SGAdapters Project Report : CS 7643

Meihan Liu
Georgia Tech
Atalanta, Georgia
liumhan@gatech.edu

Ng Wei Xuan
Georgia Tech
Atalanta, Georgia
wng33r@gatech.edu

Jianbin Chen
Georgia Tech
Atlanta, Georgia
jchen868@gatech.edu

Lee Wei Xuan
Georgia Tech
Atlanta, Georgia
wlee385@gatech.edu

Abstract

Pre-training transformer language models with domain or task data followed by fine-tuning has shown to improve performance for natural language processing tasks. However, such pre-training and fine-tuning approach suffers from high requirements of computation resources and low portability due to the large model size. Recently, adapters have demonstrated a parameter-efficient solution to achieve similar performance as full fine-tuning on the GLUE benchmark. To study adapter's effectiveness, we apply adapters to RoBERTa baseline model [9] and fine-tuning on all the tasks and domains in Gururangan et. al (2020) [4]. Adapters attained near state-of-the-art performance and achieved comparable results as TAPT and DAPT model for SCIERC and HYPERPARTISAN respectively, which demonstrates adapter's potential as parameter-efficient training strategy.

1. Introduction

The state-of-the-art performance on the benchmark datasets are achieved via fine-tuning the pre-trained models trained on large datasets [4]. However, this process is computational intensive and inefficient. It is also memory intensive to store hundreds of MB of model parameters for each task. Adapters have been demonstrated to be an effective transfer learning strategy which enables parameter-efficient model pre-training without the need to fine-tune all model weights. This is achieved by integrating a model with adapter module which is much smaller in size compared with the model (see Appendix Figure 3). Thus, Adapters are modular and easily shareable. Adapters architecture consists of one attention layer and various feedforward layers, with the number of feedforward layers and hyperparameters differing.

We are motivated by this as we encounter similar issues when it comes to text mining or Natural Language Processing tasks in the real world. It is expensive and resource intensive to conduct a standard fine-tuning of large pre-trained language models such as RoBERTa on every task domain. We want to tap on the state-of-the-art pre-trained language models but do not want to be limited by the compute resources to train the large number of parameters.

In this study, we attempt to achieve comparable or even better results via the usage of adapters as compared to pre-trained RoBERTa models fine-tuned on downstream task data. The models integrated with adapter modules will only be fine-tuned on task data for the adapter layers and the last classification layer. To study the effectiveness of adapters, we conducted experiments on all eight task datasets from four domains including BIOMED, CS, NEWS and REVIEWS in [4]. The results demonstrate that adapter-based model tuning can achieve comparable results as standard fine-tuning, which saves the hefty and computationally intensive step of fine-tuning the whole language model.

The results of this study demonstrate the potential of adapters to be applied in various real-world applications. It is possible to deploy only one base model and multiple plugable and portable adapter modules for multiple tasks instead of fine-tuned model one for each task, which is much more resource efficient. And adapter-based models have much fewer parameter to train which saves the amount of time and computing resources required and thus allows much faster iteration.

2. Standard Fine-tuning

We attempted to replicate the RoBERTa, Domain Adaptive Pre-Training (DAPT) and Task Adaptive Pre-Training (TAPT) fine-tuning results of [4]. In our experiments, we utilised the base RoBERTa model from Hugging Face[1] and the pre-trained DAPT and TAPT models shared by the authors[2]. The results will serve as the basis for comparison against adapter-based tuning performance. The main framework used is provided by Adapterhub[3], which is based on PyTorch.

[1] https://huggingface.co/roberta-base
[2] https://github.com/allenai/dont-stop-pretraining
[3] https://https://adapterhub.ml/

2.1. Experiments

Following standard practice [3], we passed the final layer [CLS] token representation to a task-specific feed forward layer for prediction. Macro-F1 is used for evaluating all the tasks except for CHEMPROT and RCT where micro-F1 is applied. The hyperparameters as listed in Table 14 of [4] was used accordingly. All tasks were run across five random seeds, and the average F1 score and its standard deviation were reported.

The same base language model RoBERTa was used to reproduce the benchmark model results for the basis of comparison. RoBERTa was also the state of the art for various benchmark data sets and would serve as a meaningful baseline to the 4 domains. This is kept similar to the base language model used in [4].

