NLP From Scratch Without Large-Scale Pretraining: A Simple and Efficient Framework

Xingcheng Yao [*1] **Yanan Zheng** [*2] **Xiaocong Yang** [3 4] **Zhilin Yang** [1 5 4]

Abstract

Pretrained language models have become the standard approach for many NLP tasks due to strong performance, but they are very expensive to train. We propose a simple and efficient learning framework TLM that does not rely on large-scale pretraining[1]. Given some labeled task data and a large general corpus, TLM uses task data as queries to retrieve a tiny subset of the general corpus and jointly optimizes the task objective and the language modeling objective from scratch. On eight classification datasets in four domains, TLM achieves results better than or similar to pretrained language models (e.g., RoBERTa-Large) while reducing the training FLOPs by two orders of magnitude. With high accuracy and efficiency, we hope TLM will contribute to democratizing NLP and expediting its development [2].

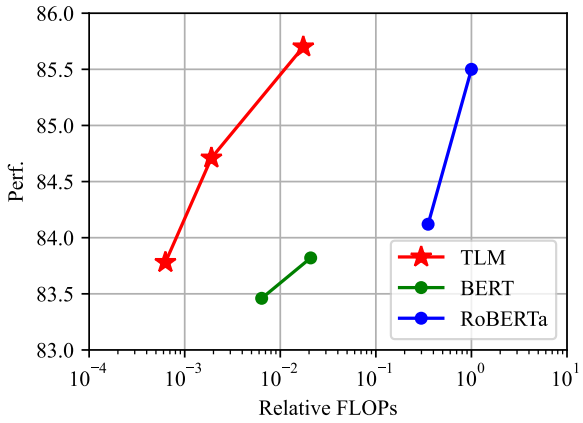


Figure 1. Average performance on eight tasks v.s. relative FLOPs w.r.t. RoBERTa-Large (Liu et al., 2019). TLM slightly outperforms RoBERTa-Large while reducing FLOPs by two orders of magnitude.

1. Introduction

Pretrained language models (PLMs) have drawn much attention from the natural language processing (NLP) community. Neural networks based on the Transformer architecture (Vaswani et al., 2017) are trained on large general corpora for self-supervised language modeling tasks such as masked language modeling (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2019), autoregressive language modeling (Radford et al., 2018; Brown et al., 2020), permutation language modeling (Yang et al., 2019), etc, and then are finetuned on a small amount of labeled data for downstream tasks. This pretraining-finetuning framework has significantly improved the performance of many NLP tasks.

However, while considered effective, large-scale pretraining is usually computationally expensive. For example, RoBERTa-Large (Liu et al., 2019), a widely-used PLM, consumes a computational cost of 4.36×10^{21} FLOPs[3]. Larger PLMs such as GPT-3 (Brown et al., 2020) consume 50 times more FLOPs for training than RoBERTa-Large. The expensiveness of large-scale pretraining prevents many research groups with limited budgets from pretraining customized language models, exploring new neural architectures, or improving pretraining loss functions. In contrast, a large number of NLP researchers resort to improving the finetuning algorithms, whose performance is largely upper-bounded by the pretraining procedure. This creates a high barrier of NLP research and might not be ideal for the long-term development of the field.

[*]Equal contribution [1]Institute for Interdisciplinary Information Sciences, Tsinghua University [2]Department of Computer Science and Technology, Tsinghua University [3]School of Economics and Management, Tsinghua University [4]Recurrent AI, Inc [5]Shanghai Qi Zhi Institute. Correspondence to: Zhilin Yang <zhiliny@tsinghua.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[1]In the broadest sense, pretraining means training on some objectives before optimizing the target tasks. In contrast, throughout the paper, we use "pretraining" to only refer to task-agnostic training of language models on a large general corpus, such as BERT (Devlin et al., 2019).

[2]Our code, model checkpoints and datasets are publicly available at: https://github.com/yaoxingcheng/TLM

[3]It was pretrained with 1,000 V100 GPUs each with 32GB memory for approximately one day.

Even though there have been efforts devoted to studying and improving the ef ciency of language model pretraining (Clark et al., 2020; So et al., 2021; Tay et al., 2021; Chen et al., 2021), most of them focus on designing sample-ef cient self-supervised tasks or discovering ef cient Transformer architectures suitable for pretraining. Their improvements are limited, with a reduction of computational costs (in terms of FLOPs) less than one order of magnitude. Another line of works target reducing the sizes of PLMs using distillation (Sanh et al., 2019; Jiao et al., 2020) to improve the ef ciency of inference, but these methods rely on pretraining a large PLM before distillation. Moreover, distilled models often do not perform as well as some of the best non-distilled PLMs such as RoBERTa-Large (Sanh et al., 2019; Jiao et al., 2020).

This work explores alternatives to the standard pretraining-netuning paradigm, aiming at more drastic ef ciency improvement without performance drop. We propose a simple, ef cient, pretraining-free framework, Task-driven Language Modeling (TLM). Given a large general corpus and some labeled task data, TLM directly trains a model from scratch without relying on PLMs. TLM is motivated by two key ideas. First, humans master a task by using only a small portion of world knowledge (e.g., students only need to review a few chapters, among all books in the world, to cram for an exam). We hypothesize that there is much redundancy in the large corpus for a speci c task. Second, training on supervised labeled data is much more data ef cient for downstream performance than optimizing the language modeling objective on unlabeled data. Based on these motivations, TLM uses the task data as queries to retrieve a tiny subset of the general corpus. This is followed by jointly optimizing a supervised task objective and a language modeling objective using both the retrieved data and the task data.

We evaluate TLM on eight different tasks covering the domains of news, review, computer science, and biomedical science, following the setting of Gururangan et al. (2020). TLM achieves results better than or similar to BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) while reducing the training FLOPs by two orders of magnitude[4].

2. Related work

Pretrained Language Models Pretrained language models have become the de-facto solution to many of the NLP tasks (Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2019; Brown et al., 2020; Yang et al., 2019). Those models are usually pretrained on a large-scale corpus in a self-supervised manner to learn a contextualized representation of tokens in natural language, and then are ne-tuned with labeled data for speci c tasks. BERT (Devlin et al., 2019), one of the most popular PLMs, is pretrained on a 16GB English corpus using a masked language modeling objective (i.e. predicting randomly masked tokens). RoBERTa (Liu et al., 2019) inherits the training objective of BERT, but is pretrained on a larger corpus consisting of 160GB English texts with larger batch size and dynamic token masking. In this work, we take both BERT and RoBERTa as our major baselines.

