Improving and Simplifying Pattern Exploiting Training

Derek Tam[*] **Rakesh R Menon**[*]
Mohit Bansal **Shashank Srivastava** **Colin Raffel**
UNC Chapel Hill
{dtredsox, rrmenon, mbansal, ssrivastava, craffel}@cs.unc.edu

Abstract

Recently, pre-trained language models (LMs) have achieved strong performance when fine-tuned on difficult benchmarks like SuperGLUE. However, performance can suffer when there are very few labeled examples available for fine-tuning. Pattern Exploiting Training (PET) is a recent approach that leverages patterns for few-shot learning. However, PET uses task-specific unlabeled data. In this paper, we focus on few shot learning without any unlabeled data and introduce ADAPET, which modifies PET's objective to provide denser supervision during fine-tuning. As a result, ADAPET outperforms PET on SuperGLUE without any task-specific unlabeled data. Our code can be found at https://github.com/rrmenon10/ADAPET.

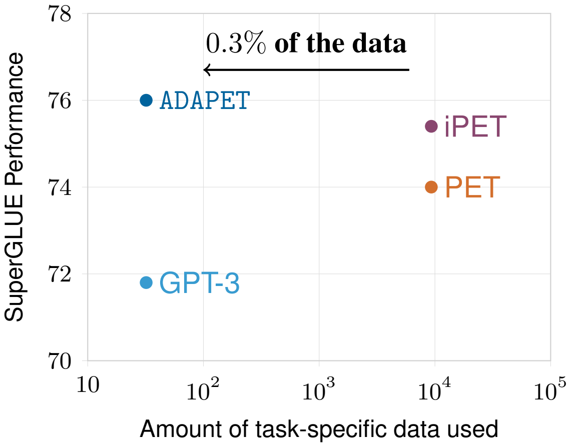


Figure 1: Performance of ADAPET vs iPET/PET and GPT-3 on SuperGLUE. While iPET/PET are parameter-efficient, they use ~9K unlabeled examples in addition to 32 labeled examples per task. ADAPET uses just 32 labeled examples, and performs better than iPET.

1 Introduction

Pre-trained language models (LMs) have shown significant gains across a wide variety of natural language processing (NLP) tasks in recent years (Devlin et al., 2019; Radford et al., 2018; Raffel et al., 2020). Most of these gains are obtained by fine-tuning language models on labeled data for a particular task. However, performance can suffer when there is very limited labeled data available for a downstream task (Xie et al., 2020; Chen et al., 2020).

Recently, GPT-3 (Brown et al., 2020) demonstrated how language models, when scaled to hundreds of billions of parameters, can learn well when primed with only a few labeled examples. However, the scale of GPT-3 (175B parameters) makes it impractical to study. There is, therefore, a need to develop smaller language models that can work equally well with limited labeled data.

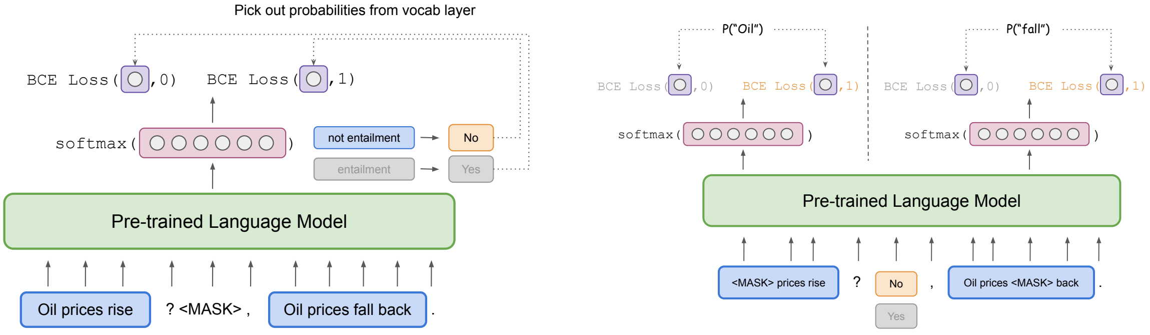
Pattern-Exploiting Training (PET; Schick and Schütze, 2020a,b) reformulates natural language understanding tasks as cloze-style questions and performs gradient-based fine-tuning. In doing so, PET outperforms GPT-3 with few labeled examples using ALBERT (Lan et al., 2020). However, PET uses additional task-specific unlabeled data.

We propose ADAPET (**A** **D**ensely-supervised **A**pproach to **P**attern **E**xploiting **T**raining) that uses more supervision by decoupling the losses for the label tokens and a label-conditioned masked language modeling (MLM) objective over the full original input. On SuperGLUE (Wang et al., 2019) with 32 labeled examples per task, ADAPET outperforms iPET without any unlabeled data.

