Pre-train or Annotate? Domain Adaptation with a Constrained Budget

Fan Bai, Alan Ritter, Wei Xu
School of Interactive Computing
Georgia Institute of Technology
`{fan.bai, alan.ritter, wei.xu}@cc.gatech.edu`

Abstract

Recent work has demonstrated that pre-training in-domain language models can boost performance when adapting to a new domain. However, the costs associated with pre-training raise an important question: given a fixed budget, what steps should an NLP practitioner take to maximize performance? In this paper, we view domain adaptation with a constrained budget as a consumer choice problem, where the goal is to select an optimal combination of data annotation and pre-training. We measure annotation costs of three procedural text datasets, along with the pre-training costs of several in-domain language models. The utility of different combinations of pre-training and data annotation are evaluated under varying budget constraints to assess which combination strategy works best. We find that for small budgets, spending all funds on annotation leads to the best performance; once the budget becomes large enough, however, a combination of data annotation and in-domain pre-training yields better performance. Our experiments suggest task-specific data annotation should be part of an economical strategy when adapting an NLP model to a new domain.[1]

1 Introduction

The conventional wisdom on semi-supervised learning and unsupervised domain adaptation is that labeled data is expensive; therefore, training on a combination of labeled and unlabeled data is an economical approach to improve performance when adapting to a new domain (Blum and Mitchell, 1998; Daume III and Marcu, 2006; Hoffman et al., 2018; Chen et al., 2020). Recent work has shown that pre-training in-domain Transformers is an effective method for unsupervised adaptation (Han and Eisenstein, 2019; Wright and Augenstein, 2020) and even boosts performance when large quantities of in-domain data are available (Gururangan et al., 2020). However, modern pre-training methods incur substantial costs (Izsak et al., 2021), and generate carbon emissions (Strubell et al., 2019; Schwartz et al., 2020; Bender et al., 2021). This raises an important question: given a fixed budget to improve a model's performance, what steps should an NLP practitioner take? On one hand, they could hire annotators to label in-domain task-specific data, while on the other, they could buy or rent GPUs or TPUs to pre-train large in-domain language models. In this paper, we empirically study the best strategy for adapting to a new domain given a fixed budget.

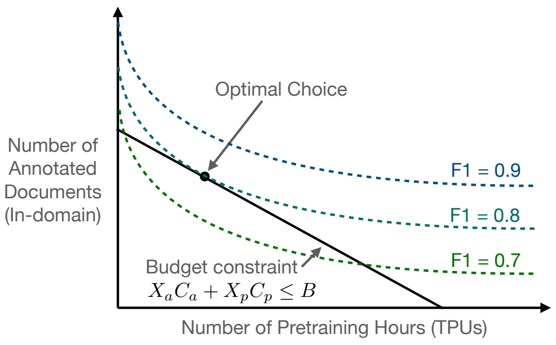


Figure 1: We view domain adaptation as a consumer choice problem (Becker, 1965; Lancaster, 1966). The NLP practitioner (consumer) is faced with the problem of choosing an optimal combination of annotation and pre-training under a constrained budget. This figure is purely for illustration and is not based on experimental data.

We view the NLP practitioner's dilemma of how to adapt to a new domain as a problem of consumer choice, a classical problem in microeconomics (Becker, 1965; Lancaster, 1966). As illustrated in Figure 1, the NLP practitioner (consumer) can obtain X_a annotated documents (by hiring annotators) at a cost of C_a each, and X_p hours of pre-training (by renting GPUs or TPUs) at a cost of C_p per hour. Given a fixed budget B, the consumer may choose

[1]Our code and data are publicly available on Github: `https://github.com/bflashcp3f/ProcBERT`.

any combination that fits within the budget constraint $X_a C_a + X_p C_P \leq B$. The goal is to choose a combination that maximizes the utility function, $U(X_a, X_p)$, which can be defined using an appropriate performance metric, such as F_1 score, that is achieved after pre-training for X_p hours and then fine-tuning on X_a in-domain documents.

To empirically estimate the cost of annotation, we hire annotators to label domain-specific documents for supervised fine-tuning in three procedural text domains: wet-lab protocols, paragraphs describing scientific procedures in PubMed articles, and chemical synthesis procedures described in patents. We choose to target natural language understanding for scientific procedures in this study, because there is an opportunity to help automate lab protocols and support more reproducible scientific experiments, yet few annotated datasets currently exist in these domains. Furthermore, annotation of scientific procedures is not easily amenable to crowdsourcing, making this an ideal testbed for pre-training-based domain adaptation. We measure the cost of in-domain pre-training on a large collection of unlabeled procedural texts using Google's Cloud TPUs.[2] Model performance is then evaluated under varying budget constraints in six source and target domain combinations.

Our analysis suggests that given current costs of pre-training large Transformer models, such as BERT (Devlin et al., 2019), and RoBERTa (Liu et al., 2019), in-domain data annotation should always be part of an economical strategy when adapting a single NLP system to a new domain. For small budgets (e.g. less than \$800 USD), spending all funds on annotation is the best policy; however, as more funding becomes available, a combination of pre-training and annotation is the best choice.

This paper addresses a specific question that is often faced by NLP practitioners working on applications: what is the most economical approach to adapt an NLP system to a new domain when no pre-trained models or task-annotated datasets are initially available? If multiple NLP systems need to be adapted to a single target domain, model costs can be amortized, making pre-training an attractive option for smaller budgets.

2 Scope of the Study

In this study, we focus on a typical scenario faced by an NLP practitioner to adapt a single NLP system to a single new domain, maximizing performance within a constrained budget. We consider only the direct benefit on the target task in our main analysis (§5), however we do provide additional analysis of positive externalities on other related tasks that may benefit from a new pre-trained model in §6.

We estimate cost based on two major expenses: annotating task-specific data (§3) and pre-training domain-specific models using TPUs (§4). Note that fine-tuning costs are not included in our analysis, as they are nearly equal whether budget is invested into pre-training or annotation.[3] We assume a generic BERT is the closest zero-cost model that is initially available, which is likely the case in real-world domain adaptation scenarios (especially for non-English languages). Our experiments are designed to simulate a scenario where no domain-specific model is initially available. We also assume that the NLP engineer's salary is a fixed cost; in other words, their salary will be the same whether they spend time pre-training models or managing a group of annotators.[4] Our primary concerns are about financial and environmental costs, rather than the overall time needed to obtain the adapted model. If the timeline is an important factor, the annotation process can be possibly sped up by hiring more annotators.

3 Estimating Annotation Cost (C_a)

In this section, we present our estimates of the annotation cost for three procedural text datasets from specialized scientific domains, which enable a comparison of model performance under varying budget constraints (§5.4).

Annotated Procedural Text Datasets. We experiment with three procedural text corpora, including Wet Lab Protocols (WLP; Tabassum et al., 2020) and two new datasets we created for this study, which include scientific articles and chemical patents. Statistics of the three datasets are shown in Table 1. The **WLP corpus** includes 726 wet lab experiment instructions collected from `protocols.io` which are annotated using an inventory of 20 entity types and 16 relation types.

[2]https://cloud.google.com/tpu

[3]These are also not a significant portion of overall costs (we estimate \$1.95 based on Google Cloud rates for P100 GPUs.)

