Cutting Down on Prompts and Parameters: Simple Few-Shot Learning with Language Models

Robert L. Logan IV[1*] **Ivana Balažević**[2†] **Eric Wallace**[3]
Fabio Petroni[4] **Sameer Singh**[1] **Sebastian Riedel**[4,5]
[1]UC Irvine [2]University of Edinburgh
[3]UC Berkeley [4]Facebook AI Research [5]University College London

Abstract

Prompting language models (LMs) with training examples and task descriptions has been seen as critical to recent successes in few-shot learning. In this work, we show that finetuning LMs in the few-shot setting can considerably reduce the need for prompt engineering. In fact, one can use *null prompts*, prompts that contain neither task-specific templates nor training examples, and achieve competitive accuracy to manually-tuned prompts across a wide range of tasks. While finetuning LMs does introduce new parameters for each downstream task, we show that this memory overhead can be substantially reduced: finetuning only the bias terms can achieve comparable or better accuracy than standard finetuning while only updating 0.1% of the parameters. All in all, we recommend finetuning LMs for few-shot learning as it is more accurate, robust to different prompts, and can be made nearly as efficient as using frozen LMs.

1 Introduction

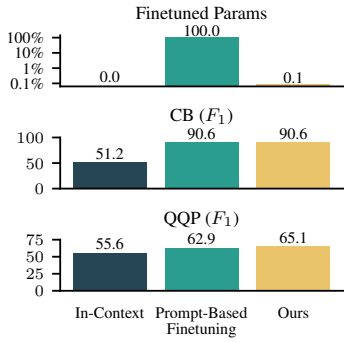
Few-shot learning—the ability to learn tasks with limited examples—is an important academic and practical challenge [5]. In state-of-the-art NLP, few-shot learning is performed by reformulating tasks as natural language "prompts" and completing those prompts with pre-trained language models [2, 15]. Prompts that are well-designed can substantially improve accuracy [21, 11]. However, finding these prompts is difficult: it requires a non-trivial combinatorial search over the prompt's wording (a.k.a. its pattern or template), whether and how to include training examples, and how to convert language model probabilities into class predictions. Consequently, prompts are often designed using human intuition that is hard to replicate and apply in a principled manner [12].

In this work, we seek to mitigate prompt engineering by identifying a class of simple prompts that are effective across many tasks for masked language models (LMs). We find that, when using prompt-based finetuning [15, 3], the prompt requires less optimization than previously thought; in fact, the pattern and training examples can be completely cut out (e.g., Figure 1, right). These *null prompts*—simple concatenations of the inputs and the [MASK] token—achieve comparable accuracy to manually-written patterns while drastically simplifying prompt design: users only need to decide the label names (a.k.a. the verbalizer) and where to place the [MASK] token. The effectiveness of null prompts also challenges the common wisdom that the success of few-shot learning is due to inductive biases present in the prompt.

A key drawback of prompt-based finetuning is that it has large memory requirements for each new downstream task (Figure 1, left). In contrast, in-context learning [2] allows reusing large-scale LMs as is, but it requires significant prompt engineering. To determine whether memory efficiency and

*Corresponding author. Email: rlogan@uci.edu
†Work done while an intern at Facebook AI Research.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.



In-Context	{What does it feel like to be on Xanax?}$_1$ and {Do 4mg Xanax bars exist?}$_2$ have different meanings. {How do you know if you're unconditionally in love with someone?}$_1$ and {How do you know if you're in love with someone and might only be denying the fact to yourself?}$_2$ have similar meanings. {Will GST affect the price level in India?}$_1$ and {Will GST effect the price level in India?}$_2$ have [MASK] meanings.
Prompt-Based Finetuning	{Will GST affect the price level in India?}$_1$? [MASK] , I want to know {Will GST effect the price level in India?}$_2$
Null Prompts (Ours)	{Will GST affect the price level in India?}$_1$ {Will GST effect the price level in India?}$_2$ [MASK]

