Parameter-efficient Weight Ensembling Facilitates Task-level Knowledge Transfer

Xingtai Lv[1*], **Ning Ding**[2*], **Yujia Qin**[2], **Zhiyuan Liu**[2,3,4,5†], **Maosong Sun**[2,3,4,5†]
[1]Department of Electronic Engineering, Tsinghua University
[2]Department of Computer Science and Technology, Tsinghua University
[3]BNRIST, Tsinghua University, [4]Institute for Artificial Intelligence, Tsinghua University
[5]International Innovation Center of Tsinghua University, Shanghai
{lvxt20, dingn18, qyj20}@mails.tsinghua.edu.cn
{liuzy, sms}@tsinghua.edu.cn

Abstract

Recent studies show that large-scale pre-trained language models could be efficaciously adapted to particular tasks in a parameter-efficient manner. The trained lightweight set of parameters, such as adapters, can be easily stored and shared as a capability equipped with the corresponding models. Owning many lightweight parameters, we focus on transferring them between tasks to acquire an improvement in performance of new tasks, the key point of which is to obtain the similarity between tasks. In this paper, we explore 5 parameter-efficient weight ensembling methods to achieve such transferability and verify the effectiveness of them. These methods extract the information of datasets and trained lightweight parameters from different perspectives to obtain the similarity between tasks, and weight the existing lightweight parameters according to the comparability to acquire a suitable module for the initialization of new tasks. We apply them to three parameter-efficient tuning methods and test them on a wide set of downstream tasks. Experimental results show that our methods show an improvement of 5%~8% over baselines and could largely facilitate task-level knowledge transfer.

1 Introduction

Increasingly large pre-trained language models (PTMs) (Bommasani et al., 2021; Han et al., 2021; Raffel et al., 2020; Brown et al., 2020) have yielded exceptional performances on a variety of tasks but also suffer from prohibitive adaptation costs with full parameter fine-tuning. It is not a feasible choice to fine-tune all parameters of a colossal model for each specific downstream task and produce a corresponding instance at the same size. To overcome this obstacle, a branch of research, namely parameter-efficient tuning, has been actively developed and explored (Ding et al., 2023; Houlsby et al., 2019; Li and Liang, 2021; Lester et al., 2021; Hu et al., 2021).

It demonstrates that only optimizing a tiny portion of parameters and keeping the PTM frozen could achieve on-par performance with full parameter fine-tuning on many tasks. After training, the set of updated parameters is lightweight and portable for storing and sharing. Although the specific structures of these parameters may be different, we treat them in a unified perspective and call them *lightweight objects*. Once lightweight objects are trained, they can be adapted to specific datasets conditioned on a large-scale PTM and be placed aside for storage in a space-efficient manner. Due to its lightweight nature, it is pragmatic to build a platform to store and share such lightweight objects for various scenarios (Beck et al., 2022).

However, the maneuverability of the platform for storing and sharing lightweight objects is still not fully exploited. In the current paradigm, one could directly access and utilize lightweight objects trained on existing datasets but face hindrances in utilizing the knowledge of these objects for new datasets (Vu et al., 2021). Such existing lightweight objects can be a valuable resource, as the knowledge contained in them has the probability of transferring to similar tasks. In this paper, we focus on the transferring of lightweight objects and aim to leverage them to boost the performance of new datasets. We assume that more similar tasks can share more knowledge, so the key point is to acquire the similarity between existing lightweight objects and new tasks. Specifically, we first assess 3 straightforward approaches to facilitate parameter-efficient tuning on new datasets. Observing unsatisfactory results on such approaches, we develop a parameter-efficient weight ensembling framework that could produce a set of parameters according to new datasets and existing lightweight objects. Under the framework, we explore 5

