Inexpensive Domain Adaptation of Pretrained Language Models: Case Studies on Biomedical NER and Covid-19 QA

Nina Poerner[*†] **and Ulli Waltinger**[†] **and Hinrich Schütze**[*]
[*]Center for Information and Language Processing, LMU Munich, Germany
[†]Corporate Technology Machine Intelligence (MIC-DE), Siemens AG Munich, Germany
poerner@cis.uni-muenchen.de | inquiries@cislmu.org

Abstract

Domain adaptation of Pretrained Language Models (PTLMs) is typically achieved by unsupervised pretraining on target-domain text. While successful, this approach is expensive in terms of hardware, runtime and CO_2 emissions. Here, we propose a cheaper alternative: We train Word2Vec on target-domain text and align the resulting word vectors with the wordpiece vectors of a general-domain PTLM. We evaluate on eight English biomedical Named Entity Recognition (NER) tasks and compare against the recently proposed BioBERT model. We cover over 60% of the BioBERT – BERT F1 delta, at 5% of BioBERT's CO_2 footprint and 2% of its cloud compute cost. We also show how to quickly adapt an existing general-domain Question Answering (QA) model to an emerging domain: the Covid-19 pandemic.[1]

1 Introduction

Pretrained Language Models (PTLMs) such as BERT (Devlin et al., 2019) have spearheaded advances on many NLP tasks. Usually, PTLMs are pretrained on unlabeled general-domain and/or mixed-domain text, such as Wikipedia, digital books or the Common Crawl corpus.

When applying PTLMs to specific domains, it can be useful to domain-adapt them. Domain adaptation of PTLMs has typically been achieved by pretraining on target-domain text. One such model is BioBERT (Lee et al., 2020), which was initialized from general-domain BERT and then pretrained on biomedical scientific publications. The domain adaptation is shown to be helpful for target-domain tasks such as biomedical Named Entity Recognition (NER) or Question Answering (QA). On the downside, the computational cost of pretraining can be considerable: BioBERTv1.0 was adapted for ten days on eight large GPUs (see Table 1), which is expensive, environmentally unfriendly, prohibitive for small research labs and students, and may delay prototyping on emerging domains.

We therefore propose a **fast, CPU-only domain-adaptation method for PTLMs**: We train Word2Vec (Mikolov et al., 2013a) on target-domain text and align the resulting word vectors with the wordpiece vectors of an existing general-domain PTLM. The PTLM thus gains domain-specific lexical knowledge in the form of additional word vectors, but its deeper layers remain unchanged. Since Word2Vec and the vector space alignment are efficient models, the process requires a fraction of the resources associated with pretraining the PTLM itself, and it can be done on CPU.

In Section 4, we use the proposed method to domain-adapt BERT on PubMed+PMC (the data used for BioBERTv1.0) and/or CORD-19 (Covid-19 Open Research Dataset). We improve over general-domain BERT on eight out of eight biomedical NER tasks, using a fraction of the compute cost associated with BioBERT. In Section 5, we show how to quickly adapt an existing Question Answering model to text about the Covid-19 pandemic, without any target-domain Language Model pretraining or finetuning.

2 Related work

2.1 The BERT PTLM

For our purpose, a PTLM consists of three parts: A tokenizer $\mathcal{T}_{\text{LM}} : \mathbb{L}^+ \rightarrow \mathbb{L}_{\text{LM}}^+$, a wordpiece embedding lookup function $\mathcal{E}_{\text{LM}} : \mathbb{L}_{\text{LM}} \rightarrow \mathbb{R}^{d_{\text{LM}}}$ and an encoder function $\mathcal{F}_{\text{LM}}$. $\mathbb{L}_{\text{LM}}$ is a limited vocabulary of wordpieces. All words from the natural language $\mathbb{L}^+$ that are not in $\mathbb{L}_{\text{LM}}$ are tokenized into sequences of shorter wordpieces, e.g., *dementia* becomes *dem ##ent ##ia*. Given a sentence $S = [w_1, \ldots, w_T]$, tokenized

[1] www.github.com/npoe/covid-qa

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1482–1490
November 16 - 20, 2020.

	size	Domain adaptation hardware	Power(W)	Time(h)	CO_2(lbs)	Google Cloud $
BioBERTv1.0	base	8 NVIDIA v100 GPUs (32GB)	1505	240	544	1421 – 4762
BioBERTv1.1	base	8 NVIDIA v100 GPUs (32GB)	1505	552	1252	3268 – 10952
GreenBioBERT (Section 4)	base	12 Intel Xeon E7-8857 CPUs, 30GB RAM	1560	12	28	16 – 76
GreenCovidSQuADBERT (Section 5)	large	12 Intel Xeon E7-8857 CPUs, 40GB RAM	1560	24	56	32 – 152

