the current hidden state $\widetilde{\mathbf{h}}^{n-1}_{\tau+1}$ is ensured to capture bidirectional contextual information of a whole input. Henceforth, the main issue is how to implement $\widehat{\mathbf{H}}^{n-1}_{1:L}$ in a memory and computation efficient way.

By rethinking the segment-level recurrence (Dai et al., 2019), we observe that the effective context being utilized can go way beyond just two segments. Consequently, the largest possible dependency length grows linearly w.r.t the number of layers. In Eq. 3.2, $\widehat{\mathbf{h}}^{n-1}_{i}$ actually contains the information from $\widehat{\mathbf{h}}^{n-1}_{i-(n-1)}$ to itself. Thus, we could discard the redundant hidden states in the skimming phase and reduce $\widehat{\mathbf{H}}^{n-1}_{1:L}$ to $[\widehat{\mathbf{h}}^{n-1}_1 \circ \widehat{\mathbf{h}}^{n-1}_{1+n} \cdots \widehat{\mathbf{h}}^{n-1}_{L-n} \circ \widehat{\mathbf{h}}^{n-1}_{L}] \in \mathbb{R}^{\lceil T/(n-1)\rceil \times d}$. Further, an additional compressive memory (Rae et al., 2019) could be used to store $\widehat{\mathbf{H}}^{n-1}_{1:L}$ in a compressed format $\widehat{\mathbf{h}}^{n-1}_{1:L} \in \mathbb{R}^{L_c \times d}$ where $L_c \ll \lceil T/(n-1)\rceil$ is the length of compressive memory. However, compressive memory still incurs extra computation and memory cost compared to the mechanism we will present below.

3.3 Enhanced Recurrence Mechanism

In order to utilize the retrospective feed mechanism in practice without extra computation and memory cost, an ideally strategy is to ensure the current hidden state $\mathbf{h}^{n-1}_{\tau+1}$ containing information of $\widehat{\mathbf{H}}^{n-1}_{1:L}$ in the skimming phase without explicitly taking as input $\widehat{\mathbf{H}}^{n-1}_{1:L}$. Essentially, we should tackle the limited effective context in segment-level recurrence mechanism. Here, we introduce an enhanced recurrence mechanism, a drop-in replacement for segment-level recurrence mechanism by changing the shifting-one-layer-downwards recurrence to the same-layer recurrence as follows,

$$\widetilde{\mathbf{h}}^{n-1}_{\tau+1} = [\, \boxed{\mathbf{SG}(\mathbf{h}^{n}_{\tau})} \circ \mathbf{h}^{n-1}_{\tau+1}] \quad (3)$$

where the only difference with Eq. 3.1 is marked in red.

As shown in Fig. 2, by combining the retrospective feed mechanism with enhanced recurrence mechanism, every segment in the retrospective phase (shown in the box with green dotted border) has bidirectional contextual information of the whole text input. We successfully model a longer effective context (shown in the box with orange dotted border) than traditional recurrence transformers without extra memory and computation cost. Another benefit coming with the enhanced recurrence scheme is that the past higher-level representations could be exploited to enrich the future lower-level representations.

3.4 Segment Reordering Objective

Besides the **Masked Language Model (MLM) Objective** (Liu et al., 2019), we introduce an additional document-aware task named **Segment Reordering Objective** for pretraining. Benefit from the much longer dependency length caused by the enhanced recurrence mechanism, the goal of segment reordering objective is to predict the correct order for the permuted set of segments of a long context, to explicitly learn the relationship among segments. During the pretraining process of this task, a long text input $\mathcal{D}$ is randomly partitioned into 1 to m chunks firstly, and then all of the combinations are shuffled by a random order. As shown in Fig. 3, $\mathcal{D}$ is partitioned into 3 chunks and then permuted, i.e., $\mathcal{D} = \{C_1, C_2, C_3\} \Longrightarrow \hat{\mathcal{D}} = \{C_2, C_3, C_1\}$, where C_i denotes the i-th chunk. Afterwards, the permuted long context $\hat{\mathcal{D}}$ is split into T sequential segments as common practice, denoted as $\hat{\mathcal{D}} = \{S_1, S_2, ..., S_T\}$. We let the pretrained model to reorganize these permuted segments, modeled as a K-class classification problem where $K = \sum_{i=1}^{m} i!$. The pretraining objective is summarized as follows for τ-th input segment,

$$\max_{\theta} \log p_\theta(S_\tau|\hat{S}_\tau) + \mathbb{1}_{\tau=T} \log p_\theta(\mathcal{D}|\hat{\mathcal{D}}) \quad (4)$$

where $\hat{S}_\tau$ is the corrupted version of S_τ by randomly setting a portion of tokens to `[MASK]`, $\hat{\mathcal{D}}$ is the permutated version of $\mathcal{D}$, θ is the model parameters and $\mathbb{1}_{\tau=T}$ indicates that segment reordering objective is only optimized at T-th step.

