From General Language Models to Medical-Domain-Specific Tasks: Final Report for NLPDL, 2022 Fall

Chuyue Tang
Yuanpei College
Peking University
2000017814@stu.pku.edu.cn

Abstract

This is the report of the final project for the course, Deep Learning in Natural Language Processing, 2022 Fall, with a focus on improving the general language model's performance on domain-specific tasks. This project explored three methods for performance improvement: post-training, task-adaption, and vocabulary-expansion. These methods were applied to the pre-trained RoBERTa model and evaluated on two medical-domain tasks, ChemProt and BioASQ. The results demonstrate that these methods can effectively improve the model's performance on these medical domain-specific tasks. In addition, our methods are also resource-efficient compared to similar approaches in the literature.

1 Introduction

In recent years, large-scale pre-trained language models have achieved impressive success. However, these models are typically pre-trained on general corpora like Wikipedia, which can lead to poor performance on some domain-specific downstream tasks. To address this issue, this project aims to improve the performance of pre-trained language models on domain-specific tasks, with a focus on the bio-medical domain.

In this project, the model, `roberta-base`, is loaded from Hugging Face[1], which is the robustly optimized version of the BERT model by Liu et al., 2019. The performance is measured on two downstream tasks, ChemProt and BioASQ.

The ChemProt task typically involves classifying the relationships between chemical compounds and proteins into one of several predefined categories or classes. In this project, we use thirteen subclasses of relationships, instead of the five classes used in the original task (see table 1). This largely increases the difficulty of this task. The metric is micro f1.

[1]https://huggingface.co/roberta-base

Class (Num)	**Sub-class(Num)**

CPR:3 (777)	UPREGULATOR(67)
	ACTIVATOR(323)
	INDIRECT-UPREGULATOR(387)

CPR:4 (2260)	DOWNREGULATOR(131)
	INHIBITOR(1642)
	INDIRECT-DOWNREGULATOR(487)

CPR:5 (170)	AGONIST(156)
	AGONIST-ACTIVATOR(10)
	AGONIST-INHIBITOR(4)

CPR:6 (235)	ANTAGONIST(235)

CPR:9 (727)	PRODUCT-OF(233)
	SUBSTRATE(480)
	SUBSTRATE_PRODUCT-OF(14)

Table 1: An illustration of the ChemProt training set labels.

The BioASQ task involves a dataset of annotated scientific articles and questions in the bio-medical domain. In this project, we treat this task as a binary classification task. Given an article-question pair, the model needs to answer 'yes' or 'no'. The metric is accuracy.

This project's work can be summarized from three aspects:

1. **post-training**: We use domain-specific data from PubMed. The pre-trained RoBERTa model is post-trained on this corpus by MLM (Masked Language Modeling).

2. **task-adaption**: We use domain-specific data from downstream tasks, ChemProt and BioASQ. Both the pre-trained RoBERTa and the post-trained model can be trained on this corpus by MLM.

3. **vocabulary-expansion**: We train a domain-specific tokenizer from PubMed data and its vocabulary is used to expand the original vocabulary of the RoBERTa tokenizer. The ex-

panded tokenizer can be further utilized in fine-tuning/post-training/task-adaption stage. As new word embeddings are introduced, we also explore different methods of initializing them.

We'll introduce our method in three separate sections, with a general description and some experiment results.

The code for this project is made public[2].

2 Domain-Specific Post-Training

In this section, we simply insert a post-training stage into the pretrain-then-finetune paradigm for downstream tasks.

There are mainly two differences between pre-training and post-training:

1. type of training data:

   Pre-training uses domain-general corpora, while post-training uses domain-specific corpora. Here we use PubMed data resized from the preprocessed data provided in a GitHub repository[3].

2. amount of training data:

   In the pre-training stage, the RoBERTa model was trained on a large dataset of about 160 GB of text, while in the post-training stage, the PubMed dataset is resized to about 130 M.

As standard practice, we adopt MLM for unsupervised training on unlabeled data.