Table 1: Domain adaptation cost. CO_2 emissions are calculated according to Strubell et al. (2019). Since our hardware configuration is not available on Google Cloud, we take an *m1-ultramem-40* instance (40 vCPUs, 961GB RAM) to estimate an upper bound on our Google Cloud cost.

as $\mathcal{T}_{\text{LM}}(S) = [\mathcal{T}_{\text{LM}}(w_1); \ldots; \mathcal{T}_{\text{LM}}(w_T)]$, $\mathcal{E}_{\text{LM}}$ embeds every wordpiece in $\mathcal{T}_{\text{LM}}(S)$ into a real-valued, trainable wordpiece vector. The wordpiece vectors of the entire sequence are stacked and fed into $\mathcal{F}_{\text{LM}}$. Note that we consider position and segment embeddings to be a part of $\mathcal{F}_{\text{LM}}$ rather than $\mathcal{E}_{\text{LM}}$.

In the case of BERT, $\mathcal{F}_{\text{LM}}$ is a Transformer (Vaswani et al., 2017), followed by a final Feed-Forward Net. During pretraining, the Feed-Forward Net predicts the identity of masked wordpieces. When finetuning on a supervised task, it is usually replaced with a randomly initialized layer.

2.2 Domain-adapted PTLMs

Domain adaptation of PTLMs is typically achieved by pretraining on unlabeled target-domain text. Some examples of such models are BioBERT (Lee et al., 2020), which was pretrained on the PubMed and/or PubMed Central (PMC) corpora, SciBERT (Beltagy et al., 2019), which was pretrained on papers from SemanticScholar, ClinicalBERT (Alsentzer et al., 2019; Huang et al., 2019a) and ClinicalXLNet (Huang et al., 2019b), which were pretrained on clinical patient notes, and AdaptaBERT (Han and Eisenstein, 2019), which was pretrained on Early Modern English text. In most cases, a domain-adapted PTLM is initialized from a general-domain PTLM (e.g., standard BERT), though Beltagy et al. (2019) report better results with a model that was pretrained from scratch with a custom wordpiece vocabulary. In this paper, we focus on BioBERT, as its domain adaptation corpora are publicly available.

	Acc@1	Acc@5	Acc@10
train (19.8K words)	53.6	63.5	65.7
heldout (2.2K words)	39.4	51.6	54.3

Table 2: $\mathbb{L}_{\text{W2V}} \rightarrow \mathbb{L}_{\text{LM}}$ alignment accuracy (%), i.e., how often the identical string is in the top-K nearest neighbors.

2.3 Word vectors

Word vectors are distributed representations of words that are trained on unlabeled text. Contrary to PTLMs, word vectors are non-contextual, i.e., a word type is always assigned the same vector, regardless of context. In this paper, we use Word2Vec (Mikolov et al., 2013a) to train word vectors. We will denote the Word2Vec lookup function as $\mathcal{E}_{\text{W2V}} : \mathbb{L}_{\text{W2V}} \rightarrow \mathbb{R}^{d_{\text{W2V}}}$.

2.4 Word vector space alignment

Word vector space alignment has most frequently been explored in the context of cross-lingual word embeddings. For instance, Mikolov et al. (2013b) align English and Spanish Word2Vec spaces by a simple linear transformation. Wang et al. (2019) use a related method to align cross-lingual word vectors and multilingual BERT wordpiece vectors. In this paper, we apply the method to the problem of domain adaptation within the same language.

3 Method

In the following, we assume access to a general-domain PTLM, as described in Section 2.1, and a corpus of unlabeled target-domain text.

3.1 Creating new input vectors

In a first step, we train Word2Vec on the target-domain corpus. In a second step, we take the intersection of $\mathbb{L}_{\text{LM}}$ and $\mathbb{L}_{\text{W2V}}$. In practice, the intersection mostly contains wordpieces from $\mathbb{L}_{\text{LM}}$ that correspond to standalone words. It also contains single characters and other noise, however, we found that filtering them does not improve alignment quality. In a third step, we use the intersection to fit an unconstrained linear transformation $\mathbf{W} \in \mathbb{R}^{d_{\text{LM}} \times d_{\text{W2V}}}$ via least squares:

$$\underset{\mathbf{W}}{\text{argmin}} \sum_{x \in \mathbb{L}_{\text{LM}} \cap \mathbb{L}_{\text{W2V}}} ||\mathbf{W}\mathcal{E}_{\text{W2V}}(x) - \mathcal{E}_{\text{LM}}(x)||_2^2$$

Intuitively, $\mathbf{W}$ makes Word2Vec vectors "look like" the PTLM's native wordpiece vectors, just