Figure 1: **Different Methods of Few-Shot Learning.** Right: We visualize different types of prompts for QQP. We denote input fields using curly brackets { }, the manually-written pattern using magenta, and verbalizers using green. We show that *null prompts*, ones that do not contain training examples or task-specific patterns, can achieve competitive accuracy. Left: We compare different methods for model finetuning. Updating only the masked LM's bias terms (using BitFit) achieves competitive accuracy while only updating 0.1% of the parameters.

simple prompt selection can be *simultaneously* achieved, we experiment with either: (a) making prompts for in-context learning similarly easy to create, or (b) making prompt-based finetuning more memory efficient. For (a), we simplify prompt engineering for in-context learning by automatically tuning the prompt's tokens or embeddings, an approach that has been successful in the non-few-shot setting [17, 7]. For (b), we study lightweight finetuning alternatives that update a smaller set of parameters: BitFit [1], Adapters [4], and calibration layers [21].

We show that the latter approach—prompt-based finetuning with lightweight updates—is considerably more successful. In particular, updating only the model's bias terms (BitFit) can achieve competitive or better few-shot accuracy than standard finetuning while only updating 0.1% of the parameters. On the other hand, automated prompt tuning for in-context learning generally fails to find prompts that are competitive with manually engineered ones. Taken together, our results show that prompt-based finetuning is preferable because it is more accurate, more robust across prompts, and can be made nearly as efficient as using frozen LMs.

2 Prompting Language Models

We use masked LMs for few-shot learning. Following Schick and Schütze [15], we have:

- a pre-trained masked LM, with T denoting its token vocabulary and T^* the set of all token sequences.
- a small set of training inputs $x_i \in X$ and their corresponding labels $y_i \in Y$.
- a *pattern* $P : X \to T^*$ that maps inputs to cloze questions containing a single [MASK] token. Additionally, a *verbalizer* $v : Y \to T$ that maps each label to a single vocabulary token. We call the pattern and verbalizer together the *prompt*.

In our work, we consider different ways of constructing the prompt (Section 2.1) and updating the masked LM's parameters (Section 2.2). Table A1contains an overview of existing prompting methods and the settings they are evaluated in.

2.1 Constructing the Prompt

The prompt is important: different prompts can cause accuracy to vary from near chance to near state-of-the-art [21]. This importance, as well as the nontrivial nature of manually tuning the prompt, has led to growing interest in methods for automatic prompt design [17, 3, 11]. These methods search for elements such as (1) the text of the pattern, (2) the tokens in the verbalizers, and (3) whether and how training examples are prepended before the test input. Unfortunately, while automated prompt search can match the accuracy of manual tuning, it introduces its own complexities. For example, the

prompts from Gao et al. [3] achieve comparable results to manually designed prompts but are found using large generative models and careful validation.

In this paper, we show that prompt-based finetuning (see Section 2.2) can considerably reduce the importance of the prompt. This does not contradict past work—the extreme importance of the prompt is only true when models are *not finetuned*.

2.2 Finetuning the LM

In-context Learning The most well-known strategy for few-shot learning is using a frozen LM [2]. This strategy relies solely on *in-context learning* (a.k.a. priming), where the LM learns by conditioning on the prompt rather than updating its parameters. In-context learning is most successful when using very large (e.g., billions of parameters) LMs, as these models better leverage the prompt [2].

Prompt-Based Finetuning Rather than using frozen LMs, *prompt-based finetuning* methods finetune all of the LM's parameters [15, 14, 3]. For masked LMs, this is done by constructing training examples that contain a [MASK] token and finetuning the masked LM to generate the correct verbalizer token in that position.

The main advantage of prompt-based finetuning over in-context learning is that it achieves higher accuracy, especially when the LM is relatively small [16]. The main downside is that the same model can no longer be reused across different tasks, thus reducing memory efficiency. In this paper, we will show an additional benefit to prompt-based finetuning—it makes prompt engineering easier. Moreover, we will show that the memory inefficiency of prompt-based finetuning can be drastically mitigated using lightweight finetuning alternatives.