We do not search extensively for better hyperparameters such as batch size b or number of epochs n. By default, $b = 8$ and $n = 3$. The post-training stage takes approximately two hours on one GPU, NVIDIA GeForce RTX 3090. The standard deviation and maximum are reported across five repeated experiments with five different seeds. Other results in this report follow the same rule.

The results are shown in table 2.

We observe that domain-specific post-training improves the performance of `roberta-base` in every aspect. Not only does the metric largely increases, but also the deviation across experiments decreases. It is effective and affordable.

[2]https://github.com/tcy63/NLPDL_project.git

[3]https://github.com/ncbi-nlp/bluebert.git

ChemProt	**mean(std)**	**max**
fine-tuning	0.7719 (0.04482)	0.7939
fine-tuning **+ post-training**	**0.8194 (0.00894)**	**0.8311**
BioASQ	**mean(std)**	**max**
fine-tuning	0.6799 (0.05388)	0.7315
fine-tuning **+ post-training**	**0.7383 (0.02610)**	**0.7685**

Table 2: Result of post-training.

3 Domain-Specific Task-Adaption

In this section, we are inspired by Gururangan et al., 2020. Their work reports that adapting to the task's unlabeled data (task-adaptive pretraining) improves performance even after domain-adaptive pretraining. But their improvements benefited from the task's unlabeled data are quite marginal. They train the model on the task's unlabeled data for 100 epochs, while our work shows that no more than 3 epochs or 500 steps of task-adaption are sufficient to produce a satisfying result.

3.1 The power of task-adaption

We use the same training strategy as post-training. The results are shown in table 3.

ChemProt	**mean(std)**
fine-tuning	0.7719 (0.04482)
fine-tuning **+ task-adaption**	**0.8159 (0.00715)**
fine-tuning + post-training	0.8194 (0.00894)
fine-tuning + post-training **+ task-adaption**	**0.8304 (0.00547)**
BioASQ	**mean(std)**
fine-tuning	0.6799 (0.05388)
fine-tuning **+ task-adaption**	**0.7463 (0.02195)**
fine-tuning + post-training	0.7383 (0.02610)
fine-tuning + post-training **+ task-adaption**	**0.7624 (0.01287)**

Table 3: Result of task-adaption.

In this experiment, we report four different cases: 1) fine-tune `roberta-base` directly, 2) train `roberta-base` on the task's unlabeled data (BioASQ) before fine-tuning, 3) train `roberta-base` on the domain's unlabeled data (PubMed) before fine-tuning, and 4) train `roberta-base` first on the domain's data and then on the task's data before fine-tuning. The main findings can be concluded as:

- Combining task-adaption and domain-specific post-training shows a significant improvement;
- Small-scale task-adaption alone can achieve comparable results with post-training.

3.2 Effect of the amount of task-adaptive training

We ask another question: how much task-adaption is enough?

To answer it, we first train `roberta-base` model on PubMed data and get the post-trained version. Then, we train the post-trained model for task-adaption with different steps and test it on two downstream tasks.

task-adaption step	ChemProt
500	**0.8222 (0.00730)**
1000	0.8269 (0.00598)
1500	0.8159 (0.00715)
2000	0.8290 (0.00275)
2500	0.8295 (0.00421)
3000	0.8287 (0.00224)
3500	0.8267 (0.00275)
4000	0.8213 (0.00859)
4500	0.8299 (0.00427)
5000	0.8274 (0.00496)
5500	0.8258 (0.00887)
6000	**0.8304 (0.00547)**

Table 4: The effect of number of task-adaption steps (evaluated in ChemProt).

As shown in table 4, the performance on ChemProt is good enough with only 500 steps of task-adaption training. The increase in performance is not obvious as the number of task-adaption steps increases.

As shown in table 5, the performance on BioASQ is also good enough with only 500 steps of task-adaption training. The increase in performance is more obvious than ChemProt as the number of task-adaption steps increases.

