Snapshot-Guided Domain Adaptation for ELECTRA

Daixuan Cheng[*], **Shaohan Huang, Jianfeng Liu**
Yuefeng Zhan, Hao Sun, Furu Wei, Denvy Deng, Qi Zhang
Microsoft Corporation
daixuancheng6@gmail.com
{shaohanh, jianfengliu, yuefzh, hasun, fuwei, dedeng, qizhang}@microsoft.com

Abstract

Discriminative pre-trained language models, such as ELECTRA, have achieved promising performances in a variety of general tasks. However, these generic pre-trained models struggle to capture domain-specific knowledge of domain-related tasks. In this work, we propose a novel domain-adaptation method for ELECTRA, which can dynamically select domain-specific tokens and guide the discriminator to emphasize them, without introducing new training parameters. We show that by re-weighting the losses of domain-specific tokens, ELECTRA can be effectively adapted to different domains. The experimental results in both computer science and biomedical domains show that the proposed method can achieve state-of-the-art results on the domain-related tasks.

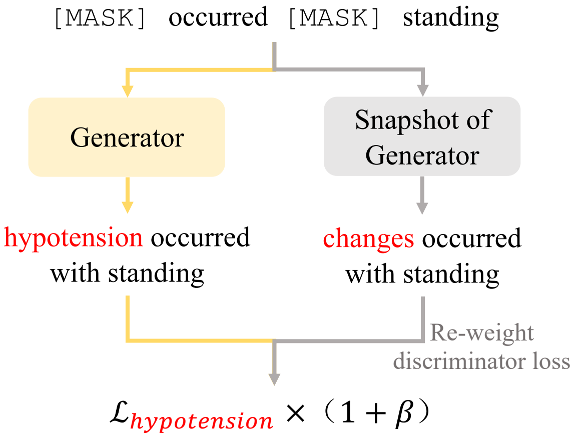
[MASK] occurred [MASK] standing
Generator
Snapshot of Generator
hypotension occurred with standing
changes occurred with standing
Re-weight discriminator loss
$\mathcal{L}_{hypotension} \times (1+\beta)$

Figure 1: Workflow of the snapshot-guided method, where $\mathcal{L}_{\text{hypotension}}$ is the discriminator loss of the token "hypotension" and β is the augmented loss weight.

1 Introduction

Pre-trained language models (Devlin et al., 2019; Clark et al., 2020) have demonstrated significant capabilities in various NLP tasks. While most language models follow the BERT-style (Devlin et al., 2019) to predict original tokens of the masked positions, ELECTRA (Clark et al., 2020) trains a discriminator to predict whether each token in a corrupted input is replaced by a generator. BERT mainly learns from the masked subset of input tokens, but ELECTRA could predict all input tokens, significantly improving sample efficiency and leading to strong results on general tasks (Chi et al., 2022; He et al., 2021; Meng et al., 2021; Xu et al., 2020; Meng et al., 2022; Bajaj et al., 2022; Shen et al., 2021). Domain adaptation of BERT-style models has been proved to consistently improve on the domain-related tasks (Gururangan et al., 2020; Lee et al., 2020; Yao et al., 2021). However, the adaptation of ELECTRA is still under-explored. This leads us to investigate domain adaptation of ELECTRA, so as to optimize its performance on the domain-related tasks.

Recent research suggests that domain-specific tokens and texts can benefit the pre-trained models in certain domains and tasks (Gururangan et al., 2020; Lee et al., 2020; Gu et al., 2020). Specifically, Gu et al. (2020) proposed a task-specific method to selectively mask domain-specific tokens in pre-training. However, the pre-trained model for one task could be a deterrent for the other tasks in the same domain. On the other hand, task-agnostic methods (Gururangan et al., 2020; Lee et al., 2020; Miolo et al., 2021) are more widely applicable—the model trained once on the domain can be used for multiple downstream tasks in the domain.