4 Experiments

4.1 Autoregressive Language Modeling

Autoregressive language modeling aims to estimate the probability distribution of an existing token/character based on the previous tokens/characters in an input sequence. To compare with previous work, we conduct experiments on word-level LM, i.e., WikiText-103 (Merity et al., 2016), which is a document-level language modeling dataset.

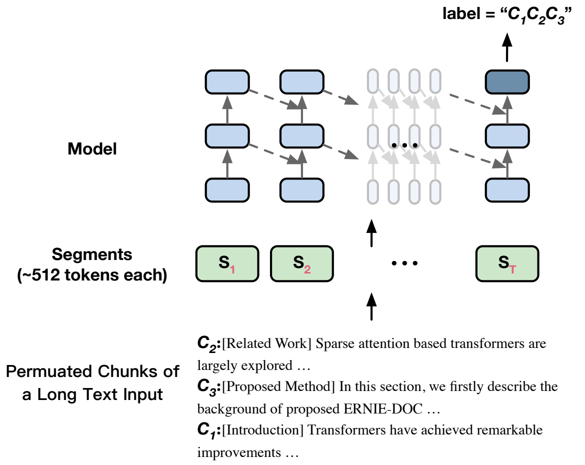

Figure 3: Illustrations of the Segment Reorder Objective.

4.1.1 Experimental Setup

For autoregressive language modeling, we use enhanced memory Transformer-XL (Dai et al., 2019): That is, we employ our enhanced recurrence mechanism instead of the primitive one. Additionally, we introduce the segment-aware mechanism to Transformer-XL, which is proposed by Segatron (Bai et al., 2020). Following Transformer-XL, we train a base size model and a large size model. The base model consists of 16 transformer layers with a hidden size of 410 and 10 self-attention heads. We train base model 200k steps with a batch size of 64. The large model consists of 18 transformer layers with a hidden size of 1024 and 16 self-attention heads. We train large model 400k steps with a batch size of 128. During training phase, the sequence length and memory length are all limited up to 150 for the base model and 384 for the large model. The remaining hyper-parameters are identical to Transformer-XL.

4.1.2 Results

Tab. 1 summarizes evaluation results on WikiText-103. ERNIE-DOC achieves an impressive improvement compared with Transformer-XL: the perplexity (PPL) decreases 3.0 for the base model and decreases 1.5 for the large model. We finally advance the state-of-the-art to **21.0** PPL with the base model and **16.8** PPL with the large model.

Models	#Param.	PPL
the results of ***Base models***		
LSTM (Grave et al., 2016)	-	48.7
LSTM+Neural cache (Grave et al., 2016)	-	40.8
GCNN-14 (Dauphin et al., 2017)	-	37.2
QRNN (Merity et al., 2018)	151M	33.0
Transformer-XL Base (Dai et al., 2019)	151M	24.0
SegaTransformer-XL Base (Bai et al., 2020)	151M	22.5
ERNIE-DOC Base	151M	**21.0**
the results of ***Large models***		
Adaptive Input (Baevski and Auli, 2018)	247M	18.7
Transformer-XL Large (Dai et al., 2019)	247M	18.3
Compressive Transformer (Rae et al., 2019)	247M	17.1
SegaTransformer-XL Large (Bai et al., 2020)	247M	17.1
ERNIE-DOC Large	247M	**16.8**

Table 1: Comparison with Transformer-XL and competitive baseline results on WikiText-103.

4.2 Pretraining and Finetuning

4.2.1 Pretraining Text Corpora

Dataset	# tokens	Avg len	Size
WIKIPEDIA	2.7B	480	8G
BOOKSCORPUS	1.2B	2010	3.5G
CC-NEWS	14B	560	42G
STORIES	7.5B	1891	22G

Table 2: English datasets used for pretraining.

English Data. To allow ERNIE-DOC to capture long dependencies in pretraining following Zaheer et al. (2020), we compiled a corpus from four standard datasets: WIKIPEDIA, BOOKSCORPUS (Zhu et al., 2015), CC-NEWS[2] and STORIES (Trinh and Le, 2018) (details listed in Tab. 2). We tokenized the corpus using RoBERTa wordpieces tokenizer following Liu et al. (2019), and the pretraining data is duplicated 10 times.