Code Repository used: https://github.com/Adapter-Hub/adapter-transformers

2.2. Results

The reproduced results from fine-tuning all layers of the respective models were shown in Table 1 (see Table 4 in Appendix for validation results). DAPT and TAPT improves over RoBERTa in almost all domains, which is consistent with the paper's results. Although DAPT does not improve performance on the HYPERPARTISAN task, the result is within one standard deviation of the ROBERTA result. And for RCT, its TAPT performance is poorer than that of base RoBERTa, which we will explain later in this section. Comparing the performance across Roberta, DAPT and TAPT, DAPT achieves best the performance for 5 of the tasks while TAPT exceeds DAPT for HYPERPARTISAN, AGNews and HELPFULNESS. This result roughly concurs with the papers' findings between DAPT and TAPT models. On average, our reproduced results are about 1 to 5% lower than those reported respectively in the paper for all tasks, except for HELPFULNESS and IMDB where we obtain about 10% discrepancy. (Insert possible reasoning for the discrepancy.)

One possible reason that the reproduced results did not match the paper's exactly could be due to the removal of degenerated seeds. The author reported that the degenerated seeds were discarded and re-sampled while our team kept all the results, and we actually observed that for some of the experiments, there could be one or two seeds that lead to much poorer performance than the other seeds in the same experiment setting. This is one of the lessons we learned that bad initialization can result in degenerated performance, and thus we should re-sample the random seed and rerun the experiment in future experiments. Another reason could be due to the different hardware used to train the models. Our models were trained on shared resources on Google Colab where the computation hardware across different models may be inconsistent across runs. In addition, due to the limited resources and time, for the RCT task, we only sampled a subset of the training, development and test data instead of using the full dataset as in [4], which may create certain discrepancies.

Dom.	Task	RoBA.	DAPT	TAPT
BM	CHEMPROT	$80.6_{0.2}$	$\mathbf{83.1_{0.6}}$	$82.3_{0.2}$
	RCT	$84.7_{0.3}$	$\mathbf{85.6_{0.6}}$	$84.1_{0.7}$
CS	ACL-ARC	$61.2_{0.8}$	$\mathbf{72.8_{1.8}}$	$63.2_{4.4}$
	SCIERC	$77.5_{2.6}$	$\mathbf{78.9_{2.8}}$	$77.6_{1.0}$
NEWS	HYP.	$86.0_{3.7}$	$85.7_{3.1}$	$\mathbf{87.1_{2.6}}$
	AGNews	$92.9_{0.4}$	$93.0_{0.3}$	$\mathbf{93.6_{0.2}}$
REV.	HELPFUL.	$54.4_{0.03}$	$55.5_{0.1}$	$\mathbf{59.2_{0.2}}$
	IMDB	$84.3_{0.02}$	$\mathbf{88.5_{0.01}}$	$87.0_{0.01}$

Table 1: Reproduced RoBERTa base / DAPT / TAPT results. Reported results are test macro-F1, except for CHEMPROT and RCT for which micro-F1 is reported following Beltagy *et al*. (2019). We report averages across five random seeds, with standard deviations as subscripts.

During our experimentation, we found that learning rate and early stopping criteria affects the models' performance the most; finding a proper combination of learning rate and stopping criteria is thus critical. Since the models were sensitive to the learning rates, we tried varying the learning rates to find the best performance we can achieve given the limited time and resources. The best results still approximately converge to a learning rate of 2e-5 as reported in [4]. No significant signs of overfitting were found for any of the replicated results (refer to Table 4 in the Appendix for the results with development datasets), signalling that the learning rate and training epochs used were adequate.

3. Adapter-based Tuning

Domain and task adaptation of the RoBERTa language model prior to task fine-tuning gives better results than task fine-tuning alone as demonstrated by the results in Table 1 and in [4]. We would like to understand if it is possible to achieve similar results with adapter-based methods. Furthermore, adapters ([5, 14]) have appeared as a parameter-efficient alternative to standard fine-tuning which uses 100% task-specific parameters. We thus wonder if adapter-base tuning could achieve similar or even better performance than fine-tuning. And inspired by [5], we explore the trade off between performance and adapter size. We show that adapters achieve parameter efficient transfer for all the tasks we have explored.