	Query	NNs of query in $\mathcal{E}_{\text{LM}}[\mathbb{L}_{\text{LM}}]$	NNs of query in $\mathbf{W}\mathcal{E}_{\text{W2V}}[\mathbb{L}_{\text{W2V}}]$
query $\in \mathbb{L}_{\text{W2V}} \cap \mathbb{L}_{\text{LM}}$ **Boldface:** Training vector pairs	**surgeon**	physician, psychiatrist, surgery	**surgeon**, urologist, neurosurgeon
	surgeon	**surgeon**, physician, researcher	neurosurgeon, urologist, radiologist
	depression	Depression, recession, depressed	**depression**, Depression, hopelessness
	depression	**depression**, anxiety, anxiousness	depressive, insomnia, Depression
	fatal	lethal, deadly, disastrous	**fatal**, lethal, deadly
	fatal	**fatal**, catastrophic, disastrous	lethal, devastating, disastrous
query $\in \mathbb{L}_{\text{W2V}} - \mathbb{L}_{\text{LM}}$	ventricular	cardiac, pulmonary, mitochondrial	atrial, ventricle, RV
	dementia	diabetes, Alzheimer, autism	VaD, MCI, AD
	suppressants	medications, medicines, medication	suppressant, prokinetics, painkillers
	anesthesiologist	surgeon, technician, psychiatrist	anesthetist, anaesthesiologist, anaesthetist
	nephrotoxicity	toxicity, inflammation, contamination	hepatotoxicity, ototoxicity, cardiotoxicity
	impairment	inability, disruption, disorders	impairments, deficits, deterioration

Table 3: Examples of within-space and cross-space nearest neighbors (NNs) by cosine similarity in GreenBioBERT's wordpiece embedding layer. Blue: Original wordpiece space. Green: Aligned Word2Vec space.

like cross-lingual alignment makes word vectors from one language "look like" word vectors from another language. In Table 2, we report word alignment accuracy when we split $\mathbb{L}_{\text{LM}} \cap \mathbb{L}_{\text{W2V}}$ into a training and development set.[2] In Table 3, we show examples of within-space and cross-space nearest neighbors after alignment.

3.2 Updating the wordpiece embedding layer

Next, we redefine the wordpiece embedding layer of the PTLM. The most radical strategy would be to replace the entire layer with the aligned Word2Vec vectors:

$$\hat{\mathcal{E}}_{\text{LM}} : \mathbb{L}_{\text{W2V}} \rightarrow \mathbb{R}^{d_{\text{LM}}} \; ; \; \hat{\mathcal{E}}_{\text{LM}}(x) = \mathbf{W}\mathcal{E}_{\text{W2V}}(x)$$

In initial experiments, this strategy led to a drop in performance, presumably because function words are not well represented by Word2Vec, and replacing them disrupts BERT's syntactic abilities. To prevent this problem, we leave existing wordpiece vectors intact and only add new ones:

$$\hat{\mathcal{E}}_{\text{LM}} : \mathbb{L}_{\text{LM}} \cup \mathbb{L}_{\text{W2V}} \rightarrow \mathbb{R}^{d_{\text{LM}}};$$
$$\hat{\mathcal{E}}_{\text{LM}}(x) = \begin{cases} \mathcal{E}_{\text{LM}}(x) & \text{if } x \in \mathbb{L}_{\text{LM}} \\ \mathbf{W}\mathcal{E}_{\text{W2V}}(x) & \text{otherwise} \end{cases} \quad (1)$$

3.3 Updating the tokenizer

In a final step, we update the tokenizer to account for the added words. Let $\mathcal{T}_{\text{LM}}$ be the standard BERT tokenizer, and let $\hat{\mathcal{T}}_{\text{LM}}$ be the tokenizer that treats all words in $\mathbb{L}_{\text{LM}} \cup \mathbb{L}_{\text{W2V}}$ as one-wordpiece tokens, while tokenizing any other words as usual.

In practice, a given word may or may not benefit from being tokenized by $\hat{\mathcal{T}}_{\text{LM}}$ instead of $\mathcal{T}_{\text{LM}}$. To give a concrete example, 82% of the words in the BC5CDR NER dataset that end in the suffix *-ia* are part of a disease entity (e.g., *dementia*). $\mathcal{T}_{\text{LM}}$ tokenizes this word as *dem ##ent ##ia*, thereby exposing this strong orthographic cue to the model. As a result, $\mathcal{T}_{\text{LM}}$ improves recall on *-ia* diseases. But there are many cases where wordpiece tokenization is meaningless or misleading. For instance *euthymia* (not a disease) is tokenized by $\mathcal{T}_{\text{LM}}$ as *e ##uth ##ym ##ia*, making it likely to be classified as a disease. By contrast, $\hat{\mathcal{T}}_{\text{LM}}$ gives *euthymia* a one-wordpiece representation that depends only on distributional semantics. We find that using $\hat{\mathcal{T}}_{\text{LM}}$ improves precision on *-ia* diseases.