2 Background

Cloze-style questions and MLM. A cloze task is a problem where certain parts of a text are removed, and the goal is to replace the missing portion based on the context (Taylor, 1953). Here, the text that has some parts removed is considered a cloze-style question. Inspired by cloze tasks, BERT introduces the MLM objective that tries to predict the original word at the masked out

[*] Equal contribution



(a) Decoupling Label Loss

(b) Label-conditioning

Figure 2: We illustrate the training with the two components of ADAPET. Here, the **blue** boxes refer to the inputs from a task (entailment, in this case). Figure 2a shows the decoupling label objective. The model has to predict the correct and incorrect labels at the masked out position, using a BCE loss over all labels. For the label conditioning objective in Figure 2b, the input text either includes the correct or incorrect label. At a randomly masked out position, the model should predict the original token when the input text has the correct label, and should not predict the original token when the input text has an incorrect label.

positions in a cloze question.

Notation. Let G represent a language model, x represent the input example converted into a cloze-style question, and y represent the label at the masked location m. We are interested in the quantity $[\![G_m(x)]\!]_z$ which represents the logit value for a specific token z at the mask location m.

2.1 Unlabeled Data Access

Schick and Schütze (2020a,b) assumes access to task-specific unlabeled data. For some applications such as sentiment analysis, unlabeled data can be cheap to acquire. But for SuperGLUE, where the examples are pairs of text with a label that is constructed to test a model's natural language understanding abilities, it might be more expensive to acquire unlabeled data. For example, the construction of BoolQ requires annotators to filter good question-article pairs before assigning labels (Clark et al., 2019). Hence, for our setup, we do not assume access to task-specific unlabeled data, which aligns with the setup in Brown et al. (2020).

2.2 PET

Our work primarily builds on top of PET (Schick and Schütze, 2020a,b). PET converts an example into a cloze-style question, similar to the input format used during pre-training. The query-form in PET is defined by a Pattern-Verbalizer Pair (PVP). Each PVP consists of

- a **pattern** which describes how to convert the inputs into a cloze-style question with masked out tokens. We illustrate this for an entailment task in Figure 2a. Here, we convert the premise ("*Oil prices fall back*") and the hypothesis ("*Oil prices rise*") into a cloze-style question with the pattern: <*premise*> ? <*mask*>, <*hypothesis*>.
- a **verbalizer** which describes the way to convert the classes into the output space of tokens. In Figure 2a, the verbalizer maps *"Not Entailment/Entailment"* to *"No/Yes"*.

After hand-designing a PVP for a given task, PET obtains logits from the model $G_m(x)$ (in the single-token label case). Given the space of output tokens $\mathcal{Y}$, (in Figure 2a {"*Yes*", "*No*"}) PET computes a softmax over $y \in \mathcal{Y}$, using the logits from $G_m(x)$. The final loss is shown in Equation 2.

$$q(y|x) = \frac{\exp([\![G_m(x)]\!]_y)}{\sum\limits_{y'\in\mathcal{Y}} \exp([\![G_m(x)]\!]_{y'})} \tag{1}$$

$$\mathcal{L} = \mathrm{CE}(q(y^*|x), y^*) \tag{2}$$

PET additionally distils knowledge from an ensemble of models trained with different patterns on both labeled and unlabeled data. iPET is an iterative variant of PET that trains models across iterations. The size of the training set gradually increases each iteration based on the labels of previous iterations. For a description of the different patterns used across the tasks (Schick and Schütze, 2020b), we refer the reader to Appendix A.1.

3 ADAPET

Our proposed approach, called ADAPET, modifies the objective from PET so that it can provide more supervision and learn without task-specific unlabeled data.

3.1 Decoupling Label Losses

PET computes class probabilities using the logits that correspond to the labels for a specific task. This discards the information from all the other logits in the vocabulary that do not correspond to a label. For example, in Figure 2a, *"oil"* is not a class token so the LM head should assign a low probability to *"oil"*. However, because PET only extracts the token logits that correspond to labels, the non-label tokens will never have any gradient signal.

One solution is to change the objective to a regular MLM objective. In that case, there would be no distinction between tokens corresponding to incorrect classes and any other token in the vocabulary. For example, in Figure 2a, the model would be trained to treat *"Yes"* (the incorrect token) the same as any other token such as *"oil"*. While we want the model to discourage *"oil"*, the training objective should still specifically suppress *"Yes"*.

In ADAPET, we penalize incorrect class tokens and encourage correct class tokens. Specifically, the model computes the probability of each token as a softmax normalized across all tokens so that each probability is influenced by the logits of all the vocabulary tokens. Then, we maximize the probability of the correct class tokens and minimize the probability of incorrect class tokens. This is equivalent to binary cross entropy, as shown in Figure 2a. Formally, if y^* is the true label for an example,

$$q(y|x) = \frac{\exp([\![G_m(x)]\!]_y)}{\sum_{v' \in \mathcal{V}} \exp([\![G_m(x)]\!]_{v'})} \quad (3)$$

$$\mathcal{L}_D = \log q(y^*|x) - \sum_{y \neq y^*} \log q(y|x) \quad (4)$$

The loss can be rewritten using binary cross entropy or regular cross entropy as:

$$\mathcal{L}_D = \texttt{BCE}(q(y^*|x), 1) + \sum_{y \neq y^*} \texttt{BCE}(q(y|x), 0) \quad (5)$$

$$= \texttt{CE}(q(y^*|x), y^*) - \sum_{y \neq y^*} \texttt{CE}(q(y|x), y) \quad (6)$$

3.1.1 Unified Loss for Different Tasks

For normal tasks where the label is exactly one token, PET uses the formulation described in Equation 2. For WSC, which does not have incorrect class labels, PET uses the original MLM objective rather than Equation 2. This is equivalent to Equation 5 without the second term in ADAPET.

