Pretrain or Annotate? Domain Adaptation with a Constrained Budget

Fan Bai, Alan Ritter, Wei Xu
School of Interactive Computing
College of Computing
Georgia Institute of Technology

Abstract

Recent work has demonstrated that pretraining in-domain transformers can boost performance when adapting to a new domain. However, the costs associated with pre-training raise an important question: given a fixed budget to improve a model's performance, what steps should an NLP practitioner take to maximize performance? In this paper, we study how to adapt to a new domain under budget constraints, and approach it as a customer choice problem between data annotation and pretraining. Specifically, we measured the annotation cost of three procedural text-domain datasets and the pretraining cost of three in-domain language models. Then we evaluated the utility of different combinations of pretraining and data annotation under varying budget constraints to see which combination strategy works best. For small budgets, we find that spending all funds on annotation leads to the best performance, while a combination of data annotation and in-domain pretraining becomes optimal once the budget becomes large enough. Therefore, we suggest that in-domain data annotation should be part of an economical strategy when adapting an NLP model to a new domain.

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; Ganin and Lempitsky, 2015; Xie et al., 2020). Recent work has shown that pretraining in-domain transformers is an effective method for unsupervised adaptation (Han and Eisenstein, 2019) and even boosts performance when large quantities of in-domain data are available (Gururangan et al., 2020). However, modern pretraining methods incur substantial computational costs. 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 data, while on the other they could buy or rent GPUs or TPUs to pretrain large in-domain models using unlabeled data. In this paper, we empirically study the best strategy for adapting to a new domain given a fixed budget: is it better to spend your funds pretraining transformers, annotating supervised data, or a combination of the two?

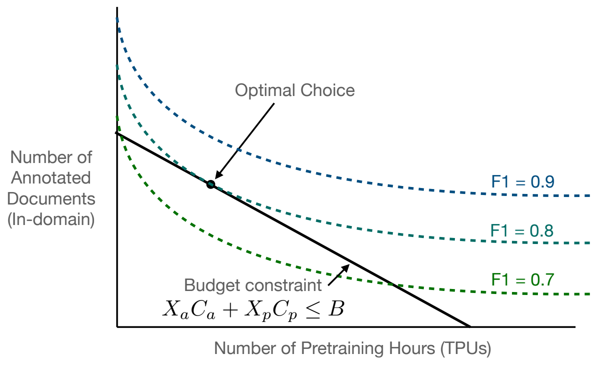

Figure 1: We view domain adaptation as a consumer choice problem, a classical problem in microeconomics. The NLP practitioner (consumer) is faced with the problem of choosing an optimal combination of annotation and pretraining under a constrained budget.

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 purchase X_a annotated documents (by hiring annotators) at a cost of C_a each, and X_p hours of pretraining (by renting GPUs or TPUs) at a cost of C_p. Given a fixed budget B, the consumer can choose any combination that fits within the budget constraint $X_aC_a + X_pC_P \leq B$. The problem faced by the consumer is then to choose an optimal combination, so as to maximize utility. We assume the utility function, $U(X_a, X_p)$ is based on an appropriate performance metric, for example, F_1 score, measured after pretraining for X_p hours and then fine-tuning on X_a annotated in-domain documents.

To empirically measure C_a, the cost of annotation, we hired annotators to manually label in-domain documents for supervised fine-tuning in 3 procedural text domains (wet-lab protocols, experimental paragraphs extracted from journal articles, and chemical synthesis procedures described in patents). We also measured the costs of in-domain pretraining, C_p, on a large collection of unlabeled procedural texts using Google's cloud TPUs. The utility of different combinations of pretraining, and annotation, $U(X_a, X_p)$, was then empirically evaluated using varying budget constraints in $\binom{3}{2} = 6$ (source $\rightarrow$ target) combinations, as described in §5.

For small budgets, we find that spending all available funds on annotation is the optimal strategy, however, as more funding becomes available, a combination of pretraining and annotation appears to be the best choice. Our results suggest that given the current costs of pretraining 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 an NLP model to a new domain.

2 Assumptions

Our analysis makes a number of important assumptions. First, we assume the goal is to adapt a single NLP system to a single target domain. This means that positive externalities due to additional tasks, or similar domains that can benefit from the availability of new pretrained models are not considered. While the entire NLP community can benefit from the availability of pretrained models in new domains, we assume that the NLP practitioner has been assigned the task of adapting an existing model to a single new domain, maximizing performance within a constrained budget.[1]

Secondly, we assume the NLP engineer's salary is a fixed cost; in other words: their salary will be paid no matter what, therefore it doesn't matter whether they spend their time pretraining models or managing a group of annotators.[2] The costs of annotators and computation (GPUs/TPUs) are the only prospective costs that are considered and the goal is to maximize performance.

Third, we assume the total wallclock time to obtain the adapted model is not an important factor. Or, if the timeline is an important factor, we assume the annotation process can be sped up simply by hiring more annotators, so that pretraining and annotation can be completed within the same timeframe.

Finally, we assume BERT is the only available zero-cost pretrained model that is initially available. While there do exist many pretrained transformers in a variety of specialized domains,[3] for example, SciBERT (Beltagy et al., 2019) and Biomed-RoBERTa (Gururangan et al., 2020), our experiments are designed to simulate a scenario where no domain-specific model is initially available in the target domain.

3 Estimating Annotation Cost C_a

In this section, we introduce our cost estimation for three procedural text-domain datasets to enable the research of consumer choice problems in NLP domain adaptation.