[4]Based on our experience, pre-training in-domain models (collecting training corpus is non-trivial) and managing a team of annotators are roughly comparable in terms of effort.

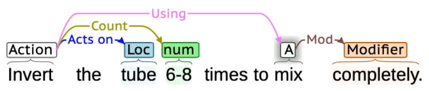
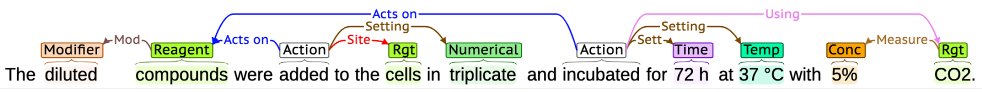
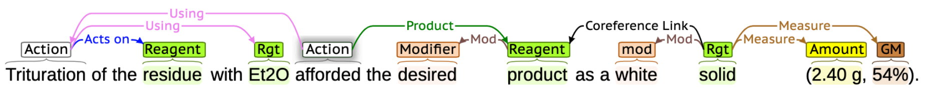
Dataset	Domain	Task	#Files (train/dev/test)	#Sentences	#Cases	#Classes	Total Cost	Price/File	Price/Sent.
WLP (Tabassum et al., 2020)	biology	NER RE	726 (492/123/111)	17,658	185,313 124,803	20 16	$7,820	$10.8	$0.44
									
PubMedM&M (this work)	biomed	NER RE	191 (100/41/50)	1,699	12,131 8,987	24 17	$1,730	$9.1	$1.02
									
ChemSyn (this work)	chemistry	NER RE	992 (793/99/100)	8,331	53,423 46,878	24 17	$5,000	$4.7	$0.60
									

Table 1: Statistics and examples of three procedural text datasets.

Following the same annotation scheme, we annotate PubMedM&M and ChemSyn. The **PubMedM&M corpus** consists of 191 double-annotated experimental paragraphs extracted from the *Materials and Methods* section of PubMed articles. The **ChemSyn corpus** consists of 992 chemical synthesis procedures described in patents, 500 of which are double-annotated. Unlike the succinct, informal language style in WLP, PubMedM&M represents an academic writing style, as it comes from published research papers (see Table 1). More details on data pre-processing, annotation and inter-annotator agreement scores can be found in Appendix A

Annotation Cost. We recruit undergraduate students to annotate the datasets using the BRAT annotation tool.[5] Annotators are paid 13 USD / hour throughout the process, which is the standard rate for undergraduate students at our university. Estimates of the cost of annotation, C_a, per-sentence are presented in Table 1.[6]

4 Estimating Pre-training Cost (C_p)

To evaluate varied strategies for combining pre-training and annotation given a fixed budget, we need accurate estimates on the cost of annotation, C_a, and pretraining, C_p. Having estimated the cost of annotating in-domain procedural text corpora in §3, we now turn to estimate the cost of in-domain pretraining. Specifically, we consider two popular approaches: 1) training an in-domain language model from scratch; 2) continued pre-training using an off-the-shelf model.

Procedure Corpus Collection. To pre-train our models, we create a novel collection of procedural texts from the same domains as the annotated data in §3, hereinafter referred to as the Procedure corpus.

Specially trained classifiers were used to identify paragraphs describing experimental procedures. For PubMed, a classifier was used to identify paragraphs describing experimental procedures by fine-tuning SciBERT (Beltagy et al., 2019) on the SciSeg dataset (Dasigi et al., 2017), which is annotated with scientific discourse structure, to extract procedures from the *Materials and Methods* section of 680k articles. For the chemical synthesis domain, the chemical reaction extractor developed by Lowe (2012) was applied to the *Description* section of 303k patents (174k U.S. and 129k European) we collected from USPTO[7] and EPO[8]. More details of our data collection process can be found in Appendix B.

Cooking recipes are also an important domain for research on procedural text understanding, therefore we include the text component of the Recipe1M+ dataset (Marín et al., 2021) in the Procedure pre-training corpus. In total, our **Procedure** collection contains around 1.1 billion words; more statistics are shown in Table 2. In addition, we create an extended version, **Procedure+**, consisting of 12 billion words, where we up-sample the procedural paragraphs 6 times and

[5]https://github.com/nlplab/brat

[6]The average time to annotate each sentence varies across datasets based on the complexity of the text, average length of sentences, etc.

[7]https://www.uspto.gov/

[8]https://www.epo.org/

combine them with the original full text of 680k PubMed articles and 303k chemical patents. This up-sampling ensures at least half of the text is procedural.

Pre-training Process and Cost. We train two procedural domain language models on the Google Cloud Platform using 8-core v3 TPUs: 1) **ProcBERT**, a BERT$_{base}$ model pre-trained from scratch using our PROCEDURE+ corpus, and 2) **Proc-RoBERTa**, for which we continued pre-training RoBERTa$_{base}$ on the PROCEDURE corpus following Gururangan et al. (2020).

We pre-train ProcBERT using the TensorFlow codebase of BERT.[9] Following Devlin et al. (2019), we deploy a two-step regime: the model is trained with sequence length 128 and batch size 512 for 1 million steps at a rate of 4.71 steps/second. Then, it is trained for 100k more steps using sequences of length 512 and a batch size of 256 at a rate of 1.83 steps/second. The pretraining process takes about 74 hours, and the total cost is about 620 USD, which includes the price for on-demand TPU-v3s (8 USD/hour)[10] plus auxiliary costs for virtual machines and data storage.

We considered the possibility of evaluating checkpoints of partially pre-trained models, for fine-grained variation of the pre-training budget, however after some investigation we chose to only report results on fully pre-trained models, using established training protocols (learning rate, number of parameter updates, model size, sequence length, etc.) to ensure fair comparison.

In addition to pre-training from scratch, we also experiment with Domain-Adaptive Pre-training, using the codebase[11] released by AI2 to train Proc-RoBERTa. Similar to Gururangan et al. (2020), we fine-tune RoBERTa on our collected PROCEDURE corpus for 12.5k steps with the averaged speed of 27.27 seconds per step, which leads to a TPU time of 95 hours.[12] Thus, the total cost of Proc-RoBERTa is around 800 USD after including the auxiliary expenses.

Finally, we estimate the cost of training for **SciBERT** (Beltagy et al., 2019), which was also trained on an 8-core TPU v3 using a two-stage training process similar to ProcBERT. The overall training of SciBERT took 7 days (5 days for the first stage and 2 days for the second stage) with an estimated cost of 1,340 USD.

Carbon Footprint. Apart from the financial cost, we also estimate the carbon footprint of each in-domain pre-trained language model for its environmental impact. We measure the energy consumption in kilowatt-hours (KWh) as in Patterson et al. (2021):

$$Energy = H \times N \times P \times PUE/1000,$$

where H is the number of training hours, N is the number of processors used, P is the average power per processor,[13] and PUE (Power Usage Effectiveness) indicates the energy usage efficiency of a data center. In our case, the average power per TPU v3 processor is 283 watts, and we use a PUE coefficient of 1.10, which is the average trailing twelve-month PUE reported for all Google data centers in Q1 2021.[14] Once we know the energy consumption, we can estimate the CO_2 emissions (CO_2e) as follows:

$$CO_2e = (Energy \times CO_2e/KWh)/1000$$

where CO_2e/KWh measures the amount of CO_2 emission when consuming 1 KWh energy, which is 474g/KWh for our pre-training.[15] For example, ProcBERT is pre-trained on a single 8-core TPU v3 for 74 hours, resulting in CO_2 emission of $(74 \times 8 \times 283 \times 1.10/1000) \times 474/1000 = 87.4$ kg. The estimated CO_2 emissions for three in-domain language models are shown in Table 3.