To combine these complementary strengths, we use a 50/50 mixture of $\mathcal{T}_{\text{LM}}$-tokenization and $\hat{\mathcal{T}}_{\text{LM}}$-tokenization when finetuning the PTLM on a task. At test time, we use both tokenizers and mean-pool the outputs. Let $o(S; \mathcal{T})$ be some output of interest (e.g., a logit), given sentence S tokenized by $\mathcal{T}$. We predict:

$$\hat{o}(S) = \frac{o(S; \mathcal{T}_{\text{LM}}) + o(S; \hat{\mathcal{T}}_{\text{LM}})}{2}$$

4 Experiment 1: Biomedical NER

In this section, we use the proposed method to create GreenBioBERT, an inexpensive and environmentally friendly alternative to BioBERT. Recall that BioBERTv1.0 (*biobert_v1.0_pubmed_pmc*) was initialized from general-domain BERT (*bert-base-cased*) and then pretrained on PubMed+PMC.

4.1 Domain adaptation

We train Word2Vec with vector size $d_{\text{W2V}} = d_{\text{LM}} = 768$ on PubMed+PMC (see Appendix for details). Then, we update the wordpiece embedding layer and tokenizer of general-domain BERT (*bert-base-cased*) as described in Section 3.

[2] Since we are not primarily interested in word alignment accuracy, we use the entire intersection as a training set in all other experiments.

Biomedical NER task	(NER task ID)	BERT (ref) (Lee et al., 2020)	BioBERTv1.0 (ref) (Lee et al., 2020)	BioBERTv1.1 (ref) (Lee et al., 2020)	GreenBioBERT (with standard error of the mean)
BC5CDR-disease (Li et al., 2016)	(1)	81.97 / 82.48 / 82.41	85.86 / 87.27 / 86.56	**86.47 / 87.84 / 87.15**	84.88 (.07) / 85.29 (.12) / 85.08 (.08)
NCBI-disease (Doğan et al., 2014)	(2)	84.12 / 87.19 / 85.63	**89.04** / 89.69 / 89.36	88.22 / **91.25 / 89.71**	85.49 (.23) / 86.41 (.15) / 85.94 (.16)
BC5CDR-chem (Li et al., 2016)	(3)	90.94 / 91.38 / 91.16	93.27 / **93.61** / 93.44	93.68 / 93.26 / **93.47**	**93.82** (.11) / 92.35 (.17) / 93.08 (.07)
BC4CHEMD (Krallinger et al., 2015)	(4)	91.19 / 88.92 / 90.04	92.23 / 90.61 / 91.41	**92.80 / 91.92 / 92.36**	**92.80** (.04) / 89.78 (.07) / 91.26 (.04)
BC2GM (Smith et al., 2008)	(5)	81.17 / 82.42 / 81.79	**85.16** / 83.65 / 84.40	84.32 / **85.12 / 84.72**	83.34 (.15) / 83.58 (.09) / 83.45 (.10)
JNLPBA (Kim et al., 2004)	(6)	69.57 / 81.20 / 74.94	**72.68** / 83.21 / **77.59**	72.24 / **83.56** / 77.49	71.93 (.12) / 82.58 (.12) / 76.89 (.10)
LINNAEUS (Gerner et al., 2010)	(7)	91.17 / 84.30 / 87.60	**93.84 / 86.11 / 89.81**	90.77 / 85.83 / 88.24	92.50 (.17) / 84.54 (.26) / 88.34 (.18)
Species-800 (Pafilis et al., 2013)	(8)	69.35 / 74.05 / 71.63	72.84 / **77.97 / 75.31**	72.80 / 75.36 / 74.06	**73.19** (.26) / 75.47 (.33) / 74.31 (.24)

Table 4: Biomedical NER test set precision / recall / F1 (%). "(ref)": Reference scores from Lee et al. (2020). **Boldface**: Best model in row. Underlined: Best model without target-domain LM pretraining.

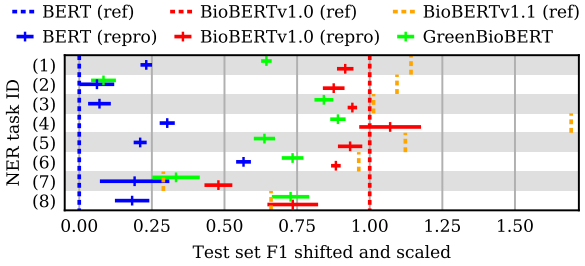

Figure 1: NER test set F1, transformed as $(x - \text{BERT}_{(\text{ref})})/(\text{BioBERTv1.0}_{(\text{ref})} - \text{BERT}_{(\text{ref})})$. This plot shows what portion of the reported BioBERT – BERT F1 delta is covered. "(ref)": Reference scores from Lee et al. (2020). "(repro)": Results of our reproduction experiments. Error bars: Standard error of the mean.