Ef cient Pretraining for NLP There is a line of work dedicated to improving the ef ciency of pretraining language models. You et al. (2020) and Shoeybi et al. (2019) utilized the data and model parallelism across different computational devices to accelerate the pretraining process. However, accelerating through parallelism does not actually reduce computational costs in terms of FLOPs for training models at large scale. Chen et al. (2021) and So et al. (2021) tried to identify ef cient neural network architectures for language model pretraining, based on the lottery ticket hypothesis and neural architecture search. Such modi cations on architecture can bring about 50% – 70% reduction in computational costs. Clark et al. (2020) and He et al. (2021) incorporated manually designed mechanisms into language model pretraining, such as adversarial training and disentangled representation of content and position, which brings about 50% – 75% reduction in computational costs. Gu et al. (2020) proposed to use task-guided pre-training with selective masking, which reduces the computation cost by around 50%. In this work, orthogonal to the aforementioned works, we investigate improving ef ciency by reducing training data redundancy. Our approach also results in more drastic improvements.

Ef cient Inference of Pretrained Models Another line of work aims at improving inference ef ciency of PLMs. Some works improve inference ef ciency by distilling large PLMs into small-sized models and using the distilled models for inference, such as DistilBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2020), MobileBERT (Sun et al., 2020), FastBERT (Liu et al., 2020), BORT (de Wynter & Perry, 2020), and BERT-of-Theseus (Xu et al., 2020). Other works speed up inference by quantizing PLMs with low-precision representations during inference, such as Q8-BERT (Zafrir et al., 2019), Q-BERT (Shen et al., 2020), and I-BERT (Kim et al., 2021). Another type of works, such as (Michel et al., 2019; Wang et al., 2020; Gordon et al., 2020), adopt pruning by removing parts of PLMs to make it smaller and faster. However, these methods rely on large PLMs, and the performance after distillation, pruning, or quantization often decreases to a certain extent compared with some of the best PLMs (e.g., RoBERTa-Large). In contrast, our approach doesn't rely on large-scale pre-training and achieves better or at least comparable performance.

[4] This effectively reduces the cost from training on 1,000 GPUs for one day to training on 8 GPUs for 42 hours.

Figure 2.Comparison between the traditional pretraining- netuning approach and our proposed framework TLM: instead of training a language model over the entire general corpus and then netuning it on task data, we rst use task data as queries to retrieve a tiny subset of the general corpus, and then perform joint learning on both the task objective and self-supervised language modeling objective.

Domain and Task Adaptation for Pretrained Models Domain-adaptive netuning is a method that netunes a pretrained model on in-domain data using a language modeling objective. It has been shown to be effective for domain- and task adaptation (Zhang et al., 2019; Gururangan et al., 2020; Li et al., 2020; Lee et al., 2020). There are a few crucial differences between domain-adaptive netuning and TLM. First, TLM is a general method to improve training ef ciency that does not use any additional domain data. It only utilizes the general corpus as in BERT and RoBERTa. In comparison, domain-adaptive netuning uses domain data to improve domain adaptation. Second, while previous works on domain-adaptive netuning are built upon a model pretrained on the general corpus, TLM learns from scratch without large-scale pretraining to substantially save computation costs.

Co-training for Semi-supervised Learning and Data-Density-Based Active Learning Additionally, we observe two techniques related to TLM. They are Co-Training (CT) (Qiao et al., 2018; Yang et al., 2021) and Data-Density-Based Active Learning (DAL) (Zhu et al., 2010; Wang et al., 2017) respectively. Both CT and TLM utilize unlabeled data to aid the learning on a certain task. The difference between TLM and CT is 2-fold: First, CT requires training distinct models from multiple views of unlabeled data, yet TLM only trains a single model through pre-text tasks such as MLM. Second, TLM takes the selection process of unlabeled data into account, which is little discussed in CT. TLM and DAL share the same avor of nding representative instances in a pool of unlabeled data. However, DAL makes the assumption that every unlabeled sample can be effectively labeled by the de nition of the task, which is not required by TLM. Also, DAL tries to nd critical instances iteratively from the whole pool of unlabeled data, yet TLM only tries to nd relevant instances in a one-shot way with respect to labeled data, which makes TLM more ef cient than classic DAL algorithms.

3. Method

3.1. TLM: Task-Driven Language Modeling

It is an interesting phenomenon that humans are able to quickly master a certain task with limited time and effort by focusing only on pieces of relevant knowledge. For example, when students cram for exams, they review a few chapters instead of going through all books in the world. Following this observation, we conjecture that one of the key aspects of learning a task is to quickly and precisely locate task-relevant information. To this end, we develop TLM that rst automatically retrieves relevant training data from a general corpus and then learns on the retrieved data and task data combined.

Formally, given a general corpus $\mathcal{D} = \{d_i\}_i$ where d_i is a document, and labeled task data $\mathcal{T} = \{(x_i; y_i)\}_i$ where x_i is text and $y_i \in Y$ is a label[5], our goal is to train a model f to estimate the conditional probability for classi cation $f(x) = \hat{p}(y|x)$.

TLM consists of two steps as shown in Figure 2.

1. Retrieve data from a general corpus using task data as queries.
2. Train a model from scratch by jointly optimizing the task objective and the language modeling objective on the retrieved data and task data.

Retrieval From General Corpus For each example in the task data $x_i \in \mathcal{T}$, we retrieve a set of documents

[5]While it is straightforward to extend our framework to generation tasks, we focus on classi cation tasks in this work.

$S_i = \{d_{i,1}, d_{i,2}, \ldots\}$ from the given general corpus $\mathcal{D}$. The set S_i represents the top k similar documents to x_i in $\mathcal{D}$. Retrieved data for all examples x's are combined $S = \cup_i S_i$. Retrieved data S is a tiny subset of the general corpus $\mathcal{D}$.

We use BM25 (Robertson & Zaragoza, 2009) for retrieval due to its efficiency. While using embedding-based dense retrievers (Karpukhin et al., 2020) might lead to better retrieval results, we do not consider these methods to keep our approach as simple as possible. Moreover, dense retrievers rely on pretraining, which might bring additional computational costs. The exploration of achieving a better tradeoff between efficiency and retrieval performance is left to future work. Moreover, for tasks with extremely long texts (e.g., Helpfulness (McAuley et al., 2015)), we find it more efficient to extract keywords (e.g., using the RAKE algorithm (Rose et al., 2010)) to form the queries for retrieval instead of using the entire input sequence. We call the retrieved data S external data and the task data T internal data.