3 Experimental Setup

3.1 Datasets and Hyperparameter Tuning

We use the following classification datasets from GLUE [19] and SuperGLUE [18]: BoolQ, CB, MNLI, MRPC, QNLI, QQP, RTE, and SST-2.[3]

To build few-shot datasets, past work collects K examples from each label for training and K examples from each label for development [3]. Despite this setup often being denoted as K-shot learning, it effectively uses $2K$ examples and splits the examples evenly into train and development. We instead propose to use cross-validation to perform more principled model selection. Concretely, we sample $2K$ examples from each label and use 4-fold cross-validation to determine the best hyperparameters. After finding the best hyperparameters, we train on the first K examples and early stop on the second K examples. We use $K = 16$ following past work [3].

We sample our examples from each dataset's original training set. Since few-shot learning can be high variance, we sample the examples with 10 different random seeds and report the mean and variance of the model performance. We use each dataset's original development set for our final evaluation and use the standard evaluation metrics (accuracy or F_1) associated with each dataset. We do not check the final evaluation metrics during any tuning of the hyperparameters to ensure that we are doing "true" few-shot learning [12].

3.2 Masked Language Models

Following past work [16], we use the RoBERTa [large, 330M params, 10] and ALBERT [xxl-v2, 223M params, 6] masked LMs provided by the HuggingFace transformers library [20].

3.3 Comparing Few-shot Methods by # Wins

The results for different few-shot learning methods can be quite different across datasets and seeds for the training set [21, 15]. To compare different methods at a high level, we use a metric denoted as *# Wins*: the number of datasets that a given method performs significantly better than all other methods

[3] We also evaluated on WiC and WNLI. We omit these results because all models achieved near-random accuracy.

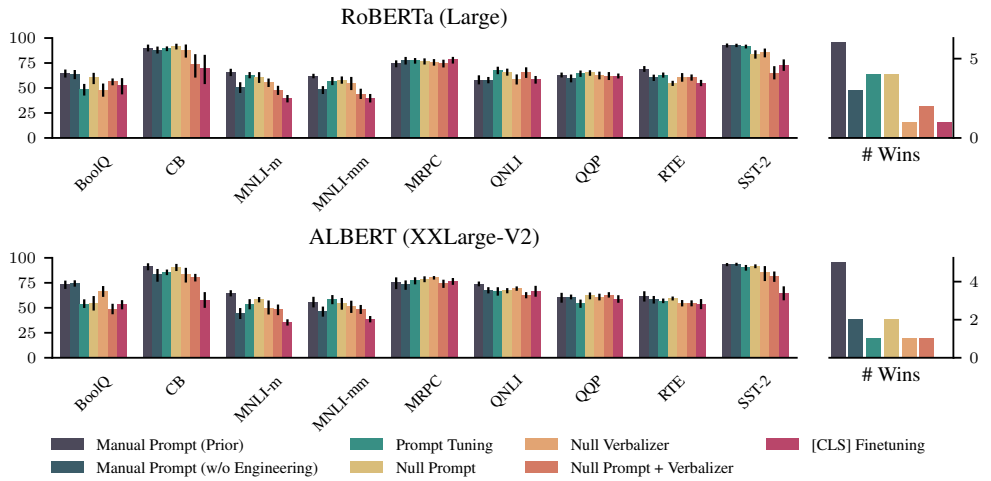


Figure 2: **Simplifying the Selection of Prompts**. We apply prompt-based finetuning in conjunction with six different types of prompts. We report accuracy or F_1 on each dataset.

on. We compute this metric for a given dataset by first performing a Welch's t-test to determine if there is a significant difference in accuracy for each pair of methods. The method which performs better than most other methods is considered the "winner" of the task and its *# Wins* is incremented by 1. There are multiple winners in the case of a tie. See Figure A5 in the Appendix for a demonstration.