For other tasks with multi-token labels (COPA, ReCoRD), PET computes the probability of the classes as the sum of the log probabilities of the individual tokens. However, it is not obvious how to convert these label probabilities into a valid probability distribution.

Rather than normalizing the probabilities, PET uses a hinge loss to ensure a margin between the correct label and the incorrect labels.

In ADAPET, for each token in the label, $\mathcal{L}_D$ discriminates the correct token from every other tokens, via the following loss:[1]

$$\mathcal{L}_D = \sum_{z^* \in y^*} \texttt{BCE}(q(z^*|x), 1) + \sum_{y \neq y^*} \sum_{z \in y} \texttt{BCE}(q(z|x), 0) \quad (7)$$

This objective splits a *single loss based on multiple tokens into multiple losses over single tokens*. As a result, we do not need to to multiply the probabilities of the individual tokens, and thus do not run into normalization issues.

3.2 Label Conditioning

The PET objective encapsulates the question: *"Given the input, what is the right label?"*. However, since the input space and output space both consist of tokens, we can also ask the reverse question, *"Given the answer, what is the correct context?"*. We implement this using an MLM objective where the model is trained to predict randomly masked-out tokens in context given the label. If the label is correct, the model is trained to predict the original token, as shown in Figure 2b. Crucially, if the label is wrong, the model is trained to *not* predict the original token. [2]

Let x' be the original input x which has been modified by randomly masking out tokens from the context and x^m be the original context tokens that have been masked out in x'. We maximize $P(x^m|x', y^*)$ and minimize $P(x^m|x', y)\ \forall\ y \neq y^*$. This objective is the same as the decoupling label losses approach described in Equation 5, except with different inputs and outputs.

$$q(x^m|x', y) = \frac{\exp([\![G_m(x', y)]\!]_{x^m})}{\sum_{v' \in \mathcal{V}} \exp([\![G_m(x', y)]\!]_{v'})} \quad (8)$$

$$\mathcal{L}_M = \texttt{BCE}(q(x^m|x', y^*), 1) + \sum_{y \neq y^*} \texttt{BCE}(q(x^m|x', y), 0) \quad (9)$$

[1] We ignore tokens that are common in all labels.

[2] This assumes the context only makes sense with the correct label. Empirically though, we find this to be reasonable.

Method	BoolQ Acc.	CB Acc./F1	COPA Acc.	RTE Acc.	WiC Acc.	WSC Acc.	MultiRC EM/F1a	ReCoRD Acc./F1	Avg -
ALBERT	55.7	68.6 / 49.1	63.0	50.5	41.4	81.7	3.6 / 49.8	84.1/83.5	57.7
GPT-3 (LAB; SINGLE)	77.5	82.1 / 57.2	92.0 ♣	72.9	55.3 ♣♦	75.0	32.5 / 74.8	89.0 / 90.1 ♣♦	73.2
sPET (LAB; SINGLE)	76.9	87.5 / 85.4	89.0	67.1	49.7	82.7 ♣♦	31.2 / 74.6	85.0 / 91.9	74.2
ADAPET (LAB; SINGLE)	80.3 ♣	89.3 / 86.8 ♣	89.0	76.5 ♣♦	54.4	81.7	39.2 / 80.1 ♣♦	85.4 / 92.1	77.3 ♣♦
PET (LAB + UNLAB; ENSEMBLE)	79.4	85.1 / 59.4	95.0 ♦	69.8	52.4	80.1	37.9 / 77.3	86.0 / 86.5	74.1
iPET (LAB + UNLAB; ENSEMBLE)	80.6 ♦	92.9 / 92.4 ♦	95.0 ♦	74.0	52.2	80.1	33.0 / 74.0	86.0 / 86.5	76.8

Table 1: Few-shot classification results on SuperGLUE with 32 labeled examples on the dev set. Note, we do not have access to the train split of GPT-3, so we follow the split provided by (Schick and Schütze, 2020b). ♣=BEST SINGLE PATTERN MODEL, ♦=BEST MODEL OVERALL, LAB=LABELED DATA, UNLAB=UNLABELED DATA

Method	BoolQ Acc.	CB Acc./F1	COPA Acc.	RTE Acc.	WiC Acc.	WSC Acc.	MultiRC EM/F1a	ReCoRD Acc./F1	Avg -
GPT-3 (LAB; SINGLE)	76.4	75.6 / 52.0	92.0 ♣♦	69.0	49.4	80.1	30.5 / 75.4	90.2 / 91.1 ♣♦	71.8
ADAPET (LAB; SINGLE)	80.0 ♣	92.0 / 82.3 ♣♦	85.4	75.0 ♣♦	53.5 ♣♦	85.6 ♣	35.7 / 76.2 ♣	85.5 / 86.1	76.0 ♣♦
PET (LAB + UNLAB; ENSEMBLE)	79.1	87.2 / 60.2	90.8	67.2	50.7	88.4 ♦	36.4 / 76.6 ♦	85.4 / 85.9	74.0
iPET (LAB + UNLAB; ENSEMBLE)	81.2 ♦	88.8 / 79.9	90.8	70.8	49.3	88.4 ♦	31.7 / 74.1	85.4 / 85.9	75.4

Table 2: Few-shot classification results on SuperGLUE with 32 labeled examples on the hidden test set. ♣=BEST SINGLE PATTERN MODEL, ♦=BEST MODEL OVERALL, LAB=LABELED DATA, UNLAB=UNLABELED DATA

The final loss for ADAPET is a sum of the decoupled label loss and the label-conditioned MLM loss.