3.1 Procedural Text Domain Dataset

We worked on three procedural text-domain datasets. The first one is Wet Lab Protocols (WLP) (Tabassum et al., 2020). It includes 726 wet lab experiment guidelines collected from `protocols.io`. Inspired by WLP, we built two new procedural text-domain datasets following the same annotation scheme. One of the two datasets is PUBMEDM&M, which consists of 191 double-annotated experimental paragraphs extracted from the *Materials and Methods* section of PubMed articles. Unlike the succinct language style in user-generated WLP, PUBMEDM&M's text uses academic language because it stems from published research papers (for an example, see Table 1). The last dataset is CHEMSYN, which includes 596 chemical synthesis procedures described in patents, but only 300 of them are double-annotated due to the dataset size. The main difference between CHEMSYN and the other two datasets is that CHEMSYN has a focused topic on chemical synthesis procedures while the other two datasets

[1]We present an additional analysis of positive externalities to other tasks that can benefit from a new pretrained model in §6.

[2]Based on our experience, pretraining in-domain models and managing a team of annotators are roughly comparable in terms of effort.

[3]`https://huggingface.co/models`

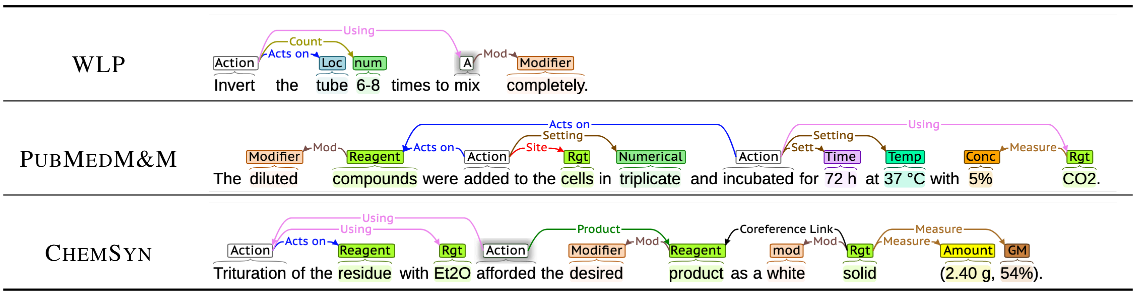
WLP
Invert the tube 6-8 times to mix completely.
PUBMEDM&M
The diluted compounds were added to the cells in triplicate and incubated for 72 h at 37 °C with 5% CO2.
CHEMSYN
Trituration of the residue with Et2O afforded the desired product as a white solid (2.40 g, 54%).

Table 1: Examples of three procedural text-domain datasets. WLP (Wet Lab Protocols) (Tabassum et al., 2020) consists of 726 user-generated wet lab experiment guidelines collected from `protocols.io`. PUBMEDM&M includes 191 experimental paragraphs extracted from PubMed papers. CHEMSYN contains 596 patent-based chemical synthesis procedures.

cover a broader range of procedures, i.e., some of the protocols in WLP are about nCoV-2019 (COVID-19) sequencing. The statistics of the three datasets are shown in Table 2. More details of data pre-processing, data annotation and inter-annotator agreement scores can be found in Appendix A.

3.2 Annotation Cost

Our annotation cost mainly came from the salary of in-house annotators, who were paid 13 USD / hour throughout the whole annotation process. For **CHEMSYN**, the overall annotation expense was roughly 5,000 USD, where each file had 8.40 sentences on average, and each sentence took around 1.83 minutes to annotate. For **PUBMEDM&M**, the average annotation speed was 2.35 minutes per sentence, and the overall annotation expense was about 1,730 USD. For **WLP**, although we did not participate in its annotation process, we estimated the annotation cost. We asked our in-house annotators to annotate randomly sampled WLP files so that we could calculate the average annotation speed, which was 1.70 minutes per sentence. Thus, the annotation cost of WLP was about 7,820 USD in our estimation.

4 Estimating Pre-training Cost C_p

In the previous section, we showed how we estimate the annotation cost C_a for three procedural text datasets. In this section, we measure the cost of in-domain pre-training C_p. Specifically, we explore two popular pre-training methods, 1) pre-training an in-domain language model from scratch; 2) domain-adaptive pre-training an off-the-shelf language model. We start with the collection of pre-training corpus in §4.1. Then in §4.2 we lay out the model configuration and implementation choices of the two pre-training methods. We close this section by showing the cost (§4.3) and carbon footprint (§4.4) of three in-domain pre-trained language models.

4.1 Corpus Collection

We collected unlabeled procedural text corpus from the same domains as the annotated data in §3. For PUBMEDM&M, we trained a procedural paragraph extractor by fine-tuning SciBERT (Beltagy et al., 2019) on the SciSeg dataset (Dasigi et al., 2017), and then applied the trained extractor to the *Materials and Methods* section of over 680k PubMed articles to obtain procedural paragraphs. Similarly, for CHEMSYN, we applied the reaction extractor developed by Lowe (2012), which is a trained Naive Bayes classifier, to the *Description* section of our collected patent corpus[4]. More details of extracting procedural paragraphs from PubMed articles and chemical patents can be found in Appendix B. Moreover, as food recipes are an important domain for research about procedural text understanding, we added the text component of the Recipe1M+ dataset (Marín et al., 2019) into the pre-training corpus, which is the largest recipe dataset available. As a result, our collected corpus PROCEDURE includes around 1.1 billion words in total, and we show its statistics together with the statistics of other corpora used in the language model pre-training in Table 3. The size of our corpus is smaller compared to other corpora, espe-

[4]We did not add wet-lab protocol data into the pre-training corpus mainly because: 1) it is too small compared to other sources as at that time there were only around 8000 available protocols (2.5 million tokens) on `protocols.io`[5] while the PubMed articles we collected has over two billion tokens; 2) over 700 high-quality wet-lab experiment-related protocols had already been used for the annotation of the WLP dataset (Tabassum et al., 2020).

