Farewell to Aimless Large-scale Pretraining: Influential Subset Selection for Language Model

Xiao Wang★*, **Weikang Zhou**★*, **Qi Zhang**★†, **Jie Zhou**★, **Songyang Gao**★,
Junzhe Wang★, **Menghan Zhang**♦, **Xiang Gao**♣, **Yunwen Chen**♣, **Tao Gui**♦ †
★ School of Computer Science, Fudan University, Shanghai, China
♦ Institute of Modern Languages and Linguistics, Fudan University, Shanghai, China
♣ DataGrand Information Technology (Shanghai) Co., Ltd.
{xiao_wang20,qz,tgui}@fudan.edu.cn

Abstract

Pretrained language models have achieved remarkable success in various natural language processing tasks. However, pretraining has recently shifted toward larger models and larger data, and this has resulted in significant computational and energy costs. In this paper, we propose Influence Subset Selection (ISS) for language model, which explicitly utilizes end-task knowledge to select a tiny subset of the pretraining corpus. Specifically, the ISS selects the samples that will provide the most positive influence on the performance of the end-task. Furthermore, we design a gradient matching based influence estimation method, which can drastically reduce the computation time of influence. With only 0.45% of the data and a three-orders-of-magnitude lower computational cost, ISS outperformed pretrained models (e.g., RoBERTa) on eight datasets covering four domains.

1 Introduction

Pretrained language models (PTMs) (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019), trained on massive and heterogeneous corpora, have significantly improved the state-of-the-art across a variety of natural language processing tasks (Wang et al., 2022, 2023). Kaplan et al. (2020) found power laws relating cross entropy loss to the sizes of language models and their training datasets. As a result, the field has recently shifted toward larger models and large data (Brown et al., 2020; Rae et al., 2021; Smith et al., 2022; Chowdhery et al., 2022) in hopes of improving performance.

However, training a state-of-the-art language model requires substantial computational resources which demand considerable energy, along with the associated financial and environmental costs (Strubell et al., 2019). For example, RoBERTa-Large (Liu et al., 2019), which was trained on 1000 V100 GPUs for approximately one day, has a computational cost of 4.36×10^{21} FLOPs. Recently, Chowdhery et al. (2022) proposes PaLM, which consumes 580 times more FLOPs than RoBERTa-Large. PaLM was trained on 6144 TPU v4 chips for more than 1200 hours, which is unaffordable for most researchers. Therefore, finding ways to speed up pretraining is crucial for the development of pretrained model research.

	PLMs	**TLM**	**ISS**
Training Data	The entire $\mathcal{D}$	Subset of $\mathcal{D}$ & task data $\mathcal{T}$	Subset of $\mathcal{D}$ & task data $\mathcal{T}$
Compute Cost	240000 GPU·hours	240 GPU·hours	80 GPU·hours
Generality	Task-Agnostic	X-Dep	$X\&Y$-Dep

Table 1: Qualitative comparison between PLMs, TLM, and ISS(ours). X/Y-Dep means the pretraining data is X/Y dependent.

In general, there are three main strategies used to speed up pretraining in NLP: parallel architectures, efficient model architectures, and novel pretraining tasks. The first one is to train a single model utilizing multiple GPUs distributed in many computational nodes (Wang et al., 2020b; Shazeer et al., 2018; Huang et al., 2019). Unfortunately, the gains in efficiency of this strategy depend entirely on the amount of computing hardware used. The second strategy is to improve model structures to reduce the computational complexity and therefore improve efficiency (Wang et al., 2020a; Katharopoulos et al., 2020; Roy et al., 2021). The last one explores more challenging pretraining tasks to accelerate a model's convergence (Clark et al., 2019; Joshi et al., 2020; Levine et al., 2020). However, their improvements are limited, with a reduction of less than an order of magnitude in computational expenses (measured in FLOPs).

In this paper, we aim to reduce the computational costs from data level (See Table 1). The PLMs are trained on the entire pretraining corpus D, which is task-agnostic. To take the downstream task

[*] Equal contribution.
[†] Corresponding Author

into account, we hope to select the most relevant samples from the pretraining corpus based on the downstream data. Recently, Yao et al. (2022) proposes TLM, which retrieves data from a pretraining corpus using task data as queries. However, TLM remains task-agnostic, because it only considers text (i.e., X) similarities and ignores the label (i.e., Y) information.

Motivated by influence function (Cook and Weisberg, 1982; Koh and Liang, 2017), we propose Influential Subset Selection (ISS) for language model, i.e. selecting the samples with the most positive influence on the downstream task. To calculate the label-aware influence value, ISS utilizes the derivation chain rule from a test objective to training samples. Nevertheless, directly applying the chain rule leads to computing the inverse of Hessian with the complexity of $O(nq^2 + q^3)$(n is the number of examples and q is parameter size), which is computationally expensive and may run out-of-memory in neural networks. To address this problem, we propose a gradient matching based influence approximation method for selecting pretraining data, which estimates the influence score by matching the gradient values of pretraining samples and end-task samples. Our method avoids the computation of the inverse of Hessian and significantly speeds up the estimation time of influence.

Our main contributions are summarized as follows:

- We propose Influential Subset Selection for language model, which explicitly utilizes knowledge of the end-task to select the pretraining corpus.
- We design a simple, efficient, gradient matching based method for influence estimation, which avoids the calculation of the inverse of Hessian and significantly speeds up the estimation time.
- We evaluate the effectiveness of our method on eight tasks covering four domains. Notably, ISS outperforms PTMs (e.g. RoBERTa) with only **0.45% of the data** and **three orders of magnitude reduced FLOPS**. Our code can be found at `https://github.com/nitwtog/ISS`.