4 Results and Analyses

We run experiments on SuperGLUE, and follow the same data split as Schick and Schütze (2020b), which consists of 32 labeled examples for each task.

Our code is implemented in Pytorch (Paszke et al., 2019) using HuggingFace (Wolf et al., 2020). We use the same pre-trained model and hyperparameters as PET, except we increased the number of training batches to 1k and choose the best checkpoint on the dev set, since it has been shown that training longer can help even with few samples (Zhang et al., 2021). For all ablation experiments, we only use the first pattern[3] and train for 250 batches. We refer the reader to Appendix B for more details.

Since we do not assume access to unlabeled data (see Section 2.1), we do not apply the three-step training procedure of PET and iPET to ADAPET. We still assume access to the full development set to choose the best masking ratio and checkpoint model, since PET presumably used the full development set to choose their hyperparameters which we copy.

4.1 Results

Table 1 and Table 2 shows our results on the validation and test sets on SuperGLUE. We compare against GPT-3 and PET/iPET. Note that PET/iPET uses unlabeled data and a three step training procedure (Schick and Schütze, 2020b). For fair comparison, we train PET with a single pattern (sPET) for 1k batches, and report scores for the best performing pattern on the validation set. We include a further analysis of how well the models perform for each pattern in Appendix A.2.

On the dev set, ADAPET outperforms all models that do not use unlabeled data, and even outperforms PET's iterative variant, iPET, by 0.5 points absolute. Surprisingly, sPET outperforms PET, but still loses to iPET by 2.6 points. But, this is in line with the ablation from Schick and Schütze (2020b), which shows that ensembling sPET models, trained with only labeled data, outperforms PET. Also, Gao et al. (2020) show that the model with the best performing pattern outperforms ensembling sPET models.

On the test set, ADAPET outperforms all other models including iPET without access to the unlabeled examples ($\sim$9k on average per task) and achieves state-of-the-art for few-shot learning on SuperGLUE.

4.2 Loss Ablation

Table 3 shows our ablation analysis for the loss functions we introduce in this paper. From the results, we see that label conditioning (LC) is extremely beneficial for ADAPET, especially on CB. Comparing our modified decoupled label objective (ADAPET W/O LC) with sPET, we see that it does worse for CB on F1, but does much better on RTE and MultiRC. Next, we compare against LC conditioned only on the correct label. We see that this

[3] The first pattern for each task can be found in App. A.1

hurts on BoolQ, but helps on CB. We ablate other model choices in Appendix C.

	BoolQ	CB	RTE	MultiRC
Method	Acc.	Acc./F1	Acc.	EM / F1a
ADAPET	**79.4**	**91.1 / 88.1**	**75.1**	**38.6 / 79.8**
ADAPET w/o LC	78.1	75.0 / 62.8	64.3	37.0 / 79.1
ADAPET LC (Pos. Ex. only)	75.4	83.9 / 80.9	72.2	31.3 / 76.9
sPet	77.5	75.0 / 72.8	57.0	26.5 / 73.2

Table 3: Ablation of ADAPET with different components. Best numbers have been **bolded**. (LC= Label Conditioning)

5 Conclusion

In this paper, we propose ADAPET, a new method for few-shot natural language understanding. Crucially, our work does not use unlabeled data and instead leverages more supervision to train the model. Assuming the same data budget, our model outperforms GPT-3 on SuperGLUE using just 0.1% as many parameters. However, our method has limitations; for example, we use a naive random masking strategy, which might not make sense for label conditioning. Future work could look into better masking strategies for labeled conditioned MLM, such as masking important tokens based on the the gradients of the logits for an example, as has been done for interpreting models (Simonyan et al., 2014).

Acknowledgments

This work was supported in part by ONR Grant N00014-18-1-2871, DARPA YFA17-D17AP00022, and NSF-CAREER Award 1846185. The views contained in this article are those of the authors and not of the funding agency.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157, Online. Association for Computational Linguistics.

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. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, page 177–190, Berlin, Heidelberg. Springer-Verlag.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*, 23(2):107–124.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut.

2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'12, page 552–561. AAAI Press.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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(140):1–67.

Melissa Roemmele, Cosmin Bejan, and Andrew Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning.

Timo Schick and Hinrich Schütze. 2020a. Exploiting cloze questions for few-shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

Timo Schick and Hinrich Schütze. 2020b. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*.

William Taylor. 1953. Cloze procedure: A new tool for measuring readability. *Journalism Bulletin*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 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, Online. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample BERT fine-tuning. In *International Conference on Learning Representations*.