[*] equal contributions
[†] corresponding authors

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 270–282

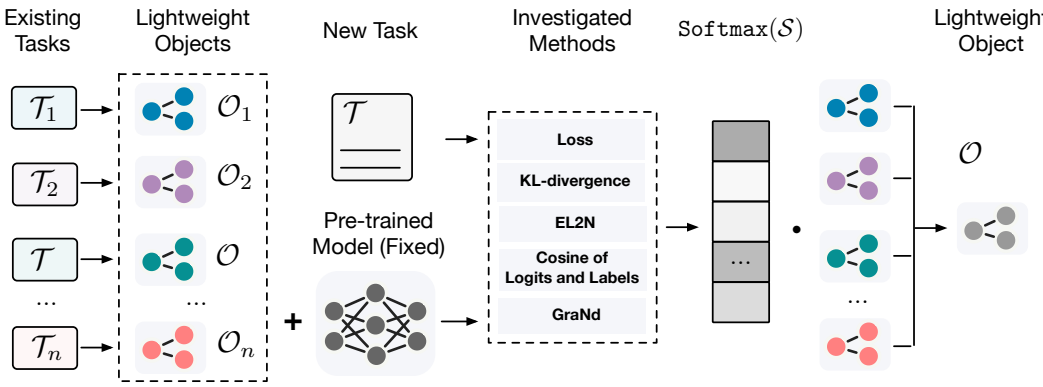

Figure 1: The framework of parameter-efficient weight ensembling. With the pre-existing lightweight objects $\{\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_n\}$ and the new task, the investigated methods are applied to obtain the indicator vector $\mathcal{S}$, the *i-th* element of which assesses the contribution of $\mathcal{O}_i$ on $\mathcal{T}_{\text{new}}$. The dot product of $\texttt{Softmax}(\mathcal{S})$ and $\{\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_n\}$ is the lightweight object for the initialization of the new task.

specific methods as a comprehensive study.

Extensive experiments across 8 transferring approaches, 51 downstream datasets, and 3 parameter-efficient tuning methods demonstrate that the parameter-based framework could considerably advance the performance compared to baseline methods. We further carry out experimental analysis to verify the compatibility and internal properties.

2 Investigated Methods

We consider a scenario where pre-trained language models $\mathcal{M}$ are *frozen*. And for downstream tasks $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_n\}$, the associated lightweight modules $\mathcal{O}^* = \{\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_n\}$ are produced via parameter-efficient fine-tuning (e.g., LoRA (Hu et al., 2021), Adapter (Houlsby et al., 2019)). Given a new task $\mathcal{T}_{\text{new}}$ with a few examples, our goal is to explore the best approach to utilize pre-existing lightweight objects to cultivate the best-performing lightweight object $\mathcal{O}_{\text{new}}$ for the initialization of $\mathcal{T}_{\text{new}}$. We investigate 9 strategies to transfer lightweight objects across tasks, including 3 baselines and 5 particular methods under our parameter-efficient weight ensembling framework.

2.1 Baselines

Straightforward methods are directly averaging all objects and further, averaging objects of similar tasks. These 2 methods and the random initialization way (From Scratch) are simple and intuitive, and we treat them as baselines.

From Scratch. This approach is the common parameter-efficient tuning pipeline. We train a randomly initialized lightweight object on the training set of the new task and evaluate it on the test set.

Avg. of Checkpoints. This approach straightforwardly takes the average of the checkpoints of all existing lightweight objects as the initialization of the new lightweight object. And the new object is trained on the training set and evaluated on the test set, formally, $\mathcal{O}_{\text{new}} \leftarrow \frac{1}{n}\sum \mathcal{O}_i$.

Manual Division. We manually select tasks that are similar to the new task based on the similarity of the task data and then average the lightweight objects corresponding to these tasks and use the result for initialization. This method is employed by Friedman et al. 2021.

2.2 Parametric Efficient Weight Ensembling

In accordance with the tenet of digging for more information about the transfer with less cost, we develop a parameter-efficient weight ensembling framework, the core of which is to obtain a similarity indicator. We explore 4 methods (Loss, KL-divergence, EL2N, and Cosine of Logits and Labels) acquiring the indicator mainly from the data and 1 method (GraNd) acquiring the indicator mainly from the weights under the framework.

We consider the procedure to exploit existing lightweight objects as a process of soft selection, similar to attention networks. Such a selection is conducted based on an indicator $\mathcal{S}_i$ that assesses the contribution of one existing lightweight object $\mathcal{O}_i$ on $\mathcal{T}_{\text{new}}$, the initialization of the new lightweight object can be obtained by

$$\mathcal{O}_{\text{new}} = \sum_{i=1}^{n} \texttt{Softmax}(\mathcal{S}_i/\tau) \cdot \mathcal{O}_i \qquad (1)$$

where τ is the hyper-parameter of the temperature indicator. Next, we introduce instantiations that are explored in this paper to construct different $\mathcal{S}$.

Loss. The output of the zero-shot loss function is a reasonable measurement under this circumstance. We directly feed examples of the valid data of $\mathcal{T}_{\text{new}}$ to $\mathcal{O}_i$ and compute the CrossEntropy loss $\mathcal{L}_i$ without any optimization. The indicator is set to the opposite of the loss output $\mathcal{S}_i = -\mathcal{L}_i$.

KL-divergence. We first train a randomly initialized lightweight object $\tilde{\mathcal{O}}_{\text{new}}$ on the training set of $\mathcal{T}_{\text{new}}$. Then we feed examples of the valid dataset of $\mathcal{T}_{\text{new}}$ separately to $\mathcal{O}_i$ and $\tilde{\mathcal{O}}_{\text{new}}$. For the two lightweight objects, we take the representation of the final layer of each output token t_{ij} through a softmax function to obtain the corresponding probability distributions $\mathcal{P}_{ij}$ and $\tilde{\mathcal{P}}_{ij}$. We then calculate the KL-divergence $\mathcal{K}_{ij}$ to assess the similarity of the two distributions that further contributes to the final indicator. After iterating the process for all the tokens, we add up the KL-divergence and take the opposite as the indicator $\mathcal{S}_i = -\sum_j \mathcal{K}_{ij}$.

