Revisiting Token Dropping Strategy in Efficient BERT Pretraining

Qihuang Zhong[1], **Liang Ding**[2], **Juhua Liu**[3]*, **Xuebo Liu**[4]
Min Zhang[4], **Bo Du**[1]*, **Dacheng Tao**[5]

[1] National Engineering Research Center for Multimedia Software, Institute of Artificial Intelligence, School of Computer Science and Hubei Key Laboratory of Multimedia and Network Communication Engineering, Wuhan University, China
[2] JD Explore Academy, China [3] Research Center for Graphic Communication, Printing and Packaging, and Institute of Artificial Intelligence, Wuhan University, China
[4] Institute of Computing and Intelligence, Harbin Institute of Technology, China [5] University of Sydney, Australia
`{zhongqihuang, liujuhua, dubo}@whu.edu.cn`, `{liangding.liam, dacheng.tao}@gmail.com`

Abstract

Token dropping is a recently-proposed strategy to speed up the pretraining of masked language models, such as BERT, by skipping the computation of a subset of the input tokens at several middle layers. It can effectively reduce the training time without degrading much performance on downstream tasks. However, we empirically find that token dropping is prone to a *semantic loss* problem and falls short in handling semantic-intense tasks (§2). Motivated by this, we propose a simple yet effective *semantic-consistent* learning method (ScTD) to improve the token dropping. ScTD aims to encourage the model to learn how to preserve the semantic information in the representation space. Extensive experiments on 12 tasks show that, with the help of our ScTD, token dropping can achieve consistent and significant performance gains across all task types and model sizes. More encouragingly, ScTD saves up to 57% of pretraining time and brings up to +1.56% average improvement over the vanilla token dropping.

1 Introduction

Masked language models (MLMs), such as BERT (Devlin et al., 2019) and its variants (Liu et al., 2019; He et al., 2020; Zhong et al., 2023a)[1], have achieved great success in a variety of natural language understanding (NLU) tasks. However, with the scaling of model size and corpus size, the pretraining of these BERT-style models becomes more computationally expensive and memory intensive (Jiao et al., 2020; Hou et al., 2022). Hence, it is crucial and green to speed up the training and reduce the computational overhead for BERT-style pretraining (Zhang and He, 2020; Schwartz et al., 2020).

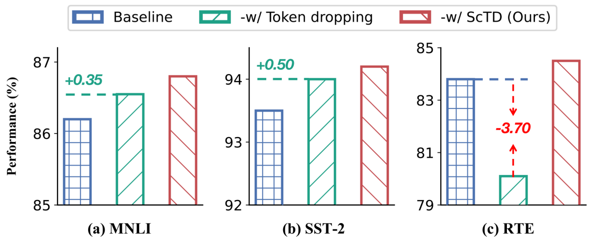

Figure 1: Performance of $\text{BERT}_{\text{base}}$ on several downstream tasks. We see that: 1) Despite the remarkable performance on general tasks (*i.e.*, MNLI and SST-2), token dropping leads to dramatically poor performance on the semantic-intense task (*i.e.*, RTE). 2) Our ScTD achieves consistent performance gains among all tasks.

To achieve this goal, various training-efficient approaches have been developed and summarized (Shoeybi et al., 2019; You et al., 2019; Zhang and He, 2020; Shen et al., 2023). Among these efforts, a recently-proposed **token dropping**[2] strategy (Hou et al., 2022) has attracted increasing attention owing to its easy-to-implement algorithm and impressive efficiency (reducing the training cost by 25% without much average performance dropping) (Yao et al., 2022; Chiang et al., 2022). Different from most previous works that focus on changing model architecture or optimization process, token dropping aims to improve training efficiency by dynamically skipping the compute of the redundant (unimportant) tokens that are less informative to the current training, at some middle layers of BERT during training. Although achieving a remarkable speedup, the performance improvement of token dropping is usually limited and unstable, compared to the baseline training scheme. More specifically, we empirically found that token dropping falls short in handling semantic-intense tasks, as shown in Figure 1. This motivates us to explore and address the limitations of token dropping in this paper.

In light of the conventional wisdom that "seman-

* Corresponding Authors: Juhua Liu (e-mail: liujuhua@whu.edu.cn), Bo Du (e-mail: dubo@whu.edu.cn)

[1]We refer to these models as BERT-style models.

[2]We also refer to it as "token drop" in some cases.

tic information is mainly encoded in the BERT's intermediate and top layers" (Jawahar et al., 2019), we suspected, apriori, that the corruption caused by the removal of unimportant tokens would break the sentence structure, and may easily lead to the semantic drift of sentence representations, as also observed in many similar scenarios (Zhang et al., 2022; Wang et al., 2021). To verify this conjecture, we conduct a series of preliminary analyses on a representative BERT model, and find that:

> ❶ The ***training dynamics*** of the token dropping show a significant semantic drift.
>
> ❷ The ***representation*** of a well-trained BERT with token dropping contains less semantics.
>
> ❸ The ***downstream semantic-intense tasks*** show a clear performance degradation.

Based on these observations, we can basically conclude that (one of) the limitation of token dropping is the *semantic loss*[3] problem, which causes vulnerable and unstable training of BERT models. To address this limitation, we propose a simple yet effective semantic-consistent learning method (referred to as **SCTD**) to improve token dropping. The principle of SCTD is to encourage the BERT to learn how to preserve the semantic information in the representation space. Specifically, SCTD first introduces two semantic constraints to align the semantic information of representations between baseline- and token dropping-based models, and then adopts a novel hybrid training approach to further improve the training efficiency.

We evaluate SCTD on a variety of benchmarks, including GLUE (Wang et al., 2018), SuperGLUE (Wang et al., 2019) and SQuAD v1/v2 (Rajpurkar et al., 2016, 2018), upon two typical MLMs: BERT-BASE and -LARGE. Results show that SCTD can not only bring consistent and significant improvements (up to +1.56% average score among all tasks) into the token dropping strategy on both BERT models, but also alleviate the semantic loss problem. Moreover, compared to the standard BERT models, SCTD can also save up to 48% of pretraining time while achieving comparable performance, and further achieve +1.42% average gain for the same training iterations.