3.1. Experiments

We applied adapter to RoBERTa baseline model and finetune only weights of the adapter and classification head

Dom.	Task	RoBA.	Default Adapter	Houlsby	Pfeiffer	Custom	Best of TAPT/DAPT
BM	CHEMPROT	$80.6_{0.2}$	$80.3_{0.7}$	$82.4_{0.6}$	$79.5_{1.9}$	$79.3_{0.7}$	$\mathbf{83.1_{0.6}}$
	RCT	$84.7_{0.3}$	$84.3_{0.5}$	$84.0_{0.2}$	$84.4_{0.4}$	$82.9_{0.1}$	$\mathbf{85.6_{0.6}}$
CS	ACL-ARC	$61.2_{0.8}$	$66.2_{4.6}$	$56.3_{4.6}$	$61.5_{4.1}$	-	$\mathbf{72.8_{1.8}}$
	SCIERC	$77.5_{2.6}$	$79.8_{1.1}$	$\mathbf{80.0_{1.3}}$	$78.1_{3.0}$	$79.7_{1.3}$	$78.9_{2.8}$
NEWS	HYP	$86.0_{3.7}$	$85.9_{0.8}$	$86.5_{1.8}$	$86.5_{4.4}$	$\mathbf{90.0_{1.8}}$	$87.1_{2.6}$
	AGNews	$92.9_{0.4}$	$\mathbf{93.7_{0.1}}$	$93.3_{0.5}$	$93.4_{0.4}$	$93.6_{0.3}$	$93.6_{0.2}$
REV.	HELPFUL	$54.4_{0.03}$	$\mathbf{64.8_{0.02}}$	$55.8_{0.5}$	$61.8_{0.02}$	-	$59.2_{0.2}$
	IMDB	$84.3_{0.02}$	$94.1_{0.1}$	$\mathbf{95.0_{0.02}}$	$93.7_{0.9}$	-	$87.0_{0.01}$

Table 2: F1 values of different adapters compared to baseline Roberta model and the best among TAPT and DAPT fine-tuned models for the test set of each task.

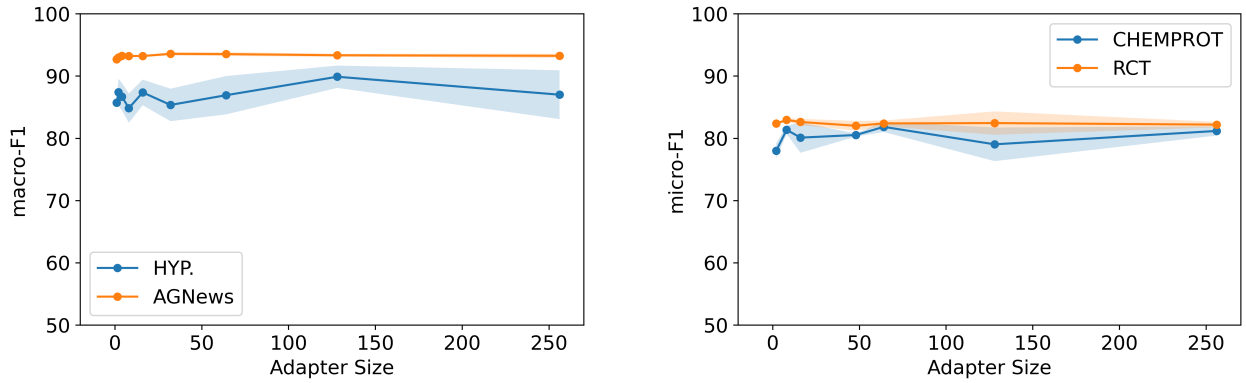

Figure 1: F1 score versus adapter size for the 4 tasks in NEWS and BIOMED domain. Reported results are for test set. We compare adapters of different sizes: (i) Adapter tuning for HYPARTISAN and AGNews with an adapter sizes 2^n for n = 0 ... 9. (ii) Adapter tuning for CHEMPROT and RCT with an adapter sizes of 2, 8, 16, 48, 64, 128 and 256. The lines and shaded areas indicate the mean and standard deviation across three random seeds.