NER task ID	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
non-aligned	-4.88	-3.50	-4.13	-3.34	-2.34	-0.56	-0.84	-4.63
random init	-4.33	-3.60	-3.19	-3.19	-1.92	-0.50	-0.84	-3.58

Table 5: Absolute drop in dev set F1 when using non-aligned word vectors or randomly initialized word vectors, instead of aligned word vectors.

4.2 Finetuning

We finetune GreenBioBERT on the eight publicly available NER tasks used in Lee et al. (2020). We also do reproduction experiments with general-domain BERT and BioBERTv1.0, using the same setup as our model. See Appendix for details on preprocessing and hyperparameters. Since some of the datasets are sensitive to the random seed, we report mean and standard error over eight runs.

4.3 Results and discussion

Table 4 shows entity-level precision, recall and F1, as measured by the CoNLL NER scorer. For ease of visualization, Figure 1 shows test set F1 shifted and scaled as

$$f(x) = \frac{x - \text{BERT}_{(\text{ref})}}{\text{BioBERTv1.0}_{(\text{ref})} - \text{BERT}_{(\text{ref})}}$$

where $\text{BERT}_{(\text{ref})}$ and $\text{BioBERTv1.0}_{(\text{ref})}$ are reported scores from Lee et al. (2020). In other words, the figure shows what portion of the reported BioBERT – BERT F1 delta is covered by our less expensive GreenBioBERT model. On average, we cover between 61% and 70% of the delta (61% for BioBERTv1.0, 70% for BioBERTv1.1, and 61% if we take our reproduction experiments as reference points).

4.3.1 Ablation study

To test whether the improvements over general-domain BERT are due to the aligned Word2Vec vectors, or just to the availability of additional word vectors in general, we perform an ablation study where we replace the aligned vectors with their non-aligned counterparts (by setting $\mathbf{W} = \mathbf{1}$ in Eq. 1) or with randomly initialized vectors. Table 5 shows that dev set F1 drops on all datasets under these circumstances, i.e., vector space alignment seems to be important.

5 Experiment 2: Covid-19 QA

In this section, we use the proposed method to quickly adapt an existing general-domain QA model to an emerging target domain: the Covid-19 pandemic. Our baseline model is SQuADBERT,[3] an existing BERT model that was finetuned on the general-domain SQuAD dataset (Rajpurkar et al., 2016). We evaluate on Deepset-AI Covid-QA (Möller et al., 2020), a SQuAD-style dataset with 2019 annotated span-selection questions about 147 papers from CORD-19 (Covid-19 Open Research Dataset).[4] We assume that there is no labeled target-domain data for finetuning on the task, and instead use the entire Covid-QA dataset as a test set. This is a realistic setup for an emerging domain without annotated training data.

[3] www.huggingface.co/bert-large-uncased-whole-word-masking-finetuned-squad

[4] https://pages.semanticscholar.org/coronavirus-research

	domain adaptation corpus	size	EM	F1	substr
SQuADBERT	——		33.04	58.24	65.87
GreenCovid-	CORD-19 only	2GB	**34.62**	60.09	**68.20**
SQuADBERT	CORD-19+PubMed+PMC	94GB	34.32	**60.23**	68.00

Table 6: Results (%) on Deepset-AI Covid-QA. EM (exact answer match) and F1 (token-level F1 score) are evaluated with the SQuAD scorer. "substr": Predictions that are a substring of the gold answer. Much higher than EM, because many gold answers are not minimal answer spans (see Appendix, "Notes on Covid-QA", for an example).

5.1 Domain adaptation

We train Word2Vec with vector size $d_{\text{W2V}} = d_{\text{LM}} = 1024$ on CORD-19 and/or PubMed+PMC. The process takes less than an hour on CORD-19 and about one day on the combined corpus, again without the need for a GPU. Then, we update SQuADBERT's wordpiece embedding layer and tokenizer, as described in Section 3. We refer to the resulting model as GreenCovidSQuADBERT.

5.2 Results and discussion

Table 6 shows that GreenCovidSQuADBERT outperforms general-domain SQuADBERT on all measures. Interestingly, the small CORD-19 corpus is enough to achieve this result (compare "CORD-19 only" and "CORD-19+PubMed+PMC"), presumably because it is specific to the target domain and contains the Covid-QA context papers.

6 Conclusion

As a reaction to the trend towards high-resource models, we have proposed an inexpensive, CPU-only method for domain-adapting Pretrained Language Models: We train Word2Vec vectors on target-domain data and align them with the wordpiece vector space of a general-domain PTLM.

On eight biomedical NER tasks, we cover over 60% of the BioBERT – BERT F1 delta, at 5% of BioBERT's domain adaptation CO_2 footprint and 2% of its cloud compute cost. We have also shown how to rapidly adapt an existing BERT QA model to an emerging domain – the Covid-19 pandemic – without the need for target-domain Language Model pretraining or finetuning.