5 Measuring Utility $U(X_a, X_p)$ under Varying Budget Constraints

Given the estimated unit cost of annotation C_a (§3) and pre-training C_p (§4), we now empirically evaluate the utility $U(X_a, X_p)$, of various budgets and pre-training strategies to find an optimal policy for domain adaptation that fits within the budget constraint $X_aC_a + X_pC_P \leq B$.

[9] https://github.com/google-research/bert

[10] https://cloud.google.com/tpu/pricing

[11] https://github.com/allenai/tpu_pretrain

[12] This is comparable to the number reported by the authors of Gururangan et al. (2020) on GitHub.

[13] Unlike Strubell et al. (2019) which measured GPU, CPU and DRAM's power separately, Patterson et al. (2021) measured the power of a processor together with other components including fans, network interface, host CPU, etc.

[14] https://www.google.com/about/datacenters/efficiency/

[15] Our models were pre-trained in the data center of Google in Netherlands: https://cloud.google.com/sustainability/region-carbon.

[16] Source: Google Flights (Patterson et al., 2021).

Corpus	#Tokens	Text Size	Pre-trained Model
Wiki + Books	3.3B	16GB	BERT
Web crawl	-	160GB	RoBERTa
PMC + CS	3.2B	-	SciBERT
BioMed	7.6B	47GB	BioMed-RoBERTa
PROCEDURE	1.05B	6.5GB	Proc-RoBERTa
- PubMed	0.32B	2.0GB	–
- Chem. patent	0.61B	3.9GB	–
- Cook. recipe	0.11B	0.6GB	–
PROCEDURE+	12B	77GB	ProcBERT
- PROCEDURE ($\times$ 6)	6.3B	39GB	–
- Full articles	5.7B	38GB	–

Table 2: Statistics of our newly created PROCEDURE and PROCEDURE+ corpora, which are used for pre-training Proc-RoBERTa and ProcBERT, respectively.

Consumption	CO_2e (kg)
Air travel, one person, SF$\leftrightarrow$NY	1200[16]
SciBERT (Beltagy et al., 2019)	198.3
Proc-RoBERTa	112.1
ProcBERT	87.4

Table 3: Carbon footprint of three in-domain pre-trained language models. CO_2e is the number of metric tons of CO_2 emissions with the same global warming potential as one metric ton of another greenhouse gas.

5.1 NLP Tasks and Models

We experiment with two NLP tasks, Named Entity Recognition (NER) and Relation Extraction (RE). For NER, we follow Devlin et al. (2019) to feed the contextualized embedding of each token into a linear classification layer. For RE, we follow Zhong and Chen (2020), inserting four special tokens specifying positions and types of each entity-pair mention, which are included as input to a pre-trained sentence encoder. Gold entity mentions are used in our relation extraction experiments, to reduce variance due to entity recognition errors.

5.2 Budget-constrained Experimental Setup

As we have three procedural text datasets (§3) annotated with entities and relations, we can experiment with six source $\Rightarrow$ target adaptation settings. For each domain pair, we compare five different pre-trained language models when adapted to the procedural text domain under varying budgets.

Based on the estimations of the annotation costs C_a (§3) and pre-training costs C_p (§4), we conduct various budget-constrained domain adaptation experiments. For example, if we have \$1,500 and the PUBMEDM&M corpus, to build an NER model that works best for the CHEMSYN domain (PUBMEDM&M$\Rightarrow$CHEMSYN), we can spend all \$1,500 to annotate 2,500 in-domain sentences to fine-tune off-the-shelf BERT. Or alternatively, we could first spend \$800 to pre-train Proc-RoBERTa, then fine-tune it on 1155 sentences annotated in the CHEMSYN domain using the remaining of \$700. Under both budgeting strategies, an additional experiment is performed to choose one of two domain adaption methods that maximizes performance: 1) a model that simply uses the annotated data in the target domain for fine-tuning; or 2) a model which is fine-tuned using a variant of EasyAdapt (Daumé III, 2007) to leverage annotated data in both the source and target domains (see below for details). We select the approach that has better development set performance and report its test set result in Table 4 and 5 (see Appendix D for more details about hyper-parameters).

5.3 EasyAdapt

In most of our experiments, we have access to a relatively large amount of labeled data from a source domain, and varying amounts of data from the target domain. Instead of simply concatenating the source and target datasets for fine-tuning, we propose a simple, yet novel variation of EasyAdapt (Daumé III, 2007) for pre-trained Transformers. More specifically, we create three copies of the model's contextualized representations: one represents the source domain, one represents the target, and the third is domain-independent. These contextualized vectors are then concatenated and fed into a linear layer that is 3 times as large as the base model's. When encoding data from a specific domain (e.g. CHEMSYN), the other domain's representations are zeroed out (1/3 of the new representations will always be 0.0). This enables the domain-specific block of the linear layer to encode information specific to that domain, while the domain-independent parameters can learn to represent information that transfers across domains. This is similar to prior work using EasyAdapt (Kim et al., 2016) for LSTMs.

5.4 Experimental Results and Analysis

We present the test set NER and RE results with five annotation and pre-training combination strategies under six domain adaptation settings in Table 4 and 5, respectively. We report averages across five random seeds with standard deviations as subscripts. If pre-training costs go over the total budget, or available data goes under the annotation budget, we