on task-specific training data, while freeze other model weights. We evaluated adapter-based models against all four domains and eight tasks as in [4], see Table 6 for details on the datasets. For all the tasks, we experimented with different adapter architectures and configurations that were published before [5, 13] and the default architecture from AdapterHub. The main difference between the adapters proposed by Houlsby [5] and Pfeiffer [13] is a) the number of feedforward layers within the adapter head, with Houlsby proposing 2 and Pfeiffer 1; b) the activation function used between the feedforward layers where Houlsby used Swish and Pfeiffer used ReLU; c) Pfeiffer added a normalization layer before the adapter. For some of the tasks, we also tried customized architecture with changes including different activation functions or changing the normalization layer setting, and achieved a new state-of-the-art results in performance (see Appendix Figure 4). In addition, as mentioned above, we anticipated that learning rate and adapter size might be important parameters for a successful model. We thus explored a wide range of learning rates and adapter size values and compared their performance for some of the tasks.

The data was prepossessed by using the Roberta base tokenizer provided by Adapterhub. The tasks are mainly classification tasks and therefore a cross-entropy loss is utilized in the classification head. Hyperparameter search on learning rate, adapter size, batch size and early stopping criteria were performed on a task-by-task basis and results reported below represent the best of each task.

Code Repository used: https://github.com/google-research/adapter-bert

3.2. Results

3.2.1 Performance of adapters across different tasks

Table 1 shows the results of the adapter based tuning, where RoBERTa refers to the base model trained without an adapter. The default adapter is the one provided by Adapterhub. As shown in Table 5, adapters-based tuning

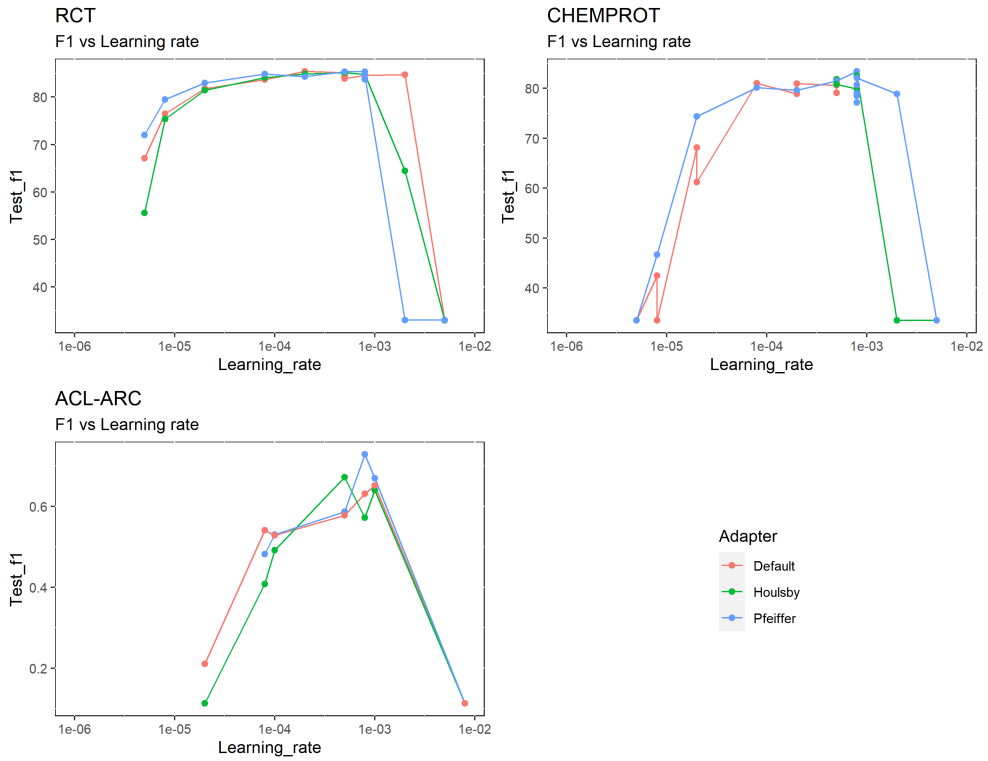

Figure 2: Test set F1 for different learning rate of the default, Houlsby and Pfeiffer adapter across three different tasks. Log-scaled xais are shown. Learning rate tested were between 5×10^{-6} to 8×10^{-3}.