Chinese Data. The same Chinese text corpora used in ERNIE 2.0 (Sun et al., 2020) are adopted to pretraining ERNIE-DOC in this paper.

4.2.2 Experimental Setup

Pretraining. We trained three sizes of models for English tasks, namely small (L=6, H=256, A=4), base (L=12, H=768, A=12), and large (L=24, H=1024, A=16)[3], one size for Chinese tasks, base (L=12, H=768, A=12). We limit the length of sentences in each mini-batch up to 512 tokens and the length of memory to 128. The models are trained for 500K/400K/100K steps using a batch size of 2560/2560/3920 sentences for small/base/large configurations. ERNIE-DOC is optimized with Adam (Kingma and Ba, 2014) using the following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-6$ and weight decay of 0.01. The learning rate is warmed up over the first 4,000 steps to a peak value of 1e-4 and then linearly decayed. The rest of the pretraining hyper-parameters are the same as RoBERTa (Liu et al., 2019) and are supplemented in Appendix A. Additionally, we apply the relative positional embedding (Shaw et al., 2018) in our model pretraining since it is necessary for hidden states reuse without causing temporal confusion (Dai et al., 2019).

[2]We use `news-please` to crawl English news articles between September 2016 and February 2019, then adopt Message Digest Algorithm5 (MD5) for deduplication.

[3]Here, we denote the number of transformer layers as L, the hidden size as H, and the number of self-attention heads as A

Models	IMDB		Hyp
	Acc	F1	F1
RoBERTa (Liu et al., 2019)	95.3	95.0	87.8
Longformer (Beltagy et al., 2020)	95.7	-	94.8
BigBird (Zaheer et al., 2020)	-	95.2	92.2
ERNIE-DOC	**96.1**	**96.1**	**96.3**
XLNet-Large (Yang et al., 2019)	96.8	-	-
ERNIE-DOC-Large	**97.1**	**97.1**	**96.6**

Table 3: Results on the IMDB and Hyperpartisan (Hyp) dataset for long text classification.

Finetune. Different from the previous models like BERT, RoBERTa, and XLNet, we apply the retrospective feed mechanism and enhanced recurrence mechanism during the finetune phase to make full use of the advantages of these two strategies.

4.2.3 Results on English Tasks

Results on Long Text Classification Tasks. We consider two datasets: IMDB reviews (Maas et al., 2011) and Hyperpartisan news detection (Kiesel et al., 2019). IMDB is a widely used sentiment analysis dataset containing 50,000 movie reviews labeled as positive or negative. Hyperpartisan contains news that takes an extreme left-wing or right-wing standpoint. Documents are extremely long (50% samples contain more than 537 tokens) in Hyperpartisan which makes it a good test for long text classification. Tab. 3 summarizes the results of ERNIE-DOC-Base and ERNIE-DOC-Large model on long text classification tasks and ERNIE-DOC achieves SOTA result. For IMDB, we observe a modest performance gain compared with RoBERTa. The reason is that nearly 90% samples in the dataset consist of less than 569 tokens. In contract with IMDB, ERNIE-DOC surpasses the

Models	TQA	HQA		
	F1	span	sup	joint
RoBERTa	74.3	73.5	83.4	63.5
Longformer	75.2	74.3	84.4	64.4
BigBird	79.5	75.5	**87.1**	67.8
ERNIE-DOC	**80.1**	**79.4**	86.3	**70.5**

Table 4: Results on TriviaQA (TQA) and HotpotQA (HQA) dev dataset for document-level question answering. HotpotQA metricis are F1.

baseline models on Hyperpartisan news detection with a large margin, demonstrating that ERNIE-DOC is capable of utilizing information from long document input. Note that we include the XLNet-Large, the previous SOTA pretraining model on IMDB dataset, as the baseline for large model setting and ERNIE-DOC achieves a comparable result with XLNet-Large.

Results on Document-level Question Answering Tasks. We utilize two document-level QA datasets (Wikipedia setting of TriviaQA (Joshi et al., 2017) and distractor setting of HotpotQA (Yang et al., 2018)) to evaluate models' reasoning ability over long documents. TriviaQA (Wikipedia setting) and HotpotQA (distractor setting) are extractive QA tasks, and we follow the simple QA model of BERT (Devlin et al., 2018) to predict an answer with the maximum sum of start and end logits across multiple segments of a sample. Also, we use a modified cross entropy loss (Clark and Gardner, 2017) for the TriviaQA dataset and conduct a two-stage model (Groeneveld et al., 2020) with the backbone of ERNIE-DOC for HotpotQA dataset. Tab. 4 shows that ERNIE-DOC-BASE outperforms RoBERTa-base by a large margin. Compared with competitive long document modelling models namely, Longformer and BigBird, ERNIE-DOC achieves SOTA result on TriviaQA dataset and significantly surpassed them on HotpotQA dataset.