Source ⇒ Target Domain	Budget	**BERT**$_{base}$ $0		**BERT**$_{large}$ $0		**ProcBERT** $620		**Proc-RoBERTa** $800		**SciBERT** $1340	
		F_1	(#sent)	F_1	(#sent)	F_1	(#sent)	F_1	(#sent)	F_1	(#sent)
PUBMEDM&M⇒CHEMSYN	$700	$92.99_{0.2}^{\dagger}$	(1166)	$\mathbf{93.32}_{0.3}^{\dagger}$	(1166)	$91.07_{0.4}^{\dagger}$	(133)	N/A		N/A	
	$1500	$93.58_{0.3}^{\dagger}$	(2500)	$94.14_{0.2}^{\dagger}$	(2500)	$\mathbf{94.73}_{0.1}^{\dagger}$	(1466)	$93.83_{0.1}^{\dagger}$	(1166)	$92.46_{0.2}^{\dagger}$	(266)
	$2300	$94.47_{0.2}$	(3833)	$94.81_{0.1}^{\dagger}$	(3833)	$\mathbf{95.17}_{0.1}^{\dagger}$	(2800)	$94.71_{0.2}^{\dagger}$	(2500)	$94.39_{0.2}$	(1600)
WLP⇒CHEMSYN	$700	$93.31_{0.3}^{\dagger}$	(1166)	$\mathbf{93.59}_{0.3}^{\dagger}$	(1166)	$90.83_{0.3}^{\dagger}$	(133)	N/A		N/A	
	$1500	$94.31_{0.3}^{\dagger}$	(2500)	$94.44_{0.2}^{\dagger}$	(2500)	$\mathbf{94.88}_{0.1}^{\dagger}$	(1466)	$94.20_{0.2}^{\dagger}$	(1166)	$92.39_{0.2}^{\dagger}$	(266)
	$2300	$94.47_{0.2}$	(3833)	$95.09_{0.3}^{\dagger}$	(3833)	$\mathbf{95.52}_{0.1}^{\dagger}$	(2800)	$94.63_{0.3}$	(2500)	$94.39_{0.2}$	(1600)
WLP⇒PUBMEDM&M	$700	$74.41_{0.8}^{\dagger}$	(686)	$\mathbf{75.41}_{0.5}^{\dagger}$	(686)	$68.30_{0.7}^{\dagger}$	(78)	N/A		N/A	
	$1500	N/A		N/A		$\mathbf{76.76}_{0.7}^{\dagger}$	(862)	$76.02_{0.5}$	(686)	$72.66_{0.4}^{\dagger}$	(156)
CHEMSYN⇒PUBMEDM&M	$700	$75.10_{0.5}^{\dagger}$	(686)	$\mathbf{75.71}_{0.4}^{\dagger}$	(686)	$72.85_{0.6}^{\dagger}$	(78)	N/A		N/A	
	$1500	N/A		N/A		$\mathbf{77.62}_{0.5}^{\dagger}$	(862)	$76.28_{0.5}^{\dagger}$	(686)	$73.93_{0.4}^{\dagger}$	(156)
CHEMSYN⇒WLP	$700	$72.23_{0.4}^{\dagger}$	(1590)	$\mathbf{73.21}_{0.2}^{\dagger}$	(1590)	$72.42_{0.4}^{\dagger}$	(181)	N/A		N/A	
	$1500	$73.30_{0.4}^{\dagger}$	(3409)	$73.46_{0.4}$	(3409)	$\mathbf{75.15}_{0.4}^{\dagger}$	(2000)	$73.90_{0.5}^{\dagger}$	(1590)	$72.48_{0.4}^{\dagger}$	(363)
	$2300	$73.18_{0.4}$	(5227)	$74.12_{0.2}^{\dagger}$	(5227)	$\mathbf{75.88}_{0.3}^{\dagger}$	(3818)	$74.67_{0.4}^{\dagger}$	(3409)	$74.68_{0.5}^{\dagger}$	(2181)
PUBMEDM&M⇒WLP	$700	$72.66_{0.3}^{\dagger}$	(1590)	$\mathbf{73.18}_{0.7}^{\dagger}$	(1590)	$72.91_{0.3}^{\dagger}$	(181)	N/A		N/A	
	$1500	$73.62_{0.4}^{\dagger}$	(3409)	$73.46_{0.4}$	(3409)	$\mathbf{75.25}_{0.1}^{\dagger}$	(2000)	$73.98_{0.2}^{\dagger}$	(1590)	$72.58_{0.2}^{\dagger}$	(363)
	$2300	$73.73_{0.3}^{\dagger}$	(5227)	$74.26_{0.2}^{\dagger}$	(5227)	$\mathbf{75.78}_{0.3}^{\dagger}$	(3818)	$74.68_{0.2}^{\dagger}$	(3409)	$74.80_{0.2}^{\dagger}$	(2181)

Table 4: Experiment results for Named Entity Recognition (NER). With higher budgets ($1500 and $2300), our in-domain pre-training of ProcBERT achieves the best results in combination with data annotation. For a smaller budget ($700), investing all funds in annotation and fine-tuning the standard BERT$_{large}$ (considered as cost-free) will yield the best outcome. **#sent** is the number of sentences from the target domain, annotated under the given budget, used for training. † indicates results using EasyAdapt (§5.3), where source domain data helps.

indicate the result as "NA". We now discuss a set of key questions regarding pre-training-based domain adaptation under a constrained budget, which our experiments can shed some light on.

Should we prioritize pre-training or data annotation for domain adaptation? For all six domain adaptation settings in Table 4, spending the entire budget on annotation and using the off-the-shelf language model BERT$_{large}$ works the best for NER when the budget is 700 USD, showing the effectiveness of data annotation in low-resource scenarios. As the budget increases, performance gains from labelling additional data diminish, and pre-training in-domain language models takes the lead. ProcBERT, which is pre-trained from scratch on the PROCEDURE+ corpus costing only 620 USD, performs best at budgets of 1500 and 2300 USD. This demonstrates that combining domain-specific pre-training with data annotation is the best strategy in high-resource settings. Similarly for RE, as shown in Table 5, using all funds for data annotation and working with off-the-shelf models achieves better performance at lower budgets, while domain-specific pre-training starts to excel as the budget increases past a certain point.

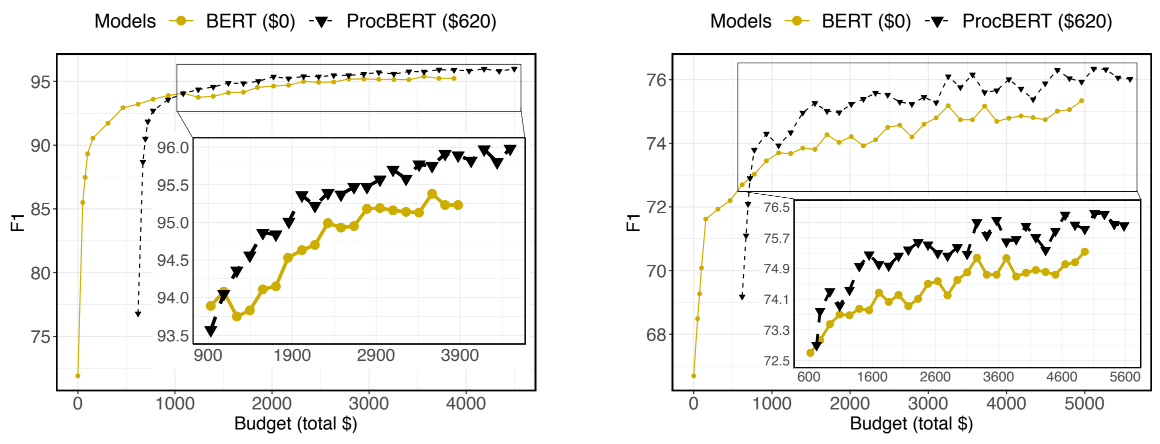
What is the starting budget to consider pre-training an in-domain language model? To answer this question, we plot test set NER performance for two strategies, BERT$_{large}$ (investing all funds on annotation) and ProcBERT (combining annotation with pre-training), against varying budgets in Figure 2. Specifically, the budget of each strategy starts with the pre-training cost of its associated language model, and is increased by 155 USD increments until the total budget reaches the total cost of available data[17]. We observe a similar trend for both PUBMEDM&M⇒CHEMSYN and CHEMSYN⇒WLP: annotation alone works better in lower budgets while in-domain pre-training (ProcBERT) excels at higher budgets. However, the intersection of the curves for these two strategies occurs at different points, which are around 1085 USD and 775 USD.

We hypothesize there are two reasons for this difference: 1) each target domain may require different amounts of labeled data to generalize well; 2) the quantity of labeled data from the source domain may also impact the need for data annotation in the target domain. To testify our hypotheses, we evaluate the utility of annotation vs. pre-training where no source-domain data is available in Figure 3. This is almost identical to the setting of Figure

[17]We also add a few points at the start of each curve to make them smoother. For BERT$_{large}$, we add 50 USD, 75 USD and 100 USD. For ProcBERT, we add 670 USD, 695 USD and 720 UDS.

