Team NLP-Crossing: Final Report CSE 481 N

Sam Lee, Shuheng Liu, Robin Yang

June 2020

Abstract

Text-based pretrained models are popular choices for people to utilize in today's NLP. We present a capstone project that replicates the study that investigate whether it is useful to apply a pretrained model to a task from a specific domain. We present a replication study across four domains (biomedical and computer science publications, news, and reviews) and five classification tasks. Additionally, we experiment with ways to improve a pretrained model's performance on tasks from target domains by finetuning on a more general purpose model. Finally, we perform analysis on the results of various attempted finetuning methods and provide suggestions on possible continuation of our work. Overall, we see consistent results that line up with the ones reported in the target paper of our replication study and explore different methods to improve the performance of the base model on a specific task.

1 Introduction

In recent years, increasing number of language models that were pretrained on enormous, diverse corpora, such as BERT [Devlin et al., 2018] and RoBERTa [Liu et al., 2019], have emerged and become the new standards. Despite the fact that they are some of the best models that produce state-of-the-art performances in various of tasks in the field of Natural Language Processing, the computational resources required to pretrain such models are often massive. In order to investigate their performances across different domains of study, we decide to do a replication study on a paper by Gururangan et al. [2020]. This leads us to ask whether general purpose models can achieve high performances in tasks from more specific domains (§2.1) and whether it is possible to finetune a base model, *i.e.* RoBERTa, without pretraining it on a specific domain to achieve similar performance in domain specific tasks (§2.2). It is essential to continue improving a model that performs well on general tasks so that it can produce competitive results even in more specific tasks. In addition, computing resources required for such high performance models may not be accessible to the general public. Therefore, in the second part of our project, we attempt to find out methods to improve a more general purpose model so that it can achieve similar performance to models pretrained on specific domains. Following the steps of Gururangan et al. [2020], we consider using the same four domains (biomedical and computer science publications, news, and reviews) and five classification tasks (two in news and one for each of the remaining domains). We adapt the terms and code for continued pretraining on the domain (which we refer to as *domain-adaptive pretraining* or **DAPT**) as well as continued pretraining on task-relevant corpus (which we refer to as *task-adaptive pretraining* or **TAPT**) so that we can fairly compare our replication results against the ones reported by Gururangan et al. [2020].

2 Experiments

In this section, we will describe the experiments that we conducted for our project, divided into the subsections of replication study and finetuning.

2.1 Replication Study

According to Gururangan et al. [2020], if we apply DAPT or TAPT to RoBERTa [Liu et al., 2019], we will be able to achieve better performances as compared to the vanilla RoBERTa on respective domains and tasks. The main objective for this part of the project is to replicate this finding.

2.1.1 Experiment Setup

There are two tasks in each of the four domains in Gururangan et al. [2020]. Due to the time constraint, our replication study only selected one task in each domain with one additional task in News. We use the code posted by the author of the paper[1], with slight modification of some hyperparameters done due to the memory restriction on the GPU that we were using, which is posted in our repository[2].

2.1.2 Results and Discussion

The selected five tasks are ChemProt (Biomedical), SciERC (CS), IMDB (Reviews), HyperPartisan and AGNews (News). Here are the results obtained from our experiments, with comparison to the results from Gururangan et al. [2020]:

Domain	**Task**	**Source**	**RoBERTa**	**DAPT**	**TAPT**	**DAPT + TAPT**
Biomedical	ChemProt	DSP	$81.5_{1.1}$	$83.0_{0.7}$	$82.0_{0.2}$	$84.8_{0.5}$
		NLP-Crossing	$81.6_{1.0}$	$83.6_{0.9}$	$82.7_{0.4}$	$83.9_{0.4}$
CS	SciERC	DSP	$78.9_{1.6}$	$79.8_{1.2}$	$79.6_{1.5}$	$80.2_{1.5}$
		NLP-Crossing	$80.4_{0.7}$	$81.4_{1.5}$	$79.9_{1.0}$	$80.5_{0.7}$
News	AGNews	DSP	$93.5_{0.4}$	$93.6_{0.2}$	$94.0_{0.2}$	$94.2_{0.1}$
		NLP-Crossing	93.3	93.6	94.3	95.2
News	HyperPartisan	DSP	$86.6_{0.9}$	$88.2_{5.9}$	$90.4_{5.2}$	$90.0_{6.6}$
		NLP-Crossing	$87.2_{4.8}$	$84.4_{10.6}$	$80.8_{8.6}$	$86.7_{5.4}$
Reviews	IMDB	DSP	$94.5_{0.1}$	$95.1_{0.1}$	$95.1_{0.2}$	$95.2_{0.1}$
		NLP-Crossing	94.5	95.0	95.7	95.3