4.2.4 Results on Chinese Tasks

We execute extensive experiments on seven Chinese language understanding tasks, including machine reading comprehension (CMRC2018, DRCD, DuReader, C^3; Cui et al. (2018), Shao et al. (2018), He et al. (2017), Sun et al. (2019a)), semantic similarity (CAIL2019-SCM; Xiao et al. (2019)) and long text classification (IFLYTEK, THUCNews; Xu et al. (2020), Sun et al. (2016)). The documents in all of the above-mentioned datasets are long enough to be used to evaluate the effectiveness of ERNIE-DOC on long context tasks. A more detailed datasets statistics and the finetune hyperparameters for all seven datasets are supplemented in Appendix A. Results of seven Chinese tasks are presented in Tab. 5. It is observed that ERNIE-DOC significantly outperforms previous models across tasks by a large margin, achieving a new state-of-the-art on these Chinese NLU tasks in base-size model group.

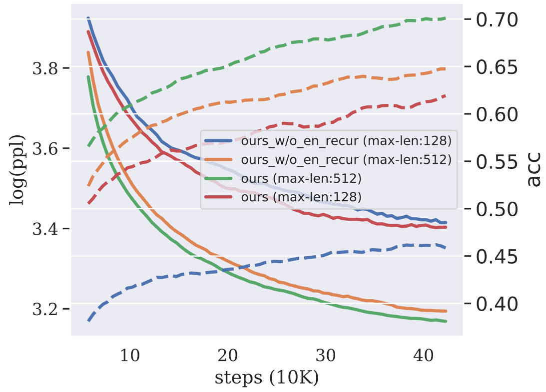

Figure 4: Acc (dotted line) and PPL (solid line) metrics for variants of our small models with different maximum sequence length during pretraining.

4.2.5 Ablation Studies

Effect of proposed components. Tab. 6 shows the performance of ERNIE-DOC small on English tasks by ablating each proposed component. All models are pretrained and finetuned with the same experimental setup, and we reported the mean results with 5 runs. From the last column in Tab. 6, we see that the segment reordering objective improves ERNIE-DOC with 0.81% point on average (#1 - #0), the retrospective feed mechanism benefits ERNIE-DOC with 0.58% point on average (#2 - #1), and the enhanced recurrence mechanism makes a large contribution of 2.55% point on average (#3 - #2). By comparing #3 to #4, we see that Segment-level recurrence is necessary in modeling the long document bringing 4.92% point improvement on average. Considering different types of tasks, we observe Hyperpartisan, an extremely long text classification dataset, gains a sound improvement using segment reordering objective (1.5% point), indicating that `[CLS]` token, pretrained using segment reordering objective is more adaptable to a more identical document-level text classification task. Moreover, we see a stable performance gain across all tasks by using the enhanced recurrence mechanism.

Models	DRCD EM/F1		CMRC2018 EM/F1	DuReader EM/F1	CAIL Acc		THU Acc		IFK Acc	C^3 Acc	
	Dev	Test	Dev	Dev	Dev	Test	Dev	Test	Dev	Dev	Test
BERT (Devlin et al., 2018)	85.7/91.6	84.9/90.9	66.3/85.9	59.5/73.1	61.9	67.3	97.7	97.3	60.3	65.7	64.5
BERT-wwm-ext*	85.0/91.2	83.6/90.4	67.1/85.7	-/-	-	-	97.6	97.6	59.4	67.8	68.5
RoBERTa-wwm-ext*	86.6/92.5	85.2/92.0	67.4/87.2	-/-	-	-	-	-	60.3	67.1	66.5
MacBERT (Cui et al., 2020)	88.3/93.5	87.9/93.2	69.5/87.7	-/-	-	-	-	-	-	-	-
ERNIE 1.0 (Sun et al., 2019b)	84.6/90.9	84.0/90.5	65.1/85.1	57.9/72/1	-	-	97.7	97.3	59.0	65.5	64.1
ERNIE 2.0 (Sun et al., 2020)	88.5/93.8	88.0/93.4	69.1/88.6	61.3/74.9	64.9	67.9	98.0	97.5	61.7	72.3	73.2
ERNIE-DOC	**90.5/95.2**	**90.5/95.1**	**76.1/91.6**	**65.8/77.9**	**65.6**	**68.8**	**98.3**	**97.7**	**62.4**	**76.5**	**76.5**

Table 5: Results on seven Chinese NLU tasks for ERNIE-DOC-base model. Results of models with "*" are from Cui et al. (2019). **CAIL**, **THU**, **IFK** is the short form of CAIL2019-SCM, THUCNews and IFLYTEK respectively. The results of the compared methods are all obtained from the corresponding released papers. Notably, the result of BERT on CAIL comes from Xiao et al. (2019), where the BERT has been post-pretrained with a legal dataset.