In this paper, we propose **S**napsh**O**t-guided **D**omain **A**daptation (SODA) for ELECTRA, which is also agnostic to downstream tasks. SODA leverages the difference between the generator at the current training step and the one at an earlier step to imitate the domain shift during adaption, which is then employed to dynamically find the domain-specific tokens. During continued

[*] Contribution during internship at Microsoft

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 2226–2232
December 7-11, 2022 ©2022 Association for Computational Linguistics

pre-training on the domain, the ELECTRA generator of an earlier training step is named as the *snapshot*. As shown in Figure 1, given the masked input, SODA finds the domain-specific token "hypotension" by comparing the generator to the snapshot, and then emphasizes the domain-specific token by re-weighting the discriminator loss. In model implementation, the snapshot is loaded from a saved checkpoint of an earlier step, thus no additional training parameters are introduced. Furthermore, SODA employs different snapshots during different training intervals, to dynamically select the tokens specific to the domain shift at hand (van der Wees et al., 2017).

We conduct experiments in both computer science and biomedical domains, SODA achieves state-of-the-art results on the domain-related tasks. We also evaluate different methods to select domain-specific tokens, to demonstrate the effectiveness of our method.

In summary, our contributions include:

- To the best of our knowledge, we are the first to explore domain adaptation for ELECTRA.
- We propose a snapshot-guided domain adaptation method to dynamically emphasize domain-specific tokens.
- According to the experimental results in two specific domains, SODA achieves promising performances on four domain-related tasks.

2 Background: ELECTRA

ELECTRA (Clark et al., 2020) trains a discriminator to predict whether each token in a corrupted input is replaced by a generator.

Given an original sequence $\boldsymbol{x} = [x_1, x_2, ..., x_n]$, 15% of the tokens are randomly replaced with `[MASK]` symbols. For each masked position i, the generator predicts a distribution $p_{\mathrm{G}}\left(x|\boldsymbol{h}_i\right)$, and then samples one token $x_i^{\mathrm{R}} \sim p_{\mathrm{G}}\left(x|\boldsymbol{h}_i\right)$ to replace the original token x_i, resulting in a corrupted sequence $\boldsymbol{x}^{\mathrm{R}}$. Here $\{\boldsymbol{h}_i\}_{i=1}^{n}$ are the contextualized representations generated by the Transformer.

Given the corrupted sequence $\boldsymbol{x}^{\mathrm{R}}$, the discriminator D is trained to distinguish each replaced token x_i^{R} against the original token x_i via the binary classification loss:

$$\begin{aligned}\mathcal{L}_{\mathrm{D}} = \mathbb{E}\Bigg(&- \sum_{x_i^{\mathrm{R}}=x_i} \log p_{\mathrm{D}}\left(x_i^{\mathrm{R}} = x_i|\boldsymbol{h}_i\right) \\ &- \sum_{x_i^{\mathrm{R}}\neq x_i} \log\left(1 - p_{\mathrm{D}}\left(x_i^{\mathrm{R}} = x_i|\boldsymbol{h}_i\right)\right)\Bigg),\end{aligned} \tag{1}$$

where $p_{\mathrm{D}}\left(x_i^{\mathrm{R}} = x_i|\boldsymbol{h}_i\right) = \mathrm{sigmoid}\left(\boldsymbol{w}^{\top}\boldsymbol{h}_i\right)$ and $\boldsymbol{w}$ is a learnable weight vector.

3 Method

Selecting Domain-specific Tokens. Grangier and Iter (2022); Moore and Lewis (2010) revealed that the domain-specific data can be selected according to the prediction differences between an in-domain model and an out-of-domain model. SODA selects domain-specific tokens with the help of a snapshot, where the snapshot represents the generator of an earlier step. We assume that the snapshot with fewer training steps is more "out-of-domain" than the current generator. Based on this assumption, we could select domain-specific tokens by comparing predictions of the current generator with those of the snapshot.