To summarize, **our contributions** are as follows:

[3]As we find that BERT models trained with token dropping are prone to losing some semantic-related polarities, e.g., less semantic knowledge in the dropped layers, we refer to this phenomenon as "semantic loss" in the paper.

- Our study reveals the semantic loss problem in the token dropping strategy, which limits its performance on downstream tasks, especially on semantic-intense tasks.
- We propose a simple yet effective, plug-in-play approach (SCTD) to alleviate the semantic loss and further improve efficiency.
- Experiments show that SCTD outperforms the vanilla token dropping with up to +1.56% average improvement and saves up to 57% of pretraining time.

2 Revisiting Token Dropping Strategy

In this section, we first review the background of token dropping strategy and then present the empirical analyses of this strategy in detail.

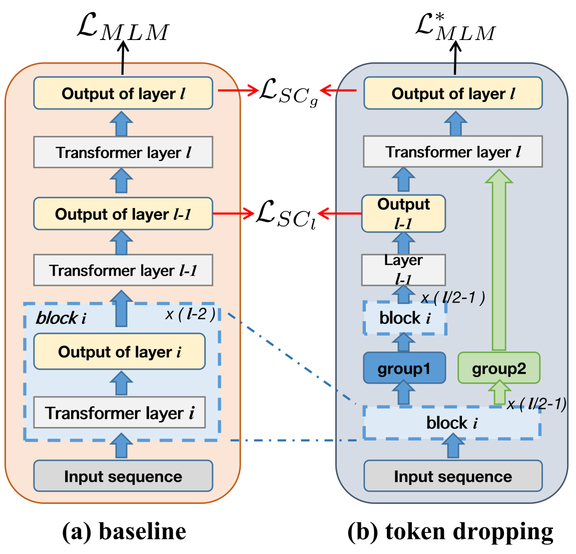

Figure 2: Illustration of BERT-style models with (a) baseline training and (b) token dropping training. In (b), the "group1" and "group2" denote the important and unimportant (skipped) tokens, respectively. The $\mathcal{L}_{SC_l}$ and $\mathcal{L}_{SC_g}$ (in red arrows) refer to the semantic-align objectives used in our SCTD.

2.1 Preliminaries

Suppose that we focus on pretraining the BERT with l transformer layers. Let L_i denote the i-th ($i \in \{1, ..., l\}$) layer, $X_i \in \mathbb{R}^{s_i \times d}$ be the output tensors of i-th layer, where s_i is the sequence length of i-th layer and d is the hidden size. Notably, X_0 denotes the input (after embedding) of the model. For the baseline training process (as illustrated in Figure 2 **(a)**), full-sequence tokens will be sequentially fed into all layers, *i.e.*, $s_0 = s_1 = ... = s_l$. In this way, we can obtain the final output tensors X_l of l-th layer, and then use a cross-entropy loss to optimize the training process as follow:

$$\mathcal{L}_{MLM} = \mathbb{E}\left(-\sum \log P(Y|X_l)\right), \qquad (1)$$

where Y denotes the ground-truths.

For token dropping (as illustrated in Figure 2 **(b)**), different from the full-sequence training, the training of a subset of unimportant tokens in middle layers will be skipped[4]. In practice, for stable training, token dropping follows the full-sequence training at several first layers (*i.e.*, from 1-th layer to $(l/2-1)$-th layer). Then, it uses several importance scores/metrics to determine the dropped tokens and divides tokens into two groups, where we denote the "group1" as important tokens and "group2" as unimportant (dropped) tokens. The group1 tokens will be fed into later layers (*i.e.*, from $(l/2-1)$-th layer to $(l-1)$-th layer), while the computations of the group2 tokens are skipped. Lastly, all tokens are merged before the last layer and then are used to obtain the final outputs[5] $\tilde{X}_l$. The loss function of token dropping is similar to Eq. 1, and we refer to it as $\mathcal{L}^*_{MLM}$.

2.2 Empirical Analyses

In this part, to verify whether removing the unimportant tokens will cause the loss of semantic information and thus hinder the performance of token dropping, we conduct systematic analyses from three aspects: 1) *revealing the semantic drift problem during* ***training dynamics***; 2) *probing the* ***representation*** *of a well-trained model with token dropping*; 3) *evaluating the* ***downstream performance*** *on semantic-intense tasks.* In practice, for comparison, we pre-train the representative BERT$_{\text{base}}$ models with baseline training scheme and token dropping, respectively. Through the above analyses, we empirically observe that:

❶ The training dynamics of the token dropping show a significant semantic drift. As suspected in §1, the corruption caused by the removal of several tokens would break the sentence structure, thus leading to semantic drift. Here, we verify this conjecture by quantitatively estimating the loss of semantic information contained in the corrupted sentence. For measuring the semantic information, we first adopt the off-the-shelf Sentence-BERT (Reimers and Gurevych, 2019) to capture the semantic representations. Then, suppose that the original sentence (without any corruption, such as masking or token dropping) contains full semantic information, we refer to the cosine similarity between semantic representations of the corrupted and original sentences as a metric to measure the semantic drift in the corrupted sentence.

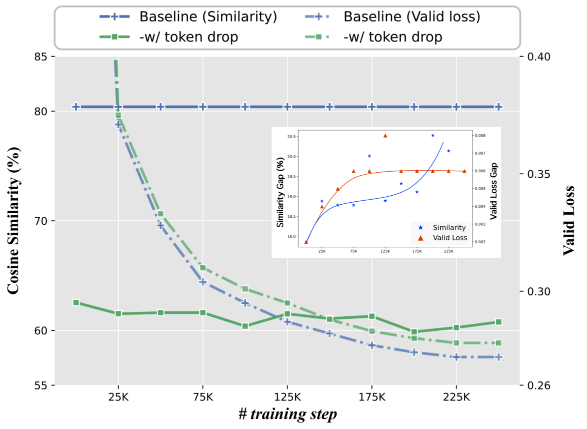

Figure 3: The comparison of similarity and validation curves between baseline and token dropping on BERT$_{\text{base}}$ pretraining. The left y-axis is the cosine similarity between corrupted- (in baseline and token dropping settings, respectively) and original sentences, while the right y-axis is the validation results. The similarity and validation gaps are illustrated in the inserted figure.