Note that our data retrieval method is task-agnostic—it only depends on text x without dependency on y. Moreover, the retrieval procedure does not assume the availability of domain-specific data. It operates on a general corpus and has the same input as the pretraining-finetuning paradigm.

Joint Training Given both the internal and external data, we train a language model f from scratch. Let $L_{mlm}(x)$ be the masked language modeling loss as in BERT (Devlin et al., 2019), and let $L_{task}(f(x), y)$ be the task loss function (e.g., cross entropy for classification). TLM optimizes the following loss function:

$$\lambda_1 E_{x \sim S}[L_{mlm}(x)] + E_{x,y \sim T}[\lambda_2 L_{mlm}(x) + L_{task}(f(x), y)]$$

where λ_1 and λ_2 are hyperparameters. The network architecture we employ is identical to BERT, where we use a CLS head for classification and an LM head for masked language modeling. TLM can also be extended to other architectures for non-classification tasks. Our implementation involves a two-stage training procedure. In the first stage, we interleave one batch of internal data with n batches of external data for mini-batch stochastic gradient descent, where n is set as an integer. In the second stage, we set both λ_1 and λ_2 as zero to only finetune the model on internal data with the task objective.

3.2. Comparison Between TLM and PLMs

Both TLM and pretraining-finetuning have two stages. In fact, the second stage of TLM equals the traditional finetuning stage. The main difference between the first stage of TLM and pretraining (PLMs) is shown in Table 1. Unlike PLMs which learn as much task-agnostic knowledge as possible at an extremely high cost, TLM learns task-related knowledge for each task with very low costs.

Table 1. Comparison between TLM and PLMs. Here we provide qualitative comparison, while quantitative comparison in terms of training data size, FLOPs, and the number of parameters is available in Table 2.

	TLM	PLMs
Loss Function	L_{task} and L_{mlm}	L_{mlm}
Training Data	A tiny subset of $\mathcal{D}$ and task data T	The entire $\mathcal{D}$
Compute Cost	8 GPUs 42 hours	1,000 GPUs one day
Generality	Task-Driven	Task-Agnostic

Given the above difference between TLM and PLMs, we will discuss the pros and cons of TLM in detail.

Democratizing NLP In pretraining-finetuning paradigm, the finetuning performance is largely upper bounded by the pretrained model. However, due to the constraints of computational resources, the majority of NLP researchers cannot afford training large-scale language models and resort to studying the finetuning algorithms. Since only a small portion of researchers are working on the architectures, loss functions, and other design choices of PLMs, there is a risk that the development of the field might be slowing down. On the other hand, TLM is efficient and highly performant. As a result, TLM has the potential of democratizing NLP and expediting its development by allowing most researchers to freely explore the architectures, loss functions, algorithms, and other design choices in the neighborhood of a state-of-the-art solution.

Efficiency TLM improves over PLMs in terms of per-task FLOPs. In many cases when there are only a few target tasks, TLM is favorable. For example, a researcher might be interested in solving four textual entailment datasets, or an industrial team might want to improve a recommender system which can be viewed as one task. However, if the goal is to solve 1,000 tasks at once (e.g., building an NLP platform to serve multiple business units within a corporate), PLMs might still be preferred.

Flexibility Since TLM is task-driven, there is a larger degree of flexibility. Researchers can use custom strategies for tokenization, sequence length, data representations, hyperparameter tuning, etc, which might improve performance and/or efficiency.

Generality PLMs learn task-agnostic general representations and can be used for few-shot and zero-shot learning (Brown et al., 2020). In comparison, TLM trades generality for efficiency by learning only task-specific representations. How to further improve TLM in terms of learning more gen-

eral representations poses a challenge for future work. We believe multi-task learning might alleviate this issue given recent observations (Wei et al., 2021; Zhong et al., 2021), especially for in-domain zero-shot generalization. It might also be possible to combine pretraining with TLM, e.g., using a small PLM with TLM to match a larger PLM, to achieve a better tradeoff between generality and ef ciency.

4. Experiments

4.1. Setup

Datasets Following (Gururangan et al., 2020), we conduct experiments on eight tasks over four domains, including biomedical science, computer science, news, and reviews (two tasks in each domain). The tasks can be categorized into high-resource and low-resource tasks. High-resource tasks has more than 5K task data, including AGNews (Zhang et al., 2015), IMDB (Maas et al., 2011), RCT (Dernoncourt & Lee, 2017), and Helpfulness (McAuley et al., 2015), while low-resource tasks include ChemProt (Kringelum et al. 2016), ACL-ARC (Jurgens et al., 2018), SciERC (Luan et al., 2018), and HyperPartisan (Kiesel et al., 2019). For the general training corpus, we collected two corpora that respectively match the original training corpora of BERT and RoBERTa. We name them respectively Corpus-BERT (C_{BERT}) and Corpus-RoBERTa ($C_{RoBERTa}$). The size of $C_{RoBERTa}$ is 10 times larger than C_{BERT}.

Baselines Our experiments focus on comparison with general PLMs. We netuned both BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) of base and large scales as the baselines. Although TLM is a general method without using addition in-domain data, it even performs close to domain-adaptive netuning methods (Gururangan et al. 2020) (see Appendix A for detailed comparison).

Evaluation Strategy We report the average performance across three random seeds, together with the standard deviation. We follow Beltagy et al. (2019) and Gururangan et al. (2020) to report the test micro-F1 for ChemProt and RCT and macro-F1 for the rest of the datasets.

For fair comparison, we evaluate TLM of different training scales. The training scale is de ned by three factors, including the number of parameters, the size of the general corpus, and the number of total training tokens. The number of total training tokens is calculated as the product of training steps, batch size, and sequence length. We report TLM at three training scales as shown in Table B.1, namely *small*, *medium*, and *large* scales. Each scale of TLM is accordingly compared to the PLM baselines with an increasing computational cost.

Training Details For each experiment of TLM, while fixing the training scale hyper-parameters (i.e., training steps, batch size and sequence length), we perform a grid search over $_1$ and $_2$. We listed the hyper-parameters used in Table B.1 in Appendix.