Dom.	Adapter FT on	Classification FT on	Test Data	Dev Data
BM	CHEMPROT	RCT	$74.1_{0.6}$	$77.1_{0.8}$
	RCT	CHEMPROT	$44.6_{0.4}$	$44.4_{1.2}$
CS	SCIERC	ACL-ARC	$26.2_{2.8}$	$36.5_{1.9}$
	ACL-ARC	SCIERC	$26.8_{2.9}$	$30.3_{4.2}$
NEWS	AGNews	HYP	$75.0_{4.4}$	$77.0_{4.4}$
	AGNews	HYP	$89.9_{0.3}$	$90.4_{0.2}$
REV.	IMDB	HELPFUL	$61_{0.02}$	$61.8_{0.02}$
	HELPFUL	IMDB	$92.1_{0.02}$	$92.0_{0.02}$

Table 3: Cross Task Transfer within domain.

achieved better performance (0.6-10.7% increase) than our baseline RoBERTa model for seven out of eight tasks, except for RCT which we achieved comparable performance. This result indicates that adapters, as a parameter-efficient tuning strategy, can achieve comparable or even better performance across domains and tasks. While there is no one golden adapter architecture suitable for all tasks, different adapter architectures exhibited different performance on the same task. For example, in the REVIEW domain, the default adapter achieves the highest performance in the HELPFULLNESS task and is 10% better than that of Houlsby adapter on the same task, while the Houlsby adapter beats other adapters in the IMDB task. We also notice that different downstream task data presents different sensitivity to adapter configuration. For example, model performance on AG is not sensitive to adapter configurations compared with other tasks.

Comparing with our reproduced pre-trained TAPT or DAPT results, the adapter-based model achieve better performance than the best fine-tuning model across RoBERTa, DAPT and TAPT for SCIERC, HYP, HELPFULNESS and IMDB, and is comparable or better than either DAPT or TAPT for all the tasks except for ACL-ARC. Considering that our reproduced fine-tuning results are slightly worse

than the results reported by the paper [4], by comparing our adapter-based model performance with the paper results, we found that our adapter-based models achieved better performance for SCIERC and HYPERPARTISAN compared with the fine-tuning results of TAPT and DAPT respectively, although the adapter-based models were not further pretrained on domain or task data. Such results indicate that applying adapter on the RoBERTa baseline model can achieve comparable or even better performance without the need of pre-training the model using domain or task specific data. We believe that it is possible to push performance further for adapter-based RoBERTa models after more comprehensive hyperparameter search, and more experimentation and study on other adapter architectures and configurations. Moreover, even with only comparable performance, adapters show advantages in probabilities since fine-tuned adapters require only 5 to 7MB disk space for parameter storage while fine-tuned RoBERTa models typically occupy more than 500MB which hinders easy distributions.

Attempts to find a custom adapter architecture that outperforms the 3 other adapter architectures were made for all tasks except for ACL-ARC, HELPFULLNESS and IMDB. For the CHEMPROT and RCT tasks, the customized adapter was build by using four different types of linearity functions (tanh, relu, leaky relu and swish) were tested on top of adding a normalization layer before or after the adapter. None of the new architectures was comparable with the previous ones even with various hyperparameter tuning. For the SCIERC task, comparable results were found with an architecture with more normalisation layers added and a 'tanh' linearity. The rationale for this change was to test the impact of different activation functions and to determine if more normalization layers would improve results, more than what was shown in Houlsby et. al (2020). However, they remain very similar to the other adapters tested.

3.2.2 Adapter Size/Performance Trade-off

The adapter size (number of units in the bottleneck) controls the parameter efficiency. Smaller adapter size introduces fewer parameter but possibly at a cost of performance. We explore this trade-off for the four tasks in NEWS and BIOMED domain, the results are shown in Figure 1. The only adapter-specific hyperparameter we tune is the adapter size, and for each size we re-run three times with different random seeds to account for training instability. As we can see from the results, adapters yielded good performance across a range of sizes for all the four tasks.

3.2.3 Learning Rate Impact

When training the adapters, we tested the effect of varying the learning rates and we observed high sensitivity of the learning rate for all the three tasks tested (RCT, CHEMPROT and ACL-ARC) across two domains (BIOMED and CS). We found that for all the tasks, the performance increased from learning rates of 10^{-5} to 10^{-4}, but decreased at larger learning rates after 0.003 (Figure 2). Interestingly, the range of learning rates that allows effective learning differ across tasks. It seems that the RCT task has a wider range while the ACL-ARC task has a narrower range. In addition, among the three adapters tested (default, Houlsby and Pfeiffer), the trends were highly consistent, indicating that such sensitivity of learning rate is affected more by the task than by the architecture of the adapter.