In practice, given some sentences randomly sampled from training data, we follow the above process and measure the (average) semantic drift during the baseline/token dropping training dynamics, respectively. For reference, we also report the validation results and illustrate all results in Figure 3. It can be found that: compared to baseline training, *i)* sentence semantics in token dropping drifts more from the original semantics; *ii)* token dropping hinders the full learning of BERT, especially in the middle and later training stages (after 75K steps). To have a closer look, we show the similarity and validation gaps between both settings in the inserted figure of Figure 3. As seen, with the training going on, both gaps have a similar tendency to increase[6], especially at the beginning of training. In general, these analyses indicate that *there is a significant semantic drift during training dynamics*

[4]Hou et al. (2022) state that such a process would not only hardly damage the effect of pretraining, but also reduce the computation costs.

[5]To distinguish from final outputs X_l of baseline training, we denote it as $\tilde{X}_l$.

[6]The curve of validation gap tends to flatten in the later training stage, as both models are going to converge.

of token dropping, which shows a correlation with the performance drop of token dropping.

❷ The representation of a well-trained BERT with token dropping contains less semantics. In addition to the analysis during training dynamics, we then investigate the semantic properties of well-trained models. Specifically, following many prior works (Conneau et al., 2018; Jawahar et al., 2019; Ding et al., 2020; Zhong et al., 2022a), we perform several semantic-aware probing tasks on the sentence representations at different layers. Taking the **Tense** and subject number (**SubjNum**) tasks as examples, we provide the comparison of semantic information between baseline and token dropping at different layers in Figure 4.

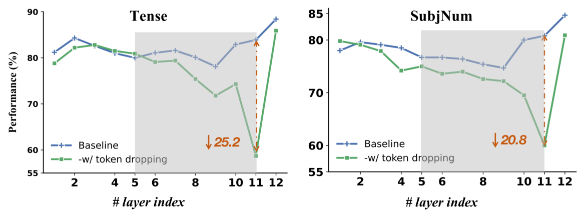

Figure 4: The comparison of semantic information between baseline and token dropping on different $\text{BERT}_{\text{base}}$ layers. We see that, for token dropping, as the number of dropped layers (from layer 5 to layer 11, illustrated in shadow areas) increases, the semantic information saved by the model is significantly reduced.

We observe that there is more semantic information in the top layers (from layer 9 to layer 12) of BERT trained with the baseline scheme, which is similar to the finding of Jawahar et al. (2019). However, when using the token dropping, the semantic information contained in BERT tends to decrease in the dropped layers (from layer 5 to layer 11). The semantic information of token dropping at 11-th layer drops dramatically, which is much lower (up to 25.2 points) than that of baseline. Moreover, due to the vulnerable and unstable training, the final representation in token dropping at the last layer is also sub-optimal. These results basically prove that *the semantic drift of token dropping damages the semantic learning ability of BERT.*

❸ The downstream semantic-intense tasks show a clear performance degradation. The aforementioned analyses mainly focus on interpreting the semantic properties of models. Here, we further evaluate the downstream performance of token dropping. Specifically, several representative semantic-intense[7] tasks are used, including OntoNotes 5.0 (Weischedel et al., 2013) (Onto. for short), CoNLL03 (Sang and De Meulder, 2003), MRPC (Dolan and Brockett, 2005) and SICK-Relatedness (Marelli et al., 2014) (SICK-R for short). Notably, for Onto. and CoNLL03, we report the few-shot (32-shot) performance to enlarge the performance difference between different models. We measure the development performance of each task using its corresponding evaluation metrics, and report the contrastive results in Table 1.

Method	Onto.	CoNLL03	MRPC	SICK-R	Avg.
	F1	*F1*	*Acc.*	*Spear.*	
Baseline	30.16	54.48	86.80	69.08	60.13
token drop	27.49	53.73	85.50	66.16	58.22
Δ (↓)	**-2.67**	**-0.75**	**-1.30**	**-2.92**	**-1.91**

Table 1: Experimental results of $\text{BERT}_{\text{base}}$ trained with different methods on several semantic-intense tasks. We observe that token dropping strategy leads to poor performance among all these tasks.

As seen, there is a great discrepancy between the downstream performance of baseline and token dropping. Overall, token dropping consistently under-performs the baseline with an average 1.91% performance drop, among all these semantic-intense tasks. Specifically, as for SICK-R (usually used to measure the semantic textual similarity), token dropping performs much worse (up to ↓2.92) than the baseline. These results indicate that, *due to the semantic drift, BERT with token dropping falls short in handling the semantic-intense tasks.*

3 Improving Token Dropping with Semantic-Consistent Learning

Based on the observations in §2, we recognize that it is essential to alleviate the side effect (i.e., *semantic loss* problem) of token dropping. To achieve this goal, we propose a simple yet effective semantic-consistent learning (ScTD) framework Specifically, our ScTD adopts two key techniques as follows:

Semantic-Consistent Learning. The principle of our ScTD is to encourage the model to preserve the semantic information in the representation space. Inspired by the success of knowledge distillation (Hinton et al., 2015; Xu et al.,

[7]We chose tasks based on whether they require semantic-related information to solve. For instance, we included MRPC (Dolan and Brockett, 2005), a task that predicts if two sentences are semantically equivalent.

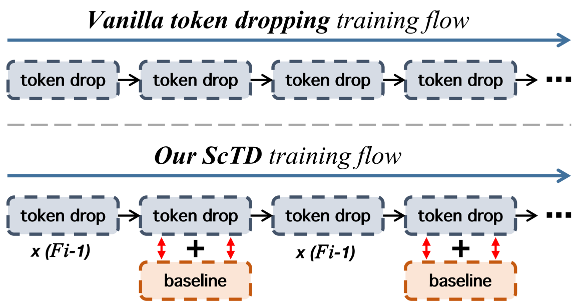

Figure 5: The comparison of training flow between the vanilla token dropping and our SCTD. The "token drop" and "baseline" modules refer to the corresponding training processes in Figure 2. For SCTD, "$\times(F_i - 1)$" means repeating the token dropping process multiple times, where F_i is a fixed interval.