2 Preliminaries

2.1 Definition

We assume an end-task dataset represented as $\mathcal{T} = (\mathcal{Z}_t)$ where $\mathcal{Z}_t = \left\{(x_t^1, y_t^1), (x_t^2, y_t^2), \ldots, (x_t^m, y_t^m)\right\}$ represents a set of texts with their ground truth labels. And we assume a large-scale pretraining corpus $\mathcal{D} = (\mathcal{Z}_p)$, where $\mathcal{Z}_p = \left\{x_p^1, x_p^2, \ldots, x_p^M\right\}$ represents unlabeled data. We define $f = f^{(\text{head})} \circ f^{(\text{feat})}$, such that $f^{(\text{feat})}(\cdot; \theta \in \Theta)$ is a feature extractor that is transferable across learning stages (e.g. pretraining to finetuning) and $f^{(\text{head})}(\cdot; \phi \in \Phi)$ is a task-specific head that is not transferable. And we assume $l_p(z_p, \theta, \phi_p)$ and $l_t(z_t, \theta, \phi_t)$ are the loss functions of pretraining and end-task.

2.2 Influence Function

Influence function (Cook and Weisberg, 1982; Koh and Liang, 2017) provides an efficient way to estimate the importance of a training sample. Considering a training sample z was weighted by a small ϵ during training, the empirical risk minimizer can be written as

$$\hat{\theta}_{\epsilon,z} = \arg\min_{\theta\in\Theta} \frac{1}{n} \sum_{z_i \in \mathcal{D}} l\left(z_i, \theta\right) + \epsilon \cdot l(z, \theta) \quad (1)$$

Assigning $-\frac{1}{n}$ to ϵ is equivalent to removing the training example z_p. Then, the influence of weighting z_p on the parameters is given by

$$\mathcal{I}_{\text{param}}\left(z\right) = \left.\frac{\mathrm{d}\hat{\theta}_{\epsilon,z}}{\mathrm{d}\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_\theta l(z, \hat{\theta}) \quad (2)$$

where $H_{\hat{\theta}} = \frac{1}{n} \sum_{z_i \in \mathcal{D}} \nabla_\theta^2 l\left(z_i, \hat{\theta}\right)$ is the Hessian and positive definite by assumption, $\mathcal{I}_{\text{param}}\left(z\right) \in R^N$, N is the number of network parameters. Then, we can linearly approximate the parameter change due to removing z without retraining the model by computing $\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n} \mathcal{I}_{\text{param}}\left(z\right)$.

3 Methodology

We investigate an influence-based subset selection method to perform efficient pretraining while attempting to minimize accuracy loss on the end-task dataset (Section 3.1). Due to the high computational costs of influence function (Koh and Liang, 2017), we design an influence approximation strategy to speed up the calculation (Section 3.2).

3.1 Influence of Pretraining Corpus

PTMs used in previous works usually adopt language modeling as pretraining tasks, lacking task-specific prior knowledge. However, we often know the end-task beforehand, so we can make specific choices about our pretraining regimen to improve end-task performance. Under this setting, we introduce Influential Subset Selection for language model, which measures the importance of pretraining samples by considering the X and Y information of the end-task simultaneously.

Specifically, pretraining sample z_p affects the prediction of end-task sample z_t by influencing the parameters of the feature encoder θ. We can apply the chain rule to measure the influence of upweighting pretraining sample z_p on the loss at end-task sample z_t.

$$\begin{aligned}\mathcal{I}\left(z_p, z_t\right) &\triangleq \left.\frac{dl\left(z_t, \hat{\theta}_{\epsilon,z}\right)}{d\epsilon}\right|_{\epsilon=0} \\ &= \nabla_\theta l\left(z_t, \hat{\theta}\right)^\top \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} \\ &= -\nabla_\theta l\left(z_t, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta l(z_p, \hat{\theta})\end{aligned} \tag{3}$$

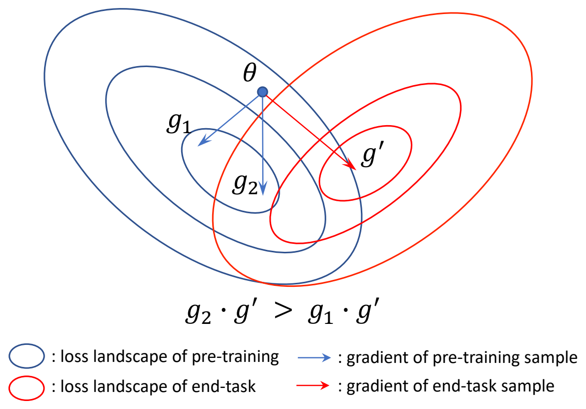
The more negative $\mathcal{I}\left(z_p, z_t\right)$ is, the more positive influence z_p can provide. However, computing the Hessian for the full training dataset is expensive, and inverting it is similarly prohibitive: with n training data points and p parameters, this computation requires $O(n * p^2 + p^3)$ operations. It means that evaluating the influence of large-scale pretrained corpus is not achievable. Thus, we propose an influence approximation algorithm to speed up the estimation time.

3.2 Influence Approximation

Motivated by calculus, the update of the model parameters is the result of cumulative updates over several training iterations. Similarly, the difference between the loss of test point z_t at the end of training versus at the beginning of training can be decomposed along the path taken by the training process. Thus, we hypothesize that the influences of all training examples on a fixed test point z_t is exactly the total reduction in loss on z_t.