	QA		Classification		
Models\Dataset	**TriviaQA**	**HotpotQA**	**IMDB**	**Hyperpartisan**	**Avg.**
#0 ERNIE-DOC	64.56	50.85	93.14	86.10	**73.66**
#1 w/o so	63.59	50.04	93.15	84.60	72.85
#2 w/o so&retro	63.38	49.87	92.56	83.27	72.27
#3 w/o so&retro&en-rec	61.09	44.05	92.07	81.67	69.72
#4 w/o so&retro&recur	58.35	31.54	91.60	77.72	64.80

Table 6: Performance of ERNIE-DOC-small by ablating each proposed components. (**so** means the segment reordering objective, **re** means the retrospective feed mechanism, **en-rec** means enhanced recurrence mechanism and **recur** means segment-level recurrence module. Acc metric for IMDB, F1 metric for TriviaQA and Hyperpartisan, Joint-F1 for HotpotQA.)

Effect of enhanced recurrence mechanism w.r.t different maximum sequence length. As depicted in Fig. 4, enhanced recurrence mechanism plays an important role in pretraining a good language model with lower PPL and performing much better on segment reordering objective with higher accuracy score at both 128 and 512 maximum sequence input length. Compared with the performance gain obtained at 512 maximum length, the effect of enhanced recurrence mechanism is more significant at 128 maximum length where there is a huge improvement gain w.r.t Acc. between ours (max-len:128) and ours_w/o_en_recur (max-len:128). This tallies with our assumption that when the maximum sequence length is long enough to contain the contextual information of a whole document, there is a little room for improvement either using enhanced recurrence mechanism or traditional segment-recurrence mechanism.

5 Conclusion

In this paper, we propose ERNIE-DOC, a document-level language pretraining model based on the recurrence transformers paradigm. Two well-designed mechanisms, namely the retrospective feed mechanism and the enhanced recurrent mechanism enable ERNIE-DOC with theoretically longest possible dependency to model bidirectional contextual information of a whole document. Additionally, ERNIE-DOC is pretrained with a document-aware segment reordering objective to explicitly learn the relationship among segments of a long context. Experiments on various downstream tasks demonstrate that ERNIE-DOC outperforms previous strong pretraining models such as RoBERTa, Longformer, BigBird by a large margin and achieves state-of-the-art results on several language modeling and language understanding benchmarks.

For future work, we will evaluate ERNIE-DOC on some language generation tasks such as generative question answering, text summarization, etc. We will also investigate its potential usage in other areas like computational biology. Another promising line is to incorporate graph neural networks into ERNIE-DOC enabling better modeling capability for tasks that require multi-hop reasoning and long document modeling ability.

References

Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*.

He Bai, Peng Shi, Jimmy Lin, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. 2020. Segabert: Pre-training of segment-aware bert for language understanding. *arXiv preprint arXiv:2004.14996*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for chinese natural language processing. *arXiv preprint arXiv:2004.13922*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2018. A span-extraction dataset for chinese machine reading comprehension. *arXiv preprint arXiv:1810.07366*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

Dirk Groeneveld, Tushar Khot, Ashish Sabharwal, et al. 2020. A simple yet strong pipeline for hotpotqa. *arXiv preprint arXiv:2004.06753*.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, et al. 2017. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 829–839.

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

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2020. Pretrained transformers for text ranking: Bert and beyond. *arXiv preprint arXiv:2010.06467*.

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*.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *CoRR*, abs/1911.05507.

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*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019a. Probing prior knowledge needed in challenging chinese machine reading comprehension. *CoRR*, abs/1904.09679.

M Sun, J Li, Z Guo, Z Yu, Y Zheng, X Si, and Z Liu. 2016. Thuctc: an efficient chinese text classifier. *GitHub Repository*.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *AAAI*, pages 8968–8975.

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

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse sinkhorn attention. *arXiv preprint arXiv:2002.11296*.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Chaojun Xiao, Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Tianyang Zhang, Xianpei Han, Heng Wang, Jianfeng Xu, et al. 2019. Cail2019-scm: A dataset of similar case matching in legal domain. *arXiv preprint arXiv:1911.08962*.