Source ⇒ Target Domain	Budget	BERT$_{base}$ $0		BERT$_{large}$ $0		ProcBERT $620		Proc-RoBERTa $800		SciBERT $1340	
		F$_1$	(#sent)	F$_1$	(#sent)	F$_1$	(#sent)	F$_1$	(#sent)	F$_1$	(#sent)
PUBMEDM&M⇒CHEMSYN	$700	$90.12_{0.4}^{\dagger}$	(1166)	**$90.71_{0.8}^{\dagger}$**	(1166)	$88.21_{0.6}^{\dagger}$	(133)	N/A		N/A	
	$1500	$91.30_{0.2}^{\dagger}$	(2500)	$91.84_{0.5}^{\dagger}$	(2500)	**$92.06_{0.4}^{\dagger}$**	(1466)	$91.25_{0.7}^{\dagger}$	(1166)	$89.58_{0.5}^{\dagger}$	(266)
	$2300	$91.81_{0.2}^{\dagger}$	(3833)	**$92.90_{0.4}^{\dagger}$**	(3833)	$92.57_{0.2}^{\dagger}$	(2800)	$92.07_{0.3}^{\dagger}$	(2500)	$91.55_{0.2}^{\dagger}$	(1600)
WLP⇒CHEMSYN	$700	**$90.20_{0.7}^{\dagger}$**	(1166)	$90.15_{0.9}$	(1166)	$88.01_{0.6}^{\dagger}$	(133)	N/A		N/A	
	$1500	$91.34_{0.3}^{\dagger}$	(2500)	$91.61_{0.5}^{\dagger}$	(2500)	**$92.27_{0.2}^{\dagger}$**	(1466)	$91.77_{0.2}^{\dagger}$	(1166)	$89.16_{0.7}^{\dagger}$	(266)
	$2300	$92.08_{0.4}^{\dagger}$	(3833)	$92.73_{0.4}^{\dagger}$	(3833)	**$92.85_{0.2}^{\dagger}$**	(2800)	$92.44_{0.6}^{\dagger}$	(2500)	$91.42_{0.4}^{\dagger}$	(1600)
WLP⇒PUBMEDM&M	$700	$77.74_{0.7}^{\dagger}$	(686)	**$79.33_{1.3}^{\dagger}$**	(686)	$74.63_{0.4}^{\dagger}$	(78)	N/A		N/A	
	$1500	N/A		N/A		**$80.10_{0.6}^{\dagger}$**	(862)	$79.33_{1.3}^{\dagger}$	(686)	$75.70_{1.1}^{\dagger}$	(156)
CHEMSYN⇒PUBMEDM&M	$700	$77.02_{0.6}^{\dagger}$	(686)	**$77.31_{0.4}^{\dagger}$**	(686)	$73.92_{0.8}^{\dagger}$	(78)	N/A		N/A	
	$1500	N/A		N/A		**$79.50_{0.7}^{\dagger}$**	(862)	$77.12_{1.2}^{\dagger}$	(686)	$75.43_{0.6}^{\dagger}$	(156)
CHEMSYN⇒WLP	$700	$78.81_{0.6}^{\dagger}$	(1590)	**$79.80_{0.6}^{\dagger}$**	(1590)	$78.97_{0.7}^{\dagger}$	(181)	N/A		N/A	
	$1500	$79.91_{0.4}^{\dagger}$	(3409)	$80.15_{0.8}^{\dagger}$	(3409)	**$80.87_{0.7}^{\dagger}$**	(2000)	$79.69_{0.4}^{\dagger}$	(1590)	$79.44_{0.8}^{\dagger}$	(363)
	$2300	$80.54_{0.6}^{\dagger}$	(5227)	$80.97_{0.2}^{\dagger}$	(5227)	$81.15_{0.6}^{\dagger}$	(3818)	**$81.33_{0.5}^{\dagger}$**	(3409)	$80.16_{0.7}^{\dagger}$	(2181)
PUBMEDM&M⇒WLP	$700	$78.34_{0.8}^{\dagger}$	(1590)	**$78.44_{1.2}^{\dagger}$**	(1590)	$77.98_{0.8}^{\dagger}$	(181)	N/A		N/A	
	$1500	$79.40_{0.6}^{\dagger}$	(3409)	$79.78_{0.6}^{\dagger}$	(3409)	**$80.45_{0.2}^{\dagger}$**	(2000)	$78.79_{1.1}^{\dagger}$	(1590)	$78.76_{0.5}^{\dagger}$	(363)
	$2300	$79.93_{0.5}^{\dagger}$	(5227)	$80.04_{0.8}^{\dagger}$	(5227)	**$80.85_{0.6}^{\dagger}$**	(3818)	$79.49_{0.7}^{\dagger}$	(3409)	$80.33_{0.3}^{\dagger}$	(2181)

Table 5: Experiment results for Relation Extraction (RE). Similar to the observations in Table 4, regardless of a small or large budget, prioritizing data annotation in the target domain is the most beneficial. † indicates results using EasyAdapt (§5.3), where source domain data helps.

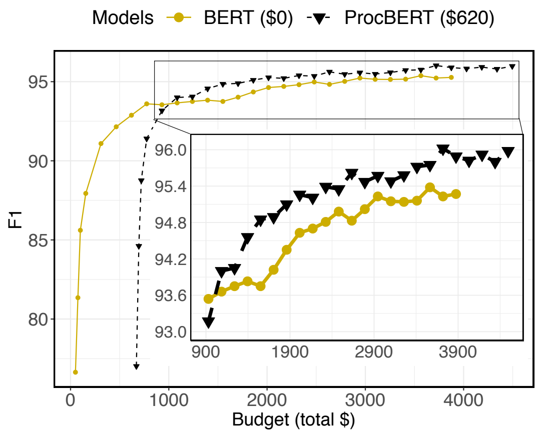


(a) PUBMEDM&M⇒CHEMSYN (b) CHEMSYN⇒WLP

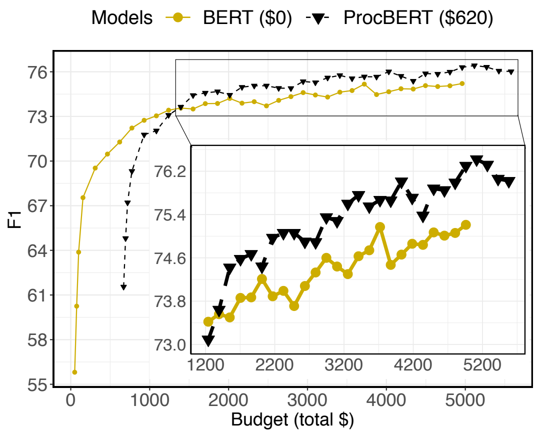
Figure 2: Comparison of two domain adaptation strategies: 1) allocate all available funds to data annotation; 2) pre-train ProcBERT on in-domain data, then use the remaining budget for annotation. For small budgets, the former yields the best performance on NER, but as the budget increases, the later becomes the best choice.