Assume that we train the feature encoder by minimizing the pertaining loss $l_p(z_p; \theta, \phi)$, via an iterative optimization procedure (such as SGD) which utilizes one training example z_p in iteration t. The parameters of the feature encoder before and after iteration t are θ_t and θ_{t+1} respectively. The influence of z_t on z_p can be approximated in the following way.

$$\mathcal{I}\left(z_p, z_t\right) = l_t\left(z_p, \theta_t\right) - l_t\left(z_p, \theta_{t+1}\right) \tag{4}$$



θ
g_1
g_2
g'
$g_2 \cdot g' > g_1 \cdot g'$
: loss landscape of pre-training
: loss landscape of end-task
: gradient of pre-training sample
: gradient of end-task sample

Figure 1: Illustration of gradient matching based influence approximation. g_1 and g_2 are the loss gradients of two different pretrained samples respectively, while g' is the loss gradient of the end-task sample. The influence of a pretrained sample is measured by how a small step based on its gradient affects the loss on the end-task sample. Compared to g_1, the update step of g_2 is more generalized.

Suppose we are at point θ_t, and we make a first-order Taylor expansion of function $l_p\left(z_p, \theta_{t+1}\right)$.

$$\begin{aligned}l_t\left(z_p, \theta_{t+1}\right) =& l_t\left(z_p, \theta_t\right) + \nabla_\theta l_t\left(z_p, \theta_t\right) \cdot \left(\theta_{t+1} - \theta_t\right) \\ &+ O\left(\left\|\theta_{t+1} - \theta_t\right\|^2\right)\end{aligned} \tag{5}$$

Assuming the model employs SGD as the optimizer, then the update in parameters is $\theta_{t+1} - \theta_t = -\eta_t \nabla_\theta l_p\left(z_t, \theta_t\right)$, where η_t is the learning rate at iteration t. Eq. (5) guarantees approximation precision as long as the update magnitude of θ is sufficiently small. By substituting the parameter update formula and disregarding the higher-order term, we arrive at the following first-order approximation.

$$l_t\left(z', \theta_t\right) - l_t\left(z', \theta_{t+1}\right) \approx \eta_t \nabla_\theta l_t\left(z', \theta_t\right) \cdot \nabla_\theta l_p\left(z_t, \theta_t\right) \tag{6}$$

We refer to this first-order approximation as gradient matching-based influence estimation. The full algorithm is provided in Algorithm 1.

Visualisation We visualize our influence estimation method in Fig 1. g_1 and g_2 are the loss gradients of two different pretrained samples respectively, while g' is the loss gradient of the end-task sample. The influence of a pretrained sample can be viewed as the dot product of its gradient and the gradient of the end-task sample. Higher influence suggests that a network is learning parameters that generalize.

Algorithm 1: Influential Subset Selection for Language Model

```
Require: Pretraining corpus $\mathcal{D}$; task training set $\mathcal{T}_t$ and validation set $\mathcal{T}_v$; learning rate $\alpha$; initial subset $S$; candidates size k.
Random initialize network $\theta, \phi_p, \phi_t$
$\hat{\theta}, \hat{\phi_p}, \hat{\phi_t} = \arg\min \frac{1}{n} \sum_{z_i \in \mathcal{T}_t} l_p(z_i) + l_t(z_i)$
for $z_p \in \mathcal{D}$ do
    Compute $\nabla_\theta l_p\left(z_p, \hat{\theta}, \hat{\phi_p}\right)$
end
for $z' \in \mathcal{T}_v$ do
    Compute $\nabla_\theta l_t\left(z', \hat{\theta}, \hat{\phi_t},\right)$
    for $z_p \in \mathcal{D}$ do
        $\mathcal{I}(z_p, z') = \nabla_\theta l_p\left(z_p, \hat{\theta}, \hat{\phi_p}\right) \cdot \nabla_\theta l_t\left(z', \hat{\theta}, \hat{\phi_t},\right)$
    end
    Sort pretraining samples based on influence
    Add top k influential samples to $S$
end
Return influential subset $S$
```

3.3 Implementation Details

Based on the influence score, we select the most relevant samples from the pretraining corpus. Following TLM, we first select a subset via a BM25 retrieval method. Then, we compute the influence score based on this subset to make ISS scalable and efficient.

Moreover, the number of parameters in large-scale language models is very large, leading to very high dimensional gradients. To tackle this problem, we adopt a last-layer gradient approximation by only considering the last layer gradients of pretrained encoder. We select a subset of mini-batches by matching the weighted sum of mini-batch pretraining gradients to the mini-batch task gradients. Let B_p and B_t be the batch size of pretraining and end-task. The use of mini-batches considerably reduces the number of selection rounds during the ISS algorithm by a factor of B, resulting in $B_p * B_t$ speed up.

4 Experimental Setup

To evaluate the efficiency and generality of our approach, we conduct experiments in two settings: pretraining from scratch, and further pretraining.

4.1 Pretraining from Scratch

Datasets. Following the setting of Gururangan et al. (2020); Yao et al. (2022), we conduct experiments on eight tasks covering four domains, including biomedical science, computer science, news, and reviews. The tasks represent both high- and low-resource ($\leq$ 5K samples) settings, including CHEMPROT (Kringelum et al., 2016), RCT (Dernoncourt and Lee, 2017), ACL-ARC (Jurgens et al., 2018), SCIERC (Luan et al., 2018), HyPERPARTISAN (Kiesel et al., 2019), AGNEws (Zhang et al., 2015), HELPFULNESS (McAuley et al., 2015), IMDB (Maas et al., 2011). Table 2 reports the statistic results of various target datasets. Similar to TLM (Yao et al., 2022), we collect two pretraining corpora that respectively match the original corpora of BERT and RoBERTa. We name them $\mathcal{C}_{BERT}$ and $\mathcal{C}_{RoBERTa}$, respectively.

Baselines. We focus on comparison with general PLMs and TLM. Following Yao et al. (2022), we finetuned both BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) of base and large scales as our baselines. And we finetuned the released TLM models as baselines.