Liang Xu, Xuanwei Zhang, Lu Li, Hai Hu, Chenjie Cao, Weitang Liu, Junyi Li, Yudong Li, Kai Sun, Yechen Xu, et al. 2020. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*.

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*, pages 5753–5763.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. 2019. Bp-transformer: Modelling long-range context via binary partitioning. *arXiv preprint arXiv:1911.04070*.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. *arXiv preprint arXiv:1905.06566*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Appendices

A.1 Tasks

Following previous work, we evaluate ERNIE-Doc on various tasks that require the ability to model the long document.

Document-level Language Modeling Task. We employ WikiText-103 (Merity et al., 2016) in language modeling experiments. WikiText-103 is the largest available word-level benchmark with long-term dependency for language modeling, which consists of 28K articles, where each article has 3.6K tokens on average, thus 103M training tokens in total.

Long Text classification. We consider two English datasets: IMDB reviews (Maas et al., 2011) and Hyperpartisan news detection (Kiesel et al., 2019) (see Tab. 7), and two Chinese datasets: IFLYTEK (Xu et al., 2020) and THUCNews (Sun et al., 2016) (see Tab. 8).

- IMDB is a widely used sentiment analysis dataset containing 50,000 movie reviews labeled as positive or negative. Training and dev dataset is equally split.
- Hyperpartisan contains news that takes an extreme left-wing or right-wing standpoint. Documents are extremely long in Hyperpartisan which makes it a good test for long text classification. We use the same split as Longformer by dividing 654 documents into train/dev/test sets.
- IFLYTEK contains 17332 app descriptions. The task is to assign each description into one of 119 categories, such as food, car rental, education, etc.
- THUCNews is generated by filtering historical data of Sina News RSS subscription channel from 2005 to 2011, including 740000 news documents and 14 categories. In this paper, we employ the subset version instead of the full one [4], which contains 10 categories, each with 5000 pieces of data.

For the above four long text classification datasets, we concatenate [CLS] token with each segment and takes as input multiple segments of a text sequentially. Each segment is generated by slicing the text with a sliding window of 128 tokens. We apply binary cross entropy loss on the [CLS] token of the last segment.

Long Text Semantic Similarity. Considering that there is no available long text semantic similarity dataset in English, we evaluate the effectiveness of ERNIE-Doc on semantic similarity task only depending on Chinese dataset CAIL2019-SCM. According to Xiao et al. (2019), CAIL2019-SCM is a sub-task of the **C**hinese **AI** and **L**aw Challenge (CAIL) competition in 2019, which contains 8964 triplets of legal documents collected from China Judgments Online. Every document in a majority of triplet has more than 512 characters, therefore, the total length of a triplet is quite long. CAIL2019-SCM requires researchers to decide which two cases are more similar in a triplet. Specifically, given a triplet (A, B, C), where A, B, C are fact descriptions of three cases. The model needs to predict whether $sim(A, B) > sim(A, C)$ or $sim(A, C) > sim(A, B)$, in which sim denotes the similarity between two cases. Instead of separately feeding the document A, B, C into the model to get the feature h, we use the combinations of (A, B) and (A, C) as input to feed into the model in this paper. we generate multiple segments for (A, B) or (A, C) with a sliding window of 128 tokens and feed them as input sequentially. The binary cross entropy loss is applied to the difference of [CLS] token output of each segment.

Document-level Question answering. We utilize two English question answering datasets (TriviaQA, HotpotQA; Joshi et al. (2017), Yang et al. (2018)) (see Tab. 7) and four Chinese question answering datasets (CMRC2018, DRCD, DuReader, C^3; Cui et al. (2018), Shao et al. (2018), He et al. (2017), Sun et al. (2019a)) (see Tab. 8) to evaluate models' reasoning ability over long documents.

TriviaQA is a large scale QA dataset that contains over 650K question-answer pairs. We evaluate models on its Wikipedia setting where documents are Wikipedia articles, and answers are named entities mentioned in multiple documents. The dataset is distantly supervised meaning that there is no golden span, thus we find all superficial identical answers in provided docu-

[4] The subset version is also released and can be downloaded from the official website of THUCTC

Datasets	IMDB		Hyperpartisan			TriviaQA		HotpotQA	
split	train	dev	train	dev	test	train	dev	train	dev
# samples	25000	25000	516	64	65	61888	7993	90432	7404
# tokens of context length in each percentile using RoBERTa wordpiece tokenizer									
50%	215	212	537	521	639	8685	8586	1279	1325
90%	569	550	1519	1539	1772	25207	24825	1725	1785
95%	745	724	1997	1979	1994	32018	32132	1888	1943
max	3084	2778	5566	2643	5566	173302	146012	3733	3618