EL2N. Similar to the foregoing method of KL-divergence, for a lightweight object $\mathcal{O}_i$, we obtain a probability distribution $\mathcal{P}_{ij}$ for each output token t_j in $\mathcal{T}_{\text{new}}$. At the same time, we construct a one-hot vector $\mathcal{V}_{ij}$ of the ground truth label. In this approach, we directly calculate the Euclidean distance of $\mathcal{P}_{ij}$ and $\mathcal{V}_{ij}$ as a measurement, denoted as d_{ij}. The final indicator is the inverse of the summation of d for all the tokens $\mathcal{S}_i = -\sum_j d_{ij}$.

Cosine of Logits and Labels. The process of this method is basically the same as the last approach, except we use the cosine function to calculate the measurement between $\mathcal{P}_{ij}$ and $\mathcal{O}_{ij}$.

GraNd. Gradients can be viewed as the amount of change a model needs to adapt to a specific task. Similar to the approach that involves loss function, we directly feed examples in $\mathcal{T}_{\text{new}}$ to a lightweight object $\mathcal{O}_i$ and calculate the CrossEntropy loss. Then for each layer of the lightweight object, we compute the gradient of parameters with respect to the cross-entropy loss. Then we calculate the sum of the squares of these gradients $\mathcal{G}_i$ and take the inverse of the rooting of the sum to get $S_i = -\sqrt{G_i}$.

The above methods consider the possible contributions of existing lightweight objects $\mathcal{O}_i$ to $\mathcal{O}_{\text{new}}$ from various technical angles for knowledge transfer. In the empirical study of the next section, we show that our approaches could substantially outperform existing baselines and tap the potential of existing lightweight objects.

3 Experiments

In this section, we apply the aforementioned approaches in different scenarios for experimental comparisons and analysis.

3.1 Experiment Settings

We use T5_{base} as the backbone model and choose 32 Question Answering (QA) tasks from *CrossFit Gym* (Ye et al., 2021) for evaluation. To further assess the generality of the investigated methods, we also experiment our methods on 19 more diverse tasks. All tasks are formulated into the text-to-text format. We iteratively treat each task as the upcoming new task and the remaining 31 tasks as existing tasks in the platform. In this way, we do 32 trials with different new tasks and get 32 results. After that, we average all 32 results as the final result. We randomly select a small amount of data from the original datasets of the new task. Specifically, the data of the *i-th* new task T_i could be represented as a tuple of ($\mathcal{D}^i_{\text{train}}$,$\mathcal{D}^i_{\text{dev}}$,$\mathcal{D}^i_{\text{test}}$), and the sizes of $\mathcal{D}^i_{\text{train}}$ and $\mathcal{D}^i_{\text{dev}}$ are both set to $16n$ for the n-classification tasks and 64 for other tasks. We apply our approaches to three popular parameter-efficient tuning methods (Adapter (Houlsby et al., 2019), LoRA (Hu et al., 2021), Prefix (Li and Liang, 2021)). Given that the PTM we mainly use is T5_{base}, and that the prompt tuning (Lester et al., 2021) method has significant convergence issues when applied to it, we choose not to experiment with prompt tuning method. Other experimental settings are shown in Appendix A.

3.2 Results and Analysis

Results on 32 QA Tasks. As reported in Table 1, we observe that: (1) the results of our approaches considerably outperform existing baselines in general, and the superiority holds for all three parameter-efficient methods. (2) The results of approaches that rely on the information from data (e.g., Cosine of Logits and Labels) are generally better than those resorting to the information from the weights (GraNd). (3) EL2N and Cosine of Logits and Labels, which directly extract information from the difference between logits and labels, perform best in knowledge transfer. We suspect that it would be easier to extract task features directly from the data under the framework.

Results on 51 Diverse Tasks. In addition to the QA tasks, we expand the size of evaluation datasets to 51 to further evaluate the knowledge transfer

Approach	Adapter	LoRA	Prefix
Baselines			
From Scratch	31.7	31.5	30.1
Avg. of Checkpoints	32.9	33.8	31.8
Manual Division	34.9	35.4	32.3
Parameter-efficient Weight Ensembling			
GraNd	34.5 (-0.4)	35.4 (+0.0)	35.1 (+2.8)
Loss	38.9 (+4.0)	38.7 (+3.3)	36.2 (+3.9)
KL-divergence	37.4 (+2.5)	37.1 (+1.7)	35.7 (+3.4)
EL2N	38.6 (+3.7)	39.1 (+3.7)	**37.3** (+5.0)
Cosine of Logits and Labels	**40.4** (+5.5)	**39.5** (+4.5)	37.2 (+4.9)