2 except models are trained on **target domain labeled data only**. Here, we observe the intersection for CHEMSYN is still around 1085 USD while the crossover point for WLP moves from the original 775 USD (in Figure 2) to around 1395 USD. Our hypothesis is that WLP is a broader domain compared to CHEMSYN (WLP covers a more diverse range of protocols that include cell cultures, DNA sequencing, etc.), so it requires more annotated data to perform well under the setting of Figure 3. However, when adapted from a large source domain dataset like CHEMSYN, the need for annotated WLP corpus is reduced so that ProcBERT can outperform BERT$_{large}$ at a lower budget.

Note that our estimated annotation cost, C_a, (per sentence) includes the annotation of both entity mentions and relations, so our analysis amortizes the cost of pre-training across both tasks. In a scenario where more tasks need to be adapted for the target domain, this could be accounted for simply by dividing the cost of pre-training among tasks, which would shift the black curves in Figure 2 and Figure 3 to the left, making pre-training an eco-



(a) ChemSyn



(b) WLP

Figure 3: Comparison of spending the entire budget on data annotation (●) and pre-training followed by in-domain annotation (▼), where models are trained on **target domain labeled data only**. The crossover point for WLP moves from 775 USD (adapted from ChemSyn) to around 1395 USD (WLP only) demonstrating that a large source domain dataset can reduce the need for target domain annotation.

Target	Entities	BERT$_{large}$ $0	ProcBERT $620
ChemSyn (budget $1085)	seen by both	**97.63**	97.45
	unseen by both	84.96	**85.65**
	seen by BERT	**94.07**	93.44
	All	93.78	93.82
WLP (budget $1395)	seen by both	**85.42**	84.78
	unseen by both	55.75	**57.19**
	seen by BERT	**76.50**	76.27
	All	73.51	73.51

Table 6: Test set F$_1$ on NER for entities seen and unseen in the training data for BERT$_{large}$ and ProcBERT, when the two achieve very similar overall performance under the same budget constraints in Figure 3. ProcBERT performs better on the unseen entities.

nomical choice at lower budgets.[18]

When using the same budget and achieving similar F$_1$, how do pre-training and annotation differ? In the previous experiments, we show that in-domain pre-training is an effective domain adaptation method especially in high-budget settings. ProcBERT can work very well when trained with less labeled data. A plausible explanation is that in-domain pre-training improves generalization to new entities in the target domain, whereas additional annotation improves the performance on entities that are observed in the training corpus. To evaluate this hypothesis, we compare model predictions of the two strategies at the crossover points in Figure 3[19], and consider each entity in the test set as "Seen" or "Unseen" based on whether it was observed in the training set. Then, we calculate the F$_1$ score for each category as shown in Table 6. Although the models that are compared achieve nearly identical overall performance, the decomposition of performance on seen and unseen entities in Table 6 clearly suggests that in-domain pre-training leads to better generalization on unseen entities, whereas allocating more budget to annotation boosts performance on entities that were seen during training. This may help provide an explanation for the main finding of this paper: in-domain pre-training results in better generalization on unseen mentions, leading to better marginal utility, but only after enough in-domain annotations are observed to fully cover the head of the distribution.

6 Positive Externalities of Pretraining

So far, we have discussed domain adaptation as a consumer choice problem where annotation and pre-training costs are balanced to maximize performance in a target domain. However, pre-training on large quantities of natural language instructions can improve performance on additional tasks in the procedural text domain, as demonstrated in the following subsections.

[18]For more discussion on our assumptions, see §2.

[19]We choose Figure 3 for this analysis instead of Figure 2 because we want to isolate the impact of source domain labeled data.

Model	X-WLP		ChEMU		Recipe	WLP		PubMedM&M		ChemSyn	
	Core	Non-Core	NER	EE	ET	NER	RE	NER	RE	NER	RE
BERT$_{base}$	$74.79_{0.6}$	$77.57_{0.6}$	$95.14_{0.1}$	$91.93_{0.6}$	$80.62_{0.7}$	$73.87_{0.4}$	$80.25_{0.4}$	$74.80_{0.7}$	$78.73_{0.9}$	$95.09_{0.2}$	$92.63_{0.2}$
BERT$_{large}$	$\underline{75.53_{1.7}}$	$76.77_{0.5}$	$95.10_{0.2}$	$92.10_{0.9}$	$81.53_{0.5}$	$74.97_{0.3}$	$81.39_{0.5}$	$77.06_{0.3}$	$78.44_{1.3}$	$95.26_{0.1}$	$92.87_{0.5}$
RoBERTa$_{base}$	$75.04_{0.8}$	$77.24_{0.6}$	$95.05_{0.1}$	$\mathbf{92.54}_{1.2}$	$83.41_{0.1}$	$74.97_{0.5}$	$80.94_{0.5}$	$76.21_{0.3}$	$78.95_{0.7}$	$95.30_{0.2}$	$93.39_{0.3}$
RoBERTa$_{large}$	$73.77_{1.6}$	$74.37_{0.2}$	$95.16_{0.2}$	$92.10_{1.1}$	$\mathbf{84.54}_{0.8}$	$\mathbf{76.37}_{0.5}$	$79.76_{0.3}$	$\mathbf{78.70}_{0.6}$	$75.66_{0.3}$	$95.66_{0.2}$	$92.87_{0.2}$
SciBERT	$75.48_{0.7}$	$\underline{78.51_{0.6}}$	$\underline{95.63_{0.0}}$	$91.80_{0.5}$	$81.75_{0.4}$	$75.89_{0.4}$	$81.29_{0.5}$	$\underline{77.81_{0.2}}$	$79.54_{0.9}$	$95.82_{0.2}$	$93.27_{0.2}$
BioMed-RoBERTa	$74.89_{0.6}$	$76.39_{0.8}$	$95.32_{0.2}$	$\underline{92.42_{0.6}}$	$82.92_{0.3}$	$75.55_{0.2}$	$\mathbf{81.56}_{0.4}$	$77.14_{0.1}$	$78.44_{2.0}$	$95.38_{0.2}$	$93.16_{0.3}$
Proc-RoBERTa	$74.76_{0.7}$	$76.12_{0.9}$	$95.49_{0.1}$	$91.55_{0.6}$	$\underline{84.19_{0.3}}$	$75.76_{0.4}$	$80.79_{0.9}$	$76.91_{0.7}$	$79.16_{0.9}$	$95.67_{0.1}$	$93.31_{0.2}$
ProcBERT	$\mathbf{76.73}_{0.9}$	$\mathbf{78.57}_{0.8}$	$\mathbf{96.19}_{\mathbf{0.1}}$	$92.32_{0.2}$	$84.10_{0.3}$	$\underline{76.04_{0.2}}$	$\underline{81.44_{0.4}}$	$77.31_{0.5}$	$\mathbf{80.19}_{0.6}$	$\mathbf{95.97}_{0.2}$	$\mathbf{93.57}_{0.2}$
SOTA	76.5	78.1	95.70	95.36	81.96	77.99	80.46	–	–	–	–

Table 7: Test set F_1 on six procedural text datasets. The best task performance is boldfaced, and the second-best performance is underlined. For the SOTA model of each dataset, we refer readers to the corresponding paper for further details: Tamari et al. (2021) for XWLP, Wang et al. (2020) for ChEMU, Gupta and Durrett (2019) for Recipe, Knafou et al. (2020) for NER on WLP, and Sohrab et al. (2020) for RE on WLP.