2020), SCTD refers to the model with baseline training (containing more semantic information) as the teacher to guide the training of students (i.e., model trained with token dropping). Considering that it is usually unavailable to obtain the pre-trained teacher model, we hereby recast it as a self-distillation process (Zhang and Sabuncu, 2020; Ding et al., 2021b). Given the same input X_0, we input X_0 into the model to perform twice forward-propagation processes, where one is for token dropping and the other is for baseline training. The outputs of baseline training (X_l) are used as the teacher distributions to teach the student (outputs $\tilde{X}_l$ of token dropping). As such, the student can learn how to align the semantic information with the teacher. More specifically, SCTD introduces two semantic constraints in a local-to-global manner (as illustrated in Figure 2). For the global one, we use the KL divergence to constrain the *global-level* semantic distributions of baseline- and token-dropping-based models at the last l-th layer, as follows:

$$\mathcal{L}_{SC_g} = \mathbf{KL}\left(p(X_l)||p(\tilde{X}_l)\right), \tag{2}$$

where $p(X_l)$ and $p(\tilde{X}_l)$ denote the corresponding distributions respectively. On the other hand, in slight of the finding that semantic loss is the most significant in the penultimate layer $(l-1)$ in token dropping setting (Figure 4), we further construct a *local-level* semantic constraint at the $(l-1)$-th layer, which is similar to Eq. 2:

$$\mathcal{L}_{SC_l} = \mathbf{KL}\left(p(X_{l-1})||p(\tilde{X}_{l-1})\right). \tag{3}$$

Hybrid Training. Since the semantic-consistent learning process requires twice forward/back-propagation, SCTD would introduce much computational overhead, leading to inefficiency. To overcome this issue, SCTD adopts a novel hybrid training strategy, as illustrated in Figure 5. Specifically, instead of using the semantic-consistent learning method throughout the training, SCTD basically follows the vanilla token dropping and adopts the semantic-consistent training intermittently. As such, SCTD can combine the advantages of semantic-consistent learning (*effectiveness*) and token dropping (*efficiency*). Let Fi be a fixed interval, SCTD first performs the vanilla token dropping training $(Fi-1)$ times and then performs once the semantic-consistent training. The overall training objective of SCTD can be formulated as:

$$\mathcal{L}_{all} = \begin{cases} \begin{array}{l} \frac{1}{2}\mathcal{L}^*_{MLM} + \frac{1}{2}\mathcal{L}_{MLM} \\ + \lambda * (\mathcal{L}_{SC_g} + \mathcal{L}_{SC_l}) \end{array}, & t \bmod Fi = 0 \\ \mathcal{L}^*_{MLM}, & t \bmod Fi \neq 0 \end{cases} \tag{4}$$

where t denotes the index of training iterations and λ is a weight factor to balance the different objectives, which is empirically[8] set as 0.05.

4 Evaluation

4.1 Setup

Downstream Tasks To investigate the effectiveness and universality of SCTD, we follow many previous studies (Zhong et al., 2022b,d) and conduct extensive experiments on various NLU tasks, covering a diversity of tasks from GLUE (Wang et al., 2018), SuperGLUE (Wang et al., 2019) and SQuAD benchmarks. Specifically, three semantic-intense tasks (MRPC (Dolan and Brockett, 2005), STS-B (Cer et al., 2017) and RTE (Giampiccolo et al., 2007)), five question answering tasks (BoolQ (Clark et al., 2019a), COPA (Roemmele et al., 2011), MultiRC (Khashabi et al., 2018), SQuAD-v1 (Rajpurkar et al., 2016) and -v2 (Rajpurkar et al., 2018)), two natural language inference tasks (MNLI (Williams et al., 2018) and CB (De Marneffe et al., 2019)), and two others (CoLA (Warstadt et al., 2019) and SST-2 (Socher et al., 2013)) are used. For evaluation, we report the performance with Accuracy ("*Acc.*") metric for most tasks, except the Pearson and Spearman correlation ("*Pear./Spea.*") for STS-B, the Matthew

[8]The detailed analysis can be found in §4.3.

Method	Budget	CoLA	MRPC		STS-B		RTE	MNLI		SST-2	GLUE
	hours	*Mcc.*	*Acc.*	*F1*	*Pear.*	*Spea.*	*Acc.*	*m.*	*mm.*	*Acc.*	*Avg.*
					BERT$_{\texttt{large}}$						
Baseline (*250K*)	34.35	61.3	90.0	92.7	**90.2**	**89.9**	83.8	86.3	86.1	93.5	84.37
token drop (*250K*)	27.33 (-20%)	64.3	88.0	91.4	89.7	89.5	80.1	86.8	86.3	94.0	84.04
-w/ ScTD (*100K*)	11.83 (-66%)	62.3	89.2	92.2	89.9	89.7	80.9	85.1	84.8	93.0	83.61
-w/ ScTD (*160K*)	17.75 (-48%)	**65.8**	88.7	91.8	89.9	89.7	81.2	86.4	86.1	94.0	84.55
-w/ ScTD (*250K*)	29.54 (-14%)	65.6	**91.4**	**93.8**	**90.2**	**89.9**	**84.5**	**87.1**	**86.5**	**94.2**	**85.63**
					BERT$_{\texttt{base}}$						
Baseline (*250K*)	15.17	56.0	86.8	90.1	**89.0**	**88.8**	77.6	83.3	83.5	**92.3**	81.11
token drop (*250K*)	12.92 (-15%)	54.1	85.5	89.6	87.8	87.8	77.6	83.4	83.3	91.7	80.35
-w/ ScTD (*100K*)	5.51 (-64%)	55.4	**87.3**	**91.1**	88.4	88.3	76.9	82.2	82.4	91.4	80.59
-w/ ScTD (*160K*)	8.79 (-42%)	58.1	87.0	90.7	88.1	88.0	78.7	83.4	83.3	90.6	81.28
-w/ ScTD (*250K*)	13.78 (-9.2%)	**58.8**	86.8	90.5	88.2	88.1	**79.4**	**83.8**	**83.6**	91.6	**81.72**

Table 2: Experimental results (dev scores) of BERT$_{\texttt{large}}$ and BERT$_{\texttt{base}}$ trained with different methods on the GLUE benchmark. Average scores on all tasks are underlined. The best results are in **bold**. We see that our ScTD improves the performance and training efficiency of token drop strategy across all task types and model sizes.

correlation ("*Mcc.*") for CoLA, the F1 score for MultiRC, and the Exact Match ("*EM*") scores for SQuAD v1/v2. We report the averaged results over 5 random seeds to avoid stochasticity. The details of all tasks and datasets are shown in Appendix A.1.