Table 1: Test results of existing baselines and our approaches. Numbers in parentheses are the difference between the method and the best-performing baseline.

across tasks, including classification, question answering, conditional generation, and others. Without losing generality, we use Adapter for the following experiments. As reported in Table 2, we could observe similar empirical conclusions as the QA experiments. In a more diverse setting, gradient information cannot reflect vital knowledge for cross-task transfer, resulting in unsatisfactory performance of GraNd. At the same time, the setting of more diverse tasks also makes knowledge transfer more difficult, which makes the gap between our approach and baseline slightly narrower.

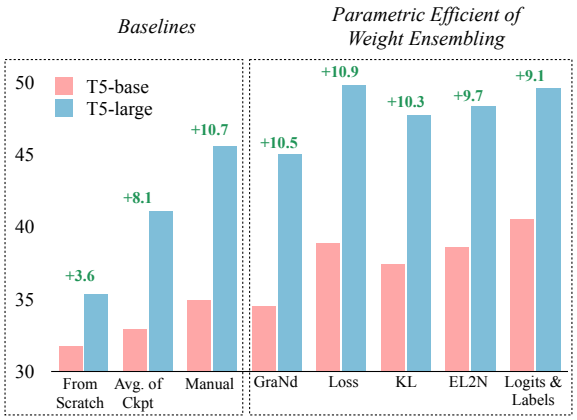

Figure 2: Performance comparisons of different backbones of each method.

Results with $T5_{large}$. We also investigate the impact of the backbone model. As illustrated in Figure 2, by directly replacing the backbone model from $T5_{base}$ to $T5_{large}$, we could observe that all the methods gain considerable improvements. Methods of parameter-efficient weight ensembling generally gain about 10% of improvement, indicating that our approaches allow for knowledge transfer under different backbone models and may achieve more significant results for large models.

Approach	All Task
Baselines	
From Scratch	41.7
Avg. of Checkpoints	41.8
Manual Division	44.6
Parameter-efficient Weight Ensembling	
GraNd	43.9 (- 0.7)
Loss	45.8 (+1.2)
KL-divergence	44.7 (+ 0.1)
EL2N	47.2 (+2.6)
Cosine of Logits and Labels	**47.8** (+3.2)

Table 2: The results on 51 downstream tasks with the adapter as the lightweight object.

Impact of the Number of Shots. In order to test whether our methods are effective under the data-rich scenario, we increase the amount of training data for the new task and conduct experiments similar to those described in **Results on 32 QA Tasks.** Specifically, we set the sizes of $\mathcal{D}^i_{\text{train}}$ and $\mathcal{D}^i_{\text{dev}}$ to $N \times k$ for the N-classification tasks and $4k$ for other tasks. In this experiment, we set k to 32, 64, 128 and 512, respectively.

Without loss of generality, we apply our approaches only to adapter-tuning, and experiment with the From Scratch, Manual Division and Cosine of Logits and Labels approaches when k is 128 and 512. It is worth mentioning that we use different hyper-parameters in experiments with different amounts of data. We use the hyper-parameters in the *few-shot* line in Table 5 in Appendix A when k is 32 and 64 and use the hyper-parameters in the *full data* line in Table 5 in Appendix A when k is 128 and 512.

The specific results are listed in Table 3, from which we conclude that (1) the results of our approaches outperform existing baselines in general; (2) the setting of more data makes the gain directly from original datasets more abundant, resulting in less gain from existing lightweight objects, which makes the gap of results between our approach and baseline narrower.

Analysis of Module Importance. To analyze the importance of different modules of the lightweight object in knowledge transfer, we experiment based on the modified GraNd approach, which can extract the information of a certain module more in-

K	32	64	128	512
Baselines				
From Scratch	33.5	36.3	38.8	45.4
Avg. of Checkpoints	35.1	37.7	-	-
Manual Division	37.3	39.2	41.5	47.6
Parameter-efficient Weight Ensembling				
Loss	40.5	42.6	-	-
KL-divergence	39.2	41.5	-	-
EL2N	**41.1**	43.0	-	-
Cosine of Logits and Labels	**41.1**	**43.7**	**43.9**	**47.9**

Table 3: The (test) results of the parallel experiment where we increase the amount of data for the new task.

dependently. Specifically, we compute the gradient of parameters with respect to the cross-entropy loss for one particular part $\mathcal{P}$ of the lightweight object. The base model we choose, T5_{base}, consists of 12 encoder blocks and 12 decoder blocks, and every decoder block has 3 sub-layers (i.e., self-attention layer, cross-attention-layer, and feedforward layer). Taking $\mathcal{P}$ as the Adapter layers in these $12 + 12$ blocks in turn, we apply the modified **GraNd** approach and acquire 24 results. The respective averages of 12 results corresponding to the encoder and 12 results corresponding to the decoder are listed in Table 4. Similarly, we take $\mathcal{P}$ as the Adapter layers in the 3×12 sub-layers in the decoder blocks, considering the results related to decoder are better than those related to encoder, in turn, and obtain 36 results. We respectively average the 12 results of self-attention layers, cross-attention layers, and feedforward layers and acquire 3 results that are listed in Table 4. All layer-wise results are shown in Appendix B.