Evaluation Strategy. The results of the experiment are the average performance of three random seeds with the standard deviations. Following Gururangan et al. (2020), we report the test micro-F1 for ChemProt and RCT, and macro-F1 for the rest of datasets. Following TLM (Yao et al., 2022), we set three pretraining scales, namely $small$, $medium$, and $large$ scales. Differently, at the same scale, our method only utilizes 20% size of the TLM data. More detailed settings are shown in Table A.1 in Appendix.

Training Details. We utilize the randomly initialized BERT of base scale as our starter models. We mostly follow optimization, and hyper-parameters choices used in Yao et al. (2022). All experiments were conducted on 4 NVIDIA GeForce RTX 3090 GPUs. Detailed hyper-parameters are provided in Table A.1 in Appendix.

Domain	Task	Train	Dev.	Test	Classes
BIOMED	CHEMPROT	4169	2427	3469	13
	† RCT	18040	30212	30135	5
CS	ACL-ARC	1688	114	139	6
	SCIERC	3219	455	974	7
NEWS	HYPERPARTISAN	515	65	65	2
	† AGNEWS	115000	5000	7600	4
REVIEWS	† HELPFULNESS	115251	5000	25000	2
	† IMDB	20000	5000	25000	2

Table 2: Statistics of various target datasets. † indicates high-resource settings.

Model	Param	Data[1]	FLOPs[2]	AGNews	Hyp.	Help.	IMDB	ACL.	SciERC	Chem.	RCT	Avg.
Bert-Base	109M	16G	2.79E19	93.50 ±0.15	91.93 ±1.74	69.11 ±0.17	93.77 ±0.22	69.45 ±2.90	80.98 ±1.07	81.94 ±0.38	87.00 ±0.06	83.46
Bert-Large	355M	16G	9.07E19	93.51 ±0.40	91.62 ±0.69	69.39 ±1.14	**94.76** ±0.09	69.13 ±2.93	**81.37** ±1.35	**83.64** ±0.41	**87.13** ±0.09	83.82
TLM(Small)	109M	0.91G	2.74E18	93.74 ±0.20	93.53 ±1.61	70.54 ±0.39	93.08 ±0.17	69.84 ±1.53	80.51 ±1.53	81.99 ±0.42	86.99 ±0.03	83.78
TLM(Small-20%)[3]	109M	0.18G	1.82E18	93.57 ±0.21	93.11 ±0.46	70.02 ±0.40	93.20 ±0.03	67.27 ±2.85	78.87 ±0.63	80.80 ±0.63	86.65 ±0.01	82.93
ISS(Small-scale)	109M	0.18G	1.82E18	**93.78** ±0.06	**93.53** ±0.00	**70.78** ±0.29	93.25 ±0.07	**72.41** ±0.66	80.56 ±0.43	81.71 ±0.10	86.99 ±0.02	**84.11**
RoBERTa-Base	125M	160G	1.54E21	**94.02** ±0.15	93.53 ±1.61	70.45 ±0.24	**95.43** ±0.16	68.34 ±7.27	81.35 ±0.63	82.60 ±0.53	87.23 ±0.09	84.12
TLM(Medium)	109M	1.21G	8.30E18	93.96 ±0.18	**94.05** ±0.96	70.90 ±0.73	93.97 ±0.10	72.37 ±2.11	81.88 ±1.92	83.24 ±0.36	87.28 ±0.10	84.71
TLM(Medium-20%)[3]	109M	0.18G	4.15E18	93.78 ±0.02	93.53 ±0.00	71.11 ±0.05	93.20 ±0.06	68.82 ±3.56	80.35 ±0.54	81.05 ±0.07	87.00 ±0.05	83.58
ISS(Medium-scale)	109M	0.18G	4.15E18	93.92 ±0.08	93.53 ±0.00	**71.51** ±0.31	93.61 ±0.06	**73.42** ±0.58	**82.20** ±0.40	**83.42** ±0.11	**87.30** ±0.02	**84.86**
RoBERTa-large	355M	160G	4.36E21	**94.30** ±0.23	**95.16** ±0.00	70.73 ±0.62	**96.20** ±0.19	72.80 ±0.62	82.62 ±0.68	**84.62** ±0.50	**87.53** ±0.13	**85.50**
TLM(Large)[4]	109M	3.64G	2.33E19	94.15 ±0.01	93.92 ±0.72	71.83 ±0.11	94.44 ±0.10	74.18 ±0.29	82.77 ±0.72	83.60 ±0.08	87.49 ±0.02	85.31
TLM(Large-20%)[3]	109M	0.72G	8.30E18	93.79 ±0.31	92.72 ±0.783	71.50 ±0.28	94.49 ±0.04	73.42 ±1.75	81.77 ±0.54	82.63 ±0.11	87.36 ±0.10	84.71
ISS(Large-scale)	109M	0.72G	8.30E18	94.22 ±0.04	93.53 ±0.00	**72.27** ±0.20	94.57 ±0.06	**74.53** ±1.38	**83.12** ±0.16	83.31 ±0.36	87.41 ±0.02	85.36

1 For ISS, data size is reported by averaging over eight tasks.
2 The training compute (FLOPs) is calculated by (6 × Training Tokens × Parameter Size) as in Kaplan et al. (2020).
3 ISS utilizes 20% of the TLM size data, so we implemented the TLM model with the same size version.
4 For a fair comparison, we implement TLM(Large) with BERT base and TLM large scale dataset.

Table 3: Evaluation results for ISS at three different training scales. For each task, we report the average F1 score across three random seeds with standard deviations as subscripts. We also show the number of parameters, the total training compute (FLOPs), and the size of training corpus for comparison.