4.2. Main Results

Table 2 shows the main results that compare TLM of three different scales and the according PLM baselines. In conclusion, TLM can achieve results that are better than or comparable to the baselines with substantial reduction in FLOPs and the size of training data. Speci cally, at a small scale, TLM achieves comparable results to BERT-Large with an average of 1/33 of FLOPs and 1/16 of the training corpus. At the medium and large scales, TLM improves the performance by 0.59 and 0.24 points on average respectively, while signi cantly reducing both FLOPs and the training data size by two orders of magnitude or more. These results con rm that TLM is highly accurate and much more ef cient than PLMs. Moreover, TLM gains more advantages in ef ciency at a larger scale. This indicates that larger-scale PLMs might have been trained to store more general knowledge that is not useful for a speci c task.

4.3. Ablation Study

4.3.1. Data Retrieval

Table 3 shows the comparison between different retrieval methods (i.e., BM25 and random retrieval) and different sizes of the general corpus. We nd that given the same general corpus, the results of BM25 signi cantly outperform those of random retrieval by a large margin on all tasks, showing that using task-relevant data for joint training is crucial for the best performance. Speci cally, BM25 shows an advantage of almost 1 point against random retrieval on high-resource tasks such as IMDB, and more signi cant advantages on low-resource tasks such as SciERC and ChemProt by around 3-4 points. This is aligned with our intuition that low-resource tasks rely more on external data.

By comparing the results of C_{BERT} and $C_{RoBERTa}$ with BM25, we observe that increasing the size of the general corpus improves performance (by 0.5, 1.34, and 1.35 points on IMDB, SciREC, and ChemProt respectively). The gains of using 10 times more data are similar to the ones observed in PLMs (Yang et al., 2019; Liu et al., 2019). This indicates that although TLM only uses a small amount of data, it is able to scale when a larger general corpus is available while maintaining ef ciency. On the other hand, the gains of using a larger corpus diminish with random retrieval, showing that random retrieval, as a task-agnostic method, is not very sensitive to the general corpus size.

Data retrieval selects the top- K -similar documents from

Table 2.Evaluation results for TLM 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 list the number of parameters, the total training compute (FLOPs), and the size of training corpus for comparison.

Model	#Param	FLOPs[1]	Data[2]	AGNews	Hyp.	Help.	IMDB	ACL.	SciERC	Chem.	RCT	Avg.
BERT-Base[3]	109M	2.79E19	16GB	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[3]	355M	9.07E19	16GB	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-scale)	109M	2.74E18	0.91GB	93.74 0.20	93.53 1.61	70.54 0.39	93.08 0.17	69.84 3.69	80.51 1.53	81.99 0.42	86.99 0.03	83.78
RoBERTa-Base[3]	125M	1.54E21	160GB	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-scale)	109M	8.30E18	1.21GB	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
RoBERTa-Large[3]	355M	4.36E21	160GB	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-scale)	355M	7.59E19	3.64GB	94.34 0.12	95.16 0.00	72.49 0.33	95.77 0.24	72.19 1.72	83.29 0.95	85.12 0.85	87.50 0.12	85.74

[1] The total training compute (FLOPs) is calculated by $6 \times \text{Total\_Training\_Tokens} \times \text{Parameter\_Size}$ as in (Brown et al., 2020). For TLM, FLOPs are reported as the averaged result over eight tasks.
[2] The size of data selected from general corpus that are actually used in training. For TLM, it is reported by averaging over eight tasks.
[3] The BERT-Base and BERT-Large are pretrained by (Devlin et al., 2019) and RoBERTa-Base and RoBERTa-Large are pretrained by (Liu et al., 2019). We netuned them to obtain the results over the eight tasks.

Table 3.Results on the development set using different retrieval methods and different general corpora on each task. We compared two data retrieval methods: random retrieval and the BM25 algorithm. We compare two source general corpora: the corpus used in BERT (C_{BERT}) and the corpus used in RoBERTa ($C_{RoBERTa}$). The size of $C_{RoBERTa}$ is 10 times larger than C_{BERT}.

	IMDB	SciERC	ChemProt
Random			
w/ C_{BERT}	93.65 0.09	83.80 0.62	80.65 0.48
w/ $C_{RoBERTa}$	94.04 0.22	83.10 1.54	80.73 0.46
BM25			
w/ C_{BERT}	94.40 0.09	86.07 0.48	83.64 0.26
w/ $C_{RoBERTa}$	94.90 0.06	87.41 0.36	84.99 0.72

Table 4.Results on the development set with different values of K. The value K is the number of retrieved documents per task example. AGNews is a high-resource task, while SciREC and ChemProt are low-resource ones. Here we use $= 20$ for all tasks. When there are external data available, we use $= 4$ for AGNews and $_1 = 1000$ for SciERC and ChemProt.

	AGNews	SciERC	ChemProt
Only Task Data	93.41 0.10	51.23 1.13	55.05 0.18
Top-50	94.51 0.15	77.61 1.75	77.21 0.47
Top-500	94.32 0.05	82.39 0.55	81.44 0.50
Top-5000	94.42 0.10	86.07 0.48	83.64 0.26

the general corpus. Table 4 shows the results of different K values. We observe that high-resource tasks such as AGNews only need a small K value, while low-resource tasks such as SciREC and ChemProt require a large K to obtain the best performance. The observation is consistent with the above analysis that low-resource tasks rely more on external data to improve from joint training.

4.3.2. Language Modeling Weights $_1$ and $_2$

The hyperparameters $_1$ and $_2$ are the weights for the LM loss on external and internal data respectively. We conduct sensitivity analysis over $_1$ and $_2$. Results are shown in Table 5 and Table 6.

For $_1$, we nd that high-resource tasks such as Helpfulness perform better with a smaller $_1$ (i.e., Helpfulness achieves best when $_1 = 1$) while low-resource tasks such as SciERC and ChemProt achieve their best when $_1$ is large (i.e., both tasks use $_1 = 999$). This is in line with conclusions in Section 4.3.1 that low-resource tasks rely more on external data. In addition, removing task data and only using external data for training (i.e., $_1 = \#C_{BERT}$), it performs worse than when incorporating the task data, proving the indispensability of small task data.

Results in Table 6 show that language modeling on internal data is necessary: consistently better results are achieved when $_2$ is non-zero. Based on our observations, competitive performance can be achieved when $_2$ is set to a proper value between 20 and 1000.

(a) TLM (Medium scale) (b) BERT-Base (c) RoBERTa-Base

Figure 3.Attention visualization of TLM and pretraining- netuning baselines, with "[CLS] crystallographic comparison with the structurally related. [SEP]" from ChemProt as the input. The positional heads (Voita et al., 2019) are highlighted in red boxes and vertical heads (Kovaleva et al., 2019) are masked in gray.