Specifically, for each masked position i in the input, the generator G and the snapshot S each predict a distribution. Suppose their predictions are $p_{\mathrm{G}}\left(x|\boldsymbol{h}_i\right)$ and $p_{\mathrm{S}}\left(x|\boldsymbol{h}_i\right)$ respectively, then we make a binary decision $b_{\mathrm{G,S}}(x_i)$ of whether token x_i is domain-specific as follows:

$$b_{\mathrm{G,S}}\left(x_i\right) = \begin{cases} 1 & x_i^{\mathrm{G}} \neq x_i^{\mathrm{S}} \\ 0 & x_i^{\mathrm{G}} = x_i^{\mathrm{S}}, \end{cases} \tag{2}$$

where x_i^{G} and x_i^{S} are sampled from the vocabulary V by:

$$\begin{aligned} x_i^{\mathrm{G}} &= \operatorname*{argmax}_{x\in V} p_{\mathrm{G}}\left(x|\boldsymbol{h}_i\right) \\ x_i^{\mathrm{S}} &= \operatorname*{argmax}_{x\in V} p_{\mathrm{S}}\left(x|\boldsymbol{h}_i\right). \end{aligned} \tag{3}$$

Emphasizing Domain-specific Tokens. Given the domain-specific tokens, we propose to emphasize them by assigning them a higher loss weight in the discriminator loss. Suppose ω_i is the loss weight assigned to the token x_i, then we set $\omega_i = 1 + \beta$ if x_i is domain-specific, and $\omega_i = 1$ otherwise, where β is the augmented loss weight. Based on Eq. 2, the loss weight of x_i is formulated as:

$$\omega_i = 1 + b_{\mathrm{G,S}}\left(x_i\right)\beta. \tag{4}$$

The re-weighted loss function based on Eq. 1 is

$$\begin{aligned}\mathcal{L}_{\mathrm{D}}^{\mathrm{SODA}} = \mathbb{E}\Bigg(&- \sum_{x_i^{\mathrm{R}}=x_i} \omega_i \log p_{\mathrm{D}}\left(x_i^{\mathrm{R}} = x_i|\boldsymbol{h}_i\right) \\ &- \sum_{x_i^{\mathrm{R}}\neq x_i} \omega_i \log\left(1 - p_{\mathrm{D}}\left(x_i^{\mathrm{R}} = x_i|\boldsymbol{h}_i\right)\right)\Bigg).\end{aligned} \tag{5}$$

Method	Computer Science			Biomedical		
	ACL-ARC	**SCIERC**	**Average**	**ChemProt**	**RCT**	**Average**
BERT as source model						
BERT	$70.96_{3.00}$	$81.13_{0.80}$	76.05	$84.76_{0.29}$	$87.64_{0.14}$	86.20
DAPT	$74.41_{2.80}$	$81.69_{1.29}$	78.05	$85.09_{0.52}$	$87.81_{0.04}$	86.45
AdaLM◇	73.61	81.91	77.76	-	-	-
RoBERTa as source model						
RoBERTa♡	$63.00_{5.80}$	$77.30_{1.90}$	70.15	$81.90_{1.00}$	$87.20_{0.10}$	84.55
DAPT♡	$75.40_{2.50}$	$80.80_{1.50}$	78.10	$84.20_{0.20}$	$87.60_{0.10}$	85.90
ELECTRA as source model						
ELECTRA	$73.93_{3.70}$	$81.95_{0.42}$	77.94	$83.99_{0.44}$	$87.84_{0.07}$	85.92
From Scratch	$70.05_{2.76}$	$79.08_{0.85}$	74.57	$85.08_{0.80}$	$87.76_{0.08}$	86.42
DAPT	$76.57_{2.00}$	$82.67_{0.76}$	79.62	$85.93_{0.28}$	$88.02_{0.07}$	86.98
DAPT with Random G	$74.07_{4.27}$	$82.97_{1.59}$	78.52	$86.14_{0.19}$	$87.98_{0.04}$	87.06
SODA	$\mathbf{77.13}_{2.14}$	$\mathbf{83.10}_{0.84}$	**80.12**	$\mathbf{86.20}_{0.61}$	$\mathbf{88.08}_{0.09}$	**87.14**

Table 1: Results of different model-based strategies on the domain-related tasks (◇ from (Yao et al., 2021) and ♡ from (Gururangan et al., 2020)). We report averages across five random seeds, with standard deviations as subscripts.