Appendix

A Patterns and Pattern Performances

A.1 Pattern Verbalizer Pairs

We list the patterns and the verbalizers used by the PET and ADAPET models for the SuperGLUE dataset here. For improved readability of the patterns, we first list a legend for the different letter combinations that we use throughout the patterns and then proceed to enumerate the patterns for each dataset.

- `p`: passage/paragraph/pronoun
- `q`: question
- `h`: hypothesis
- `e`: entity
- `w`: word
- $\mathtt{c_i}$: choice i
- $\mathtt{s_i}$: sentence i

A.1.1 BoolQ (Clark et al., 2019)

For this QA task, we are given a paragraph pand a yes/no question q. We use two forms of labels for this task yes/no and true/false.

- Pattern : p. Question: q? Answer: ___.
  Verbalizer: yes/no
- Pattern : p. Based on the previous passage, q? ___.
  Verbalizer: yes/no
- Pattern : Based on the following passage, q? ___. p
  Verbalizer: yes/no
- Pattern : p. Question: q? Answer: ___.
  Verbalizer: true/false
- Pattern : p. Based on the previous passage, q? ___.
  Verbalizer: true/false
- Pattern : Based on the following passage, q? ___. p
  Verbalizer: true/false

A.1.2 CB (de Marneffe et al., 2019)

In this textual entailment task, given a premise p and hypothesis h we need to determine if the h entails/contradicts/is neutral with respect to the p. The labels for this task are mapped to yes/no/maybe respectively.

- Pattern : h? | ___,p
  Verbalizer: yes/maybe/no
- Pattern : “h"? | ___,“p"
  Verbalizer: yes/maybe/no
- Pattern : h? | ___.p
  Verbalizer: yes/maybe/no
- Pattern : “h?" | ___.“p"
  Verbalizer: yes/maybe/no

A.1.3 RTE (Dagan et al., 2005)

This is a textual entailment task similar to CB, except that we have just two labels for classification, entailment and not entailment. We map these two labels to yes and no respectively in the PVPs.

- Pattern : h? | ___,p
  Verbalizer: yes/no
- Pattern : “h"? | ___,“p"
  Verbalizer: yes/no
- Pattern : h? | ___.p
  Verbalizer: yes/no
- Pattern : “h?" | ___.“p"
  Verbalizer: yes/no

A.1.4 COPA (Roemmele et al., 2011)

Given a premise p, we need to find which of the options c_1 or c_2 is the responsible cause/effect for this task. For effect examples:

- Pattern : “c_1" or “c_2”? p, so ___.
  Verbalizer: c_1/c_2
- Pattern : c_1 or c_2? p, so ___.
  Verbalizer: c_1/c_2

For cause examples:

- Pattern : “c_1" or “c_2”? p, because ___.
  Verbalizer: c_1/c_2
- Pattern : c_1 or c_2? p, because ___.
  Verbalizer: c_1/c_2

A.1.5 WiC (Pilehvar and Camacho-Collados, 2019)

In this task, we are given two sentences s_1 and s_2 and we need to identify if a word w occurs in the same sense in both sentences.

- Pattern : “s_1" / “s_2" Similar sense of “w"? ___ .
  Verbalizer: yes/no
- Pattern :
  s_1 s_2 Does w have the same meaning in both sentences?_.
  Verbalizer: yes/no
- Pattern : w. Sense (1) (a) “s_1" (_) “s_2"
  Verbalizer: b/2

A.1.6 WSC (Levesque et al., 2012)

Here, we are given a sentence s that contains some nouns and pronouns. We are tasked with finding the correct noun that a specific pronoun p refers to. Within the FewGLUE dataset, we are provided with the only positive examples and hence our verbalizer contains just the correct noun phrase.

- Pattern : s The pronoun ‘*p*’ refers to . ___
  Verbalizer: correct noun
- Pattern :
  s In the previous sentence, the pronoun ‘*p*’ refers to ___.
  Verbalizer: correct noun
- Pattern :
  s In the passage above, what does the pronoun ‘*p*’ refer to? Answer: ___.
  Verbalizer: correct noun

A.1.7 MultiRC (Khashabi et al., 2018)

In this task, we are given a passage pand multiple questions q. We are tasked with finding the right answer from a list of candidate answers e. Here, we pose it as a binary classification task where we predict yes if the e answers q with context p, else no.

- Pattern : p. Question: q? Is it e? ___.
  Verbalizer: yes/no
- Pattern : p. Question: q? Is the correct answer "e"? ___.
  Verbalizer: yes/no
- Pattern :
  p. Based on the previous passage, q? Is "e" a correct answer? ___.
  Verbalizer: yes/no

A.1.8 ReCoRD (Zhang et al., 2018)

For this task, given a passage p and cloze question q, we are supposed to find the right replacement for a '@placeholder' token in the question. Since the task itself is already framed in a cloze-style format, we merely concatenate the passage with the cloze question to form the input to the language model.

A.2 Results on Individual Patterns

We train the sPET and ADAPET models using the same experimental setup mentioned in Section 4 and report results across all patterns for all datasets on the validation dataset of SuperGLUE. Note that the numbers in Table 1 contains the best numbers from this table for the dev results. Our results can be found in Table 4. Overall, ADAPET outperforms sPET on 25 out of 29 patterns across datasets.