Table 5.Results on the development set with different weights on external data (i.e., $_1$). We assign different values for $_1$ for the rst stage, and report the nal performance after two-stage joint learning. "Ext only" means using only external data for training (i.e., $_1 = 1$). Helpfulness is a high-resource task, and the others are low-resource ones. For all tasks, we $_2 = 20$.

	Helpfulness	SciERC	ChemProt
$_1$ =1	71.02 0.51	80.72 3.32	73.27 0.30
$_1$ =3	70.41 0.52	80.01 0.72	79.43 1.03
$_1$ =99	69.56 0.23	84.95 0.57	83.30 0.30
$_1$ =999	69.35 0.72	86.07 0.48	83.64 0.26
Ext only	69.76 0.50	85.66 1.58	82.50 0.27

Table 6.Results on the development set with different language modeling weights on internal data (i.e $_2$). Here we set $_1 = 1000$ for SciERC and ChemProt, and $_1 = 4$ for RCT

	RCT	SciERC	ChemProt
$_2$ =0	85.75 0.11	83.31 0.88	83.41 0.33
$_2$ =20	88.08 0.02	86.07 0.48	83.64 0.26
$_2$ =100	88.16 0.15	85.48 1.01	83.77 0.77
$_2$ =1000	88.02 0.04	85.29 1.86	83.63 0.90

4.3.3. Second Stage of Training

TLM contains two training stages— rst training on all three terms combined and then netuning using only the task objective. To validate the effectiveness of the second stage of TLM, we compare the performance of two-stage training against using only stage one. Results are shown in Table 7. We nd that removing the second stage hurts the ultimate performance consistently, proving its indispensability. Particularly, the second stage has much more in uence on low-resource tasks (with a huge decrease of 19.37 points on ACL-ARC and 14.34 points on ChemProt) than on high-resource tasks (with a performance decrease of 0.53 points on AGNews and 2.17 points on IMDB).

Table 7. Results on the development set of two-stage training and one-stage training (removing stage 2).

	AGNews	IMDB	ChemProt	ACL-ARC
two-stage	94.51	94.40	83.64	76.37
wo/ stage-2	93.98#	92.23#	69.30#	57.00#

Table 8.Results of adding MLM loss on task data into PLM. Results are based on RoBERTa-base.

Model	AGNews	Hyp.	Help.	IMDB	ACL.	SciERC	Chem.	RCT	Avg.
PLM	94.02	93.53	70.45	95.43	68.34	81.35	82.60	87.23	84.12
PLM+MLM	93.83	93.50	71.12	95.54	70.94	80.90	82.53	87.09	84.43
TLM	93.96	94.05	70.90	93.97	72.37	81.88	83.24	87.28	84.71

4.3.4. MLM Loss on Task Data

During the rst training stage, TLM uses masked language loss on task data. To examine whether the trick attains the main improvements, we compare results on PLM, PLM with additional MLM loss on task data (PLM+MLM) and TLM. Results in Table 8 show that adding MLM loss on task data into PLM has only marginal gains and does not affect the main conclusion of the paper. In addition, results in Table 3 and Table 4 show that retrieving appropriate relevant data is also essential for the performance of TLM.

4.4. Analysis

4.4.1. Attention Weight Visualization

We also study the difference between the model behaviors of TLM and pretraining- netuning by visualizing their attention weights. Voita et al. (2019) found that a speci c kind of heads, referred to as "positional head" in which at least 90% of the maximum attention weights are assigned to adjacent tokens, have vital contributions to nal predictions of the model. Another sort of heads we are interested in are those in which most maximum attention weights are assigned to [CLS],[SEP] or the period token("."), which potentially encode less semantic or syntactic information (Kovaleva et al., 2019). In our experiments, if more than 90% maximum weights are assigned to [CLS], [SEP] or the period token, we categorize this head as a "vertical head". Results in Figure 3 show that on the task ChemProt, more

Table 9. Examples of retrieved data. The overlap between queries and retrieved data are highlighted in blue in italics.

Task	Task Data as Query	Retrieved General Data
Hyp.	"A Republican student association at San Diego State University (SDSU) is facing backlash for sending a letter demanding Muslim students condemn last week's terror attacks in Barcelona. ... "	Example 1: "...The SDSU Aztecs intercollegiate water polo, swimming and diving teams are based at the Aztec Aquaplex..." Example 2: The Daily Aztec is a not-for-pro t, independent student newspaper serving San Diego State University (SDSU) and the surrounding College Area in San Diego, California. ...
Help.	Poor Quality. The case broke after dropping it on the tile oor. ...	Example 1: ...a collaborative algorithm will be able to recommend it, the quality of those recommendations will be poor. ... Example 2: ... Books that're of poor quality will quickly cease to sell. ...
ChemProt	FCEO signi cantly inhibited nitric oxide (NO) and prostaglandin E2 (PGE2) by suppressing the protein expression of inducible nitric oxide synthase (iNOS) and cyclooxygenase (COX)-2, respectively.	Example 1: ... They regulate the development of sperm by controlling their cell division and survival. Other immune factors found in the testis include the enzyme inducible nitric oxide synthase (iNOS) ... Example 2: These compounds have been shown "in vivo" to reduce two proteins that mediate in ammation: cyclooxygenase-2 (COX-2) and inducible nitric oxide synthase (iNOS).
SciERC	Image sequence processing techniques are used to study exchange , growth , and transport processes and to tackle key questions in environmental physics and biology.	Example 1: ... Driving forces in signal processing for data parallelism are video encoding, image and graphics processing, wireless communications to name a few. Example 2: They have applications in many disciplines such as biology, chemistry, ecology, neuroscience, physics, image processing ...

Table 10. Evaluation results on the GLUE benchmark. Model size, data, and FLOPs are similar to Table 2.

Method	CoLA	RTE	STS-B	MRPC	QQP	SST-2	QNLI	MNLI	Avg.
BERT-Base	59.3	68.2	89.8/89.4	86.0/90.5	91.1/88.1	92.5	91.8	84.5/84.5	82.97
TLM (small-scale)	59.8	67.1	89.0/88.7	86.8/90.4	91.1/88.1	92.2	91.0	83.3/83.9	82.60

positional heads and less vertical heads are observed in TLM than in PLMs. We also observe similar patterns across various tasks (see Appendix C). These phenomena suggest that TLM learns different (probably more informative) attention patterns compared to PLMs.