Training Framework. SODA continues to pre-train ELECTRA on the domain corpus using different snapshots for each training interval. Assume the minimum gap between the snapshot and the current generator is W, and the snapshot interval is T. Our strategy is to utilize the snapshot taken at step $n\mathrm{T}$ to assist token selection during the interval from $\mathrm{W}+n\mathrm{T}$ to $\mathrm{W}+(n+1)\mathrm{T}$.

For example, at training step $\mathrm{W}+n\mathrm{T}$, the generator at step $n\mathrm{T}$ is loaded as the snapshot to assist token selection. As the training progresses, we expect that the selection should prefer the tokens that are more specific to the current domain shift (van der Wees et al., 2017). Therefore, at the beginning of the next interval (step $\mathrm{W}+(n+1)$ T), we replace the snapshot with the generator at the step $(n+1)\mathrm{T}$ which is closer to the current generator.

4 Experiment

4.1 Datasets

We use the same pre-training corpora as AdaLM (Yao et al., 2021); the computer science corpus is collected from arXiv[1] and the biomedical corpus is the latest collection from PubMed[2]. For the downstream tasks, we use ACL-ARC (Jurgens et al., 2018) and SCIERC (Luan et al., 2018) for computer science, chemProt (Kringelum et al., 2016) and RCT (Dernoncourt and Lee, 2017) for biomedical domain. Specifications of these datasets are shown in Appendix A.

4.2 Implementation

We use ELECTRA$_{\text{BASE}}$ (Clark et al., 2020) as our source model. Our pre-training code is built upon Fairseq[3]. Detailed experimental settings of the continued pre-training are listed in Appendix B. For snapshot settings, the minimum gap W is 30K steps. We set the interval T as 35K steps for computer science domain and 20K steps for biomedical domain. Analysis of the snapshot interval is in Section 4.4. In the re-weighed loss function, the augmented loss weight β is 0.2 for computer science domain and 0.5 for biomedical domain. We recommend using β values less than 1 because a too high β value will negatively impact the learning of other tokens.

Our fine-tuning code is based on AdaLM[4]. We run hyperparameter search to find the best-performing models. The search settings and results are listed in Appendix B.

4.3 Main results

We present the downstream tasks results of different competitive methods in Table 1. For each of the source models, the first row is the general model without continued pre-training, and DAPT (Gururangan et al., 2020) is the vanilla continued pre-training on the domain corpus.

SODA achieves the best performances across the tasks in both domains when ELECTRA is the source model, demonstrating the effectiveness of

[1] https://www.kaggle.com/Cornell-University/arxiv

[2] https://pubmed.ncbi.nlm.nih.gov/

[3] https://github.com/facebookresearch/fairseq

[4] https://github.com/microsoft/unilm/tree/master/adalm

	Computer Science		Biomedical	
	ACL-ARC	**SCIERC**	**ChemProt**	**RCT**
DAPT	$76.57_{2.00}$	$82.67_{0.76}$	$85.93_{0.28}$	$88.02_{0.07}$
Rand	$76.28_{1.74}$	$82.29_{1.39}$	$86.03_{0.36}$	$88.01_{0.03}$
Know	$76.33_{2.56}$	$81.62_{0.81}$	$86.16_{0.25}$	$88.03_{0.06}$
Freq	$76.73_{2.00}$	$82.59_{0.68}$	$86.06_{0.28}$	$88.02_{0.08}$
SODA	**$77.13_{2.14}$**	**$83.10_{0.84}$**	**$86.20_{0.61}$**	**$88.08_{0.09}$**

Table 2: Results of different token selection methods. We report averages across five random seeds, with standard deviations as subscripts.

emphasizing domain-specific tokens in the continued pre-training. ELECTRA outperforms BERT- and RoBERTa-based methods through vanilla continued pre-training, which suggests the great potential of ELECTRA as the source model for domain adaptation. We also observe that whether to randomly initialize the generator has an insignificant impact on the continued pre-training of ELECTRA: DAPT with Random G performs better than DAPT in biomedical domain but worse than DAPT in computer science domain.