Table 7: English Datasets statistics

Datasets	IFLYTEK		THUCNews		CAIL		CMRC2018		DuReader		C^3		DRCD		
split	train	dev	train	dev	train	dev	train	dev	train	dev	train	dev	train	dev	test
# samples	12133	2599	50000	5000	5102	1500	10121	3219	15763	1628	11869	3816	26936	3524	3493
# tokens of context length in each percentile using BERT tokenizer															
50%	243	242	656	578.5	1837	1834	423	426	163	182	96	89	397	421	405
90%	506.8	508	1821	1599	1965	1962	745	771	550	567	591.2	554	616	666	626
95%	563.4	560.2	2455	2245	2008	1995	827	840	652	667	697	691.5	709	740	736
max	3153	1698	26659	9128	2400	2310	970	961	1021	854	1534	1167	1678	989	950

Table 8: Chinese Datasets statistics

ments. We use the following input format for each segment: "`[CLS] context [q] question [/q]`" where context is generated by slicing multi-documents input with a sliding window of 128 tokens. We take as input multiple segments of a sample sequentially and attach a linear layer to each token in a segment to predict the answer span. We use a modified cross entropy loss (Clark and Gardner, 2017) assuming that each segment contains at least one correct answer span. The final prediction for each question is a span with the maximum sum of start and end logit across multiple segments.

HotpotQA is a QA dataset where golden spans of an answer and sentence-level supporting facts are provided. Thus, it contains two tasks namely, answer span prediction and supporting facts prediction. In the distractor setting, each question is associated with 10 documents where only 2 documents contain supporting facts. It requires the model to find and reason over multiple documents to find answers, and explain the predicted answers using predicted supporting facts. Following Groeneveld et al. (2020), we implemented a two-stage model based on ERNIE-Doc and use the following input format for each segment: "$\texttt{[CLS] title}_1$ $\texttt{[p] sent}_{1,1}$ $\texttt{[SEP] sent}_{1,2}$ $\texttt{[SEP]} \ldots$ $\texttt{title}_2$ $\texttt{[p] sent}_{2,1}$ $\texttt{[SEP] sent}_{2,2}$ `[SEP] ... [q] question [/q]`" For evidence prediction, we apply 2 layer feedforward networks over the special token [SEP] and [p] representing a sentence and a paragraph separately. Then we use binary cross entropy loss to do binary classification. For answer span prediction, we train the model with a multi-task objective: 1) question type (yes/no/span) classification on the [CLS] token. 2) supporting evidence prediction on [SEP] and [p]. 3) span prediction on the start and end token of a golden span.

CMRC2018, DRCD and DuReader are common Chinese QA datasets with same format, which have been evaluated in numerous popular pretraining models, such as BERT (Devlin et al., 2018), ERNIE 1.0 (Sun et al., 2019b), ERNIE 2.0 (Sun et al., 2020) and etc. The detailed descriptions of three datasets can refer to (Cui et al., 2018), (Shao et al., 2018) and He et al. (2017). We adopt the same input format as TriviaQA for each segment, denotes as "`[CLS] context [SEP] question [SEP]`" where context is generated by slicing multi-documents input with a sliding window of 128 tokens. We take as input multiple segments of a sample sequentially and attach a linear layer to each token in a segment to predict the answer span. Then, we apply a softmax and use the cross entropy loss with the correct answer. The final prediction for each question is a span with the maximum sum of start and end logit across multiple segments.

The multiple **C**hoice **C**hinese machine reading **C**omprehension dataset (C^3) (Sun et al., 2019a) is the first Chinese free-form multi-choice dataset where each question is associated with at most four choices and a single document. According to (Sun et al., 2019a), m segments are constructed for a question, in which m denotes the number of choice for that question. We use the following input format for each segment: "`[CLS] context [SEP] question [SEP] choice`$_i$ `[SEP]` " where context is generated by slicing document input with a sliding window of 128 tokens stride. We take as input multiple segments of a sample in a single batch and attach a linear layer to `[CLS]` that outputs an unnormalized logit. Then we obtain the final prediction for a question by applying a softmax layer over the unnormalized logits of all choices associated with it.

A.2 Hyperparameters for Language Modeling

In Tab. 9, we present the detailed hyperparameters used for our experiments, which are the same as the hyperparameters employed in Transformer-XL (Dai et al., 2019).