4 Simplifying Prompt Engineering

In this section, we run prompt-based finetuning and ablate different elements of the prompt. We consider the following ablations:

- **Manual Prompt (Prior)**: We use manually-written prompts from Schick and Schütze [15, 16], and Gao et al. [3]. We show the patterns and verbalizers in Appendix A2.
- **Manual Prompt (w/o Engineering)**: We simulate standard prompt design by manually writing one prompt for each task using our intuition. We show the prompts in Appendix A3.
- **Prompt Tuning**: Inspired by Liu et al. [9] and Lester et al. [7], we use the pattern from Manual Prompt (Prior) but randomly initialize the embeddings of the tokens in the pattern and learn them using gradient-based optimization. This ablates the gains from human-designed patterns.
- **Null Prompt**: We use the same verbalizer as Manual Prompt (Prior) but use a pattern that consists of only the input fields and a [MASK] token. This ablates the pattern entirely.
- **Null Verbalizer**: We use the same pattern as Manual Prompt (Prior) but select tokens at random from the vocabulary for the verbalizer. This ablates the gains from a human-designed verbalizer.
- **Null Prompt + Verbalizer** We use both null prompts and random tokens for the verbalizer.

In all cases, we finetune all of the masked LM parameters. We show the accuracy of the above prompts as well as traditional finetuning (using a [CLS] token and a classification head) in Figure 2.

Manual Prompts Perform Best The manually-written prompts from prior work perform best on average for both models. On the other hand, our manual prompts (w/o Engineering) are noticeably worse than the ones from prior work and are outperformed by many other methods.

Null Prompts Are Competitive In many cases, prompt tuning and null prompts perform comparably to manually-written prompts, especially for RoBERTa. For instance, both of these methods outperform our manually-written prompts in terms of # Wins. These results show that one can achieve competitive few-shot results without resorting to any tuning of the prompt.

From an analysis perspective, these results also show that effective few-shot learning can be accomplished without any inductive bias from a manually written pattern. In fact, combining null prompts

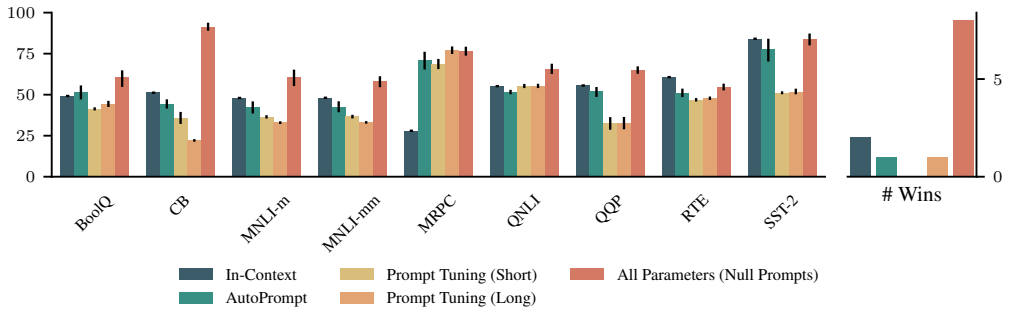


Figure 3: **Prompt-Only Tuning**. We try to simplify prompt engineering for in-context learning (i.e., using frozen models) by directly learning the prompt. The performance (accuracy/F_1) for prompt-only tuning is substantially lower than finetuning the LM parameters for RoBERTa-large.

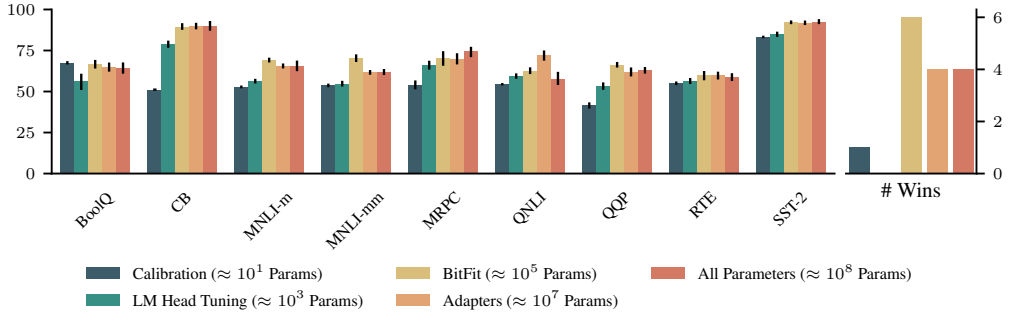


Figure 4: **Parameter-Efficient Prompt-based Finetuning**. We perform prompt-based finetuning using different lightweight finetuning schemes. We show the accuracy or F_1 on each dataset for RoBERTa-large.

with null verbalizers, which involves no human design at all, still significantly outperforms standard [CLS] finetuning for numerous tasks (3 for RoBERTa and 5 for ALBERT at $p = 0.05$). This shows that some of the effectiveness of prompt-based finetuning is due to its basic setup, i.e., predicting on a [MASK] token with an MLM head.