We hope that our approach will benefit practitioners with limited time or resources, and that it will encourage environmentally friendlier NLP.

Acknowledgements

This research was funded by Siemens AG. We thank our anonymous reviewers for their helpful comments.

References

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, USA.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *EMNLP-IJCNLP*, pages 3606–3611, Hong Kong, China.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, Minneapolis, USA.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A Smith. 2019. Show your work: Improved reporting of experimental results. In *EMNLP-IJCNLP*, pages 2185–2194, Hong Kong, China.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Martin Gerner, Goran Nenadic, and Casey M Bergman. 2010. LINNAEUS: a species name identification system for biomedical literature. *BMC bioinformatics*, 11(1):85.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *EMNLP-IJCNLP*, pages 4229–4239, Hong Kong, China.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019a. ClinicalBERT: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*.

Kexin Huang, Abhishek Singh, Sitong Chen, Edward T Moseley, Chih-ying Deng, Naomi George, and Charlotta Lindvall. 2019b. Clinical XLNet: Modeling sequential clinical notes and predicting prolonged mechanical ventilation. *arXiv preprint arXiv:1912.11975*.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 70–75.

Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. 2015. The CHEMDNER corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7(1):1–17.

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

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. BioCreative V CDR task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in Adam.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. 2020. Covid-qa: A question & answer dataset for covid-19.

Evangelos Pafilis, Sune P Frankild, Lucia Fanini, Sarah Faulwetter, Christina Pavloudi, Aikaterini Vasileiadou, Christos Arvanitidis, and Lars Juhl Jensen. 2013. The SPECIES and ORGANISMS resources for fast and accurate identification of taxonomic names in text. *PloS one*, 8(6).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392, Austin, USA.

Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, et al. 2008. Overview of BioCreative II gene mention recognition. *Genome biology*, 9(2):S2.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *ACL*, pages 3645–3650, Florence, Italy.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008, Long Beach, USA.

Hai Wang, Dian Yu, Kai Sun, Janshu Chen, and Dong Yu. 2019. Improving pre-trained multilingual models with vocabulary expansion. In *CoNLL*, pages 316–327, Hong Kong, China.

Inexpensive Domain Adaptation of Pretrained Language Models (Appendix)

Word2Vec training

We downloaded the PubMed, PMC and CORD-19 corpora from:

- https://ftp.ncbi.nlm.nih.gov/pub/pmc/oa_bulk/ [20 January 2020, 68GB raw text]
- https://ftp.ncbi.nlm.nih.gov/pubmed/baseline/ [20 January 2020, 24GB raw text]
- https://pages.semanticscholar.org/coronavirus-research [17 April 2020, 2GB raw text]

We extract all abstracts and text bodies and apply the BERT basic tokenizer (a rule-based word tokenizer that standard BERT uses before wordpiece tokenization). Then, we train CBOW Word2Vec[5] with negative sampling. We use default parameters except for the vector size (which we set to $d_{\text{W2V}} = d_{\text{LM}}$).

Experiment 1: Biomedical NER

Pretrained models

General-domain BERT and BioBERTv1.0 were downloaded from:

- www.storage.googleapis.com/bert_models/2018_10_18/cased_L-12_H-768_A-12.zip
- www.github.com/naver/biobert-pretrained

Data

We downloaded the NER datasets by following instructions on www.github.com/dmis-lab/biobert#Datasets. For detailed dataset statistics, see Lee et al. (2020).

Preprocessing

We use Lee et al. (2020)'s preprocessing strategy: We cut all sentences into chunks of 30 or fewer whitespace-tokenized words (without splitting inside labeled spans). Then, we tokenize every chunk S with $\mathcal{T} = \mathcal{T}_{\text{LM}}$ or $\mathcal{T} = \hat{\mathcal{T}}_{\text{LM}}$ and add special tokens:

$$X = \textit{[CLS]}\ \mathcal{T}(S)\ \textit{[SEP]}$$

Word-initial wordpieces in $\mathcal{T}(S)$ are labeled as *B(egin)*, *I(nside)* or *O(utside)*, while non-word-initial wordpieces are labeled as *X(ignore)*.

[5] www.github.com/tmikolov/word2vec

Modeling, training and inference

We follow Lee et al. (2020)'s implementation (www.github.com/dmis-lab/biobert): We add a randomly initialized softmax classifier on top of the last BERT layer to predict the labels. We finetune the entire model to minimize negative log likelihood, with the AdamW optimizer (Loshchilov and Hutter, 2018) and a linear learning rate scheduler (10% warmup). All finetuning runs were done on a GeForce Titan X GPU (12GB).

At inference time, we gather the output logits of word-initial wordpieces only. Since the number of word-initial wordpieces is the same for $\mathcal{T}_{\text{LM}}(S)$ and $\hat{\mathcal{T}}_{\text{LM}}(S)$, this makes mean-pooling the logits straightforward.