4.4.2. Case Study of Retrieved Data

We have shown several casess of retrieved data in Table 9. TLM retrieves relevant data from a general corpus using BM25 (Robertson & Zaragoza, 2009). Since BM25 is based on sparse features, it focuses more on lexical similarity instead of semantic similarity. This might be speci cally bene cial for professional domains, e.g., SciERC for computer science and ChemProt for biomedical science), since there are a large number of proper nouns in these domains. For other domains, it seems BM25 also performs reasonably well for retrieving related documents.

4.5. Results on More Datasets

So far we have followed the setting of Gururangan et al. (2020) and adopted the datasets therein. In this section, we additionally experiment with the GLUE benchmark (Wang et al., 2018) following the setting of BERT (Devlin et al., 2019) to examine the performance of TLM on a more diverse set of tasks including natural language understanding. We follow the small-scale setting in Section 4.2 in terms of model size, data, and FLOPs. Results in Table 10 show that given the advantages in ef ciency, the average performance of TLM is comparable to BERT across 8 tasks, which is consistent with our previous ndings and demonstrates the effectiveness of TLM.

5. Conclusions

In this paper, we have proposed a simple, ef cient, pretraining-free framework, TLM. The core idea is to only use a tiny, task-relevant subset of the general corpus for language model training. Our experiments show that TLM achieves results similar to or even better than PLMs, with a reduction of training FLOPs by two orders of magnitude. TLM opens the possibility of reducing the heavy reliance on large-scale PLMs and training a model from scratch in an ef cient manner, while not hurting the overall performance. We hope TLM will contribute to democratizing NLP and

expediting its development by allowing most researchers to freely explore the architectures, loss functions, algorithms, and other design choices in the neighborhood of a state-of-the-art solution.

As discussed in Section 3.2, there are several potential directions for future work. It will be interesting to study how to use TLM to match the performance even larger-scale PLMs. Moreover, further extending and improving TLM for few-shot and zero-shot learning is a crucial problem.

References

Beltagy, I., Lo, K., and Cohan, A. SciBERT: A pretrained language model for scienti c 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)*, pp. 3613–3618, Hongkong, China, 2019. Association for Computational Linguistics.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020.

Chen, X., Cheng, Y., Wang, S., Gan, Z., Wang, Z., and Liu, J. EarlyBERT: Ef cient BERT training via early-bird lottery tickets. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

de Wynter, A. and Perry, D. J. Optimal subarchitecture extraction for BERT. *CoRR*, abs/2010.10499, 2020.

Dernoncourt, F. and Lee, J. Y. Pubmed 200k RCT: a dataset for sequential sentence classi cation in medical abstracts. In *IJCNLP(2)*, pp. 308–313. Asian Federation of Natural Language Processing, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

Gordon, M., Duh, K., and Andrews, N. Compressing bert: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pp. 143–155. Association for Computational Linguistics, 2020.

Gu, Y., Zhang, Z., Wang, X., Liu, Z., and Sun, M. Train no evil: Selective masking for task-guided pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6966–6974, Online, November 2020. Association for Computational Linguistics.

Gururangan, S., Marasovic, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*, pp. 8342–8360. Association for Computational Linguistics, 2020.

He, P., Liu, X., Gao, J., and Chen, W. DeBERTa: Decoding-enhanced bert with disentangled attention. In *2021 International Conference on Learning Representations*, May 2021.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, Online, November 2020. Association for Computational Linguistics.

Jurgens, D., Kumar, S., Hoover, R., McFarland, D. A., and Jurafsky, D. Measuring the evolution of a scienti c eld through citation frames. *Trans. Assoc. Comput. Linguistics*, 6:391–406, 2018.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics.

Kiesel, J., Mestre, M., Shukla, R., Vincent, E., Adineh, P., Corney, D. P. A., Stein, B., and Potthast, M. Semeval-2019 task 4: Hyperpartisan news detection. In *SemEval@NAACL-HLT*, pp. 829–839. Association for Computational Linguistics, 2019.

Kim, S., Gholami, A., Yao, Z., Mahoney, M. W., and Keutzer, K. I-BERT: integer-only BERT quantization. In *International Conference on Machine Learning*, 2021.

Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. Revealing the dark secrets of BERT. 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), pp. 4365–4374, Hongkong, China, 2019. Association for Computational Linguistics.

Kringelum, J., Kjærulff, S. K., Brunak, S., Lund, O., Oprea, T. I., and Taboureau, O. Chemprot-3.0: a global chemical biology diseases mapping. Database J. Biol. Databases Curation, 2016, 2016.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics, 36(4):1234–1240, 2020.

Li, J., Zhang, Z., Zhao, H., Zhou, X., and Zhou, X. Task-speci c objectives of pre-trained language models for dialogue adaptation. arXiv preprint arXiv:2009.04984, 2020.

Liu, W., Zhou, P., Wang, Z., Zhao, Z., Deng, H., and Ju, Q. FastBERT: a self-distilling BERT with adaptive inference time. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 6035–6044, Online, July 2020. Association for Computational Linguistics.

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

Luan, Y., He, L., Ostendorf, M., and Hajishirzi, H. Multi-task identi cation of entities, relations, and coreference for scienti c knowledge graph construction. In EMNLP, pp. 3219–3232. Association for Computational Linguistics, 2018.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In ACL, pp. 142–150. The Association for Computer Linguistics, 2011.

McAuley, J. J., Targett, C., Shi, Q., and van den Hengel, A. Image-based recommendations on styles and substitutes. In SIGIR, pp. 43–52. ACM, 2015.

Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one? In Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.

Qiao, S., Shen, W., Zhang, Z., Wang, B., and Yuille, A. Deep co-training for semi-supervised image recognition. Lecture Notes in Computer Science, pp. 142–159, 2018. ISSN 1611-3349. doi: 10.1007/978-3-030-01267-0_9. URL http://dx.doi.org/10.1007/978-3-030-01267-0_9.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2018. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

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

Robertson, S. E. and Zaragoza, H. The probabilistic relevance framework: BM25 and beyond. Found. Trends Inf. Retr., 3(4):333–389, 2009.

Rose, S., Engel, D., Cramer, N., and Cowley, W. Automatic keyword extraction from individual documents. Text mining: applications and theory, 1:1–20, 2010.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019.

Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. Q-BERT: Hessian based ultra low precision quantization of BERT. Proceedings of the AAAI Conference on Arti cial Intelligence, 34(05): 8815–8821, Apr. 2020.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-LM: training multi-billion parameter language models using model parallelism, 2019.