5 Achieving Simplicity *and* Efficiency

Thus far, we have shown that prompt-based finetuning can simplify prompt engineering at the cost of memory inefficiency—a new set of parameters must be learned for each task. This is in contrast to in-context learning, which holds all model weights fixed but is heavily influenced by small prompt modifications [21, 11]. In this section, we investigate how to achieve *both* memory efficiency and simple prompts. Concretely, in Section 5.1 we try to simplify prompt engineering for in-context learning by tuning the prompt, and in Section 5.2, we reduce the number of learned parameters for prompt-based finetuning.

5.1 Simplifying In-Context Learning With Prompt-Only Tuning

Here, we try to make prompt engineering for in-context learning as simple as prompt-based finetuning by automatically finding the prompt. We focus on the emerging class of methods that do *prompt-only tuning*: learning the prompt while keeping the rest of the model fixed [17, 7]. We consider:

- **AUTOPROMPT**: Following [17], we search for discrete tokens to use in the input instead of manually-designed patterns.
- **Prompt Tuning (Short)**: We use the same prompt tuning approach described in the previous section but we keep the masked LM fixed.
- **Prompt Tuning (Long)**: We increase the number of learned prompt embeddings to 20 in order to expand the learning capacity.

For reference, we also report the results from prompt-based finetuning with null prompts. We show the results for RoBERTa in Figure 3. We find that only tuning the prompt is relatively unsuccessful. First, on average it fails to match the performance of manually designed prompts. Second, all methods struggle to match the accuracy of prompt-based finetuning. In fact, for many of the datasets, prompt-only methods perform worse by a wide margin (e.g., 40% absolute difference in F_1 score on CB). This shows that finetuning masked LMs in the few-shot setting leads to substantially higher accuracy than prompt-only tuning.

Our Results versus Recent Prompt Tuning Work We find that only tuning the prompt performs substantially worse than finetuning the entire LM. This is in contrast to recent work, which argues that prompt-only tuning is competitive with finetuning [7, 8]. We believe these are not contradictions but rather differences in the models and settings. Li and Liang [8] focus on *left-to-right* LMs for *generation* tasks, whereas we focus on masked LMs for classification tasks. This may explain the difference in the results. Moreover, Lester et al. [7] show that prompt-only tuning becomes less competitive as models get smaller. Consequently, although we find that finetuning a masked LM is superior to prompt-only tuning, there may be other settings in which they fair similarly.

5.2 Memory-Efficient Finetuning

Given the inadequacies of prompt-only tuning, we next study if prompt-based finetuning can be made memory-efficient. To do so, we focus on reducing the number of trainable parameters, taking inspiration from recent work in the non-few-shot setting. We consider four methods:

- **Adapters**: We use Adapters [4], neural networks layers that are inserted between the feedforward portion of the Transformer architecture. We use the default Adapters hyperparameters from Houlsby et al. [4] ($\approx 10^7$ parameters per task).
- **BitFit**: Following Ben-Zaken et al. [1], we only update the bias terms inside the Transformer ($\approx 10^5$ parameters per task).
- **LM Head Tuning**: We update the embeddings in the MLM output layer that are associated with the verbalizer tokens ($\approx 10^3$ parameters per task).
- **Calibration**: Following Zhao et al. [21], we learn an affine transformation on top of the logits associated with the verbalizer tokens ($\approx 10^1$ parameters per task).

We run prompt-based finetuning for each method with the prompts from Manual Prompts (Prior). We also report the accuracy of finetuning all of the parameters for reference.