These results could reflect the importance of different modules in parameter-efficient knowledge transfer. We observe (1) the results of the decoder blocks are higher than those of the encoder blocks, indicating more importance of the decoder; (2) in decoder blocks, cross-attention layers produce lower results than self-attention layers and feed-forward layers, and it demonstrates that information that is propagated in the decoder is crucial for knowledge transfer.

4 Conclusion

This paper investigates task-level knowledge transfer under the scenario of parameter-efficient tuning. We empirically explore 8 strategies to use existing lightweight objects to perform knowledge transfer for the adaptation of new tasks. Experimental results and analysis show that our methods could effectively utilize knowledge distributed in lightweight objects. We expect our exploration could facilitate the development and application of parameter-efficient tuning of large language models.

Encoder Block	Decoder Block	Self-Attention Layer	Cross-Attention Layer	FF Layer
32.7	36.2	36.95	33.88	37.01

Table 4: The (test) results of the parallel experiment where we apply the modified GraNd approach on different blocks and layers of the PLM.

Limitations

Our approaches that are developed in the parameter-efficient weight ensembling framework, and experiments have the following limitations. First of all, our framework cannot efficiently extract information from the parameters of the trained lightweight objects, resulting in relatively unsatisfactory performance of the approach resorting to the information from the weights, i.e., GraNd. Furthermore, the modules that we focus on in our analysis of module importance are only blocks and sub-layers of the blocks. We have not probed finer modules, in which we speculate more precise information about transferring lightweight objects across tasks is concealed. Last, all tasks in our experiments are formulated into the text-to-text format, and we have not conducted analysis on tasks in other formats.

Acknowledgements

This work is supported by the National Key R&D Program of China (No. 2020AAA0106502), National Natural Science Foundation of China(No. 62236011).

References

Francesco Barbieri, Jose Camacho-Collados, Leonardo Neves, and Luis Espinosa-Anke. 2020. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. *arXiv preprint arXiv:2010.12421*.

Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the ai:

Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678.

Tilman Beck, Bela Bohlender, Christina Viehmann, Vincent Hane, Yanik Adamson, Jaber Khuri, Jonas Brossmann, Jonas Pfeiffer, and Iryna Gurevych. 2022. AdapterHub playground: Simple and flexible few-shot learning with adapters. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–75, Dublin, Ireland. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Michael Chen, Mike D'Arcy, Alisa Liu, Jared Fernandez, and Doug Downey. 2019a. Codah: An adversarially-authored question answering dataset for common sense. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 63–69.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019b. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2020. Climate-fever: A dataset for verification of real-world climate claims. *arXiv preprint arXiv:2012.00614*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, pages 1–16.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*.

Dan Friedman, Ben Dodge, and Danqi Chen. 2021. Single-dataset experts for multi-dataset question answering. *ArXiv preprint*, abs/2109.13880.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79.

Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. 2012. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. 2021. Pre-trained models: Past, present and future. *AI Open*, 2:225–250.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of ICML*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos qa: Machine reading comprehension with contextual commonsense reasoning. *arXiv preprint arXiv:1909.00277*.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Neema Kotonya and Francesca Toni. 2020. Explainable automated fact-checking for public health claims. *arXiv preprint arXiv:2010.09926*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of EMNLP*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of ACL*, pages 4582–4597, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020. Birds have four legs?! numersense: Probing numerical commonsense knowledge of pre-trained language models. *arXiv preprint arXiv:2005.00683*.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65(4):782–796.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

Clara H McCreery, Namit Katariya, Anitha Kannan, Manish Chablani, and Xavier Amatriain. 2020. Effective transfer learning for identifying similar questions: matching user questions to covid-19 faqs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3458–3465.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. 2020. Getting closer to ai complete question answering: A set of prerequisite real tasks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8722–8731.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. Dream: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7:217–231.

Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019a. Quarel: A dataset and models for answering questions about qualitative relationships. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7063–7071.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019b. Quartz: An open-domain dataset of qualitative relationship questions. *arXiv preprint arXiv:1909.03553*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Niket Tandon, Bhavana Dalvi Mishra, Keisuke Sakaguchi, Antoine Bosselut, and Peter Clark. 2019. Wiqa: A dataset for" what if..." reasoning over procedural text. *arXiv preprint arXiv:1909.04739*.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. *arXiv preprint arXiv:2104.08835*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. " going on a vacation" takes longer than" going for a walk": A study of temporal commonsense understanding. *arXiv preprint arXiv:1909.03065*.