Hyper-parameters For pretraining, we train the BRET-BASE and -LARGE models with different methods[9] from scratch. We basically follow the original paper (Devlin et al., 2019) (*e.g.*, the same pretraining corpus), except that we do not use the next sentence prediction (NSP) objective, as suggested in (Liu et al., 2019). In practice, we train each model for 250K steps, with a batch size of 1024 and a peak learning rate of 2e-4. For fine-tuning, the learning rate is selected in {1e-5, 2e-5, 3e-5, 5e-5}, while the batch size is in {12, 16, 32} depending on tasks. The maximum length of input sentence is 384 for SQuAD v1/v2 and 256/512 for other tasks. The detailed hyper-parameters for fine-tuning are provided in Appendix A.2. We use AdamW (Loshchilov and Hutter, 2018) as the optimizer for both pretraining and fine-tuning processes. All experiments are conducted on NVIDIA A100 (40GB) GPUs.

4.2 Compared Results

Results of GLUE are shown in Table 2, while those of SuperGLUE and SQuAD are in Table 3. Based on these results, we can find that:

Method	Boolq	CB	MultiRC	COPA	SQ-v1	SQ-v2
	Acc.	*Acc.*	*F1*	*Acc.*	*EM*	*EM*
			BERT$_{\texttt{large}}$			
Baseline	78.1	91.1	70.3	**72.0**	85.53	79.16
token drop	79.9	91.1	**72.8**	68.0	86.35	81.50
-w/ ScTD	**79.7**	**92.9**	**72.8**	**72.0**	**86.54**	**81.67**
			BERT$_{\texttt{base}}$			
Baseline	**74.4**	83.9	68.1	63.0	81.97	72.18
token drop	73.0	83.9	67.7	64.0	81.67	72.68
-w/ ScTD	73.8	**87.5**	**68.9**	**68.0**	**82.47**	**72.79**

Table 3: Experimental results of BERT$_{\texttt{large}}$ and BERT$_{\texttt{base}}$ trained with different methods on the SuperGLUE (Wang et al., 2019) benchmark and SQuAD (Rajpurkar et al., 2016) (SQ for short) tasks. We see that our ScTD achieves consistent and significant improvements on SuperGLUE and SQuAD tasks as well.

ScTD consistently improves performance on all types of tasks. First, results on the semantic-intense tasks (MRPC, STS-B and RTE) show that ScTD effectively alleviates the semantic loss problem of token dropping. Specifically, for the RTE task, ScTD brings significant improvement (up to +3.4%) against the vanilla token dropping, and even outperforms the full-sequence training baseline. On the other hand, we observe that ScTD is also beneficial to the other general tasks (*e.g.*, question answering). With the help of ScTD, token dropping strategy achieves up to +1.56% average gains among all types of tasks, proving the effectiveness and universality of ScTD.

[9]Following Hou et al. (2022), we implement the token dropping and our approach under the same settings, *e.g.*, dropping 50% of the tokens.

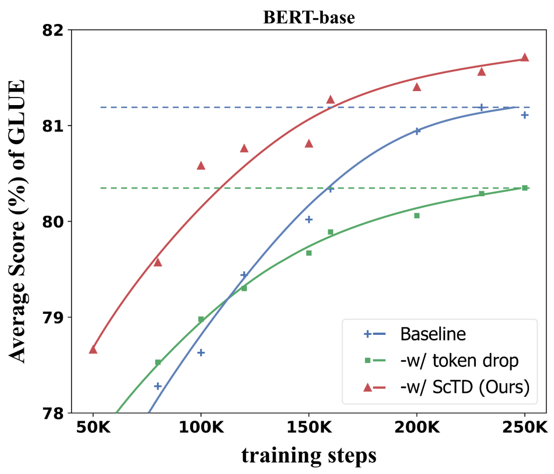

Figure 6: Average scores (%) on GLUE benchmark of $\text{BERT}_{\text{base}}$ models trained with different methods for the full pretraining process. Our method achieves comparable performance with baseline at 150K training steps.

ScTD improves performance on both model sizes. Extensive results show that ScTD works well on both Large and Base BERT models. Specifically, compared to the vanilla token dropping, ScTD brings +1.59% and +1.37% average gains on GLUE tasks, respectively. Results on the other tasks also show a similar phenomenon. Thus, we could recommend our ScTD to speed up the training of all discriminative MLMs regardless of the regime in model capacity.

ScTD effectively improves the training efficiency. Results in Table 2 show that, with our ScTD, BERT models can achieve comparable or even better performance with much fewer training steps, *i.e.*, improving the training efficiency[10]. Specifically, compared to the full training (250K steps) BERT models, ScTD can save up to 48% pretraining time while achieving comparable performance. We attribute it to the higher data efficiency, since ScTD not only takes full advantage of the token dropping's ability to learn important words but also alleviates the semantic loss problem in the token dropping. This can be further proved by the illustration of Figure 6, as ScTD always shows better performance against the other counterparts during the training dynamics. Furthermore, when training with the same iterations, our ScTD can even outperform the standard BERT by a clear margin. We attribute this to the regularization effect of token dropping[11].

[10]While the semantic-consistent learning process in ScTD will introduce extra computation overhead, ScTD performs much better in terms of training efficiency. That is, the relatively little computation overhead is acceptable.

$\mathcal{L}_{MLM}$	$\mathcal{L}_{SC_l}$	$\mathcal{L}_{SC_g}$	GLUE	SGLUE	SQuAD
			Avg.	*Avg.*	*Avg.*
Baseline			77.73	69.11	74.15
token drop			76.58	68.01	72.28
-w/ ScTD (Ours)					
✓			78.30	68.73	75.56
	✓		78.06	69.49	75.66
		✓	79.27	68.64	75.80
✓	✓		78.51	69.64	75.51
✓		✓	79.26	69.32	75.59
	✓	✓	79.36	69.89	75.91
✓	✓	✓	**79.58**	**70.29**	**76.01**

Table 4: Ablation study on different training objectives ($\{\mathcal{L}_{MLM}, \mathcal{L}_{SC_l}, \mathcal{L}_{SC_g}\}$) introduced in our ScTD.