Table 1: Results on different phases of adaptive pretraining as well as RoBERTa, comparing to the results from *Don't Stop Pretraining* (DSP, [Gururangan et al., 2020]). In each cell, the upper row represents the result from Gururangan et al. [2020] and the lower row represents the result from our experiments. Reported results are the test macro-F_1, except for the task ChemProt, which is in test micro-F_1 (accuracy). The result is in the form of $\text{mean}_{\text{std. dev.}}$. If the standard deviation is not present, the number represents the result from one run of the experiment. The original results are generated by running each experiment over 6 points across 5 seeds, while our results, if not only from one run, are generated by running each experiment once across 5 seeds.

The reason for us to run some experiments only once while multiple times for the others is that AGNews and IMDB both have relatively small standard deviations for all models (less than 0.5), and therefore it seems reasonable to run the experiments only once to confirm that the reported result is accurate. From Table-1, we can see that most of our results fall into one standard deviation of the reported results. In the other results, for which we calculate the means and the standard deviations, it can be observed that most of the results are close to the results reported in Gururangan et al. [2020], and the only discrepancy, which occurs while running HyperPartisan, can be attributed to the small size of the dataset that it was trained and tested on, as indicated in Gururangan et al. [2020].

[1]https://github.com/allenai/dont-stop-pretraining

[2]https://github.com/minhsuanlee/cse481n-20sp-NLP-Crossing

Nevertheless, it can be concluded that our results match those reported in Gururangan et al. [2020], as both of them show that additional pretraining, whether it is domain-specific or task specific, improves the model's performance, and a combined domain-specific and task-specific pretraining results in a better performance compared to only applying only one type of pretraining. Seeing that pretraining benefits the models performance, we are incentivized to carry out the second part of our project to finetune the base model (RoBERTa in this case), which could potentially be less computationally expensive, to achieve comparable results in the tasks above when there is limited computational resource for pretraining.

2.2 Finetuning

2.2.1 Experiment Setup

RoBERTa [Liu et al., 2019] has shown its reliability on many languages tasks, but pretraining such a model can be computationally expensive. While users are often unable to find available task specific pretrained models, they can find a pretrained model for general purpose easily. A popular approach is to use transfer learning, by which a user selects a related base model and performs additional training on a different dataset. This approach can be effective but still requires extensive time and computing power. Instead, we freeze original model's parameters and add additional layers to the end of the model, and train on the added layers.

We select RoBERTa as our base model and experiment on approaches to improve the model performance on the IMDB dataset [Maas et al., 2011]. This dataset contains 50,000 IMDB reviews split evenly into 25,000 train and 25,000 test sets. The overall distribution of labels is balanced (25,000 positive and 25,000 negative). We choose this dataset because it is easily understandable. We choose RoBERTa as our base model since the replication study provides us with a baseline to compare. To get control over layers, instead of using the code from Gururangan et al. [2020], we implement our own code with RoBERTa as one of the modules.

2.2.2 Results

We experiment on different types of layers including linear layer, attention layer, convolutional layer, and recurrent layer. The model has an output dimension of 768. The baseline approach is to add a dropout layer after the RoBERTa module [Liu et al., 2019], with an arbitrarily selected drop rate of 0.3, followed by a linear layer with an output dimension of 2, indicating the two labels (positive and negative). In the rest of the experiments, additional layers are added before the dropout layer. We train until the F_1 score converges or if there is no trend of converging. Because we use self-written code that has RoBERTa as one of the modules, the baseline F_1 score is slightly different than the score in our previous section, which is 94.0 after 10 epochs.

One linear layer 389 We add one additional linear layer with input dimension 768 and output dimension 389. We have a tanh layer after this linear layer. The F_1 score after 30 epochs is 92.8.