4 Domain-Specific Vocabulary-Expansion

In this section, we expand the vocabulary of the RoBERTa tokenizer with a domain-specific tokenizer. In addition, we analyze the initialization problem of new word embeddings and explore four different initialization methods. Our experiment shows that vocabulary-expansion can result in 95% performance of post-training with almost no computation overhead. Moreover, we combine post-training and vocabulary-expansion together and observe a marginal improvement.

task-adaption step	BioASQ
500	**0.7342 (0.01265)**
1000	0.7374 (0.01353)
1500	0.7450 (0.00919)
2000	0.7458 (0.00924)
2500	0.7436 (0.00808)
3000	0.7483 (0.00856)
3500	0.7510 (0.00765)
4000	0.7517 (0.01210)
4500	0.7537 (0.00907)
5000	0.7530 (0.00735)
5500	0.7550 (0.00671)
6000	**0.7624 (0.01287)**

Table 5: The effect of number of task-adaption steps (evaluated in BioASQ).

4.1 Domain-general tokenizer and domain-specific tokenizer

The RoBERTa tokenizer is a tool that separates text into tokens for use with the RoBERTa language model. It is based on the WordPiece model, which uses subword units instead of individual words to represent language.

However, the RoBERTa tokenizer may not perform well on domain-specific tasks, because it may not be able to split domain-specific words into meaningful sub-parts.

One solution to this issue is to expand the vocabulary of the RoBERTa tokenizer to include more domain-specific terms or sub-parts. This is expected to improve the performance of the tokenizer and ultimately the performance on domain-specific tasks.

We describe this process as 'vocabulary expansion'. It contains the following steps. First, we train a tokenizer on PubMed data, which is used in domain-specific post-training. This can be easily done with Hugging Face. Additional parameters can be provided such as the maximum number of tokens or the minimum frequency of a word to be included in the vocabulary. Second, we add new tokens from the vocabulary of the domain-specific tokenizer into the original tokenizer's vocabulary.

In table 6, we can get an intuitive view of the difference between the original domain-general tokenizer and our domain-specific tokenizer. A high overlapping ratio indicates the generality of language, while the difference ratio represents domain-specific knowledge different from the domain-general text. As the vocabulary size of the domain-specific tokenizer expands, the overlapping ratio decreases.

domain-specific vocabulary size	overlapping words	ratio
500	408	81.6%
1000	747	74.7%
2000	1392	69.6%
3000	1995	66.5%
5000	3069	61.4%
10000	5256	52.6%

Table 6: Difference between the original domain-general tokenizer and our domain-specific tokenizer.

4.2 Initialization of new word embeddings

The problem of expanding word embeddings of a pre-trained model comes along with an expanded tokenizer. Even if the additional words are important and relevant to the domain-specific task, the model has not been pre-trained on them. So the model will start with pre-trained embeddings belonging to the RoBERTa tokenizer vocabulary, as well as randomly initialized embeddings belonging to extra words.

Two possible solutions to this problem will be discussed in this report.

First, we can change the initialization method of the word embeddings. For example, we can use the pre-trained `<unk>` token's embedding. We can also generate these extra embeddings using the distribution of pre-trained embeddings. Details can be found in section 4.3.

Second, we can learn these extra embeddings during the post-training stage. Previously, we only evaluate it in direct fine-tuning, which may not be enough to learn these embeddings. Details can be found in section 4.4.

Each word embedding in the model can be represented as a vector $\mathbf{x} \in \mathcal{R}^d$. In the `roberta-base` model, $d = 768$. Denote the size of the original vocabulary as $N = 50265$. Then $\mathbf{x}_1 \cdots \mathbf{x}_N$ are pre-trained and optimized for the `roberta-base` model. Original word embeddings can be represented as $E \in \mathcal{R}^{N \times d}$. Increasing the vocabulary size with m extra words will add newly initialized vectors $\mathbf{y}_1 \cdots \mathbf{y}_m$ at the end. In this report, we experiment with four initialization approaches.

1. **rand**: default initialization provided by Hugging Face.

   $\mathbf{y}_i \sim \mathcal{N}(\mu, \sigma)$, $\mu = 0$, $\sigma = 0.02$

2. **zero**: all extra word embeddings are initialized as zero.

   $\mathbf{y}_i = 0$

3. **unk**: all extra word embeddings are initialized by `<unk>` token in the original vocabulary.

   $\mathbf{y}_i = \mathbf{x}_{unk}$

4. **mean**: use the average of the original embeddings to sample new embeddings.