Hyperparameters

We tune the batch size and peak learning rate on the development set (metric: F1), using the same hyperparameter space as Lee et al. (2020):

Batch size: $[10, 16, 32, 64]$[6]

Learning rate: $[1 \cdot 10^{-5}, 3 \cdot 10^{-5}, 5 \cdot 10^{-5}]$

We train for 100 epochs, which is the upper end of the 50–100 range recommended by the original authors. After selecting the best configuration for every task and model (see Table 7), we train the final model on the concatenation of training and development set, as was done by Lee et al. (2020). See Figure 2 for expected maximum development set F1 as a function of the number of evaluated hyperparameter configurations (Dodge et al., 2019).

Experiment 2: Covid-19 QA

Pretrained model

We downloaded the SQuADBERT baseline from:

- www.huggingface.co/bert-large-uncased-whole-word-masking-finetuned-squad

Data

We downloaded the Deepset-AI Covid-QA dataset from:

- www.github.com/deepset-ai/COVID-QA/blob/master/data/question-answering/COVID-QA.json [24 June 2020]

[6] Since LINNAEUS and BC4CHEM have longer maximum tokenized chunk lengths than the other datasets, our hardware was insufficient to evaluate batch size 64 on them.

At the time of writing, the dataset contains 2019 questions and gold answer spans. Every question is associated with one of 147 research papers (contexts) from CORD-19.[7] Since we do not do target-domain finetuning, we treat the entire dataset as a test set.

Preprocessing

We tokenize every question-context pair (Q, C) with $\mathcal{T} = \mathcal{T}_{\text{LM}}$ or $\mathcal{T} = \hat{\mathcal{T}}_{\text{LM}}$, which yields $(\mathcal{T}(Q), \mathcal{T}(C))$. Since $\mathcal{T}(C)$ is usually too long to be digested in a single forward pass, we define a sliding window with width and stride $N = \text{floor}(\frac{509 - |\mathcal{T}(Q)|}{2})$. At step n, the "active" window is between $a_n^{(l)} = (n-1)N + 1$ and $a_n^{(r)} = \min(|C|, nN)$. The input is defined as:

$$X^{(n)} = \textit{[CLS]}\ \mathcal{T}(Q)\ \textit{[SEP]}\ \mathcal{T}(C)_{a_n^{(l)} - p_n^{(l)} : a_n^{(r)} + p_n^{(r)}}\ \textit{[SEP]}$$

$p_n^{(l)}$ and $p_n^{(r)}$ are chosen such that $|X^{(n)}| = 512$, and such that the active window is in the center of the input (if possible).

Modeling and inference

Feeding $X^{(n)}$ into the QA model yields start logits $\mathbf{h}'^{(\text{start},n)} \in \mathbb{R}^{|X^{(n)}|}$ and end logits $\mathbf{h}'^{(\text{end},n)} \in \mathbb{R}^{|X^{(n)}|}$. We extract and concatenate the slices that correspond to the active windows of all steps:

$$\mathbf{h}^{(*)} \in \mathbb{R}^{|\mathcal{T}(C)|}$$

$$\mathbf{h}^{(*)} = [\mathbf{h}'^{(*,1)}_{a_1^{(l)}:a_1^{(r)}}; \ldots; \mathbf{h}'^{(*,n)}_{a_n^{(l)}:a_n^{(r)}}; \ldots]$$

Next, we map the logits from the wordpiece level to the word level. This allows us to mean-pool the outputs of $\mathcal{T}_{\text{LM}}$ and $\hat{\mathcal{T}}_{\text{LM}}$ even when $|\mathcal{T}_{\text{LM}}(C)| \neq |\hat{\mathcal{T}}_{\text{LM}}(C)|$.

Let c_i be a word in C and let $\mathcal{T}(C)_{j:j+|\mathcal{T}(c_i)|}$ be the corresponding wordpieces. The start and end logits of c_i are:

$$o_i^{(*)} = \max_{j \leq j' \leq j+|\mathcal{T}(c_i)|} [h_{j'}^{(*)}]$$

Finally, we return the answer span $C_{k:k'}$ that maximizes $o_k^{(\text{start})} + o_{k'}^{(\text{end})}$, subject to the constraints that k' does not precede k and the answer contains no more than 500 characters.

[7] www.github.com/deepset-ai/COVID-QA/issues/103

Notes on Covid-QA

There are some important differences between Covid-QA and SQuAD, which make the task challenging:

- The Covid-QA contexts are full documents rather than single paragraphs. Thus, the correct answer may appear several times, often with slightly different wordings. But only a single occurrence is annotated as correct, e.g.:

  Question: What was the prevalence of Coronavirus OC43 in community samples in Ilorin, Nigeria?