4.2 Further Pretraining

Datasets. We perform further pretraining in biomedical science and computer science domains. Specifically, we conduct experiments on four datasets, including CHEMPROT (Kringelum et al., 2016), RCT (Dernoncourt and Lee, 2017), ACL-ARC (Jurgens et al., 2018), SCIERC (Luan et al., 2018). For the pretraining stage, we collect the unlabeled datasets from S2ORC (Lo et al., 2020).

Baselines. We select general PTMs (Devlin et al., 2019; Liu et al., 2019) and domain-specific further pretraining models (Lee et al., 2020; Beltagy et al., 2019; Gururangan et al., 2020) as our baselines. Finetuning on the end-task occurs after further pretraining on domain unlabeled corpora.

Evaluation Strategy. Similar to pretraining from scratch, we report the average performance across three random seeds. And we report the micro-F1 for ChemProt and RCT, and macro-F1 for ACL-ARC and SCIERC.

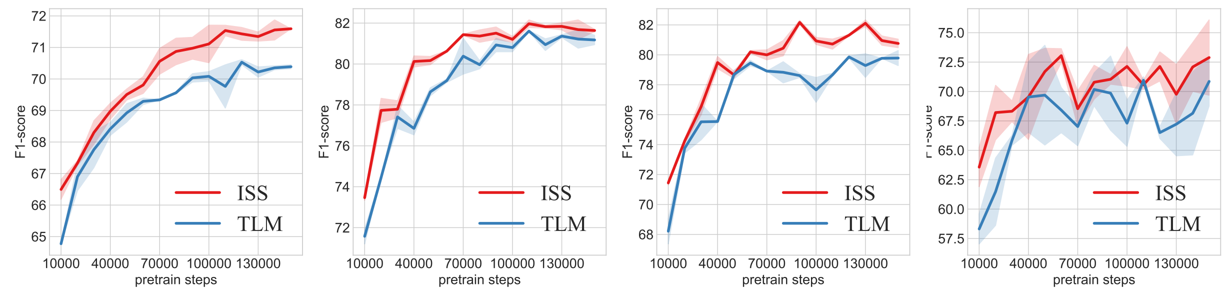
Training Details. In this setting, we perform further pretraining on off-the-shelf pretrained models, such as BERT and RoBERTa. All experiments were conducted on 4 NVIDIA GeForce RTX 3090 GPUs. Detailed hyper-parameters are provided in Table A.2 in Appendix.

5 Experimental Results

In this section, we will discuss the results of comparing our methods against other baselines.

5.1 Pretraining from Scratch

Table 3 shows the main results of ISS with the according TLM and PLMs baselines at three different scales. The followings are the related comparison and analysis we conducted: **1)** ISS could achieve re-



(a) Helpfullness (b) Chemprot (c) SciERC (d) ACL-ARC

Figure 2: Comparison of ISS and TLM at different pretraining steps. The experiments were conducted on the small-scale dataset, and notably, the data scale of TLM was five times larger than ours.

Model	**Param**	**Data**	**FLOPs**	**BIOMED**		**CS**		**Avg.**
				RCT	Chem	ACL	SciERC	
BERT-Base	109M	16G	2.79E19	87.00	81.94	69.45	80.98	79.84
RoBERTa-base	125M	160G	1.54E21	87.23	82.60	68.34	81.35	79.88
SciBERT	109M	15G	2.65E19	-	83.64	70.98	79.97	-
BioBERT	109M	96G	1.80E20	-	76.46	-	-	-
DAPT	125M	47G	1.58E18	87.6	84.2	75.4	80.8	82.00
DAPT+TAPT	125M	47G	1.77E18	**87.8**	84.4	75.6	81.3	82.28
ISS-DAPT(BERT)	109M	1.7G	6.9E17	87.36 ±0.02	83.90 ±0.10	76.06 ±0.70	**83.91** ±0.38	82.81
ISS-DAPT(RoBERTa)	125M	1.7G	7.9E17	87.57 ±0.06	**84.88** ±0.10	**76.70** ±0.25	82.23 ±0.30	**82.85**

Table 4: Evaluation results for ISS in further pretraining. We report the average F1 score across three random seeds with standard deviations as subscripts.

	AGNews		SciERC		Chemprot	
	ISS	TLM	ISS	TLM	ISS	TLM
10%	94.34 ±0.08	94.08 ±0.07	80.82 ±0.41	81.41 ±0.16	80.80 ±0.34	80.15 ±0.32
20%	**94.40** ±0.06	94.16 ±0.09	**83.70** ±0.31	81.21 ±0.44	**82.82** ±0.41	81.51 ±0.55
40%	94.14 ±0.05	94.05 ±0.18	83.16 ±0.07	82.48 ±0.43	81.98 ±0.14	81.75 ±0.04
60%	94.08 ±0.02	94.07 ±0.09	82.51 ±0.29	**83.05** ±0.20	82.08 ±0.22	81.80 ±0.41
80%	94.17 ±0.04	**94.27** ±0.09	81.71 ±0.24	81.75 ±0.15	81.83 ±0.30	**81.86** ±0.47

Table 5: Results on the development set with different data scales.

sults that are better than or comparable to the PLM baselines with significant reductions in FLOPs and the size of training data. At the large scale, ISS achieves comparable results to RoBERTa-large, with an average of 0.19% of FLOPs and 0.45% of the training corpus. At the small and medium scales, ISS improves the performance by 0.29 and 0.74 points on average respectively; **2)** At the same data scale, ISS significantly outperforms TLM, which indicates that task label information is crucial. And the influence-based subset selection can select more influential pertaining samples; **3)** ISS could offer limited performance gains on high-resource datasets. It demonstrates that the influence of the pretraining samples would be decreased as the task data grows sufficiently.