B (More) Experiment Details

B.1 Decoupled Label Objective

All our experiments followed the same setup as PET (Schick and Schütze, 2020a). We use a random seed of 42, maximum text length of 256 [4], AdamW optimizer, learning rate of $1e^{-5}$, weight decay of $1e^{-2}$, and linear decay scheduler with a warmup over the first 10% of batches.

B.2 Label Conditioning Objective

For all datasets, we mask out up to 10.5% of tokens in the text. For COPA, because the pattern contains both the correct and incorrect choice, we use a different pattern where we only feed in one choice for the label conditioning objective.

For the cause examples:

- Pattern : Because p, ___.
  Verbalizer: c_1/c_2

For the effect examples:

- Pattern : Because ___ , p.
  Verbalizer: c_1/c_2

C Ablations

C.1 Duration of Training

We trained ADAPET for 1k batches and compared to PET/iPET which were trained for 250 batches. In this section, we compare sPET and ADAPET trained for 250 and 1k batches in Table 6. Note that training for 1k batches is not guaranteed to outperform training for 250 batches, even if we checkpoint every 250 batches, since the learning rate scheduler will have to accommodate for a different number of total batches. Overall, ADAPET gets a boost by training longer, especially on ReCoRD, while sPET peaks at 250 batches.

C.2 Multi-Task Multi-Pattern Training

We also tried training the model with multiple patterns at once, as compared to ensembling and distilling them. We formulated this as a multi-task training problem, where different patterns are viewed as different tasks, and the model would sample a pattern to train from each batch. We compare sPET, ADAPET, and ADAPET without the label conditioning objective. The results are shown in Table 7. In general, multi-task multi-pattern training hurts performance for ADAPET, is mixed on sPET, and is beneficial for ADAPET with the label conditioning objective.

C.3 Replacement Token Detection (RTD)

In our formulation, the decoupled label objective can be viewed as a binary classifier that seeks to assign high probability to the correct label token, and low probability to the incorrect label token. In reality though, the model has a softmax classifier head on top that is converted into a one-vs-all classifier.

Another way to achieve the same objective would be to use a binary classifier head on top. Rather than feeding in the *"[MASK]"* token, we would feed in either the correct label token or the incorrect label token, and the model must distinguish whether these tokens make sense in context

[4] Note: for MultiRC and ReCoRD we use 512 tokens as per (Schick and Schütze, 2020b).

	Pattern/ Model	1	2	3	4	5	6
BoolQ	sPET	75.8	76.9 ♦	74.6	76.0	76.3	68.4
Acc.	ADAPET	80.3 ♣♦	78.6 ♣	80.0 ♣	78.1 ♣	78.0 ♣	79.9 ♣
CB	sPET	75/72.8	87.5/85.4 ♦	83.9/68.9	85.7/82.3	-	-
Acc./F1	ADAPET	89.3/81.4 ♣	89.3/86.8 ♣♦	89.3/85.2 ♣	89.3/86.8 ♣	-	-
COPA	sPET	89 ♦	85 ♣	-	-	-	-
Acc.	ADAPET	89 ♦	77	-	-	-	-
MultiRC	sPET	30.6/73.7	29.9/73.2	19.1/65	30.8/74.6 ♦	15/65.2	23.1/69.6
F1a/EM	ADAPET	35.8/79.1 ♣	34.7/78.3 ♣	39.2/80.1 ♣♦	35.7/78.2 ♣	35.5/78.9 ♣	31.5/76.8 ♣
RTE	sPET	56	53.8	59.9	67.1 ♦	-	-
Acc.	ADAPET	76.2 ♣	75.1 ♣	74.4 ♣	76.5 ♣♦	-	-
WiC	sPET	49.7 ♣♦	47.5	49.7	-	-	-
Acc.	ADAPET	49.4	52.4 ♣	54.5 ♣♦	-	-	-
WSC	sPET	82.7 ♣♦	79.8	81.7 ♣	-	-	-
Acc.	ADAPET	81.7 ♦	79.8	79.8	-	-	-
ReCoRD	sPET	85.0/91.9 ♦	-	-	-	-	-
Acc./F1	ADAPET	85.4/92.1 ♣♦	-	-	-	-	-

Table 4: Performance of sPET and ADAPET models on the validation set of SuperGLUE for different patterns after training for 1000 batches. The patterns we use are the same as PET (Schick and Schütze, 2020b). Note that Table 1 uses the best pattern (♦) results from this table for each model to report validation set scores.♣ = Best Model for Each Pattern