Dataset	Domain	Task	#Files	#Sentences	#Cases	File split	Cost ($)	Price ($/sent.)
WLP (Tabassum et al., 2020)	biology	NER RE	726	17,658	185,313 124,803	492/123/111	7,820	0.44
PUBMEDM&M (this work)	biomed.	NER RE	191	1,699	12,131 8,987	100/41/50	1,730	1.02
CHEMSYN (this work)	chem.	NER RE	992	8,331	53,423 46,878	793/99/100	5,000	0.60

Table 2: Statistics of three procedural text-domain datasets.

Corpus	# Tokens	Text Size	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	-
- PubMed	0.32B	2.0GB	-
- Chem. patent	0.61B	3.9GB	-
- Recipe	0.11B	0.6GB	-
PROCEDURE+	12B	77GB	-
- Procedural text	6.3B	39GB	-
- Full articles	5.7B	38GB	-

Table 3: The statistics of the pre-training corpora.

cially for those used to pre-train a language model from scratch, so we created an extended version PROCEDURE+, where we up-sampled the procedural paragraphs 6 times and combined them with the original full text of PubMed articles and chemical patents to make sure at least half of the data is procedural text, leading to a larger corpus containing around 12 billion words.

4.2 Pre-training Process

In this section, we outline the details of pre-training two procedural-text-specific language models: 1) ProcBERT, a language model pre-trained from scratch using our procedural corpus PROCEDURE+; 2) Proc-RoBERTa, an in-domain RoBERTa-based language model fine-tuned with the more focused procedural corpus PROCEDURE.

4.2.1 Configuration and Training Objective

For ProcBERT, we used the configuration BERT-base, which has 12 attention layers with 12 attention heads in each layer and the hidden dimension of 768. The training of ProcBERT involves two objectives, Masked Language Model (MLM) and Next Sentence Prediction (NSP). For Proc-RoBERTa, we continued pre-training RoBERTa-base using our collected procedural text following Gururangan et al. (2020). In this case, Proc-RoBERTa had the same configuration as ProcBERT, but its training only required the MLM objective.

4.2.2 Implementation

We pre-trained ProcBERT using the TensorFlow codebase of BERT[6] (Devlin et al., 2019) from Google Research. We used the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $L2$ weight decay of 0.01. Following Devlin et al. (2019), we deployed the two-step regime to speed up our pre-training. In the first step, we pre-trained the model with sequence length 128 and batch size 512 for 1 million steps. The learning rate was warmed up over the first 100k steps to a peak value of 1e-4, then linearly decayed. In the second step, we trained other 100k steps of sequence 512 and batch size 256 to learn the positional embeddings with peak learning rate 2e-5. We used the original sub-word mask as the masking strategy, and we masked 15% of tokens in the sequence for both training steps.

For Proc-RoBERTa, we used the codebase of AI2[7], which enables language model pre-training on TPU with PyTorch. Similar to Gururangan et al. (2020), we trained RoBERTa on our collected procedural text for 12.5k steps with a learning rate of 3e-5 and an effective batch size 2048, which was achieved by accumulating the gradient of 128 steps with a basic batch size of 16. The input sequence length was 512 throughout the whole process, and 15% of words were masked for prediction as well.

4.3 Pre-training Cost

All our pre-training experiments were conducted on Google Cloud Platform (GCP) using on-demand 8-core v3 TPUs. For **ProcBERT**, as introduced in Section 4.2.2, the whole training process consisted of two stages. In the first stage, the model was trained for 1 million steps with an approximate speed of 4.71 step/second. In the second stage, the model was further trained on the longer input sequence for 100k steps with a speed of 1.83

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

[7]`https://github.com/allenai/tpu_pretrain`

Consumption	CO_2e (kg)
Air travel, one person, SF↔NY	1200[11]
Pre-training	
ProcBERT	87.4
Proc-RoBERTa	112.1
SciBERT	198.3

Table 4: The carbon footprint of three in-domain pre-trained language models.

step/second. Therefore, the training took around 74 hours in total, and the cost on TPU was around 590 USD as the price of an on-demand TPU v3 was 8 USD / hour[8]. After including auxiliary expenses on other cost items like virtual machines and data storage, the total cost for the pre-training of ProcBERT was around 620 USD[9]. For **Proc-RoBERTa**, following Gururangan et al. (2020), we fine-tuned RoBERTa on our pre-training corpus PROCEDURE for 12.5k steps with the averaged speed of 27.27 seconds per step, which led to a TPU time of 95 hours[10]. Thus, the total cost of Proc-RoBERTa was around 800 USD after including the auxiliary expenses. To extend the scope of our research, we estimated the cost of two off-the-shelf pre-trained language models BERT (Devlin et al., 2019) and SciBERT (Beltagy et al., 2019). **BERT**, as assumed in §2, is an initially available pre-trained model, so the total cost is 0 USD. **SciBERT** uses TPU v3 with 8 cores, and its training is a two-stage process similar to ProcBERT. The overall training of SciBERT takes 7 days (5 days for the first stage and 2 days for the second stage) and costs 1,340 USD in terms of TPU usage. Because we don't know the details of how Beltagy et al. (2019) set up the virtual machines and because this part of the expense is relatively trivial compared to TPU cost, we use 1,340 USD as the final cost of SciBERT.

4.4 Carbon Footprint

Apart from the financial cost (§4.3), we also estimate the carbon footprint of each in-domain pre-trained language model to measure its environmental impact. We first calculate the energy consumption in kilowatt-hours (KWh) following Patterson et al. (2021):

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

where H is the training hours, N denotes the number of processors we used, P means the average power per processor[12], 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, the average trailing-twelve-month PUE for all Google data centers in Q4 2019[13]. 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[14]. For example, ProcBERT was per-trained on a TPU v3-8 (8 processors) for 74 hours, so its CO_2 emission was $(74 \times 8 \times 283 \times 1.10/1000) \times 474/1000 = 87.4$ kg. The estimated CO_2 emissions of three in-domain language models are shown in Table 4.