So, D. R., Mańke, W., Liu, H., Dai, Z., Shazeer, N., and Le, Q. V. Primer: Searching for ef cient transformers for language modeling, 2021.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 2158–2170, Online, July 2020. Association for Computational Linguistics.

Tay, Y., Dehghani, M., Rao, J., Fedus, W., Abnar, S., Chung, H. W., Narang, S., Yogatama, D., Vaswani, A., and Metzler, D. Scale ef ciently: Insights from pre-training and ne-tuning transformers, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. Analyzing multi-head self-attention: Specialized heads

do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

Wang, M., Min, F., Zhang, Z.-H., and Wu, Y.-X. Active learning through density clustering. *Expert Systems with Applications*, 85:305–317, 2017. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2017.05.046. URL https://www.sciencedirect.com/science/article/pii/S095741741730369X.

Wang, Z., Wohlwend, J., and Lei, T. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6151–6162, Online, November 2020. Association for Computational Linguistics.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Xu, C., Zhou, W., Ge, T., Wei, F., and Zhou, M. BERT-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7859–7869, Online, November 2020. Association for Computational Linguistics.

Yang, L., Wang, Y., Gao, M., Shrivastava, A., Weinberger, K. Q., Chao, W.-L., and Lim, S.-N. Deep co-training with task decomposition for semi-supervised domain adaptation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2021. doi: 10.1109/iccv48922.2021.00878. URL http://dx.doi.org/10.1109/iccv48922.2021.00878.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

You, Y., Li, J., Reddi, S. J., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations (ICLR 2020)*, Addis Ababa, Ethiopia, 2020. OpenReview.net.

Zafrir, O., Boudoukh, G., Izsak, P., and Wasserblat, M. Q8BERT: quantized 8bit BERT. *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, Dec 2019.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, pp. 649–657. Curran Associates, Inc., 2015.

Zhang, X., Shapiro, P., Kumar, G., McNamee, P., Carpuat, M., and Duh, K. Curriculum learning for domain adaptation in neural machine translation. *Proceedings of the 2019 Conference of the North*, 2019.

Zhong, R., Lee, K., Zhang, Z., and Klein, D. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2856–2878, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

Zhu, J., Wang, H., Tsou, B. K., and Ma, M. Active learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1323–1331, 2010. doi: 10.1109/TASL.2009.2033421.

A. Comparison to Domain Adaptation

Our work is different from domain adaptation such as Gururangan et al. (2020). While domain adaptation aims to address how to effectively adapt a pretrained LM into one domain-speci c task with suf cient domain data, this work targets to provide a method that is general enough to solve any task without domain data. Nevertheless, we still compare TLM with (Gururangan et al., 2020) as Table A.2 shows. We hope to gure out that, under the harsh but practical condition that no domain data is accessible, whether our proposed framework TLM can still match or even outperform the traditional domain adaptation methods with large pretrained language models as well as domain data.

From results in Table A.2, we have observations:

1. We reproduced the RoBERTa-Base results using the hyper-parameters reported by Gururangan et al. (2020) as well as our own hyper-parameters. Results show that the baseline RoBERTa-Base results are underestimated in the paper with a gap of around 3 points. We list our hyper-parameters for ne-tuning RoBERTa in Table A.1.

2. We also reproduced the DAPT+TAPT results using our own hyper-paraemters. Results show that DAPT+TAPT with new hyper-parameters also performs slightly better than it was reported by Gururangan et al. (2020).

3. From the perspective of total training computes (FLOPs), DAPT+TAPT consumes a comparable FLOPs with TLM (large-scale), and TLM (large-scale) achieved comparable results with DAPT+TAPT (i.e., 85.70 vs 85.57). However, from the perspective of data usage, DAPT+TAPT uses large amounts of domain data, the amount of which for each domain almost equals the amount of BERT total training corpus. TLM does not rely on it.

Table A.1.Comparison between the hyperparameters for ne-tuning from our implementation and from Gururangan et al. (2020).

Hyper-parameters	Ours	Reported
Epochs	-	3 or 10
Training steps	3e4	-
Patience	-	3
Learning rate	2e-5	2e-5
Batch size	32	16
Max. grad. norm	-	1
Weight decay	0	0.1

Table A.2.Comparison results of TLM and Gururangan et al. (2020).

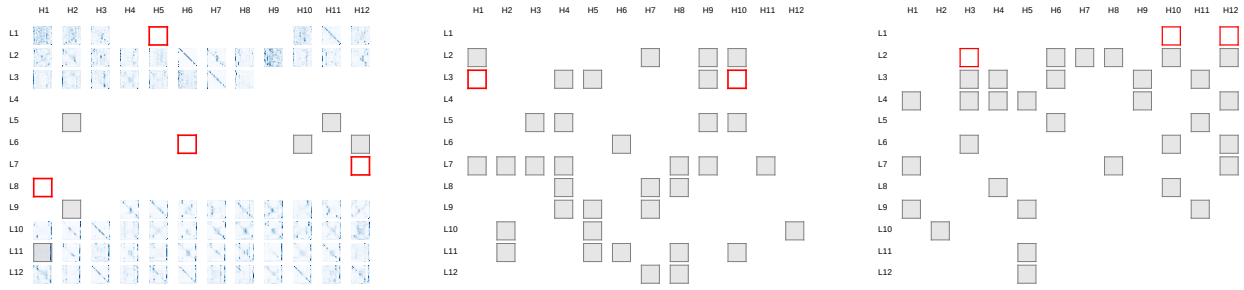
	AGNews	Hyp.	Help.	IMDB	ACL.	SciERC	Chem.	RCT	Avg.
RoBERTa-Base[1]	93.90 0.20	86.60 0.90	65.10 3.40	95.00 0.20	63.00 5.80	77.30 1.90	81.90 1.00	87.20 0.10	81.25
RoBERTa-Base[2]	93.97 0.13	88.50 4.18	67.45 0.49	95.43 0.07	63.87 1.24	79.97 1.29	81.50 0.94	87.26 0.08	82.24
RoBERTa-Base[3]	94.02 0.15	93.53 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
DAPT[1]	93.90 0.20	88.20 5.90	66.50 1.40	95.40 0.10	75.40 2.50	80.80 1.50	84.20 0.20	87.60 0.10	84.00
DAPT+TAPT[1]	94.60 0.10	90.00 6.60	68.70 1.80	95.60 0.10	75.60 3.80	81.30 1.80	84.40 0.40	87.80 0.10	84.75
DAPT+TAPT[3]	94.07 0.07	93.59 0.00	71.44 0.99	95.65 0.14	75.62 1.77	82.06 0.90	84.45 0.68	87.67 0.11	85.57
TLM (large-scale)	94.32 0.07	95.16 0.00	72.49 0.33	95.77 0.24	72.19 1.72	83.29 0.95	85.12 0.85	87.50 0.12	85.74

[1] Results reported by Gururangan et al. (2020)
[2] Our reproduced results with the hyper-parameters reported by Gururangan et al. (2020)
[3] Results obtained by our own hyper-parameters

Table B.1.Detailed hyper-parameters for TLM of different scales for each task.