4.4 Ablation analysis

Token Selection Method. We compare our snapshot-guided token selection method with three alternatives: (1) Rand: randomly select 10% input tokens; (2) Know: select the tokens that are wrongly predicted by generator, because such tokens contain more knowledge (Wang et al., 2022); (3) Freq: select the tokens with high frequency differences between the target and source domain corpus, where we use Wikipedia corpus (Zhu et al., 2015) to represent the source domain. As shown in Table 2, SODA consistently outperforms all the alternatives. This suggests that dynamics is a crucial factor for token selection in domain adaptation.

Snapshot Interval Length. We test the snapshot interval lengths T at 70K, 35K, 20K steps to analyze the effects. As shown in Figure 2, compared to 70K, a relatively shorter interval (i.e., 35K for computer science and 20K for biomedical domain) can improve performance, because the shorter interval makes it possible to change the snapshots more frequently, so as to dynamically select the tokens that are more specific to the domain shift at hand.

We also record ratios of the selected tokens of all masked tokens in each interval. Figure 3 shows the records of the best-performing models in computer science and biomedical domains, of which the interval lengths are 35K and 20K respectively. The snapshot is loaded for the first time at step 30K, at which the ratio jumps from 0 to a high value. After that, the ratio drops every T steps until the end (95K). This is intuitive because as the model converges, there should be fewer domain-specific tokens to learn.

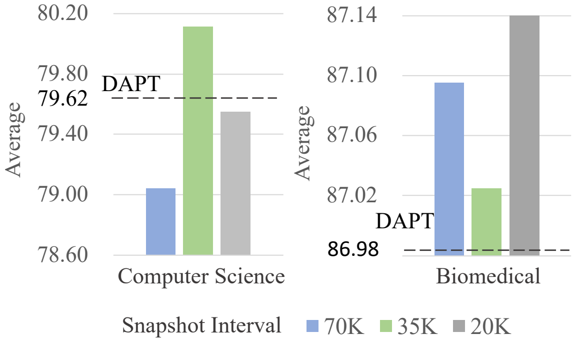

Figure 2: Experiments with different snapshot intervals. The Y-axis represents the average score of tasks in the domain.

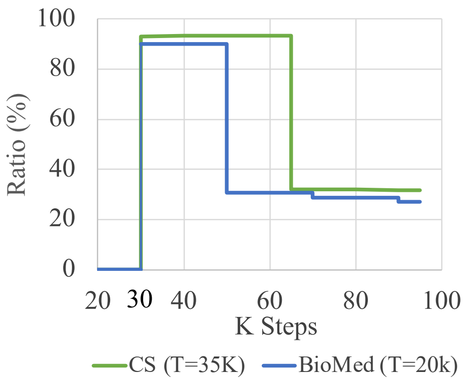

Figure 3: Changes of the selected ratio, the Y-axis represents the ratio of the selected tokens in all masked tokens.

Domain Specificity. In this paper, we use domain specificity to summarize the prediction differences due to different pre-training steps for domain adaptation. We also analyze the domain-specificity from the perspective of token frequency: first, make a domain-specific token set of the tokens with high frequency differences between the target and source (Wikipedia (Zhu et al., 2015)) domains. Second, measure specificity by calculating the ratio of the tokens belonging to the specific token set. Specificity of the predicted tokens by the snapshot and generator, and specificity of the selected tokens by SODA are as in Table 3.

From the results, specificity of the generator is higher than that of the snapshot, and most tokens

	Computer Science	Biomedical
Snapshot	0.45	0.39
Generator	0.69	0.66
Selected tokens	0.71	0.73

Table 3: Specificity of the predicted tokens by the snapshot and generator, and specificity of the selected tokens by SODA.

selected by SODA belong to the domain-specific token set. We also filter out the tokens not in the specific token set to check their impacts. From the results, filtering out such tokens leads to performance degradation, with an average score drop of 0.50 in the computer domain and 0.14 in the biomedical domain, proving that SODA could dynamically select tokens that are beneficial to continued pre-training, even if some of them are not specific in terms of token frequency.