4.3 Ablation Study

We evaluate the impact of each component of our ScTD, including *i*) semantic-consistent learning objectives, *ii*) coefficient λ and *iii*) fixed interval Fi in the hybrid training process. Notably, due to the limited computational budget, we conduct experiments on the $\text{BERT}_{\texttt{large}}$ models trained with different methods for 5 epochs (35K steps).

Impact of different training objectives. As shown in §3, in addition to the original MLM objective $\mathcal{L}^{*}_{MLM}$ of token dropping, we introduce several extra training objectives ($\mathcal{L}_{MLM}, \mathcal{L}_{SC_l}, \mathcal{L}_{SC_g}\}$) to align the semantic information. Here, we conduct experiments to analyze the impact of different objectives and show the results in Table 4. It can be seen that all objectives are beneficial to our ScTD, where the $\mathcal{L}_{SC_g}$ is the most helpful. This indicates the semantic alignment in the global-level representation space is more critical. Also, we can observe that the combination of all objectives performs best, thus leaving as the default setting.

Impact of Coefficient λ. The factor λ in Eq. 4, which is used to balance different objectives, is an important hyper-parameters. In this study, we analyze its influence by evaluating the performance with different λ spanning {0, 0.01, 0.05, 0.25, 0.5} on several GLUE tasks. Figure 7 illustrates the average results. Compared with the baseline, our ScTD consistently brings improvements across all ratios

[11]BERT-style PLMs are often over-parameterized and prone to overfitting. Using regularization methods like token dropping and LayerDrop (Fan et al., 2020) during training can improve model generalization and even boost performance.

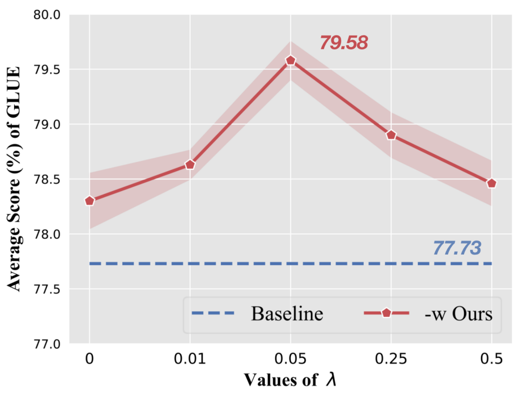

Figure 7: Parameter analysis of λ on $\text{BERT}_{\text{large}}$.

Method	Budget	GLUE	SQuAD
	training time (hours)	*Avg.*	*Avg.*
Baseline	4.93	77.73	74.15
token drop	3.87 (-21.5%)	76.58	72.28
-w/ ScTD (Ours)			
$Fi = 5$	4.69 (-4.9%)	78.96	75.49
$Fi = 10$	4.25 (-13.8%)	**79.58**	**75.80**
$Fi = 20$	4.04 (-18.1%)	78.45	75.74
$Fi = 50$	3.92 (-20.5%)	79.01	75.04

Table 5: Ablation study on different fixed intervals Fi for performing the semantic-align process.

of λ, basically indicating that the performance of ScTD is not sensitive to λ. More specifically, the case of $\lambda = 0.05$ performs best, and we thereby use this setting in our experiments.

Impact of Fixed Interval Fi. In our ScTD, we use a fixed interval Fi to control the frequency for performing the semantic-align process. To verify its impact, we evaluate the performance of ScTD on different Fi and show the results in Table 5. Observably, too small Fi not only causes much computational overhead, but also affects the stability of hybrid training, thus leading to sub-optimal performance. On the contrary, for the larger Fi (*e.g.*, 50), it may be difficult to make full use of the semantic-consistent learning process, hindering the effect of ScTD. In the case of $Fi = 10$, ScTD achieves a better trade-off between costs and performance, which we suggest as the best setting[12].

[12]Some readers may wonder why the teacher (i.e., model with baseline training) trained with only $1/Fi$ steps is strong enough to guide the training of student model. One possible reason for this question is that training with hard-to-learn tokens (Fi-1) times and training with easy-to-learn tokens once is sufficient to obtain remarkable teacher models, similar to the Lookahead Optimizer (Zhang et al., 2019), which updates fast weights k times before updating slow weights once.

4.4 Does ScTD indeed alleviate the semantic loss problem?

Here, we examine whether ScTD can alleviate the limitation of token dropping. Specifically, following the preliminary analyses in §2, we compare our ScTD with other counterparts by probing the trained BERT models (as illustrated in Figure 8) and pertinently evaluating on several semantic-intense tasks (as shown in Table 6).

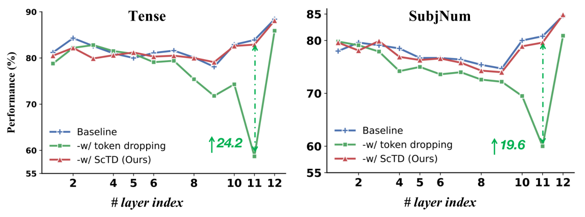

Figure 8: The comparison of semantic information on different $\text{BERT}_{\text{base}}$ layers. We see that ScTD preserves more semantic information than vanilla token dropping.

Method	Onto.	CoNLL03	MRPC	SICK-R	Avg.
	F1	*F1*	*Acc.*	*Spear.*	
Token drop	27.49	53.73	85.50	66.16	58.22
ScTD ($\Delta\uparrow$)	**+2.04**	**+2.59**	**+1.30**	**+2.38**	**+2.08**

Table 6: Experimental results of $\text{BERT}_{\text{base}}$ models on several semantic-intense tasks. We observe that our ScTD brings consistent performance gains.

It can be found that, with our ScTD, BERT learns more semantic information among most layers, especially in dropped layers. Also, ScTD brings consistent and significant performance gains on all semantic-intense tasks against the vanilla token dropping. These results can prove that ScTD is beneficial to address the semantic loss problem.