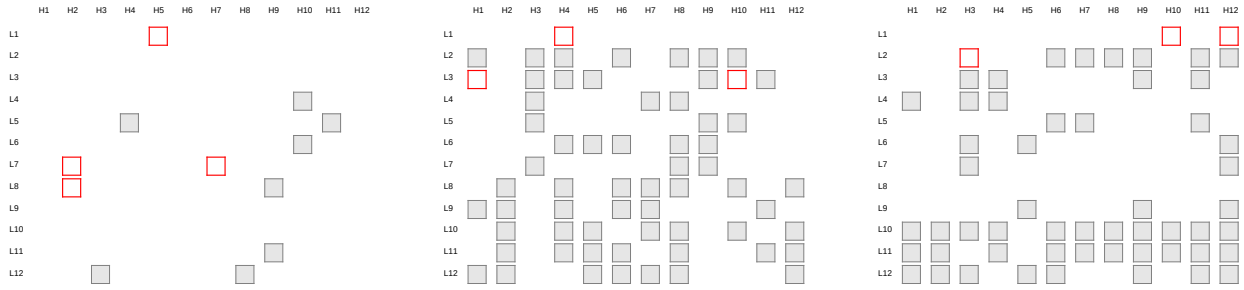
	Hyper-Parameters	AGNews	Hyp.	Help.	IMDB	ACL.	SciERC	Chem.	RCT
Small Scale	Top-K	50	5000	50	500	5000	5000	5000	50
	[1]	1	99	1	19	999	999	999	3
	[2]	100	20	100	100	100	20	20	20
	Source Corpus[2]	C_{BERT}	C_{BERT}	C_{BERT}	C_{BERT}	C_{BERT}	C_{BERT}	C_{BERT}	C_{BERT}
	Training Data Size[3]	1.1GB	0.2GB	0.5GB	0.9GB	1.5GB	1.6GB	0.7GB	0.8GB
	Training Steps	1E5	5E4	1.5E5	1.5E5	1.5E5	1.5E5	1.5E5	1E5
	Batch Size	256	256	256	256	256	256	256	256
	Sequence Length	128	128	128	128[1]	128	128	128	128
Medium Scale	Top-K	50	5000	50	500	5000	5000	5000	50
	[1]	3	99	1	99	999	999	999	3
	[2]	100	100	1000	100	20	20	100	100
	Source Corpus[2]	C_{BERT}	C_{BERT}	C_{BERT}	C_{BERT}	C_{BERT}	C_{BERT}	C_{BERT}	C_{BERT}
	Training Data Size[3]	1.1GB	0.2GB	0.5GB	3.3GB	1.5GB	1.6GB	0.7GB	0.8GB
	Training Steps	3E5	1E5	3E5	3E5	3E5	3E5	3E5	3E5
	Batch Size	256	256	256	256	256	256	256	256
	Sequence Length	128	128	128	512	128	128	128	128
Large Scale	Top-K	100	10000	100	1000	10000	10000	10000	100
	[1]	3	499	7	99	1999	1999	1999	7
	[2]	100	20	100	1000	20	20	20	100
	Source Corpus[2]	$C_{RoBERTa}$	$C_{RoBERTa}$	$C_{RoBERTa}$	$C_{RoBERTa}$	$C_{RoBERTa}$	$C_{RoBERTa}$	$C_{RoBERTa}$	$C_{RoBERTa}$
	Training Data Size[3]	3.1GB	0.9GB	1.7GB	11GB	3.5GB	4.2GB	2.5GB	2.2GB
	Training Steps	5E5	3E5	5E5	5E5	5E5	3E5	5E5	5E5
	Batch Size	256	512	512	512	512	512	256	256
	Sequence Length	128	128	128	512	128	128	128	128

[1] At a small scale on IMDB, we use a sequence length of 512 for internal data and a sequence length of 128 for external data.
[2] C_{BERT} and $C_{RoBERTa}$ are our collected corpus that respectively match the original training corpus of BERT and RoBERTa.
[3] TLM only uses a tiny subset of the source general corpus for training. We list the data size that are actually used for TLM training.



(a) TLM (*Medium scale*) (b) BERT-Base (c) RoBERTa-Base

Figure C.1. task: RCT ; input: "[CLS] twenty-eight individuals from outpatient physiotherapy departments were randomized. [SEP]"



(a) TLM (b) BERT-Base (c) RoBERTa-Base

Figure C.2. task: SciERC ; input: "[CLS] multi-view constraints associated with groups of patches are combined. [SEP]"

B. Detailed Experiment Settings

Table B.1 lists the detailed hyperparameters for TLM at stage 1 of different scales for each task. At small and medium scales, for tasks with less than 5K training examples (HyperPartisan, ChemProt, SciERC, ACL-ARC), we set $K = 5000$; for tasks with more than 100K training examples (RCT, AGNews, Helpfulness), we set $K = 50$, for the rest of the tasks (IMDB), we set $K = 500$. At the large scale, K is doubled for each task. At each scale on every task, we conduct grid search for $\rho_1 \in \{1, 3, 7, 19, 99, 499, 999, 1999\}$ and $\rho_2 \in \{20, 100, 1000\}$, and adjust training steps, batch size and sequence length to minimize the training cost while preserving competitive performance. We observe that for almost all the tasks, the larger the training scale, the more reliance on external data, indicated by the increasing trend of ρ_1 and ρ_2 as the total training tokens goes up.

C. Attention visualization on other tasks

Besides ChemProt (Figure 3), we also experimented on RCT (Figure C.1) and SciERC (Figure C.2) to get attention visualizations. We find TLM consistently contains more positional heads (in red box) and less vertical heads (in gray mask). These results reveal that the aforementioned pattern generally holds for TLM.