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

As we now know the annotation cost C_a (§3) and pre-training cost C_p (§4), in the next step we evaluate the utility of combining annotation and pre-training $U(X_a, X_p)$ to find an optimal strategy that adapts to a new domain under budget constraints. In this section, we first introduce the two involved tasks in §5.1. Then we describe how we use labeled data from both source and target domains via the supervised domain adaptation framework EasyAdapt in §5.2. Lastly, we show our experimental setup for the utility measurement (§5.3) and give a detailed analysis of the measured utilities under different budgets (§5.4).

[8] `https://cloud.google.com/tpu/pricing`

[9] The type of virtual machine we used for all pre-training experiments was `n1-standard-8`, which has 8 virtual CPUs with 3.75 GB of memory per CPU and costs 213.8 USD per month. However, we later found that the pre-training of ProcBERT could have used cheaper machines like `n1-standard-1` because its TensorFlow code is less CPU-demanding.

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

[12] 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.

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

[14] Our models were pre-trained in the data center Goolge Netherlands.

Source ⇒ Target Domain	**Budget**	**BERT-base** \$0		**BERT-large** \$0		**ProcBERT** \$620		**Proc-RoBERTa** \$800		**SciBERT** \$1340	
		F1	(#sent)	F1	(#sent)	F1	(#sent)	F1	(#sent)	F1	(#sent)
PUBMEDM&M⇒CHEMSYN	\$700	$92.58_{0.2}$	(1166)	$\mathbf{93.32}_{0.2}^{\dagger}$	(1166)	$90.96_{0.4}^{\dagger}$	(133)	NA		NA	
	\$1500	$93.32_{0.2}$	(2500)	$93.94_{0.4}^{\dagger}$	(2500)	$\mathbf{94.68}_{0.2}$	(1466)	$93.60_{0.3}^{\dagger}$	(1166)	$92.44_{0.3}^{\dagger}$	(266)
	\$2300	$94.49_{0.2}$	(3833)	$94.79_{0.2}$	(3833)	$\mathbf{95.11}_{0.2}$	(2800)	$94.89_{0.2}$	(2500)	$94.22_{0.2}$	(1600)
WLP⇒CHEMSYN	\$700	$92.94_{0.2}^{\dagger}$	(1166)	$\mathbf{93.06}_{0.1}^{\dagger}$	(1166)	$90.40_{0.3}^{\dagger}$	(133)	NA		NA	
	\$1500	$93.32_{0.2}$	(2500)	$94.23_{0.1}^{\dagger}$	(2500)	$\mathbf{94.73}_{0.2}^{\dagger}$	(1466)	$94.19_{0.1}^{\dagger}$	(1166)	$92.09_{0.3}^{\dagger}$	(266)
	\$2300	$94.49_{0.2}$	(3833)	$94.79_{0.3}^{\dagger}$	(3833)	$\mathbf{95.37}_{0.1}^{\dagger}$	(2800)	$94.82_{0.4}^{\dagger}$	(2500)	$94.22_{0.2}$	(1600)
WLP⇒PUBMEDM&M	\$700	$71.88_{0.4}$	(686)	$\mathbf{72.93}_{0.5}^{\dagger}$	(686)	$63.80_{0.7}^{\dagger}$	(78)	NA		NA	
	\$1500	NA		NA		$\mathbf{74.57}_{0.3}^{\dagger}$	(862)	$73.75_{0.1}$	(686)	$69.94_{0.4}^{\dagger}$	(156)
	\$2300	NA		NA		NA		NA		NA	
CHEMSYN⇒PUBMEDM&M	\$700	$73.14_{0.5}^{\dagger}$	(686)	$\mathbf{74.24}_{0.6}^{\dagger}$	(686)	$71.16_{0.4}^{\dagger}$	(78)	NA		NA	
	\$1500	NA		NA		$\mathbf{75.81}_{0.7}^{\dagger}$	(862)	$74.78_{0.4}^{\dagger}$	(686)	$72.87_{0.3}^{\dagger}$	(156)
	\$2300	NA		NA		NA		NA		NA	
CHEMSYN⇒WLP	\$700	$72.91_{0.3}^{\dagger}$	(1590)	$\mathbf{73.30}_{0.3}^{\dagger}$	(1590)	$72.69_{0.2}^{\dagger}$	(181)	NA		NA	
	\$1500	$73.69_{0.1}^{\dagger}$	(3409)	$73.88_{0.6}^{\dagger}$	(3409)	$\mathbf{75.21}_{0.2}^{\dagger}$	(2000)	$74.24_{0.3}^{\dagger}$	(1590)	$72.52_{0.9}^{\dagger}$	(363)
	\$2300	$73.85_{0.4}^{\dagger}$	(5227)	$74.23_{0.5}^{\dagger}$	(5227)	$\mathbf{75.51}_{0.4}^{\dagger}$	(3818)	$74.76_{0.3}^{\dagger}$	(3409)	$74.78_{0.5}^{\dagger}$	(2181)
PUBMEDM&M⇒WLP	\$700	$73.21_{0.1}^{\dagger}$	(1590)	$\mathbf{73.43}_{0.3}^{\dagger}$	(1590)	$72.85_{0.1}^{\dagger}$	(181)	NA		NA	
	\$1500	$73.78_{0.3}^{\dagger}$	(3409)	$74.18_{0.4}^{\dagger}$	(3409)	$\mathbf{75.36}_{0.1}^{\dagger}$	(2000)	$73.93_{0.3}^{\dagger}$	(1590)	$72.89_{0.3}^{\dagger}$	(363)
	\$2300	$74.09_{0.2}^{\dagger}$	(5227)	$74.32_{0.3}^{\dagger}$	(5227)	$\mathbf{75.73}_{0.3}^{\dagger}$	(3818)	$75.08_{0.3}^{\dagger}$	(3409)	$74.72_{0.3}^{\dagger}$	(2181)

Table 5: Experiment results for NER. With higher budgets (\$1500 and \$2300), our in-domain pre-training of ProcBERT can achieve the best results in combination with data annotation. For a smaller budget (\$700), investing all funds in data annotation and using the standard BERT-base and 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 (§5.2), where additional source domain data helps.