5.2 Further Pretraining

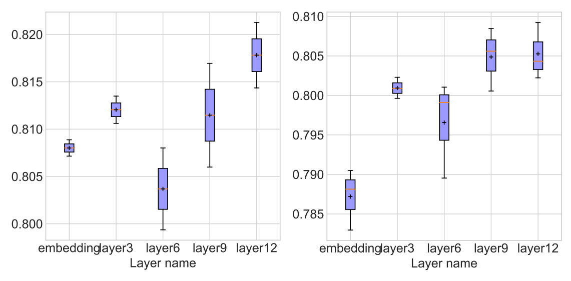
We compared ISS with other domain-specific further pretraining methods. Differently, we initialize the network with off-the-shelf pretrained models to provide initialization and select influential subsets from the domain corpus. Table 4 shows the main results. In conclusion, our method outperforms all the baselines, with significant reductions in FLOPs and the size of training data by one order of magnitude or more. It proves our approach is feasible.

5.3 Comparison of Pretraining Steps

To validate the effect of pretraining steps, we compare the performance of ISS with TLM at different pretraining steps. The test results on the four tasks with different pretraining steps are shown in Figure 3. We observe that ISS could achieve the best performance with fewer steps on most of the datasets.

5.4 Subset Size for Pretraining

To compare the performance at different data scales, we extracted subsets from the TLM small-scale corpus at different scales via ISS and TLM, respectively. The results are shown in Table 5. We can observe that the performance of TLM becomes better as the dataset grows, but the best results are still lower than those of our method. In ISS, the F1-score would reach the top at the 20%-40% scale and gradually decrease as the data size grows. We



(a) Chemprot (b) SciERC

Figure 3: F1-score results of ISS with gradients of different layers (i.e., Embedding layer, 3/6/9/12-th Transformer Block) over Chemprot and SciERC.

	Related-Label	PMI	AGNews	
			ISS(small) /%	TLM(small) /%
immigration	World	1.341	0.0072	0.0070
policy	World	1.187	0.0493	0.0401
china	World	0.382	0.0836	0.0695
medals	Sports	1.400	0.0139	0.0136
golds	Sports	1.400	0.0009	0.0008
sports	Sports	1.293	0.0459	0.0454
financial	Business	1.054	0.0717	0.0567
commerce	Business	0.844	0.0097	0.0081
business	Business	0.710	0.1170	0.0952
automation	Sci/Tech	1.420	0.0043	0.0028
internet	Sci/Tech	1.224	0.0729	0.0524
technology	Sci/Tech	1.115	0.0864	0.0661

Table 6: Comparison of the frequency of task influential words in different subsets.

believe that as the dataset expands, task-irrelevant or noisy data is added.

5.5 Last Better than First

As explained in Section 3.3, the last layer of gradients of the model encoder is only considered to speed up the computation. We have studied the relationship between the gradients at the different layers used in ISS and the corresponding performances. Table 3 shows the results on Chemprot and SciERC. We can observe that the closer the layer, to the task head, the better the selected subset works. The phenomena suggest that different layers in the language model can capture different information, with layers closer to the task head learning more information about the task.

Table 7 shows the times required by ISS calculating influences at the different layers. Overall, the time cost of selecting a subset is negligible compared to pretraining. In addition, the computational speed based on the last layer would be nearly double, compared to that at the embedding layer.

Layer name	Cost times	
	Small	Large
Embedding	2.0 hours	5.2 hours
3-th Transformer	1.8 hours	4.8 hours
6-th Transformer	1.6 hours	4.4 hours
9-th Transformer	1.4 hours	4.0 hours
12-th Transformer	1.1 hours	3.6 hours

Table 7: Comparison of the speed of computing influences using different layers. The experiments were conducted on Chemport dataset.

6 Analysis

6.1 Visualization of Pretrained Model

We visualize the task data on ISS-small, BERT, and RoBERTa, using the t-SNE algorithm (Van der Maaten and Hinton, 2008). The results are shown in Figure 4. We can observe that the different classes of deep features in ISS-small formed tighter clusters, suggesting that ISS provides better initialization for downstream tasks. In contrast, the features learned by BERT and Roberta are distributed respectively in separate clusters with overlapping parts that could not be distinguished.

6.2 Analyzing of Task-influential Words

We compute the point-wise mutual information (PMI) (Levy and Goldberg, 2014) between words and their corresponding labels in the task dataset. Briefly, PMI is used to measure the likelihood of two events occurring together, so the higher the PMI a word has, the more likely it is to be task-influential. We select words with high PMI as task-influential words, and compare their frequency in ISS-small and TLM-small datasets, respectively. As shown in Table 6, the word frequency in the ISS-small dataset is higher than that in the TLM-small dataset. Thus, ISS may focus more on task-influential words.

7 Related Work

7.1 Efficient Pretraining for PLMs

Many attempts have been made to improve the efficiency of pretraining. Parallel architectures (Shazeer et al., 2018; Wang et al., 2020b) are commonly used in pretraining. However, parallelism