   $\mathbf{y}_i \sim \mathcal{N}(\mu, s\Sigma)$, $s = 10^{-5}$,

   $\mu = \frac{1}{N}\sum_{j=1}^{N} \mathbf{x}_j$,

   $\Sigma = (E - \mu)^T (E - \mu)/N$

4.3 Comparison between post-training and vocabulary-expansion

In this experiment, we add 500 tokens from the domain-specific tokenizer's vocabulary into the original vocabulary, which means $m = 500$. We use the expanded tokenizer directly in downstream tasks without any post-training. New word embeddings are initialized in four different ways introduced in section 4.2.

BioASQ	**mean(std)**
fine-tuning	0.6799 (0.05388)
fine-tuning + post-training	**0.7383 (0.02610)**
fine-tuning **+ vocab-expansion**	**0.7007 (0.07021)**

Table 7: Comparison between post-training and vocabulary-expansion.

As shown in table 7, vocabulary-expansion achieves 95% of the post-training result, without any extra computational cost. The only extra time is spent on training tokenizers, which can be done only in minutes.

Next, we will see the effect of different initialization methods in table 8. The default random

ChemProt	**mean(std)**
fine-tuning	0.7719 (0.04482)
fine-tuning + post-training	**0.8194 (0.00894)**
fine-tuning **+ vocab-expansion(rand)**	0.7109 (0.16460)
fine-tuning **+ vocab-expansion(zero)**	0.7785 (0.02610)
fine-tuning **+ vocab-expansion(unk)**	**0.7901 (0.02336)**
fine-tuning **+ vocab-expansion(mean)**	0.7798 (0.02618)

Table 8: Effect of different initialization methods in vocabulary-expansion.

initialization appears more unstable than the other three initialization methods.

Post-training and vocabulary-expansion provide us with a trade-off between performance and efficiency. Yet both methods prove to be useful.

4.4 Domain-specific post-training with the expanded tokenizer

Further, we combine post-training with vocabulary-expansion. This approach is similar to the last subsection, except for a post-training stage. Results are shown in table 9.

BioASQ	**mean(std)**
fine-tuning	0.6799 (0.05388)
fine-tuning + post-training	0.7383 (0.02610)
fine-tuning + post-training **+ vocab-expansion(rand)**	0.7181 (0.01678)
fine-tuning + post-training **+ vocab-expansion(zero)**	0.7403 (0.02143)
fine-tuning + post-training **+ vocab-expansion(unk)**	0.7275 (0.02218)
fine-tuning + post-training **+ vocab-expansion(mean)**	**0.7463 (0.00654)**

Table 9: Result of post-training with the expanded tokenizer and different methods of new embedding initialization.

5 Conclusion and Future Work

In this project, we explore how post-training, task-adaption, and vocabulary-expansion can be helpful to improve the domain-general pre-trained language model's performance on domain-specific downstream tasks.

Gururangan et al., 2020 shares a similar goal with our work. Its 'domain-adaptive pre-training (DAPT)' is close to our post-training and 'task-adaptive pre-training (TAPT)' is close to our task-adaption. However, our method takes much less training time and still achieves a comparable result on ChemProt.

Sachidananda et al., 2021 also considers the initialization of new word embeddings and provides two useful algorithms. Both take hours to run, involving indexing the corpora and conducting subtoken sequence counts. Although our four approaches (rand, zero, unk, and mean) are relatively simple, they only take minutes to run and can still have some performance gains.

Since it is only a course project, we do not do enormous experiments to explore every variable. But we believe these open up many directions for future discovery, such as the choice of post-training corpus, the choice of added vocabulary, and the combination of task-adaption and vocabulary-expansion.

Although we focus on the medical domain this time, it is believed that these methods can be generalized to other domains. Such pre-trained models with expert domain-specific knowledge will have many applications in the future. To this end, we are looking forward to more robust and effective methods.

References

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 ACL*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Vin Sachidananda, Jason S. Kessler, and Yi'an Lai. 2021. Efficient domain adaptation of language models via adaptive tokenization. *CoRR*, abs/2109.07460.