Source ⇒ Target Domain	Budget	BERT-base \$0		BERT-large \$0		ProcBERT \$620		Proc-RoBERTa \$800		SciBERT \$1340	
		F1	(#sent)	F1	(#sent)	F1	(#sent)	F1	(#sent)	F1	(#sent)
PUBMEDM&M⇒CHEMSYN	\$700	$90.12_{0.4}^{\dagger}$	(1166)	$90.71_{0.8}^{\dagger}$	(1166)	$88.21_{0.6}^{\dagger}$	(133)	NA		NA	
	\$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)	NA		NA	
	\$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)	NA		NA	
	\$1500	NA		NA		$80.10_{0.6}^{\dagger}$	(862)	$79.33_{1.3}^{\dagger}$	(686)	$75.70_{1.1}^{\dagger}$	(156)
	\$2300	NA		NA		NA		NA		NA	
CHEMSYN⇒PUBMEDM&M	\$700	$77.02_{0.6}^{\dagger}$	(686)	$77.31_{0.4}^{\dagger}$	(686)	$73.92_{0.8}^{\dagger}$	(78)	NA		NA	
	\$1500	NA		NA		$79.50_{0.7}^{\dagger}$	(862)	$77.12_{1.2}^{\dagger}$	(686)	$75.43_{0.6}^{\dagger}$	(156)
	\$2300	NA		NA		NA		NA		NA	
CHEMSYN⇒WLP	\$700	$78.81_{0.6}^{\dagger}$	(1590)	$79.80_{0.6}^{\dagger}$	(1590)	$78.97_{0.7}^{\dagger}$	(181)	NA		NA	
	\$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)	NA		NA	
	\$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 6: Experiment results for RE.

5.1 Tasks

We experimented on two tasks, Named Entity Recognition (NER) and Relation Extraction (RE), to evaluate the utility of different combination strategies. We used a different fine-tuning architecture for each task. For NER, we followed Devlin et al. (2019) to feed the final contextualized embedding of each token into a linear classification layer with softmax output. For RE, our architecture was similar to Soares et al. (2019), where four special tokens `[E1]`, `[\E1]`, `[E2]`, `[\E2]` were used to specify the positions of two entities, and the contextualized embeddings of `[E1]` and `[E2]` were concatenated as input to a final linear layer that was used to predict the relation.

In terms of the evaluation, a NER prediction is correct only if its label and boundary both match with the gold entity mention. For RE, the input is the gold entity mentions, and a relation prediction is correct when its label matches with the gold relation. We used F_1 score as the evaluation metric for both NER and RE.

5.2 EasyAdapt

In our setting, we had access to two labeled corpora from source and target domains, and we wanted to use both corpora to achieve the best target domain performance. Here we focused on the simple but effective feature augmentation framework EasyAdapt (Daumé III, 2007), which sets up a distinct feature space for each domain and creates a separate feature space shared across all domains, where the source domain knowledge can be transferred to the target domain.

The first step of EasyAdapt is to generate the feature representation of the input sequence. In this case, for NER, it was the hidden vector generated from the last layer of the pre-trained language model for each word. For RE, it was the concatenation of contextual embeddings of special tokens `[E1]` and `[E2]` as introduced in §5.1. Once we have the feature vector, we replicate it three times, where the first two vectors represent the source and target domain respectively, and the last vector represents the shared feature space. Given a data point from one specific domain (source or target), we concatenate the feature vector of its own domain and the feature vector of the shared domain as the final feature representation. In this last step, the feature representation is used as the input for a linear classification layer for NER and RE. This linear classification layer is created by concatenating two linear layers corresponding to the two selected feature vectors.

5.3 Experimental Setup

As we had three procedural text-domain datasets, we experimented with 6 (source $\rightarrow$ target) domain adaptation settings, where we used the data from both domains for training, and the data only from the target domain for evaluation. For each source-target pair, we tried out four combination strategies of pre-training and annotation with three different budgets to imitate the real-world situations, in which different NLP practitioners may have different amounts of resources. Given a budget, a target domain dataset and a pre-trained language model, we first calculated how much of the budget we could use for data annotation based on the pre-training cost (calculated in §4.3). Then we used the remaining budget to "buy" target domain annotated data, considering the price ($ /) listed in Table 2 (here we assume every sentence has the same cost). For instance, if the remaining budget is able to buy N annotated sentences, we will take the first N sentences from a set of ordered files, and truncate the last file to have exactly N sentences when necessary. However, if the pre-training cost went over the total budget, or the annotation cost went under the annotation budget, we left the result as "NA" (Not Applicable). Once we had the target domain training data ready, we trained two models: 1) a model that uses EasyAdapt to leverage the training data from both domains; 2) a model that only uses the target domain training data. Then we selected the model that had a better development set performance and reported its test set result. The main reason for creating the second model was that EasyAdapt is more effective in low-resource scenarios where the annotated target corpus is smaller. However, the annotated target dataset in some of the combination strategies was larger. In this case, utilizing the source domain data (with EasyAdapt) can hurt the performance because there is already sufficient data in the target domain to learn an efficient model.