4.5 Case study

We conduct case study to analyze the tokens selected at different training steps. Table 4 shows the results. We observe that the snapshot could help find the domain-specific tokens such as "paper" and "sorting" in the computer science domain and "inhibit" and "chemical" in the biomedical domain. Compared with the tokens selected at step 50K, the tokens at step 85K are fewer and more domain-specific, which proves SODA could dynamically select the domain-specific tokens as the training progresses.

5 Conclusion

In this paper, we design a snapshot-guided domain adaptation method for ELECTRA to capture the token-level domain knowledge by comparing generators of different training steps. Our method can dynamically select and emphasize the domain-specific tokens, which can benefit domain adaptation. Experimental results show that our method achieves state-of-the-art results without introducing additional training parameters.

Acknowledgements

The first author would like to thank Junyu Bi, Shitao Xiao, Yuanmeng Yan, Haifeng Sun and Chunning Du for the inspirational discussions and support. Many thanks to our anonymous reviewers and area chairs for their detailed and helpful suggestions and comments.

	Input: The **paper** is divided into **two** parts. Given a **graph** with vertices, **sorting** number...
50K	G_{50K}: The network is divided into two parts. Given a graph with vertices, sorting number...
	S_{0K}: The model is divided into two parts. Given a basis with vertices, such number...
85K	G_{85K}: The network is divided into two parts. Given a graph with vertices, sorting number...
	S_{35K}: The problem is divided into two parts. Given a way with vertices, sorting number...

	Input: Transport was **inhibit** ##ed **by** an applied **chemical gradient**... **clinical** investigation...
50K	G_{50K}: Transport was inhibit ##ed by an applied proton gradient... clinical investigation...
	S_{0K}: Transport was affect ##ed by an applied electrical force... good investigation...
85K	G_{85K}: Transport was inhibit ##ed by an applied chemical gradient... clinical investigation...
	S_{35K}: Transport was inhibit ##ed by an applied proton gradient... clinical investigation...

Table 4: The tokens selected at training step 50K/85K (shown in red) of the computer science and biomedical domains. Input is the original text, where tokens in boldface are masked. S and G stand for the snapshot and the generator respectively, with the trained steps as subscripts.

Limitations

We only conduct experiments on ELECTRA, future research may experiment on BERT-style models by replacing the discriminator with a BERT-style model to effectively adapt the model. Besides, we set the loss weights for the domain-specific tokens as static values, future research may explore dynamic loss weights to improve the performance.

Ethics Statement

All the pre-training and fine-tuning datasets, and the pre-trained models used in this work are publicly available.

References

Payal Bajaj, Chenyan Xiong, Guolin Ke, Xiaodong Liu, Di He, Saurabh Tiwary, Tie-Yan Liu, Paul Bennett, Xia Song, and Jianfeng Gao. 2022. METRO: efficient denoising pretraining of large scale autoencoding language models with model generated signals. *CoRR*, abs/2204.06644.

Zewen Chi, Shaohan Huang, Li Dong, Shuming Ma, Bo Zheng, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, Heyan Huang, and Furu Wei. 2022.

XLM-E: cross-lingual language model pre-training via ELECTRA. In *ACL*, pages 6170–6182. Association for Computational Linguistics.

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

Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k RCT: a dataset for sequential sentence classification in medical abstracts. In *IJCNLP*, pages 308–313. 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 *NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics.

David Grangier and Dan Iter. 2022. The trade-offs of domain adaptation for neural language models. In *ACL*, pages 3802–3813. Association for Computational Linguistics.

Yuxian Gu, Zhengyan Zhang, Xiaozhi Wang, Zhiyuan Liu, and Maosong Sun. 2020. Train no evil: Selective masking for task-guided pre-training. In *EMNLP*, pages 6966–6974. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasovic, 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 *ACL*, pages 8342–8360. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *CoRR*, abs/2111.09543.

David Jurgens, Srijan Kumar, Raine Hoover, Daniel A. McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *Trans. Assoc. Comput. Linguistics*, 6:391–406.