3.2.4 Cross-Task Learning

The idea of performing cross-task transfer is to explore whether the representations learned by training adapters on one tasks can be helpful on another task in the same domain. For this experiment, we train adapter-based RoBERTa model on one task, but only fine-tune the classification layer on another task and examine its performance on this second task. The results are overall mixed. For the tasks in REVIEW domain, we achieved comparable performance although slightly worse as their corresponding adapter-based model, while the results were much worse for the other tasks (Table 3). We speculate that cross-task transfer is possible but largely for domains containing a narrower set of specialized linguistic features, or it may depends on if the task data encapsulates a wide range of domain related features.

4. Conclusion

Overall, we have demonstrated the effectiveness of adapters as a parameter-efficient way of model fune-tuning on downstream tasks. Adapter-based models achieve comparable or even better performance as fine-tuning. Especially for SCIERC, HYP, HELPFULNESS and IMDB, models using adapters achieved better performance as corresponding RoBERTa models pre-trained on domain or task data [4]. The trade-off between adapter size and learning rates were also studied, and the results showed that adapters yield good performance across a large range of adapter sizes from 1 to 256 for all the four tasks we studied. Future work can be done on studying the effectiveness of adapters as a substitute of standard pre-training to see if it can learn comparable or even better distributed representation than state-of-the-art pre-trained models. We can explore the ability of Adapters to fuse information learned from multiple tasks, with the proposed adapter fusion approach [13] that combines pre-trained adapters across various tasks.

5. Appendix

5.1. Work Division

We reported our results on four domains in this study. Each team member is responsible for one domain as specified in Table 8, summary of the team's contributions and details are provided in Table 8.

5.2. Project Code Repository

We used the pretrained and further pretrained models from the paper what is uploaded onto huggingface.
We used the documentation codes from AdapterHub as a reference for training the adapters.

The link to the code associated with the Don't Stop Pretraining ACL 2020 paper is at: *Github Link*
The Google Drive link to the jupyter notebooks for all experiments are at: *Google Drive Folder Link*.

5.3. Data sheet

Training, development and test data was downloade from *Github Link* [4] with original data sources of each task listed below: CHEMPROT [8], RCT [2], ACL-ARC [6], SCIERC [10], HYPERPARTISAN [7], AGNEWS [15], HELPFULNESS [12], IMDB [11].

5.4. Configuration of the best customized adapter

Table 7 listed the configuration of the best customized adapter we tested for each task.

For what purpose was the dataset created? Was there a specific task in mind? Was there a specific gap that needed to be filled? Please provide a description.

BIOMED - Academic papers are an important textual domain for NLP research. The format of academic papers present characteristics worth studying . Existing academic papers corpera covers small research domains and may not preserve the full text structure. The dataset was collected for named entity recognition, PICO elements, Dependency parsing, relationship extraction tasks.

CS - Academic papers are an important textual domain for NLP research. The format of academic papers present characteristics worth studying . Existing academic papers corpera covers small research domains and may not preserve the full text structure. The dataset was collected for named entity recognition, Dependency parsing, relationship extraction tasks.

NEWS - Text generative models have great applications but it could also give way to adversarial articles. The dataset was created with the intention of discriminating between real news and fake news.

REVIEWS - Amazon reviews was collected with the intention of training better recommender systems. The dataset was collected for recommendation and ranking tasks.

What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)? All datasets are text data representing documents of different domains.

How many instances are there in total (of each type, if appropriate)? The data composition utilised is the same as the task data used in [4] as shown in Table 6, with the exception of the RCT task data under the BIOMED domain which was subsampled randomly due to the limited time and computation resources that impedes full dataset training. The training and test datasets were subsampled randomly with 4000 entries while the development set was subsampled randomly with 2000 entries. Significant performance degradation was not found from the result of the original paper [4] (Table 1).

References

[1] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[2] Franck Dernoncourt and Ji Young Lee. PubMed 200k RCT: a dataset for sequential sentence classification in medical abstracts. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 308–313, Taipei, Taiwan, Nov. 2017. Asian Federation of Natural Language Processing.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019.

[4] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019.