A Experiments Details

In this section, we describe the experimental settings in detail. The 32 Question Answering tasks and 19 diverse tasks are listed in Table 6. All datasets are publicly-available and downloaded from huggingface datasets[1]. The lightweight objects which we use to produce the lightweight module for initialization are trained with full data, and the hyper-parameters of this experiment are listed in the line of *full data* in Table 5. For the approaches which require tuning the lightweight objects for a small number of steps first, we train 200 steps first and evaluate every 50 steps. For the other approaches, we choose 30 examples for the classification tasks or 50 examples (for other tasks) to generate indicators that will be used to synthesize the initialization for the upcoming task. The particular hyper-parameters are reported in Table 5. We also do all parameter fine-tuning experiment with full data, the test result of which is 54.2

We use Huggingface Transformers (Wolf et al., 2020) and PyTorch (Paszke et al., 2019) for all the experiments. Our experiments are done with NVIDIA A100 (maximum GPU memory=39.58GB). Producing the lightweight object $\mathcal{O}_{\text{new}}$ of $\mathcal{T}_{\text{new}}$ through our investigated methods takes approximately 15 minutes and occupies 10 GB GPU memory on average, while testing on 32 QA tasks takes approximately 11 hours and occupies 18 GB GPU memory on average. T5_{base} model (checkpoints released by Lester et al. (2021)) contains 248 million parameters.

hyper-parameter	*few-shot*	*full data*
learning rate	5e-4	5e-4
batch size	8	16
earlystop steps	10	20
evaluation interval	100	1000
adapter size	12	12
lora size	10	10
prefix r	24	24
prefix num	120	120

Table 5: Hyper-parameter setting. The line of *few-shot* shows hyper-parameter of the experiments in which we test our approaches, while the line of *full data* shows hyper-parameter of the experiments in which we get the lightweight objects.

[1] https://huggingface.co/datasets

Question Answering / Machine Reading Comprehension
adversarialqa (Bartolo et al., 2020)
hotpot_qa (Yang et al., 2018)
superglue-record (Zhang et al., 2020)
Question Answering / Multiple-choice Question Answering
ai2_arc (Clark et al., 2018)
codah (Chen et al., 2019a)
commonsense_qa (Talmor et al., 2018)
cosmos_qa (Huang et al., 2019)
dream (Sun et al., 2019)
hellaswag (Zellers et al., 2019)
openbookqa (Mihaylov et al., 2018)
qasc (Khot et al., 2020)
quail (Rogers et al., 2020)
quarel (Tafjord et al., 2019a)
quartz-no_knowledge (Tafjord et al., 2019b)
quartz-with_knowledge (Tafjord et al., 2019b)
race-high (Lai et al., 2017)
race-middle (Lai et al., 2017)
sciq (Welbl et al., 2017)
superglue-copa (Gordon et al., 2012)
swag (Zellers et al., 2018)
wino_grande (Sakaguchi et al., 2021)
wiqa (Tandon et al., 2019)
Question Answering / Binary
boolq (Clark et al., 2019)
mc_taco (Zhou et al., 2019)
Question Answering / Long-form Question Answering
eli5-askh (Fan et al., 2019)
eli5-asks (Fan et al., 2019)
eli5-eli5 (Fan et al., 2019)
Question Answering / Closed-book Question Answering
lama-conceptnet (Petroni et al., 2019)
lama-google_re (Petroni et al., 2019)
numer_sense (Lin et al., 2020)
search_qa (Dunn et al., 2017)
web_questions (Berant et al., 2013)
Classification / Sentiment Analysis
amazon_polarity (McAuley and Leskovec, 2013)
financial_phrasebank (Malo et al., 2014)
Classification / Nli
anli (Nie et al., 2019)
scitail (Khot et al., 2018)
Classification / Fact Checking
climate_fever (Diggelmann et al., 2020)
health_fact (Kotonya and Toni, 2020)
liar (Wang, 2017)
tab_fact (Chen et al., 2019b)
Classification / Emotion
tweet_eval-offensive (Barbieri et al., 2020)
tweet_eval-sentiment (Barbieri et al., 2020)
tweet_eval-irony (Barbieri et al., 2020)
Classification / Paraphrase
glue-mrpc (Dolan and Brockett, 2005)
glue-qqp
medical_questions_pairs (McCreery et al., 2020)
Conditoinal Generation / Summarization
samsum (Gliwa et al., 2019)
xsum (Narayan et al., 2018)
Others / Linguistic Phenomenon
blimp-ellipsis_n_bar_1 (Warstadt et al., 2020)
blimp-irregular_past_participle_adjectives (Warstadt et al., 2020)
blimp-sentential_negation_npi_scope (Warstadt et al., 2020)

Table 6: All the tasks which we use in the experiments. The first 32 tasks are Question Answering tasks, and the the last 19 tasks are other diverse tasks.