  Correct: 13.3% (95% CI 6.9-23.6%) *# from main text*

  Predicted: 13.3%, 10/75 *# from abstract*

- SQuAD gold answers are defined as the "shortest span in the paragraph that answered the question" (Rajpurkar et al., 2016, p. 4), but many Covid-QA gold answers are longer and contain non-essential context, e.g.:

  Question: When was the Middle East Respiratory Syndrome Coronavirus isolated first?

  Correct: (MERS-CoV) was first isolated in 2012, in a 60-year-old man who died in Jeddah, KSA due to severe acute pneumonia and multiple organ failure

  Predicted: 2012

These differences are part of the reason why the exact match score is lower than the word-level F1 score and the substring score (see Table 6, bottom, main paper).

Biomedical NER task	(ID)	BERT (repro) hyperparams	BERT (repro) dev set F1	BioBERTv1.0 (repro) hyperparams	BioBERTv1.0 (repro) dev set F1	GreenBioBERT hyperparams	GreenBioBERT dev set F1
BC5CDR-disease	(1)	$32, 3 \cdot 10^{-5}$	82.12	$10, 1 \cdot 10^{-5}$	85.15	$32, 1 \cdot 10^{-5}$	83.90
NCBI-disease	(2)	$32, 3 \cdot 10^{-5}$	87.52	$32, 1 \cdot 10^{-5}$	87.99	$10, 3 \cdot 10^{-5}$	88.43
BC5CDR-chem	(3)	$64, 3 \cdot 10^{-5}$	91.00	$32, 1 \cdot 10^{-5}$	93.36	$10, 1 \cdot 10^{-5}$	92.59
BC4CHEMD	(4)	$16, 1 \cdot 10^{-5}$	88.02	$32, 1 \cdot 10^{-5}$	89.35	$16, 1 \cdot 10^{-5}$	88.53
BC2GM	(5)	$32, 1 \cdot 10^{-5}$	83.91	$64, 3 \cdot 10^{-5}$	85.54	$64, 3 \cdot 10^{-5}$	84.25
JNLPBA	(6)	$32, 5 \cdot 10^{-5}$	85.18	$32, 5 \cdot 10^{-5}$	85.30	$10, 3 \cdot 10^{-5}$	85.10
LINNAEUS	(7)	$16, 1 \cdot 10^{-5}$	96.67	$32, 1 \cdot 10^{-5}$	97.22	$10, 1 \cdot 10^{-5}$	96.49
Species-800	(8)	$32, 1 \cdot 10^{-5}$	72.70	$32, 1 \cdot 10^{-5}$	77.34	$16, 1 \cdot 10^{-5}$	75.93

Table 7: Best hyperparameters (batch size, peak learning rate) and best dev set F1 per NER task and model. BERT (repro) and BioBERTv1.0 (repro) refer to our reproduction experiments.

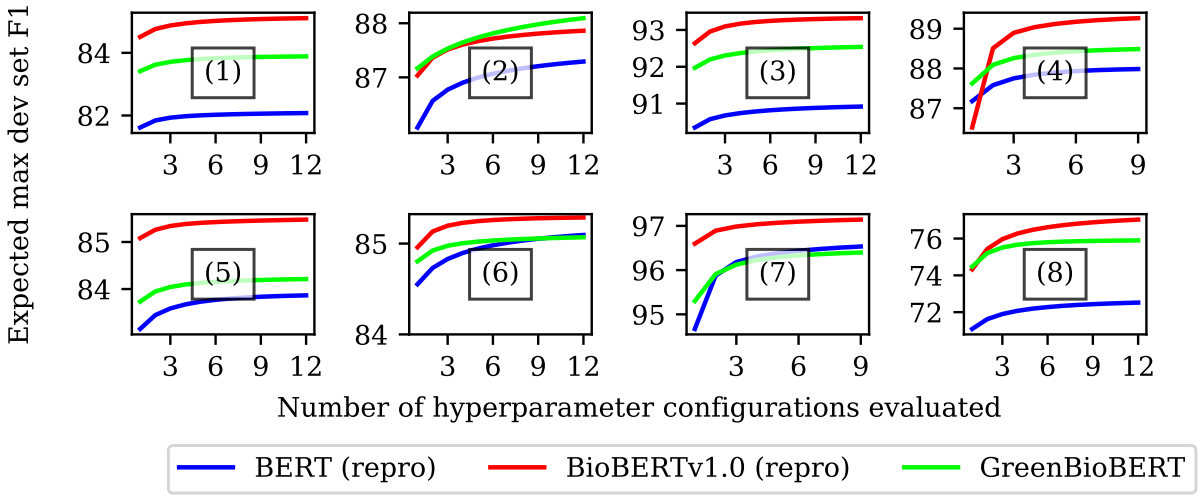

Figure 2: Expected maximum F1 on NER development sets as a function of the number of evaluated hyperparameter configurations. Numbers in brackets are NER task IDs (see Table 7).