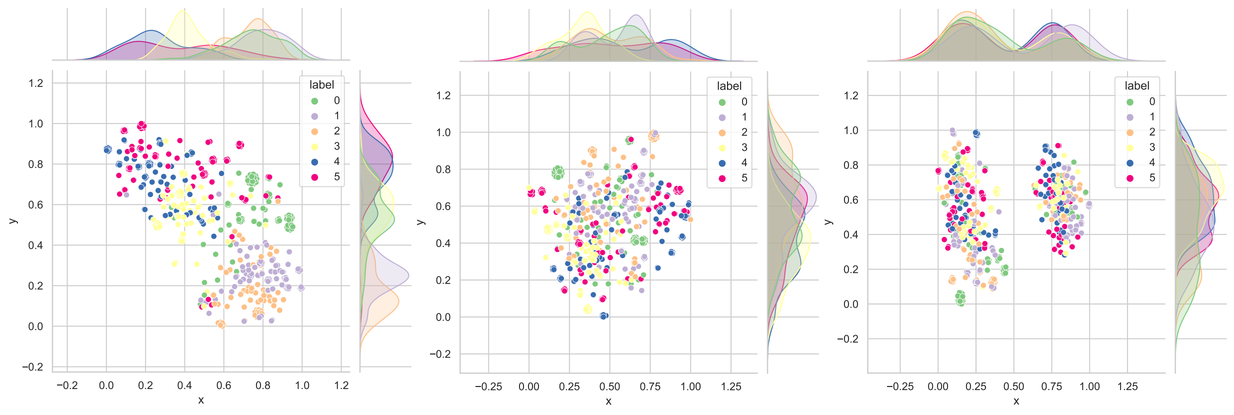

(a) ISS (b) BERT (c) RoBERTa

Figure 4: Visualization of sentence representation on Chemprot using the t-SNE algorithm (Van der Maaten and Hinton, 2008). Each color denotes a class.

would not actually reduce computational costs in terms of FLOPs. For most Transformer-based PTMs, as their input sequence goes longer, their efficiency is limited by the computation of attention weights. Choromanski et al. (2020) and Wang et al. (2020a) design low-rank kernels to theoretically approximate the original attention weights. Child et al. (2019) and Roy et al. (2021) introduce sparsity into attention mechanisms by limiting the view of each token to a fixed size and separating tokens into several chunks. ELECTRA (Clark et al., 2019) applies the replaced token detection which is more challenging. PMI-Masking (Levine et al., 2020) selectively masks tokens based on their importance. However, their improvements are limited, with less than an order of magnitude reduction in computational expenses (measured in FLOPs). Orthogonal to these works, ISS investigates reducing training data redundancy by the influence of pretraining data points.

7.2 Further Pretraning in NLP

Continually pretraining can effectively improve PTMs' performance on new domains or downstream tasks (Gururangan et al., 2020). To achieve it, most previous works continually optimize the pretrained model parameters on a large number of corpora collected from the target domain (e.g., scientific (Beltagy et al., 2019), finance(Araci, 2019) and bio-media (Lee et al., 2020)). However, it is computationally expensive to further pretrain the model on a large amount of unlabeled data and it may not be feasible to collect such a large scale of unlabeled data on certain domains. In contrast, ISS does not need any additional domain data and only utilizes the general corpus. In addition, our approach can also be employed for further pretraining, as we demonstrate in our experiments.

7.3 Dataset Pruning

Dataset pruning is closely related to the coreset selection methods (Mirzasoleiman et al., 2020; Agarwal et al., 2004), which try to identify the most representative training samples. Several works (Killamsetty et al., 2021; Rebuffi et al., 2017; Toneva et al., 2018) have studied dataset pruning for efficient training of deep learning models in supervised learning and active learning scenarios. Dataset pruning methods typically rely on a predefined criterion to compute a scalar score for each training example, e.g. the compactness (Rebuffi et al., 2017), diversity (Sener and Savarese, 2017), and forgetfulness (Toneva et al., 2018), and then rank and select the training data according to the computed score. Recently, Yao et al. (2022) proposed TLM for transfer learning, which retrieves a subset from the pretraining corpus that is more similar to the task corpus. However, these methods are heuristic and lack of generalization guarantee, they also discard the influence interaction between the collected samples. Our proposed method overcomes these shortcomings.

8 Conclusion

In this paper, we propose Influential Subset Selection for language model, which aims to reduce the computational costs of pretraining from data level. Specifically, we introduce influence function to measure the importance of each pretraining sample. Moreover, we design a simple, efficient, gradient matching-based method for influence estimation, which significantly speeds up the estimation time. Experiments on various datasets demonstrate that our method achieves comparable performance with PTMs, with a reduction of training FLOPs by three orders of magnitude.

Limitations

There are two potential risks with our method. First, ISS trades generality for efficiency by learning only task-specific representations. Consequently, it may not be suitable for other tasks. Secondly, our method is hardly practical for few-shot or zero-shot learning, as few or no task data are available as anchor points. These potential risks are left to future work.

Ethics Statement

Pretraining from scratch and further pretraining such as DAPT need large-scale unlabeled corpus to learn general knowledge, which results in corresponding greenhouse emissions due to energy consumption (Strubell et al., 2019). However, as shown in Section 5, our new efficient algorithms greatly increase the data efficiency of PTMs, reducing these harms as well as the various harms associated with labor for data collection. Our work introduces a new subset selection algorithm but leverages pre-existing datasets and models. Overall, this work inherits some of the risks of the original work upon which it is implemented, (such as bias (Bender et al., 2021) or privacy leakage (Carlini et al., 2021).

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No.62206057,61976056,62076069), Shanghai Rising-Star Program (23QA1400200), Natural Science Foundation of Shanghai (23ZR1403500), Program of Shanghai Academic Research Leader under grant 22XD1401100, and CCF-Zhipu AI Large Model Fund.

References

Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. 2004. Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635.

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

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. *Advances in neural information processing systems*, 33:1877–1901.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. In *International Conference on Learning Representations*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

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

R Dennis Cook and Sanford Weisberg. 1982. *Residuals and influence in regression*. New York: Chapman and Hall.

Franck Dernoncourt and Ji Young Lee. 2017. PubMed 200k RCT: a dataset for sequential sentence classification in medical abstracts. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 308–313, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR.