6.1 Ancillary Procedural NLP Tasks

In addition to the procedural text datasets discussed in §5, we experiment with three ancillary procedural text corpora, to explore how in-domain pretraining can benefit other tasks.

The **ChEMU corpus** (Nguyen et al., 2020) contains NER and event annotations for 1500 chemical reaction snippets collected from 170 English patents. Its NER task focuses on identifying chemical compounds, and its event extraction (EE) task aims at detecting chemical reaction events including trigger detection and argument role labeling.

The **XWLP corpus** (Tamari et al., 2021) provides the Process Event Graphs (PEG) of 279 wet-lab biochemistry protocols. The PEG is a document-level graph-based representation specifying the involved experimental objects.

The **Recipe corpus** (Kiddon et al., 2016) includes annotation of entity states for 866 cooking recipes. It supports Entity Tracking (ET) task which predicts whether or not a specific ingredient is involved in each step of the recipe.

6.2 Experiments on Ancillary Tasks

For ChEMU, gold arguments are provided, so we only need to identify the event trigger and predict the role of the gold arguments. An event prediction is correct if the event trigger, associated arguments, and their roles match with the gold event mention. We tackle this task using a pipeline model similar to Zhong and Chen (2020). For XWLP, we focus on the operation argument role labeling task, where gold entities are provided as input. Following Tamari et al. (2021), we decompose the results into "Core" and "Non-Core" roles. For the Recipe task, we follow the data splits and fine-tuning architecture of Gupta and Durrett (2019). The state of an ingredient in each cooking step is correct if it matches with the gold labels, as either present or absent.

Results. Test set results of eight pre-trained language models on six procedural text datasets are presented in Table 7.[20] ProcBERT, performs best in most tasks and even achieves the state-off-the-art performance on operational argument role labeling ("Core" and "Non-Core") of XWLP, showing the effectiveness of in-domain pre-training.

7 Conclusion

In this paper, we address a number of questions related to the costs of adapting an NLP model to a new domain (Blitzer et al., 2006; Han and Eisenstein, 2019), an important and well-studied problem in NLP. We frame domain adaptation under a constrained budget as a problem of consumer choice. Experiments are conducted using several pre-trained models in three procedural text domains to determine when it is economical to pre-train in-domain transformers (Gururangan et al., 2020), and when it is better to spend available resources on annotation. Our results suggest that when a small number of NLP models need to be adapted to a new domain, pre-training, by itself, is not an economical solution.

Acknowledgments

We are grateful to the anonymous reviewers for helpful feedback on an earlier draft of this paper. We also thank Vardaan Pahuja for assis-

[20]For ChEMU, we report the development set results because its test set is not publicly available.

tance with extracting experimental paragraphs from PubMed, and John Niekrasz for sharing the output of Daniel Lowe's reaction extraction tool on European patents. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001119C0108, in addition to the NSF (IIS-1845670) and IARPA via the BETTER program (2019-19051600004). The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense, IARPA or the U.S. Government. This work is approved for Public Release, Distribution Unlimited.

References

Gary S Becker. 1965. A Theory of the Allocation of Time. *The economic journal*, (299):493–517.

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.

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*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128.

Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Pradeep Dasigi, G. Burns, E. Hovy, and A. D. Waard. 2017. Experiment Segmentation in Scientific Discourse as Clause-level Structured Prediction using Recurrent Neural Networks. *ArXiv*.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic.

Hal Daume III and Daniel Marcu. 2006. Domain Adaptation for Statistical Classifiers. *Journal of artificial Intelligence research*.

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*, pages 4171–4186, Minneapolis, Minnesota.

Aditya Gupta and Greg Durrett. 2019. Effective Use of Transformer Networks for Entity Tracking. 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 759–769, Hong Kong, China.

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.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised Domain Adaptation of Contextualized Embeddings for Sequence Labeling. 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 4238–4248, Hong Kong, China.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. 2018. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *International Conference on Machine Learning (ICML)*.

Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to Train BERT with an Academic Budget. *arXiv*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally Coherent Text Generation with Neural Checklist Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly Easy Neural Domain Adaptation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Julien Knafou, Nona Naderi, Jenny Copara, Douglas Teodoro, and Patrick Ruch. 2020. BiTeM at WNUT 2020 Shared Task-1: Named Entity Recognition over Wet Lab Protocols using an Ensemble of Contextual Language Models. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 305–313, Online.

Kelvin J Lancaster. 1966. A New Approach to Consumer Theory. *Journal of political economy*, (2):132–157.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*.

Daniel M. Lowe. 2012. Extraction of Chemical Structures and Reactions from the Literature.

Javier Marín, A. Biswas, Ferda Ofli, Nick Hynes, A. Salvador, Yusuf Aytar, Ingmar Weber, and A. Torralba. 2021. Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 187–203.

Dat Quoc Nguyen, Zenan Zhai, Hiyori Yoshikawa, Biaoyan Fang, Christian Druckenbrodt, Camilo Thorne, Ralph Hoessel, S. Akhondi, Trevor Cohn, Timothy Baldwin, and K. Verspoor. 2020. ChEMU: Named Entity Recognition and Event Extraction of Chemical Reactions from Patents. *Advances in Information Retrieval*, pages 572 – 579.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training. *arXiv*.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge Enhanced Contextual Word Representations. 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 43–54, Hong Kong, China.

Roy Schwartz, Jesse Dodge, N. A. Smith, and Oren Etzioni. 2020. Green AI. *Communications of the ACM*, pages 54 – 63.

Mohammad Golam Sohrab, Anh-Khoa Duong Nguyen, Makoto Miwa, and Hiroya Takamura. 2020. mgsohrab at WNUT 2020 Shared Task-1: Neural Exhaustive Approach for Entity and Relation Recognition Over Wet Lab Protocols. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 290–298, Online.

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.

Jeniya Tabassum, Wei Xu, and Alan Ritter. 2020. WNUT-2020 Task 1 Overview: Extracting Entities and Relations from Wet Lab Protocols. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 260–267, Online.

Ronen Tamari, Fan Bai, Alan Ritter, and Gabriel Stanovsky. 2021. Process-Level Representation of Scientific Protocols with Interactive Annotation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2190–2202, Online.

Jingqi Wang, Yuankai Ren, Zhi Zhang, and Yaoyun Zhang. 2020. Melaxtech: A Report for CLEF 2020 - ChEMU Task of Chemical Reaction Extraction from Patent. In *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*, volume 2696 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Dustin Wright and Isabelle Augenstein. 2020. Transformer Based Multi-Source Domain Adaptation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7963–7974, Online.

Zexuan Zhong and Danqi Chen. 2020. A Frustratingly Easy Approach for Joint Entity and Relation Extraction. *ArXiv*.

A Data Annotation

We annotate two datasets PUBMEDM&M and CHEMSYN in the domain of scientific articles and chemical patents mainly following the annotation scheme of the Wet Lab Protocols (WLP; Tabassum et al., 2020). On top of 20 entity types and 16 relation types in WLP, we supplement four entity types (COMPANY, SOFTWARE, DATA-COLLECTION and INFO-TYPE) and one relation type (BELONG-TO) due to two key features of our corpus: 1) scientific articles usually specify the provenance of reagents for better reproducibility; 2) it covers a broader range of procedures such as computer simulation and data analysis.