Results We show the results in Figure 4. There are diminishing returns as the parameter count is increased. In particular, substantial gains are made when going from calibration to LM head tuning to BitFit, however, there is either a marginal improvement or even a decrease in performance when going to Adapters or All Parameters. The BitFit method provides the best accuracy-efficiency trade-off, and it even outperforms finetuning all of the parameters in terms of # Wins. This suggests that updating all of the LM's hundreds of millions of parameters on only 16 data points is suboptimal.

6 Conclusion

Two high-level methods exist in few-shot prompting: using a frozen LM (in-context learning) and finetuning the LM on the few training examples (prompt-based finetuning). In this work, we demonstrate two new advantages of prompt-based finetuning. First, we show that it is robust to different choices of the prompt. In fact, there is a simple class of prompts—null prompts—that can be flexibly applied to different tasks without degrading performance relative to manually written and learned prompts. Second, we demonstrate that prompt-based finetuning can be made memory efficient: finetuning only the bias terms (BitFit) achieves comparable or better accuracy than finetuning all the parameters while being 1000x more memory efficient. Taken together, using null patterns with BitFit is an approach that is efficient, simple-to-tune, and competitive in accuracy. Code and instructions for reproducing our results are available at: https://www.github.com/ucinlp/null-prompts.

References

[1] Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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 *NeurIPS*.

[3] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *ACL*.

[4] 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 *ICML*.

[5] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. In *Science*.

[6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*.

[7] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

[8] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

[9] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. *arXiv preprint arXiv:2103.10385*.

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

[11] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

[12] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *arXiv preprint arXiv:2105.11447*.

[13] Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *NAACL*.

[14] Teven Le Scao and Alexander M. Rush. 2021. How many data points is a prompt worth? In *NAACL*.

[15] Timo Schick and Hinrich Schütze. 2021. Exploiting cloze questions for few-shot text classification and natural language inference. In *EACL*.

[16] Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *NAACL*.

[17] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*.

[18] 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 *NeurIPS*.

[19] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

[20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2020. HuggingFace's Transformers: State-of-the-art natural language processing. In *EMNLP Demo Track*.

[21] Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *ICML*.

[22] Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *NAACL*.

A Appendix

Method	Finetuned Params	Prompt Design	Few-shot
AUTOPROMPT [17]	None	Learned (Discrete)	✗
Prompt Tuning [7]	Prompt Token Embeds	Learned (Continuous)	✗
OPTIPROMPT [22]	Prompt Token Embeds	Learned (Continuous)	✗
Soft Prompts [13]	All Contextualized Embeds	Learned (Continuous)	✗
GPT-3 [2]	None	Manual	✓
PET [15]	All	Manual	✓
LM-BFF [3]	All	Learned (Discrete)	✓
P-Tuning [9]	All + Prompt Token Embeds	Learned (Continuous)	✓
Null Prompts + Bitfit (Ours)	Bias Terms	None	✓

Table A1: **Overview of Existing Work on Prompting**. *Finetuned Params* indicates the parameters altered during training. *Prompt Design* indicates how prompts are created; we use *null prompts*. *Few-Shot* indicates using few-shot training and validation sets.

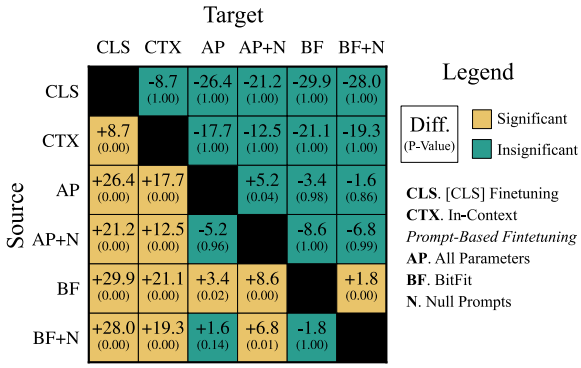


Figure A5: **How # Wins are Computed**. For a given dataset, we perform a Welch's t-test to determine if there is a significant difference in accuracy for each pair of methods. The method which performs better than most other methods (i.e., the row with the most yellow squares; BitFit in this case) is considered the "winner" of the task, and its *# Wins* is incremented by 1. In the figure above, we show a subset of methods evaluated on a single dataset.