5.4 Results

The test set NER results with four annotation-pre-training combination strategies under six domain adaptation settings are shown in Table 5. We can see that, for all six domain adaptation settings, the cost-free language models (BERT-base and BERT-large) work better than in-domain language models when the budget is 700 USD, showing the effectiveness of data annotation in low-resource scenarios. Using a larger model like BERT-large can further boost the performance. As the budget increases, the benefits of adding more labeled training data diminish for cost-free language models, and in-domain language models start to take the lead. Our language model ProcBERT, which is pre-trained from scratch on the self-collected procedural corpus while costing only 620 USD, performs the best at the budgets 1500 USD and 2300 USD (even out-

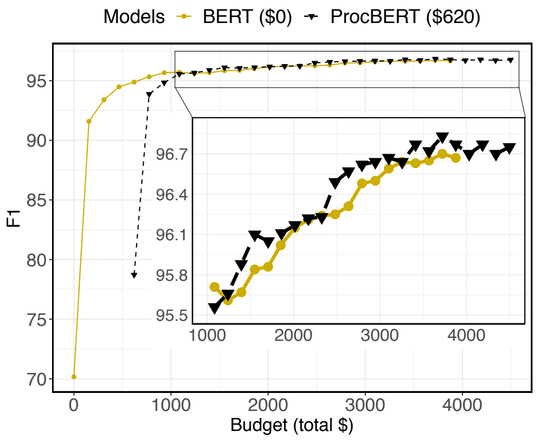

Figure 2: Results.

	Seen entities	Unseen entities	All
BERT ($0)	**97.96**	89.01	95.59
ProcBERT ($620)	97.57	**91.35**	**95.61**

Table 7: Comparison (F1) between annotation and pre-training.

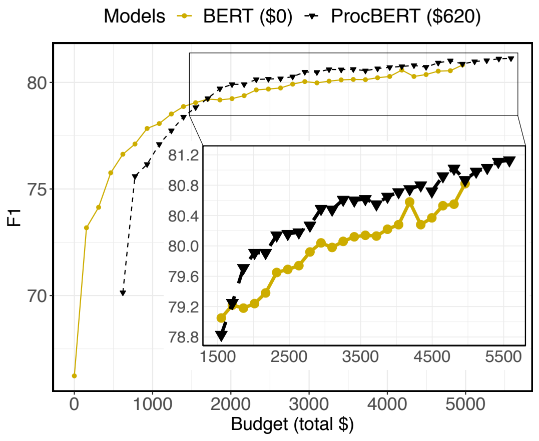

Figure 3: Results.

performing much larger BERT-large). This demonstrates that domain-specific pre-training will be more beneficial in high-resource settings where there is a reasonably large labeled dataset to learn the task-specific knowledge. Note that the advantage of ProcBERT over other in-domain language models in this budget-aware comparison partially comes from the fact that it has more training data due to its lower pre-training cost. Therefore, we compare all involved models in the same experimental setting in §6 to see how useful our procedural text-based pre-training is for procedural tasks.

Table 6 presents the test set RE results under six domain adaptation settings. Similarly, the domain-agnostic language model like BERT-base performs competitively at a lower budget while the domain-specific language model starts to excel as the budget increases. However, one thing we notice is that when the target domain dataset is WLP, our in-domain language model ProcBERT can perform on par or even better than BERT-base at the budget of 700 USD. There are two reasons for this result: 1) The cost of per sentence annotation for WLP is less than the cost of the other two datasets (Table 2), which gives us more in-domain labeled data compared to the other two datasets with the same budget; 2) Using EasyAdapt to take advantage of labeled data from other domains can alleviate the shortage of in-domain labeled data. This means, in practice, if we can find existing labeled data from domains close to our target domain, we can spend more money on the domain-specific pre-training over data annotation.

5.5 Analysis

In the previous section, we experimented with four different combination strategies for six distinct domain adaptation settings. We showed that annotating in-domain data is crucial to domain adaptation especially in low-resource scenarios while domain-specific pre-training leads to a greater performance gain under higher budgets. However, there are two important remaining questions: 1) What is the budget threshold that we should start to consider with in-domain pre-training when approaching a new domain? 2) Given these two styles of domain adaptation methods ("supervised" data annotation V.S. "unsupervised" pre-training), how do they impact the target task differently?

To explore the first question, we took the domain adaptation setting PUBMEDM&M⇒CHEMSYN as an example. We compared two combination strategies BERT-base and ProcBERT on NER. The budgets of these two strategies started with 0 USD and 620 USD respectively. We increased them 155 USD each time until their annotation portion reached a capacity of 1,850 USD, which is the annotation expense of the whole training set of CHEMSYN. The development set results are shown in Figure 3. We can see that BERT-base has a clear advantage over ProcBERT when the budget is less than 1000 USD while ProcBERT performs better starting from 1240 USD.

For the second question, we first searched for the budget where two strategies have the closest performances in Figure 3, and then decomposed their performances by whether the gold or predicted entity mentions exist in the training data (seen) or not (unseen). The decomposed results are presented in Table 7. We can see that ProcBERT has slightly worse performance on "seen" entities and performs much better on "unseen" entities. This signals that data annotation can provide more task-specific knowledge while in-domain pre-training can help improve the efficiency of learning task-specific knowledge from limited annotated data.

6 Positive Externalities on Other Tasks

So far, we have discussed the topic of domain adaptation as a consumer choice problem (between pre-training and annotation), and we experimented with multiple pre-trained language models to test the effectiveness of different combination strategies. In this section, we research the impact of pre-training on procedural tasks by comparing our pre-trained procedural text-domain language models with other off-the-shelf language models using three external procedural text datasets.