5 Related Works

Pretraining with Transformer-based architectures like BERT (Devlin et al., 2019) has achieved great success in a variety of NLP tasks (Devlin et al., 2019; Liu et al., 2019; He et al., 2020; Joshi et al., 2020). Despite its success, BERT-style pretraining usually suffers from unbearable computational expenses (Jiao et al., 2020; Zhang and He, 2020). To this end, several training-efficient approaches are proposed to speed up the pretraining and reduce the computational overhead, such as mixed-precision training (Shoeybi et al., 2019), distributed training (You et al., 2019), curriculum learning (Nagatsuka et al., 2021; Ding et al., 2021a) and designing efficient model architectures and optimizers (Gong et al., 2019; Clark et al., 2019b; Zhang and He, 2020; Zhang et al., 2023; Zhong et al., 2022c; Sun et al., 2023). These works mainly focus on efficient optimization processes or model architecture changes.

More recently, Hou et al. (2022) propose the token dropping strategy, which exposes a new mode to speed up the BERT pretraining. Without modifying the original BERT architecture or training setting, token dropping is inspired by the dynamic halting algorithm (Dehghani et al., 2018) and attempts to skip the computations on part of (unimportant) tokens in some middle BERT layers during the forward-propagation process. Owing to its impressive efficiency, token dropping has recently attracted increasing attention (Yao et al., 2022; Chiang et al., 2022). For instance, Yao et al. (2022) apply the token dropping strategy to broader applications, *e.g.*, both NLP and CV communities.

Along with the line of token dropping, we take a further step by exploring and addressing its limitations. To be specific, we first reveal the semantic loss problem (§2) in the token dropping, and then propose a novel semantic-consistent learning method (§3) to alleviate this problem and further improve performance and training efficiency.

6 Conclusion

In this paper, we reveal and address the limitation of token dropping in accelerating language model training. Based on a series of preliminary analyses, we find that removing parts of tokens would lead to a semantic loss problem, which causes vulnerable and unstable training. Furthermore, experiments show such a semantic loss will hinder the performance of token dropping in most semantic-intense scenarios. To address this limitation, we improve token dropping with a novel semantic-consistent learning algorithm. It designs two semantic constraints to encourage models to preserve semantic information. Experiments show that our approach consistently and significantly improves downstream performance across all task types and model architectures. In-depth analyses prove that our approach indeed alleviates the problem, and further improves training efficiency.

In future work, we will explore the effectiveness of our method on more advanced discriminative language models (He et al., 2020; Zhong et al., 2023b). Also, it will be interesting to revisit and address the semantic loss problem in efficient training methods for generative language models (such as GPT3 (Brown et al., 2020)).

Limitations

Our work has several potential limitations. First, given the limited computational budget, we only validate our ScTD on the Large and Base sizes of BERT models. It will be more convincing if scaling up to the larger model size and applying ScTD to more cutting-edge model architectures. On the other hand, besides the downstream performance, we believe that there are still other properties, *e.g.*, generalization and robustness, of MLMs that can be improved by our ScTD approach, which are not fully explored in this work.

Ethics and Reproducibility Statements

Ethics We take ethical considerations very seriously, and strictly adhere to the ACL Ethics Policy. This paper proposes a semantic-consistent algorithm to improve the existing token dropping strategy. The proposed approach aims to speed up the pretraining of BERT-style models, instead of encouraging them to learn privacy knowledge that may cause the ethical problem. Moreover, all pretraining datasets used in this paper are publicly available and have been widely adopted by researchers. Thus, we believe that this research will not pose ethical issues.

Reproducibility We will publicly release our code in https://github.com/WHU-ZQH/ScTD and the pretrained models in https://huggingface.co/bert-sctd-base to help reproduce the experimental results of this paper.

Acknowledgements

We are grateful to the anonymous reviewers and the area chair for their insightful comments and suggestions. This work was supported in part by the National Natural Science Foundation of China under Grants 62225113 and 62076186, and in part by the Science and Technology Major Project of Hubei Province (Next-Generation AI Technologies) under Grant 2019AEA170. Xuebo Liu was supported by Shenzhen Science and Technology Program (Grant No. RCBS20221008093121053). The numerical calculations in this paper have been done on the supercomputing system in the Supercomputing Center of Wuhan University.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval*.

Cheng-Han Chiang, Yung-Sung Chuang, and Hung-Yi Lee. 2022. Recent advances in pre-trained language models: Why do they work and how do they work. In *AACL*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019a. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019b. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single\ &!#* vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F Wong, Dacheng Tao, and Zhaopeng Tu. 2021a. Progressive multi-granularity training for non-autoregressive translation. In *Findings of the ACL*.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F Wong, Dacheng Tao, and Zhaopeng Tu. 2021b. Understanding and improving lexical choice in non-autoregressive translation. In *ICLR*.

Liang Ding, Longyue Wang, Di Wu, Dacheng Tao, and Zhaopeng Tu. 2020. Context-aware cross-attention for non-autoregressive translation. In *COLING*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *ICLR*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *ACL*.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Efficient training of bert by progressively stacking. In *ICML*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *ICLR*.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. In *NeurIPS*.

Le Hou, Richard Yuanzhe Pang, Tianyi Zhou, Yuexin Wu, Xinying Song, Xiaodan Song, and Denny Zhou. 2022. Token dropping for efficient bert pretraining. In *ACL*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Findings of EMNLP*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *TACL*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL-HLT*.

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*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *ICLR*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*.

Koichi Nagatsuka, Clifford Broni-Bediako, and Masayasu Atsumi. 2021. Pre-training a BERT with curriculum learning by increasing block-size of input text. In *RANLP*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI*.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *HLT-NAACL*.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green ai. *Communications of the ACM*.

Li Shen, Yan Sun, Zhiyuan Yu, Liang Ding, Xinmei Tian, and Dacheng Tao. 2023. On efficient training of large-scale deep learning models: A literature review. *arXiv*.

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

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 *EMNLP*.

Hao Sun, Li Shen, Qihuang Zhong, Liang Ding, Shixiang Chen, Jingwei Sun, Jing Li, Guangzhong Sun, and Dacheng Tao. 2023. Adasam: Boosting sharpness-aware minimization with adaptive learning rate and momentum for training deep neural networks. *arXiv*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP*.

Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, Yicheng Zou, Xin Zhou, Jiacheng Ye, Yongxin Zhang, Rui Zheng, Zexiong Pang, et al. 2021. Textflint: Unified multilingual robustness evaluation toolkit for natural language processing. In *ACL*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *TACL*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Guodong Xu, Ziwei Liu, Xiaoxiao Li, and Chen Change Loy. 2020. Knowledge distillation meets self-supervision. In *ECCV*.