[5] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[6] David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406, 2018.

[7] Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. SemEval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 829–839, Minneapolis,

Minnesota, USA, June 2019. Association for Computational Linguistics.

[8] Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Taboureau. Chemprot-3.0: a global chemical biology diseases mapping. *Database*, 2016, 2016.

[9] 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. *arXiv:1907.11692.*, 2019.

[10] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[11] 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.

[12] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes, 2015.

[13] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning, 2021.

[14] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters, 2017.

[15] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2016.

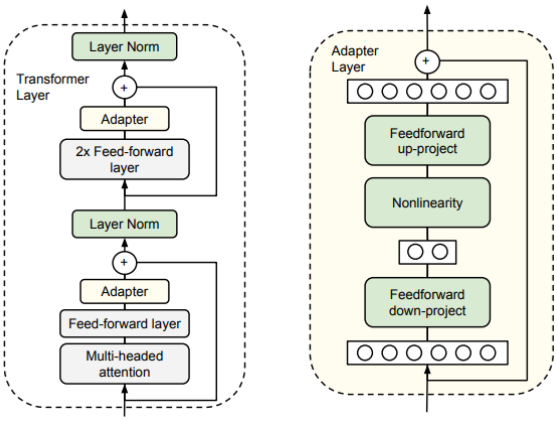

Figure 3: Architecture of the adapter module and its integration with the Transformer. Adapted from [5].

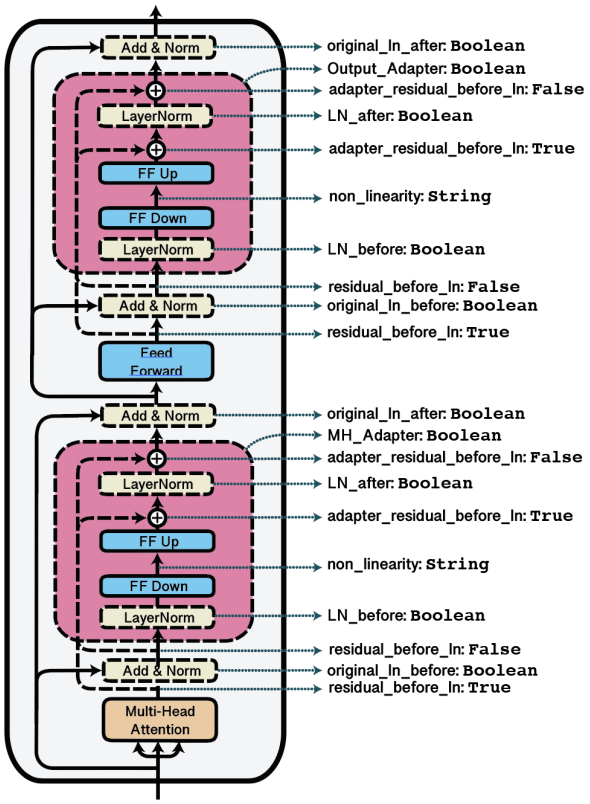

Figure 4: Architecture of the adapter and possible configurations for customization.

Dom.	Task	RoBA.	DAPT	TAPT
BM	CHEMPROT	$81.9_{1.1}$	$83.6_{0.3}$	$81.7_{1.8}$
	RCT	$84.2_{0.4}$	$85.8_{0.5}$	$84.1_{0.5}$
CS	ACL-ARC	$67.1_{0.6}$	$70.7_{5.8}$	$63.4_{7.1}$
	SCIERC	$81.6_{1.6}$	$85.2_{1.6}$	$82.2_{1.6}$
NEWS	HYP.	$83.5_{4.6}$	$83.4_{2.6}$	$81.9_{4.1}$
	AGNews	$93.4_{0.4}$	$93.3_{0.3}$	$93.9_{0.2}$
REV.	HELPFUL.	$54.4_{0.03}$	$59.9_{0.1}$	$56.2_{0.2}$
	IMDB	$84.6_{0.02}$	$88.6_{0.01}$	$87.6_{0.01}$

Table 4: Reproduced RoBERTa base / DAPT / TAPT results. Reported results are development macro-F1, except for CHEMPROT and RCT for which micro-F1 is reported following [1]. We report averages across five random seeds, with standard deviations as subscripts.