6.1 Datasets & Tasks

In this section, we experiment with 3 new procedural text datasets CHEMU, XWLP and RECIPE to reveal the effectiveness of our pre-trained language model.

The **CHEMU** corpus (Nguyen et al., 2020) provides NER and Event Extraction (EE) annotations for 1500 chemical reaction snippets collected from 170 English patents. Its NER task targets identifying chemical compounds in context, and its EE task involves detecting event triggers and identifying the roles of provided arguments in the events.

The **XWLP** (Tamari et al., 2021) dataset contains the Process Event Graphs (PEG) of 279 real-world wet-lab biochemistry protocols. The PEG is a document-level representation we need to extract, specifying the involved experimental objects (like reagents and equipment) and how they are operated in a sequence of lab procedure steps.

The **RECIPE** (Kiddon et al., 2016) dataset includes the annotation of ingredient states for 866 recipes. The task we need to tackle is called Entity Tracking (ET), where we need to predict whether a specific ingredient is used in each cooking step or not.

6.2 Evaluation

We follow the evaluation method of the respective paper for each new task. For CHEMU, following Nguyen et al. (2020), an event prediction is correct if its event trigger and argument semantic roles both match with the gold event mention. For XWLP, we evaluate the PEG extraction task by evaluating its three sub-tasks, which are operation classification (OC), argument role labeling (ARL) and temporal ordering (TO). The evaluation of OC is similar to NER while the evaluations of ARL and TO are similar to RE introduced in §5.1. For the entity tracking task in RECIPE, a state of ingredient in each cooking step is correct if it matches with the gold state (presence (1) or absence (0)).

6.3 Models

We use different model architectures for different tasks.

6.4 Implementation

We report averages and standard deviations across five random seeds for all our experiments.

7 Related Work

A number of recent studies have explored current and potential future negative externalities associated with the recent trend towards large-scale pre-training in NLP. These include the carbon emissions generated by training models with larger and larger computational costs (Strubell et al., 2019; Schwartz et al., 2020), and the increased barriers to entry created by reliance on specialized hardware (Bender et al., 2021). This line of work has called on the NLP community to evaluate not only a method's accuracy on held-out test data but also the amount of computation that is required during learning and inference.

8 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 framed the problem of domain adaptation under a constrained budget as a problem of consumer choice, a classic problem in microeconomics. Experiments were conducted using several pre-trained transformer-based models in three procedural text domains to determine when

	X-WLP		ChEMU		Recipe	WLP		PubMedM&M		ChemSyn	
Model Name	Core	Non-Core	NER	EE	ET	NER	RE	NER	RE	NER	RE
BERT-base			$94.34_{0.3}$		$80.62_{0.7}$	$74.33_{0.1}$	$80.25_{0.4}$	$70.71_{0.3}$	$78.73_{0.9}$	$95.08_{0.2}$	$92.63_{0.2}$
BERT-large			$94.67_{0.1}$			$74.68_{0.5}$	$81.39_{0.5}$	$72.42_{0.3}$		$95.12_{0.3}$	
RoBERTa-base			$94.63_{0.1}$		$83.41_{0.1}$	$75.29_{0.3}$	$80.94_{0.5}$	$72.20_{0.5}$	$78.95_{0.7}$	$95.08_{0.2}$	$93.39_{0.3}$
RoBERTa-large			$95.41_{0.1}$			$75.48_{0.6}$		$\mathbf{74.28_{0.7}}$		$95.30_{0.3}$	
SciBERT	$\mathbf{00.00_{0.0}}$	$\mathbf{00.00_{0.0}}$	$95.42_{0.2}$		-	$76.36_{0.3}$	$81.29_{0.5}$	$73.61_{0.3}$	$79.54_{0.9}$	$95.56_{0.2}$	$93.27_{0.2}$
BioMed-RoBERTa			$94.88_{0.3}$			$75.50_{0.3}$	$81.56_{0.4}$	$73.26_{0.4}$	$78.44_{2.0}$	$95.31_{0.2}$	$93.16_{0.3}$
Proc-RoBERTa			$95.52_{0.1}$		$84.19_{0.3}$	$75.57_{0.5}$	$80.79_{0.9}$	$72.91_{0.6}$	$79.16_{0.9}$	$95.64_{0.1}$	$93.31_{0.2}$
ProcBERT			$95.92_{0.2}$		$84.10_{0.3}$	$\mathbf{76.56_{0.5}}$	$81.44_{0.4}$	$73.75_{0.5}$	$80.19_{0.6}$	$\mathbf{95.67_{0.2}}$	$93.57_{0.2}$
SOTA	(Tamari et al., 2021)				81.96 (Gupta and Durrett, 2019)	77.99 (Knafou et al., 2020)	80.46 (Sohrab et al., 2020)	-	-	-	-

Table 8: Results.

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 pre-training, by itself, is never an economical solution in this setting. Furthermore, annotation has a much lower carbon footprint (Patterson et al., 2021), and will also create more jobs than if equivalent resources were allocated to pre-training.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001119C0108. 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 or the U.S. Government.

References

Gary S Becker. 1965. A theory of the allocation of time. *The economic journal*, 75(299):493–517.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *EMNLP/IJCNLP*.

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.

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*, abs/1702.05398.

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. Association for Computational Linguistics.

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

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR.

Aditya Gupta and Greg Durrett. 2019. Effective use of transformer networks for entity tracking. *ArXiv*, abs/1909.02635.

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 *ACL*.

X. Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *EMNLP/IJCNLP*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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. Association for Computational Linguistics.

Chaitanya Kulkarni, Wei Xu, Alan Ritter, and R. Machiraju. 2018. An annotated corpus for machine reading of instructions in wet lab protocols. In *NAACL-HLT*.