Zhewei Yao, Xiaoxia Wu, Conglong Li, Connor Holmes, Minjia Zhang, Cheng Li, and Yuxiong He. 2022. Random-ltd: Random and layerwise token dropping brings efficient training for large-scale transformers. *arXiv*.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. In *ICLR*.

Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. 2019. Lookahead optimizer: k steps forward, 1 step back. In *NeurIPS*.

Minjia Zhang and Yuxiong He. 2020. Accelerating training of transformer-based language models with progressive layer dropping. In *NeurIPS*.

Zheng Zhang, Donglin Yang, Yaqi Xia, Liang Ding, Dacheng Tao, Xiaobo Zhou, and Dazhao Cheng. 2023. Mpipemoe: Memory efficient moe for pre-trained models with adaptive pipeline parallelism.

Zhilu Zhang and Mert Sabuncu. 2020. Self-distillation as instance-specific label smoothing.

Zhuosheng Zhang, Hai Zhao, and Ming Zhou. 2022. Instance regularization for discriminative language model pre-training. In *EMNLP*.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2022a. E2s2: Encoding-enhanced sequence-to-sequence pretraining for language understanding and generation. *arXiv*.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2022b. Panda: Prompt transfer meets knowledge distillation for efficient model adaptation. *arXiv*.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2023a. Self-evolution learning for discriminative language model pretraining. In *Findings of ACL*.

Qihuang Zhong, Liang Ding, Keqin Peng, Juhua Liu, Bo Du, Li Shen, Yibing Zhan, and Dacheng Tao. 2023b. Bag of tricks for effective language model pretraining and downstream adaptation: A case study on glue. *arXiv*.

Qihuang Zhong, Liang Ding, Li Shen, Peng Mi, Juhua Liu, Bo Du, and Dacheng Tao. 2022c. Improving sharpness-aware minimization with fisher mask for better generalization on language models. In *Findings of EMNLP*.

Qihuang Zhong, Liang Ding, Yibing Zhan, Yu Qiao, Yonggang Wen, Li Shen, Juhua Liu, Baosheng Yu, Bo Du, Yixin Chen, et al. 2022d. Toward efficient language model pretraining and downstream adaptation via self-evolution: A case study on superglue. *arXiv*.

A Appendix

A.1 Details of Tasks and Datasets

In this work, we conduct extensive experiments on parts of tasks from GLUE and SuperGLUE. In addition, two widely-used commonsense question answering tasks are also used. Here, we introduce the descriptions of the used tasks and datasets in detail. Firstly, we present the statistics of all datasets in Table 7. Then, each task is described as:

CoLA Corpus of Linguistic Acceptability (Warstadt et al., 2019) is a binary single-sentence classification task to determine whether a given sentence is linguistically "acceptable".

MRPC Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) is a task to predict whether two sentences are semantically equivalent.

STS-B Semantic Textual Similarity (Cer et al., 2017) is a task to predict how similar two sentences are on a 1-5 scale in terms of semantic meaning.

RTE Recognizing Textual Entailment (Giampiccolo et al., 2007), given a premise and a hypothesis, is a task to predict whether the premise entails the hypothesis.

MNLI The Multi-Genre Natural Language Inference Corpus (Williams et al., 2018) is a task to predict whether the premise entails the hypothesis, contradicts the hypothesis, or neither, given a premise sentence and a hypothesis sentence.

SST-2 The Stanford Sentiment Treebank (Socher et al., 2013) is a binary classification task to predict the sentiment of a given sentence.

CB CommitmentBank (De Marneffe et al., 2019) is a task that can be framed as three-class textual entailment on a corpus of 1,200 naturally occurring discourses.

BoolQ Boolean Question (Clark et al., 2019a) is a question answering task where each sample consists of a short passage and a yes/no question about the passage.

MultiRC Multi-Sentence Reading Comprehension (Khashabi et al., 2018) is a QA task where each example consists of a context paragraph, a question about that paragraph, and a list of possible answers. The model need to predict which answers are true and which are false.

COPA Choice of Plausible Alternatives(Roemmele et al., 2011) is a causal reasoning task in which a system is given a premise sentence and must determine either the cause or effect of the premise from two possible choices.

SQuAD v1 The Stanford Question Answering Dataset (Rajpurkar et al., 2016) is a popular reading comprehension benchmark, where the answer to each question is a segment of text from the corresponding reading passage.

SQuAD v2 The latest version of the Stanford Question Answering Dataset (Rajpurkar et al., 2018) is one of the most widely-used reading comprehension benchmarks that require the systems to acquire knowledge reasoning ability.

A.2 Hyper-parameters of Fine-tuning

For fine-tuning, we use the BERT models as the backbone PLMs and conduct experiments using the open-source toolkit fairseq[13] and transformers[14]. Notably, we apply the same hyper-parameters to all PLMs for simplicity. The training epochs/steps, batch size, and learning rate for each downstream task are listed in Table 7.

[13]https://github.com/facebookresearch/fairseq
[14]https://github.com/huggingface/transformers

Task		**#Train**	**#Dev**	**#Class**	**LR**	**BSZ**	**Epochs/Steps**
GLUE	CoLA	8.5K	1,042	2	2e-5	32	2,668 steps
	MRPC	3.7K	409	2	1e-5	32	1,148 steps
	STS-B	5.7K	1,501	-	2e-5	32	1,799 steps
	RTE	2.5K	278	2	1e-5	16	2,036 steps
	MNLI	392K	9,815	3	1e-5	256	15,484 steps
	SST-2	63.3K	873	2	1e-5	64	10,467 steps
SuperGLUE	BoolQ	9.4K	3,270	2	1e-5	16	10 epochs
	CB	250	57	2	2e-5	16	20 epochs
	MultiRC	5.1K	953	2	2e-5	32	10 epochs
	COPA	400	100	2	2e-5	16	10 epochs
Commonsense QA	SQuAD v1	87.6K	10,570	-	3e-5	12	2 epochs
	SQuAD v2	130K	11,873	-	3e-5	12	2 epochs

Table 7: Data statistics and fine-tuning hyper-parameters of all used tasks in this paper. “Class” refers to the label class, “LR” means the learning rate and “BSA” denotes the batch size.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 6

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4 and Appendix A1

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.