Dom.	Task	Default Adapter	Houlsby	Pfeiffer	Custom
BM	CHEMPROT	$80.9_{0.5}$	$83.0_{0.6}$	$80.5_{1.6}$	$80.3_{0.6}$
	RCT	$84.3_{0.7}$	$84.2_{0.7}$	$84.0_{0.7}$	$84.9_{0.6}$
CS	ACL-ARC	$64.9_{7.6}$	$58.8_{10.2}$	$65.9_{5.2}$	-
	SCIERC	$82.0_{1.5}$	$84.5_{1.1}$	$82.6_{3.3}$	$83.5_{1.9}$
NEWS	HYP	$87.7_{2.4}$	$84.4_{3.9}$	$85.1_{2.9}$	$84.8_{2.4}$
	AGNews	$93.9_{0.1}$	$\mathbf{93.7_{0.4}}$	$93.6_{0.4}$	$93.8_{0.3}$
REV.	HELPFUL	$64.8_{0.02}$	$\mathbf{58.8_{0.05}}$	$60.0_{0.3}$	-
	IMDB	$94.1_{0.1}$	$\mathbf{95.0_{0.03}}$	$93.6_{0.5}$	-

Table 5: F1 values of different adapters compared to baseline Roberta model and the best among TAPT and DAPT fine-tuned models for the DEVELOPMENT set of each task.

Domain	Task	Label Type	Train	Dev.	Test	Classes
BIOMED	CHEMPROT	relation classification	4169	2427	3469	13
	RCT	abstract sent. roles	4000 (original 180041)	2000 (original 30212)	4000 (original 30135)	5
CS	ACL-ARC	citation intent	1688	114	139	6
	SCIERC	Relation classification	3219	455	974	7
NEWS	HYPERPARTISAN	partisanship	515	65	65	2
	AGNEWS	topic	115000	5000	7600	4
REVIEWS	HELPFULNESS	review helpfulness	115251	5000	25000	2
	IMDB	review sentiment	20000	5000	25000	2

Table 6: Dataset used in this work

Task	Configuration
CHEMPROT	Houlsby Architecture with adapter size = 64 (reduction_factor = 12).
RCT	Pfeiffer Architecture with adapter size = 64 (reduction_factor = 12).
ACL-ARC	Default Architecture with adapter size = 64 (reduction_factor = 12).
SCIERC	Houlsby Architecture with adapter size = 64 (reduction_factor = 12).
HYP.	Houlsby Architecture with adapter size = 128 (reduction_factor = 6).
AGNews	Default Architecture with adapter size = 48 (reduction_factor = 16).
Helpfulness	Houlsby Architecture with adapter size = 64 (reductionfactor = 12)
IMDB	Default Architecture with adapter size = 64 (reduction_factor = 12)

Table 7: Adapter Configuration for the best performance adapters in each task. Unless specified, the configurations for above listed adapters are the same as default setting in https://docs.adapterhub.ml/classes/adapter_config.html

Student Name	Contributed Aspects	Details
Jianbin CHEN	Implementation and Analysis	1. Finetuned the DAPT and TAPT models and trained the adapters for the BIOMED Domain. Analyzed adapter-based tuning results. 2. Contributed to setting up adapter training code and cross-task training code 3. Report writing and revision for experiments, conclusion and created plots too.
Wei Xuan LEE	Implementation, analysis, report writing	1. Finetuned the DAPT and TAPT models and trained the adapters for the CS Domain. Analyzed adapter-based tuning results. 2. Contributed to setting up adapter training code and cross-task training code 3. Report writing and revision for abstract, section 1 to 3 and data sheet.
Meihan LIU	Implementation, analysis, report writing	1. Code implementation for training pipelines including standard fine-tuning, adapter-base tuning, changing adapter size/performance trade-off and cross-task learning. 2. Figured out how to run early stopping using adapter-transformers library. 3. Actively participated in discussion and troubleshooting. Contributed in experiments design and data analysis. 4. Report writing and revision for abstract, section 1 to 3 and conclusion. Created plots.
Wei Xuan NG	Implementation and Analysis	1. Finetuned the DAPT and TAPT models and trained the adapters for the REVIEWS Domain. Analyzed adapter-based tuning results. 2. Contributed to setting up adapter training code and cross-task training code 3. Report writing and revision for experiments, conclusion

Table 8: Contributions of team members.