Kelvin J Lancaster. 1966. A new approach to consumer theory. *Journal of political economy*, 74(2):132–157.

Y. 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*, abs/1907.11692.

D. M. Lowe. 2012. Extraction of chemical structures and reactions from the literature.

J. Marín, A. Biswas, Ferda Ofli, Nicholas Hynes, A. Salvador, Y. Aytar, Ingmar Weber, and A. Torralba. 2019. Recipe1m: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE transactions on pattern analysis and machine intelligence*.

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*, 12036:572 – 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 preprint arXiv:2104.10350*.

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

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and T. Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *ACL*.

Mohammad Golam Sohrab, Anh-Khoa Duong Nguyen, Makoto Miwa, and H. Takamura. 2020. mgsohrab at wnut 2020 shared task-1: Neural exhaustive approach for entity and relation recognition over wet lab protocols. In *W-NUT@EMNLP*.

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

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. Association for Computational Linguistics.

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: Volume 1, Long Papers*. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc.

A Data Annoataion

To facilitate the comparison between the pre-training based approach and the supervised domain adap-tation approach, we created two new 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) framework (Kulkarni et al., 2018). In WLP, there are 18 Entity types (including the Action) and 14 Relation types. On top of it, we supplemented four new Entity types (COMPANY, SOFTWARE, COLLECTION(DATA) and INFO-TYPE) and one new 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) Other than wet-lab experiments, our corpus also includes the detail of data processing/analysis and computer-simulated experiments. For the PUBMEDM&M dataset, we annotated a corpus of 191 of the procedural text paragraphs extracted from a subset of PubMed articles. Similarly, for the CHEMSYN dataset, we annotated 600 sampled chemical synthesis-related paragraphs obtained from chemical patents. The statistics of these two datasets are presented in Table 9 and Table 10 respectively.

The annotation process is conducted with BRAT annotation tool, and it consists of two steps. In the first step, each file is distributed to two annotators, and they will annotate the file independently. Then in the second step, the coordinator will discuss the annotation with each annotator making sure their annotation follows the guideline and dissolve the disagreement. We double-annotated all files in the PUBMEDM&M dataset and half of the files in the CHEMSYN dataset. 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 can be found in Table 11. We can see that the CHEMSYN dataset has higher IAA scores for two potential reasons: 1) we annotated the PUBMEDM&M dataset first, so the annotators might be more experienced when they annotated the CHEMSYN dataset; 2) the PUBMEDM&M dataset contains more diverse content like wet lab experiments or computer simulation procedures while the CHEMSYN dataset is mainly about chemical synthesis.

	Total	per Protocol	per Sentence
# of sentences	1508	7.90	-
# of words	29121	152.47	19.31
# of entities	9845	51.54	6.53
# of relations	8987	47.05	5.96
# of actions	2286	11.97	1.52

Table 9: Statistics of the PUBMEDM&M Corpus.

	Total	per Protocol	per Sentence
# of sentences			
# of words			
# of entities			
# of relations			
# of actions			

Table 10: Statistics of the CHEMSYN Corpus.

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

Table 11: Inter-Annotator Agreement (F_1 scores on Entity/Action Identification and Relation Extraction) of two annotation steps.

B Procedural Corpus Collection

PubMed Articles The first source is PubMed articles, and we chose to work with PubMed data because it is a large repository of articles containing descriptions of experimental procedures that can be freely accessed and shared. Specifically, we extracted paragraphs describing procedures (including chemical synthesis procedures) from the *Materials and Methods* section of articles within the open access subset of PubMed. XML files containing full-text of the articles were downloaded from NCBI[15], and they were processed to extract the PubMed ID for each article in addition to all paragraphs within a section of a paper with the title "Materials and Methods."

To filter out paragraphs within the materials and methods section that are not describing procedures, we developed a scientific discourse tagger using the dataset SciSeg (Dasigi et al., 2017). This dataset annotates clauses within PubMed articles with a set of discourse labels: `{Goal, Fact, Result, Hypothesis, Method, Problem, Implication}`. We fine-tuned SciBERT (Beltagy et al., 2019) using this dataset to automatically label PubMed sentences, achieving an average F1 score of 72.65% in a five-fold cross validation.

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

After training, we ran the scientific discourse tagger described above on each paragraph extracted from a *Materials and Methods*. We consider a paragraph to describe a valid procedure if at least 40% of clauses are labeled as `Method`. The threshold was obtained by manual inspection of the randomly sampled subset of the data.

In total the PubMed OpenAccess subset contains 2,542,736 articles, of which about 650k contain a *Materials and Methods* section. After filtering as described above, we retained a set of 1,785,923 paragraphs describing procedures. 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 procedural text is the patent data as chemical patents include the detailed procedures of chemical synthesis. Similarly, to extract paragraphs describing experimental steps from the *Description* section of chemical patents, we first downloaded US patent data (1976-2016) from USPTO[16] and European data (1978-2020) from EPO[17] as XML files. Then we applied the reaction extractor developed by Lowe (2012), which is a trained Naive Bayes classifier, to all our patent corpus. Note that the US patent data has two subsets, "Grant" (1976-2016) and "Application" (2001-2016). The "Application" subset contains the "Grant" subset for the same year, so we only use the US patents from the "Application" subset for those overlapping periods (2001-2016). As a result, we can get 2,435,999 paragraphs from US patents and 4,039,605 paragraphs from European patents. What's more, as our focus is the English data, we used the tool **langid**[18] to identify the source language for each extracted paragraph, and only kept English data, which has 3,671,482 paragraphs (90.9%).

C Hyper-parameter for Downstream Tasks

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

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

[18] `https://github.com/saffsd/langid.py`