Jens Kringelum, Sonny Kim Kjærulff, Søren Brunak, Ole Lund, Tudor I. Oprea, and Olivier Taboureau. 2016. Chemprot-3.0: a global chemical biology diseases mapping. *Database J. Biol. Databases Curation*, 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. *Bioinform.*, 36(4):1234–1240.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *EMNLP*, pages 3219–3232. Association for Computational Linguistics.

Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2021. COCO-LM: correcting and contrasting text sequences for language model pretraining. In *NeurIPS*, pages 23102–23114.

Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2022. Pre-training text encoders with adversarial mixture of training signal generators. *CoRR*, abs/2204.03243.

Giacomo Miolo, Giulio Mantoan, and Carlotta Orsenigo. 2021. Electramed: a new pre-trained language representation model for biomedical nlp. *arXiv preprint arXiv:2104.09585*.

Robert C. Moore and William D. Lewis. 2010. Intelligent selection of language model training data. In *ACL*, pages 220–224. The Association for Computer Linguistics.

Jiaming Shen, Jialu Liu, Tianqi Liu, Cong Yu, and Jiawei Han. 2021. Training ELECTRA augmented with multi-word selection. In *ACL/IJCNLP (Findings)*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 2475–2486. Association for Computational Linguistics.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *EMNLP*, pages 1400–1410. Association for Computational Linguistics.

Cunxiang Wang, Fuli Luo, Yanyang Li, Runxin Xu, Fei Huang, and Yue Zhang. 2022. On effectively learning of knowledge in continual pre-training. *CoRR*, abs/2204.07994.

Zhenhui Xu, Linyuan Gong, Guolin Ke, Di He, Shuxin Zheng, Liwei Wang, Jiang Bian, and Tie-Yan Liu. 2020. MC-BERT: efficient language pre-training via a meta controller. *CoRR*, abs/2006.05744.

Yunzhi Yao, Shaohan Huang, Wenhui Wang, Li Dong, and Furu Wei. 2021. Adapt-and-distill: Developing small, fast and effective pretrained language models for domains. In *ACL/IJCNLP (Findings)*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 460–470. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, pages 19–27. IEEE Computer Society.

Appendices

A Downstream Tasks Details

Dom.	Task	Train	Dev.	Test	Classes
CS	ACL-ARC	1688	114	139	6
	SCIERC	3219	455	974	7
Bio.	ChemProt	4169	2427	3469	13
	RCT	18040	30212	30135	5

Table 5: Specifications of the fine-tuning task datasets in computer science (CS) and biomedical (Bio.) domains.

B Pre-training and Fine-tuning Settings

Computing Infrastructure	8 A100 GPUs
Runtime	10.5h
Number of Parameters	1.7e8
FLOPs of Snapshot	5.4e17
FLOPs of ELECTRA	1.66e19

Hyperparameter	Assignment
Number of steps	95K
Batch size	512
Maximum sequence length	512
Maximum learning rate	1e-4 (CS) or 2e-4 (Bio.)
Learning rate optimizer	Adam
Adam epsilon	1e-6
Adam beta weights	0.9, 0.98
Learning rate scheduler	warmup linear
Weight decay	0.01
Warmup steps	10K
Learning rate decay	linear

Table 6: Pre-training hyperparameters on computer science (CS) and biomedical (Bio.) domains.

Search Method	Uniform sampling
Criterion	macro-F1 (CS) or micro-F1 (Bio.)
Search Trails	16
LR bound	1e-5∼1e-4
WR bound	0.01∼0.15

Dom.	Task	Epochs	LR	WR	Batch Size
CS	ACL-ARC	30	7e-5	0.15	32
	SCIERC	30	9e-5	0.1	32
Bio.	ChemProt	30	7e-5	0.01	32
	RCT	4	2e-5	0.01	128

Table 7: Fine-tuning hyperparameter search settings for learning rate (LR) and warmup ratio (WR) for all the comparative methods, and configurations for the best-performing SODA, the weight decay is 0.1.