Hyperparameters	WikiText-103 Base	WikiText-103 Large
Layers	16	18
Hidden size	410	1024
Attention heads	10	16
Training sequence length	150	384
Training Memory length	150	384
Testing sequence length	64	128
Testing sequence length	640	1600
Batch size	64	128
Learning rate	2.5e-4	2.5e-4
Warmup steps	0	16000
Training steps	200k	400k

Table 9: Hyperparameters used for WikiText-103.

A.3 Hyperparameters for PreTraining

As shown in Tab. 10, we present the detailed hyperparameters adopted to pretraining ERNIE-DOC on English text corpora and Chinese text corpora. For comparisons, we follow the same optimization hyperparameters of RoBERTA$_{\text{BASE}}$ or RoBERTA$_{\text{LARGE}}$ (Liu et al., 2019) for base-size or large-size model in English domain. As for Chinese ERNIE-DOC, we follow the same optimization hyperparameters of ERNIE 2.0$_{\text{BASE}}$.

A.4 Hyperparameters for FineTune

A.4.1 Long Text Classification tasks

The finetuning hyperparameters for IMDB (Maas et al., 2011) and Hyperpartisan (Kiesel et al., 2019) are presented in Tab. 11.

Hyperparameters	English		Chinese
	BASE	**LARGE**	**BASE**
Layers	12	24	12
Hidden size	768	1024	768
Attention heads	12	16	12
Training steps	400K	100K	300K
Batch size	2560	3920	2560
Learning rate	1e-4	1e-4	1e-4
Warmup steps	4000	4000	4000
Adam (beta1,beta2)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Adam (epsilon)	1e-6	1e-6	1e-6
Learning rate schedule	Linear	Linear	Linear
Weight decay	0.01	0.01	0.01
Dropout	0.1	0.1	0
GPU (Nvidia V100)	40	80	40

Table 10: Hyperparameters used for ERNIE-DOC pretraining.

Hyperparameters	BASE		LARGE	
	IMDB	**Hyp**	**IMDB**	**Hyp**
Batch size	32	32	32	16
Learning rate	7e-5	1e-4	1e-5	4e-6
Epochs	3	15	3	15
LR schedule	linear	linear	linear	linear
Layerwise LR decay	1	0.7	0.9	1
Warmup proportion	0.1	0.1	0.1	0.1
Weight decay	0.01	0.01	0.01	0.01

Table 11: Hyperparameters used for finetuning on IMDB and Hyperpartisan (Hyp).

A.4.2 Document-level Question answering tasks

The finetuning hyperparameters for TriviaQA (Welbl et al., 2018) and HotpotQA (Yang et al., 2018) are presented in Tab. 12.

Hyperparameters	TriviaQA	HotpotQA
Batch size	64	32
Learning rate	3e-5	1.5e-4
Epochs	5	6
LR schedule	linear	linear
Layerwise LR decay	0.8	0.8
Warmup proportion	0.1	0.1
Weight decay	0.01	0.01

Table 12: Finetuning hyperparameters on the TriviaQA and HotpotQA for base-size ERNIE-DOC.

A.4.3 Chinese NLU tasks

Tab. 13 lists the finetuning hyperparameters for Chinese NLU tasks including IFLYTEK (Xu et al., 2020), THUCNews (Sun et al., 2016), CMRC2018 (Cui et al., 2018), DRCD (Shao et al., 2018), DuReader He et al. (2017), C^3 (Sun et al., 2019a) and CAIL2019-SCM (Xiao et al., 2019).

Tasks	Batch size	Learning rate	Epochs	Dropout
DRCD	64	2.25-4	5	0.1
CMRC2018	64	1.75e-4	5	0.2
DuReader	64	2.75e-4	5	0.1
C3	24	1e-4	8	0.1
CAIL	48	5e-5	15	0.1
THU	16	1.5e-4	16	0.1
IFK	16	1.5e-4	5	0.1

Table 13: Hyperparameters used for finetuning on Chinese NLU tasks. Note that the warmup proportion are set to 0.1 and the layerwise lr decay rate are set to 0.8 for all tasks.

B Attention Complexity

Theoretically, the attention in Longformer and Bigbird scales linearly with respect to the input length while the complexity of ERNIE-DOC is quadratic. However, the maximum length is set as 4096 in BigBird and Longformer with the local window size as 512 which results in $4096 * 512$ token-to-token calculations. For ERNIE-DOC, the maximum input length is set as 512 with a commonly used memory size as 128 which results in $512 * (512 + 128)$ token-to-token calculations. Thus, the attention complexity of ERNIE-DOC is on par with BigBird and Longformer in practice. However, the limitation of ERNIE-DOC is that the retrospective feed mechanism doubles the time of a forward pass since the input needs to be fed twice. We will attempt to tackle this limitation in future work.