B Details and Extension of the Analysis of Module Importance

All the 60 experimental results described in **Analysis of Module Importance.** in Experiments 3.2 are listed in Table 7 (24 results corresponding to blocks) and Table 8 (36 results corresponding to layers). There exist some results that are satisfactory compared to the results listed in Table 1, which indicates the potential of the GraNd approach.

Block Number	Encoder Block	Decoder Block
0	31.9	37.5
1	32.6	35.9
2	31.7	37.1
3	32.4	35.9
4	33.9	35.1
5	32.3	37.4
6	33.8	36.0
7	33.0	34.5
8	32.6	36.9
9	32.1	36.9
10	32.5	34.6
11	33.7	36.6
average	32.7	36.2

Table 7: The (test) results of parallel experiment where we apply the modified GraNd approach on different blocks of the PLM.

Block Number	SelfAttention Layer	CrossAttention Layer	FF Layer
0	37.9	37.2	35.7
1	37.1	36.8	37.2
2	35.6	34.1	37.6
3	35.4	35.1	38.3
4	37.5	32.0	37.0
5	33.3	37.0	38.3
6	38.1	35.4	38.3
7	38.0	31.6	37.6
8	37.5	30.9	38.3
9	36.8	32.2	36.5
10	37.9	32.7	35.4
11	38.3	31.5	33.9
average	36.95	33.88	37.01

Table 8: The (test) results of parallel experiment where we apply the modified GraNd approach on different layers in the decoder blocks of the PLM.

Beyond the experiments based on the GraNd approach, we also apply the modified Cosine of Logits and Labels approach to analyze the module importance from the perspective of the model output. Specifically, we first train a randomly initialized lightweight object on $\mathcal{T}_{\text{new}}$ to obtain the fine-tuned lightweight object $\tilde{\mathcal{O}}$. Then we feed examples of valid dataset of $\mathcal{T}_{\text{new}}$ separately to $\mathcal{O}_i$ and $\tilde{\mathcal{O}}$. For the two lightweight objects, we take the output hidden states $\mathcal{H}_i$ and $\tilde{\mathcal{H}}$ of one particular module $\mathcal{P}$ through a cosine function to acquire the indicator $\mathcal{S}_i$. Then under our framework of testing tasks, we obtain a result corresponding to $\mathcal{P}$. Taking $\mathcal{P}$ as the adapters in the 12 + 12 blocks of the base model we choose, i.e., $T5_{base}$, we acquire 24 results, which are shown in Table 9. These results could reflect the importance of different modules, from which we observe the results of the decoder blocks are higher than those of the encoder blocks, indicating more importance of the decoder.

Block Number	Encoder Block	Decoder Block
0	31.9	33.9
1	32.6	34.2
2	33.0	34.1
3	33.5	34.9
4	33.5	33.7
5	33.5	34.0
6	33.4	34.7
7	33.7	34.8
8	34.4	35.2
9	34.0	34.2
10	34.2	33.7
b 11	32.0	34.0
average	33.31	34.28

Table 9: The (test) results of the parallel experiment where we apply the modified Cosine of Logits and Labels approach on different blocks of the PLM.

C Approaches Extracting Information from the Weights

Beyond the GraNd approach, we also explore 3 specific methods resorting to the information from the weight, Cosine, Euclidean, and Performance.

Cosine. We train a randomly initialized lightweight object $\tilde{\mathcal{O}}_{\text{new}}$ on the training set of $\mathcal{T}_{\text{new}}$. Then for existing lightweight object $\mathcal{O}_i \in \mathcal{O}^*$, we calculate the cosine similarity between $\tilde{\mathcal{O}}_{\text{new}}$ and $\mathcal{O}_i$ as the indicator. The final initialization of the new lightweight object is the weighted average according to the cosine similarity after softmax $\mathcal{O}_{\text{new}} \leftarrow \sum_{i=1}^{n} \texttt{Softmax}[\cos(\tilde{\mathcal{O}}_{\text{new}}, \mathcal{O}_i)] \cdot \mathcal{O}_i$.

Approach	Adapter	LoRA	Prefix
Cosine	33.3 (-1.6)	32.8 (-2.6)	32.1 (-0.2)
Euclidean	34.8 (-0.1)	34.4 (-1.0)	32.7 (+0.4)
Performance	35.1 (+0.2)	34.4 (-1.0)	34.8 (+2.5)

Table 10: Test results of Cosine, Euclidean and Performance. Numbers in parentheses are the difference between the method and the best-performing baseline listed in Table 1.