Dataset	Pattern	Verbalizer
BoolQ	{passage}. Question: {question}? Answer: [MASK].	True: "Yes" False: "No"
CB	{premise}? [SEP] [MASK], {hypothesis}	entailment: "Yes" contradiction: "No" neutral: "Maybe"
MNLI	{sentence1}? [SEP] [MASK], {sentence2}	entailment: "Yes" contradiction: "No" neutral: "Maybe"
MNLI-mm	{sentence1}? [SEP] [MASK], {sentence2}	entailment: "Yes" contradiction: "No" neutral: "Maybe"
MRPC	{sentence1} and {sentence2} have [MASK] meanings.	0: "different" 1: "similar"
QNLI	{question}? [SEP] [MASK], {sentence}	entailment: "Yes" not_entailment: "No"
QQP	{question1} and {question2} have [MASK] meanings.	0: "different" 1: "similar"
RTE	{sentence1}? [SEP] [MASK], {sentence2}	entailment: "Yes" not_entailment: "No"
SST-2	{sentence} It was [MASK] .	0: "terrible" 1: "great"

Table A2: **Prompts denoted as "Manual Prompts (Prior)"**. We use prompts inspired from past work [15, 3]. The fields between curly brackets indicate dataset-specific inputs. Predictions are made on the [MASK] token in each prompt. For prompt tuning, we tune the tokens in the pattern.

Dataset	Pattern	Verbalizer
BoolQ	Passage: {passage} Question: {question} Answer: [MASK].	True: "true" False: "false"
CB	Premise: {premise} Hypothesis: {hypothesis} Label: [MASK]	entailment: "yes" contradiction: "no" neutral: "maybe"
MNLI	Premise: {sentence1} Hypothesis: {sentence2} Label: [MASK]	entailment: "yes" contradiction: "no" neutral: "maybe"
MNLI-mm	Premise: {sentence1} Hypothesis: {sentence2} Label: [MASK]	entailment: "yes" contradiction: "no" neutral: "maybe"
MRPC	{sentence1} and {sentence2} are the [MASK].	0: "different" 1: "same"
QNLI	Question: {question} Sentence: {sentence} Label: [MASK]	entailment: "yes" not_entailment: "no"
QQP	{question1} and {question2} are the [MASK].	0: "different" 1: "same"
RTE	Premise: {sentence1} Hypothesis: {sentence2} Label: [MASK]	entailment: "yes" not_entailment: "no"
SST-2	{sentence} Overall my impression is [MASK] .	0: "bad" 1: "good"

Table A3: **Prompts denoted as "Manual Prompts (w/o Engineering)"**. We manually write one prompt for each task, using only our intuition, and do not tune or edit them in any way after evaluating them. Fields between curly brackets indicate dataset-specific inputs. Predictions are made on the [MASK] token in each prompt. For prompt tuning, we tune the tokens in the pattern.

Dataset	Pattern	Verbalizer
BoolQ	{passage} {question} [MASK]	True: "Yes" False: "No"
CB	{premise} [MASK] {hypothesis}	entailment: "Yes" contradiction: "No" neutral: "Maybe"
MNLI	{sentence1} [MASK] {sentence2}	entailment: "Yes" contradiction: "No" neutral: "Maybe"
MNLI-mm	{sentence1} [MASK] {sentence2}	entailment: "Yes" contradiction: "No" neutral: "Maybe"
MRPC	{sentence1} {sentence2} [MASK]	0: "different" 1: "similar"
QNLI	{question} [MASK] {sentence}	entailment: "Yes" not_entailment: "No"
QQP	{question1} {question2} [MASK]	0: "different" 1: "similar"
RTE	{sentence1} [MASK] {sentence2}	entailment: "Yes" not_entailment: "No"
SST-2	{sentence} [MASK]	0: "terrible" 1: "great"

Table A4: **Null Prompts** used for results in Sections 4 and 5.