	Pattern/ Model	1	2	3	4	5	6
BoolQ	sPET	77.5	77.1	73.9	75.6	74.2	66.8
Acc.	ADAPET	79.4 ♣	78.3 ♣	78.7 ♣	77.7 ♣	78.2 ♣	76.8 ♣
CB	sPET	75/72.8	85.7/83.5	83.9/68.9	85.7/82.3	-	-
Acc./F1	ADAPET	91.1/88.1 ♣	87.5/85.5 ♣	87.5/78.7 ♣	89.3/85 ♣	-	-
COPA	sPET	90 ♣	87	-	-	-	-
Acc.	ADAPET	73	89 ♣	-	-	-	-
MultiRC	sPET	29.9/72.8	30.2/73.3	23.6/69.0	27.4/72.8	16.1/65.7	23.9/70.3
F1a/EM	ADAPET	36.4/79.4 ♣	36.0/78.6 ♣	38.1/79.0 ♣	34.6/77.9 ♣	33.2/77.8 ♣	31.4/75.1 ♣
RTE	sPET	57	54.5	56.7	71.7	-	-
Acc.	ADAPET	74.7♣	69.7 ♣	75.1 ♣	73.6 ♣	-	-
WiC	sPET	49.8	47.8	49.5	-	-	-
Acc.	ADAPET	51.1 ♣	49.5 ♣	50.8 ♣	-	-	-
WSC	sPET	82.7 ♣	78.8 ♣	79.8	-	-	-
Acc.	ADAPET	76.9	74	79.8	-	-	-
ReCoRD	sPET	82.3/91 ♣	-	-	-	-	-
Acc./F1	ADAPET	77.4/87.2	-	-	-	-	-

Table 5: Performance of sPET and ADAPET models on the validation set of SuperGLUE for different patterns after training for 250 batches. The patterns we use are the same as PET (Schick and Schütze, 2020b). ♣ = Best Model for Each Pattern

	Method	**BoolQ** Acc.	**CB** Acc./F1	**COPA** Acc.	**RTE** Acc.	**WiC** Acc.	**WSC** Acc.	**MultiRC** EM/F1a	**ReCoRD** Acc./F1	**Avg** -
dev	sPET (LAB; SINGLE; 250 BATCHES)	**77.5**	85.7 / 83.5	**90.0**	**71.7**	**49.8**	82.7	30.2 / 73.3	82.3 / 91.0	**74.3**
	sPET (LAB; SINGLE; 1K BATCHES)	76.9	**87.5 / 85.4**	89.0	67.1	49.7	82.7	**31.2 / 74.6**	**85.0 / 91.9**	74.2
	ADAPET (LAB; SINGLE; 250 BATCHES)	79.4	**91.1 / 88.1**	89.0	75.1	51.1	79.8	38.1 / 79.0	77.4 / 87.2	75.6
	ADAPET (LAB; SINGLE; 1K BATCHES)	**80.3**	89.3 / 86.8	89.0	**76.5**	**54.4**	**81.7**	**39.2 / 80.1**	**85.4 / 92.1**	**77.3**
test	ADAPET (LAB; SINGLE; 250 BATCHES)	78.4	**93.6 / 86.4**	**86.0**	**75.0**	49.6	**90.4**	**37.3 / 75.4**	78.5 / 79.5	75.6
	ADAPET (LAB; SINGLE; 1K BATCHES)	**80.0**	92.0 / 82.3	85.4	**75.0**	**53.5**	85.6	35.7 / 76.2	**85.5 / 86.1**	**76.0**

Table 6: Performance of the models trained with 250 batches vs 1k batches

Method	**BoolQ** Acc.	**CB** Acc./F1	**RTE** Acc.	**MultiRC** EM / F1a
sPET	**77.5**	**85.7/83.5**	71.7	30.2 / 73.3
sPET (MTMP)	77.3	87.5/78.7	**74**	**30.1 / 74.3**
ADAPET	**79.4**	**91.1 / 88.1**	**75.1**	**38.1 / 79.0**
ADAPET (MTMP)	78.9	89.3/86.8	73.3	35.9/78.3
ADAPET W/O LC	77.8	78.6 / 54.9	71.5	**32.5 / 74.8**
ADAPET W/O LC (MTMP)	**79.9**	**89.3/83.6**	**77.3**	27.7/72.6

Table 7: Comparison of sPET and ADAPET with Multi-Pattern Multi-Task training MPMT = MULTI PATTERN MULTI TASK. Best numbers have been **bolded**. (LC= LABEL CONDITIONING)

or not. This objective would be very similar to the RTD objective for ELECTRA (Clark et al., 2020). Inference would be slower since the number of forward passes would scale up by the number of labels. For multi token labels though, because there is not need to condition on other label tokens, the number of forward passes would scale down by the number of tokens in the labels.

Table 8 shows the results of using the RTD objective with a binary classifier. Overall, the RTD objective seems to perform worse than the decoupled label objective. There are several reasons why using a RTD head might perform worse. First, the RTD head would have $|V|$ times fewer parameters, but relative to the whole model, the change in number of parameters is not substantial. Second, the softmax classifier has been pretrained, and contains lots of information, which is now lost when we discard the softmax classifier and randomly initialize a binary classifier head from scratch.

We also experiment with using a binary classifier head initialized with ELECTRA, but the results were the same and so we omit them from the table. We note that ALBERT (xxlarge-v2) is a much better performing model than BERT, and ELECTRA is more comparable to BERT than ALBERT (xxlarge-v2).