This approach is also adopted in the transfer of soft prompts by Vu et al. 2021.

Euclidean. We develop this approach by following a basic intuition that the more a lightweight object's parameters change after training on the new dataset $\mathcal{T}_{\text{new}}$, then the less relevant it is to $\mathcal{T}_{\text{new}}$. We first train $\mathcal{O}_i$ on the new task $\mathcal{T}_{\text{new}}$ for m steps and select the best-performing lightweight object $\tilde{\mathcal{O}}_i$ in this process. Then, we directly calculate the Euclidean distance between each layer of $\mathcal{O}_i$ and $\tilde{\mathcal{O}}_i$ to get the distance after summation d_i. In this method, the indicator is the inverse of the distance $\mathcal{S}_i = -d_i$. The final lightweight object $\mathcal{O}_{\text{new}}$ is obtained by Eq. 1.

Performance. This approach follows the foregoing insight and uses the change in performance to measure the correlation between a lightweight object $\mathcal{O}_i$ and $\mathcal{T}_{\text{new}}$. In the beginning, we directly produce the zero-shot performance of $\mathcal{O}_i$ on $\mathcal{T}_{\text{new}}$ without any training, which is denoted as z_i. Then we train $\mathcal{O}_i$ on $\mathcal{T}_{\text{new}}$ for m steps and select the best performance b_i. The indicator is computed by the difference between two numbers $\mathcal{S}_i = b_i - z_i$. Although the indicator $\mathcal{S}_i$ of the Performance approach is the difference between two performances, the original cause of this difference is the change of parameters of the lightweight object, because of which we treat the Performance method as the approach extracting information from the weights.

We apply these 3 approaches to three parameter-efficient tuning methods and experiment on 32 QA tasks, the results of which are shown in Table 10. The performance of these approaches is slightly poor. We suspect the reason may be that the information about transferring lightweight objects across tasks contained in weights is more covert, and our framework cannot efficiently extract it.

D Analysis of Utilizing Best Single Lightweight Object

In this section, we probe the performance of transferring knowledge with best single lightweight object $\mathcal{O}_{\text{best}}$. For a new task $\mathcal{T}_{\text{new}}$, $\mathcal{O}_{\text{best}}$ refers to the existing lightweight object $\mathcal{O}_i$ with the highest similarity indicator $\mathcal{S}_i$, and we utilize $\mathcal{O}_{\text{best}}$ for the initialization of $\mathcal{T}_{\text{new}}$ (i.e., $\mathcal{O}_{\text{new}} = \mathcal{O}_{\text{best}}$).

Approach	Adapter	LoRA	Prefix
GraNd	34.3 (-0.6)	32.6 (-2.8)	34.8 (+2.5)
Loss	39.8 (+4.9)	37.9 (+2.5)	37.2 (+4.9)
KL-divergence	37.4 (+2.5)	36.5 (+1.1)	35.6 (+3.3)
EL2N	39.9 (+5.0)	**38.2** (+2.8)	37.3 (+5.0)
Cosine of Logits and Labels	**40.0** (+5.1)	38.1 (+2.7)	**37.7** (+5.4)

Table 11: The (test) results of the parallel experiment where we utilize $\mathcal{O}_{best}$ for the initialization of $\mathcal{T}_{\text{new}}$. Numbers in parentheses are the difference between the method and the best-performing baseline listed in Table 1.

We modify the original framework in accordance with the above description, and experiment with our parameter-efficient weight ensembling approaches, the results of which are shown in Table 11. Generally, the performance of obtaining $\mathcal{O}_{\text{new}}$ by $\mathcal{O}_{\text{best}}$ is similar to that of the primal framework, which is consistent with the experimental phenomenon that $\mathcal{O}_{\text{best}}$ occupies basically 95% weight when cultivating $\mathcal{O}_{\text{new}}$ in our original framework.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Page 5, after the Conclusion, and before the References.

☒ A2. Did you discuss any potential risks of your work?
Our work is only about algorithms, it is almost impossible to exist potential risks, so we didn't discuss them in our paper.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
The abstract is at the beginning of the paper, and the section number of the introduction is 1.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

We use the datasets of 51 tasks in Experiments(the section number is 3), and we list the name of thedatasets in Appendix A(in page 8).

☑ B1. Did you cite the creators of artifacts you used?
We cite the datasets in Appendix A(in page 8).

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
In Appendix A(in page 8)

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☑ Did you run computational experiments?

The section number is 3(Experiments).

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
We report them in Appendix A(in page 8).

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
We discuss the experimental setup in Experiments(the section number is 3) and Appendix A(in page 8).

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
We report them in Experiments(the section number is 3) and Appendix B.C,D(in page 8-10).

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
We discuss the experimental setup in Appendix A(in page 8).

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.