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, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Krishnateja Killamsetty, S Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. 2021. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Taboureau. 2016. Chemprot-3.0: a global chemical biology diseases mapping. *Database*, 2016.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Yoav Levine, Barak Lenz, Opher Lieber, Omri Abend, Kevin Leyton-Brown, Moshe Tennenholtz, and Yoav Shoham. 2020. Pmi-masking: Principled masking of correlated spans. In *International Conference on Learning Representations*.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27.

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*.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Andrew L. 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, Portland, Oregon, USA. Association for Computational Linguistics.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. 2020. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. 2018. Mesh-tensorflow: Deep learning for supercomputers. *Advances in neural information processing systems*, 31.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. 2018. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020a. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020b. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.

Xiao Wang, Shihan Dou, Limao Xiong, Yicheng Zou, Qi Zhang, Tao Gui, Liang Qiao, Zhanzhan Cheng, and Xuanjing Huang. 2022. MINER: Improving out-of-vocabulary named entity recognition from an information theoretic perspective. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5590–5600, Dublin, Ireland. Association for Computational Linguistics.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, et al. 2023. Instructuie: Multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*.

Xingcheng Yao, Yanan Zheng, Xiaocong Yang, and Zhilin Yang. 2022. Nlp from scratch without large-scale pretraining: A simple and efficient framework. In *International Conference on Machine Learning*, pages 25438–25451. PMLR.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

A Detailed Experiment Settings

Table A.1 lists the detailed hyperparameters of ISS at different scales for each task on the pre-training task. On each task, we perform a grid search for $B_p \in \{1, 2, 4, 8\}$ and Batch size$(task)$ $\in \{1,2,4,8,16\}$ and adjust the training step, batch size, and sequence length to minimize the training cost while maintaining competitive performance.

Table A.2 lists the detailed hyperparameters of ISS for each task on further pretraining task.

	Hyper-Parameters	AGNews	Hyp.	Help.	IMDB	ACL.	SciERC	Chem.	RCT
Small Scale	B_p	4	1	4	4	4	4	4	4
	B_t	16	1	16	16	2	4	8	16
	Source Corpus[1]	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$
	Training Data Size[2]	0.22GB	0.04GB	0.1GB	0.18GB	0.3GB	0.32GB	0.13GB	0.16GB
	Training Steps	5E4	2E4	1E5	1E5	1E5	1E5	1E5	5E4
	$\rho 1$	1	99	1	19	999	999	999	3
	$\rho 2$	100	20	100	100	100	20	20	20
	Batch Size	256	256	256	256	256	256	256	256
	Sequence Length	128	128	128	512	128	128	128	128
Medium Scale	B_p	4	1	4	4	4	4	4	4
	B_t	16	1	16	16	2	4	8	16
	Source Corpus[1]	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$	$\mathcal{C}_{TLM-small}$
	Training Data Size[2]	0.22GB	0.04GB	0.1GB	0.18GB	0.3GB	0.32GB	0.13GB	0.16GB
	Training Steps	1.5E5	5E4	1.5E5	1.5E5	1.5E5	1.5E5	1.5E5	1.5E5
	$\rho 1$	1	99	1	19	999	999	999	3
	$\rho 2$	100	20	100	100	100	20	20	20
	Batch Size	256	256	256	256	256	256	256	256
	Sequence Length	128	128	128	512	128	128	128	128
Large Scale	B_p	4	1	4	4	4	4	4	4
	B_t	16	1	16	16	2	4	8	16
	Source Corpus[1]	$\mathcal{C}_{TLM-large}$	$\mathcal{C}_{TLM-large}$	$\mathcal{C}_{TLM-large}$	$\mathcal{C}_{TLM-large}$	$\mathcal{C}_{TLM-large}$	$\mathcal{C}_{TLM-large}$	$\mathcal{C}_{TLM-large}$	$\mathcal{C}_{TLM-large}$
	Training Data Size[2]	0.62GB	0.18GB	0.34GB	2.20GB	0.70GB	0.84GB	0.5GB	0.44GB
	Training Steps	3E5	1E5	3E5	3E5	3E5	3E5	3E5	3E5
	$\rho 1$	3	99	1	99	999	999	999	3
	$\rho 2$	100	100	1000	100	20	20	100	100
	Batch Size	256	256	256	256	256	256	256	256
	Sequence Length	128	128	128	512	128	128	128	128

1 $\mathcal{C}_{TLM-small}$ and $\mathcal{C}_{TLM-large}$ are provided by TLM(Yao et al., 2022).
2 ISS only uses a tiny subset of the source general corpus for training. We list the data size that are actually used for ISS training.

Table A.1: Detailed hyper-parameters for ISS of different scales for each task

Hyper-Parameters	RCT	Chem.	Acl.	SciERC
B_p	16	8	2	4
B_t	4	4	4	4
Source Corpus[1]	$\mathcal{C}_{\text{S2ORC}}$	$\mathcal{C}_{\text{S2ORC}}$	$\mathcal{C}_{\text{S2ORC}}$	$\mathcal{C}_{\text{S2ORC}}$
Train Data Size[2]	1.5G	1.5G	1.9G	1.9G
Training Steps	5E4	5E4	5E4	5E4
Batch Size	256	256	256	256
Sequence Length	128	128	128	128

1 $\mathcal{C}_{\text{S2ORC}}$ is provided by S2ORC(Lo et al., 2020).
2 ISS only uses a tiny subset of the source general corpus for training. We list the data size that are actually used for ISS training.

Table A.2: Detailed hyper-parameters for ISS in further pretraining

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations section

☑ A2. Did you discuss any potential risks of your work?
Limitations section

☑ 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?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ 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)?
No response.

☐ 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?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ 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.
No response.

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?
Section 4

☑ 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.?
Section 4

☑ 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)?
Section 4

☑ 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?
Section 4

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.