Method	**BoolQ** Acc.	**CB** Acc./F1	**RTE** Acc.	**MultiRC** EM / F1a
ADAPET W/O LC	**77.8**	78.6 / 54.9	**71.5**	**32.5 / 74.8**
ADAPET RTD	69.8	**82.1 / 80.2**	57.8	21.7 / 72.2

Table 8: Comparison of decoupled label objective and with the replacement token detection (RTD) objective. Best numbers have been **bolded**. (LC= LABEL CONDITIONING)

C.4 Label Conditioning with Important Words Masked Out

For the label conditioning component, we randomly mask out tokens in the input text, and the model tries to predict the original token when conditioned on the correct label, and not predict the original token when conditioned on an incorrect label. This makes sense if the masked out token is an influential token that affects the label, like *"Yes"* in Figure 2a, but makes less sense if the masked out token is an unimportant word like *"the"*. We experiment with only masking out important words, using TFIDF as an approximation of how important a word is. The results are shown in table 9. Overall, using TFIDF as an approximation for masking out important words hurts performance.

Method	**BoolQ** Acc.	**CB** Acc./F1	**RTE** Acc.	**MultiRC** EM / F1a
ADAPET	**79.4**	**91.1 / 88.1**	**74.7**	**36.4 / 79.4**
ADAPET TFIDF	76.1	76.8/61.8	72.9	31.1 / 77.1

Table 9: Comparison of ADAPET with random masking and masking tokens based on TFIDF. Best numbers have been **bolded**. (LC= LABEL CONDITIONING)

C.5 Ensembles

PET/iPET ensemble and distill with unlabeled data. However, it is not clear how beneficial unlabeled data is for ensembling, so we show results of ensembling models trained only on labeled data with different patterns and different seeds. For ensembling, we average the logits across the different models.

C.5.1 Across Patterns

Table 10 shows our results ensembling across patterns. In general, ensembling across patterns provides mixed results for ADAPET and sPET. This corroborates the finding in Gao et al. (2020) where sometimes the best performing model performs better than ensembling across patterns.

C.5.2 Across Seeds

Table 11 shows our results ensembling across seeds. We fix the pattern (pattern 1) and train with different seeds. For this experiment, we ensemble across models for seeds 41, 42, 43. From our results in Table 11, we find that ensembling patterns across seeds provides mixed results. Hence, we do not apply ensembling for our final results.

	BoolQ	**CB**	**RTE**	**MultiRC**
Method	Acc.	Acc./F1	Acc.	EM / F1a
ADAPET	79.4	**91.1 / 88.1**	**75.1**	38.1/ 79.0
ADAPET (ENS; PAT)	**79.5**	89.3/86.8	**75.1**	**38.2/79.2**
sPET	77.5	**85.7/83.5**	71.7	30.2 / 73.3
sPET (ENS; PAT)	**78.2**	71.4 / 77.8	**74.3**	**30.7 / 73.8**

Table 10: Ensemble of sPET and ADAPET across patterns. We use the best pattern (instead of pattern 1) numbers for ADAPET and sPET here. (ENS= ENSEMBLE) (PAT= PATTERN) Best numbers have been **bolded**.

	BoolQ	**CB**	**RTE**	**MultiRC**
Method	Acc.	Acc./F1	Acc.	EM / F1a
ADAPET	**79.4**	**91.1 / 88.1**	**75.1**	**38.1/ 79.0**
ADAPET (ENS; SEED)	79	**91.1 / 88.1**	69	35.9 / 79.3
sPET	77.5	**75.0 / 72.8**	**57.0**	26.5 / 73.2
sPET (ENS; SEED)	**77.8**	78.6 / 64.1	53.1	**30.5 / 73.8**

Table 11: Ensemble of sPET and ADAPET across seeds. Best numbers have been **bolded**.

C.6 Masking Ratio

We experiment with several different masking schemes, where we mask out a fixed percentage (FIXED), or *up to* a fixed percentage (VARIABLE) in Table 12. If x is the number of tokens masked out in FIXED masking, we mask out between 1 and x tokens for VARIABLE masking. For the ablation, we tested with multiples of 1.5 for the masking ratio (in addition to 10%), to match the 15% ratio of ALBERT pre-training. From our results in Table 12, we find that 10.5% VARIABLE mask ratio provided the best trade-off between scores for all models. Hence, we choose that for our final experiments in the main paper.

	BoolQ	**CB**	**RTE**	**MultiRC**
Masking Ratio	Acc.	Acc./F1	Acc.	EM / F1a
15% (FIXED)	80.7	91.1/87.7	70.8	35.8/79.1
10.5% (FIXED)	80.1	89.3/85.0	72.9	35.8/79.1
10% (FIXED)	79.9	81.1/87.5	69.0	33.9/78.4
7.5% (FIXED)	78.3	85.7/79.8	74	36.9/78.8
15% (VARIABLE)	78.9	87.5/80.0	75.1	35.9/78.7
10.5% (VARIABLE)	79.4	91.1/88.1	74.7	36.4/79.4
10% (VARIABLE)	80.0	89.3/86.8	71.1	33.9/78.4
7.5% (VARIABLE)	79.7	89.3/86.8	70.8	36.9/78.8

Table 12: Results with different masking strategies for label-conditioned MLM in ADAPET.