One linear layers 768 We add one additional linear layer with input dimension 768 and output dimension 768. We have a tanh layer after this linear layer. The F_1 score after 30 epochs is 91.2.

Two linear layers We add two additional linear layers. Both of them have input dimension 768 and output dimension 768. We have tanh layers between linear layers. The model does not show a trend of converging after 25 epochs.

Attention layers We add multi-head attention layer [Vaswani et al., 2017]. We arbitrarily choose the number of heads to be 8. The dimensions of query, value, and key are all 768. We expect attention to take a longer training time, but the model does not show a trend of converging after 50 epochs.

Convolutional and recurrent We do not find convolutional and recurrent layers to be appropriate because the input is one-dimensional. We manually add one additional dimension with size 1 but the model does not show a trend of converging.

2.2.3 Discussion

In the experiments above, we first perform a replication study and obtain similar performance with multiple pretrained models on various tasks of the targeted domains. Then, we explore further with different methods of finetuning on RoBERTa to see if there exists a more efficient way to enable a general purpose model to perform as well as a domain or task specific pretrained model on the relevant task in that domain. We find that a good general approach with finetuning to achieve our intent is unexpectedly difficult. We expect the two linear layers trial to perform better than the one linear layer trial, but the experiment result shows otherwise. This might be due to loss of features in the tanh layers between linear layers. However, non-linear layers are necessary between linear layers and should not be omitted. It is expected that convolutional layers does not work because it is not a suitable layer to use in NLP tasks. Recurrent layer is commonly used in NLP tasks, but the 768 dimension input at this point is very likely to have already lost the positional relationship that is critical to recurrent layers.

In general, the vector output by the pretrained module before the original projection layer is a very limited representation of the initial input. We expect attention layers to boost our model performance as attention layers are critical components in transformers including RoBERTa. One significant reason for its low performance is that the input vector for this attention layer contains too little information.

Having this in mind, we have a potential explanation for the relatively good performance of a general pretrained model on domain-specific tasks. The pretrained model is trained and released with the last linear layer removed. It is designed in a way such that the user could get similar model performance by training only on a linear layer. Then a change in the linear layer could possibly have a negative influence on the final result because the pretrained model is not designed to support that layer structure. One way to improve the model's ability to generalize could be to train a model and release the model without the complex layer. The user would take the model, add the same complex layer, and train on the complex layer. This would be something worth investigating in the future.

3 Related Work

We referred to the website Valkov [2020] to implement our code for classification using Hugging Face[3] models for finetuning.

4 Conclusion & Future Work

In the replication study, we replicate the results from Gururangan et al. [2020] which uses RoBERTa as a baseline and compares the domain and task specific pretrained models on four different domains. We found that our results match the results in Gururangan et al. [2020] and models that are pretrained on a desired domain or task perform better than using RoBERTa without further pretraining. This becomes the incentive for us to attempt to finetune RoBERTa, which, if successful, will be more accessible to the general public as it will be less computationally expensive.

Regarding finetuning, we have not yet found a good approach to improve the performance of RoBERTa on specific domains, but we conclude that an output with higher dimension and greater size from the pre-trained model allows for more flexibility and potentially better performance.

For replication study, one can carry out the three tasks remaining in Gururangan et al. [2020] and other experiments to confirm their findings. Future work for finetuning may include finding the types and number of layers that suit the purpose of our project the best, as well as conducting hyperparameter search on the added layers.

[3]https://huggingface.co/allenai

Acknowledgments

The authors would like to thank Jesse Dodge and Kelvin Luu for helping us formulate and scope this project, and thank Suchin Gururangan, one of the authors of Gururangan et al. [2020], for clarifying details in his paper. The authors would also like to thank Tobias Rohde for giving us suggestions on finetuning, thank members of Team Null for providing us feedback on our Second Advanced Solution Attempt, and thank the CSE481N (Spring 2020) staff members and our classmates for giving us feedback and comments along the way.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks, 2020.

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

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P11-1015`.

Venelin Valkov. Sentiment analysis with bert and transformers by hugging face using pytorch and python, Apr 2020. URL `https://www.curiousily.com/posts/sentiment-analysis-with-bert-and-hugging-face-using-pytorch-and-python/`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.