We recruit four undergraduate students to annotate the datasets using the BRAT annotation tool.[21] We double-annotate all files in PUBMEDM&M and half of the files in CHEMSYN. For those double-annotated files, the coordinator will discuss the annotation with each annotator making sure their annotation follows the guideline and dissolve the disagreement. As for the inter-annotator agreement (IAA) score, we treat the annotation from one of the two annotator as the gold label, and the other annotation as the predicted label, and then use the F1 scores of Entity(Action) and Relation evaluations as the final inter-annotator agreement scores, which are shown in Table 8. We can see that CHEMSYN has higher IAA scores, and there are two potential reasons: 1) we annotate PUBMEDM&M first, so the annotators might be more experienced when they annotate CHEMSYN; 2) PUBMEDM&M contains more diverse content like wet lab experiments or computer simulation procedures while CHEMSYN is mainly about chemical synthesis.

Dataset	Entities/Actions	Relations
PUBMEDM&M	70.54	51.94
CHEMSYN	79.87	87.20

Table 8: Inter-Annotator Agreement (F_1 scores on Entity/Action Identification and Relation Extraction).

B Procedural Corpus Collection

PubMed Articles. The first source of our procedural corpus is PubMed articles because they contain a large number of freely accessible experimental procedures. Specifically, we extract procedural paragraphs from the *Materials and Methods* section of articles within the Open Access Subset of PubMed. XML files containing full text of articles are downloaded from NCBI[22] and then processed to obtain all the paragraphs within the *Materials and Methods* section.

To improve the quality of our collected corpus, we develop a procedural paragraph extractor by fine-tuning SciBERT (Beltagy et al., 2019) on the SciSeg dataset (Dasigi et al., 2017), which includes discourse labels (`{Goal, Fact, Result, Hypothesis, Method, Problem, Implication}`) for PubMed articles. This extractor achieves an average F1 score of 72.65% in a five-fold cross validation, and we run it on all acquired paragraphs. We consider a paragraph as a valid procedure if at least 40% of clauses are labeled as `Method`. This threshold is obtained by manual inspection of the randomly sampled subset of the data.

In total, the PubMed Open Access Subset contains 2,542,736 articles, of which about 680k contain a *Materials and Methods* section. After running our trained procedural paragraph extractor, we retain a set of 1,785,923 procedural paragraphs. Based on a manual inspection of the extracted paragraphs, we estimate that 92% consist of instructions for carrying out experimental procedures.

Chemical Patents. The second source of our corpus is the patent data because chemical patents usually include detailed procedures of chemical synthesis. We download U.S. patent data (1976-2016) from USPTO[23] and European data (1978-2020) from EPO[24] as XML files. Then we apply the reaction extractor developed by Lowe (2012), a trained Naive Bayes classifier, to the *Description* section of our collected patents. Note that the U.S. patent data has two subsets, "Grant" (1976-2016) and "Application" (2001-2016). The "Application" subset covers the "Grant" subset from the same year, so for those overlapping years (2001-2016), we only use the U.S. patents from the "Application" subset. As a result, we get 2,435,999 paragraphs from 174,554 U.S. patents and 4,039,6065 paragraphs from 129,035 European patents. Lastly, we use

[21] https://github.com/nlplab/brat

[22] https://www.ncbi.nlm.nih.gov/

[23] https://www.uspto.gov/learning-and-resources/bulk-data-products

[24] https://www.epo.org/searching-for-patents/data/bulk-data-sets.html

the language identification tool **langid**[25] to build an English-only corpus, which includes 6,107,481 paragraphs.

C Pre-training Details

We pre-train ProcBERT using the TensorFlow codebase of BERT (Devlin et al., 2019).[26] We use the Adam optimizer (Kingma and Ba, 2015) with β_1 = 0.9, β_2 = 0.999 and $L2$ weight decay of 0.01. Following Devlin et al. (2019), we deploy the two-step regime. In the first step, we pre-train the model with sequence length 128 and batch size 512 for 1 million steps. The learning rate is warmed up over the first 100k steps to a peak value of 1e-4, then linearly decayed. In the second step, we train 100k more steps of sequence length 512 and batch size 256 to learn the positional embeddings with peak learning rate 2e-5. We use the original sub-word mask as the masking strategy, and we mask 15% of tokens in the sequence for both training steps.

For Proc-RoBERTa, we use the codebase from AI2,[27] which enables language model pre-training on TPUs with PyTorch. Similar to Gururangan et al. (2020), we train RoBERTa on our collected procedural text corpus for 12.5k steps with a learning rate of 3e-5 and an effective batch size 2048, which is achieved by accumulating the gradient of 128 steps with a basic batch size of 16. The input sequence length is 512 throughout the whole process, and 15% of words are masked for prediction.

D Hyper-parameters for Downstream Tasks

We use the same five random seeds as Peters et al. (2019) for all our experiments in §5 and §6.[28] We select the best hyperparameter values based on the average development set performances over five random seeds by grid search. For models with BERT$_{base}$ or RoBERTa$_{base}$ architecture, the search range includes learning rate (1e-5, 2e-5), batch size (16, 48, 64, 128), max sequence length (128, 256, 512) and epoch number (5, 20, 60), and the used hyperparameter values on budget-constrained domain adaptation experiments (denoted as "BUDGET") (§5) and ancillary tasks (§6) are shown in Table 9. For BERT$_{large}$ and RoBERTa$_{large}$, the search range is different in learning rate (5e-6, 1e-5), batch size (4, 8, 12, 24, 64) and epoch number (3, 5, 10, 20), and the used values are shown in Table 10.

[25]https://github.com/saffsd/langid.py

[26]https://github.com/google-research/bert

[27]https://github.com/allenai/tpu_pretrain

[28]https://github.com/allenai/kb/blob/master/bin/run_hyperparameter_seeds.sh

Hyperparam.	BUDGET		X-WLP	CHEMU		RECIPE	WLP		PUBMEDM&M		CHEMSYN	
	NER	RE	ARL	NER	EE	ET	NER	RE	NER	RE	NER	RE
Learning Rate	1e-5	1e-5	1e-5	1e-5	1e-5	2e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5
Batch Size	16	48	16	16	16	16	16	128	16	64	16	48
Max Seq. Length	512	256	512	512	512	512	256	128	512	256	512	256
Max Epoch	20	5	5	20	5	20	20	5	60	5	20	5

Table 9: Hyperparameters for models with BERT$_{base}$ or RoBERTa$_{base}$ architecture on budget-constrained domain adaptation experiments (denoted as "BUDGET") (§5) and ancillary tasks (§6).

Hyperparam.	BUDGET		X-WLP	CHEMU		RECIPE	WLP		PUBMEDM&M		CHEMSYN	
	NER	RE	ARL	NER	EE	ET	NER	RE	NER	RE	NER	RE
Learning Rate	1e-5	5e-6	5e-6	1e-5	5e-6	5e-6	1e-5	5e-6	1e-5	5e-6	1e-5	5e-6
Batch Size	4	12	4	4	4	4	4	64	4	24	4	12
Max Seq. Length	512	256	512	512	512	512	256	128	512	256	512	256
Max Epoch	10	3	3	10	3	5	10	3	20	5	10	3

Table 10: Hyperparameters for BERT$_{large}$ and RoBERTa$_{large}$ on budget-constrained domain adaptation experiments (§5) and ancillary tasks (§6).