# Journal Pre-proof



GPT understands, too

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, Jie Tang

PII: S2666-6510(23)00014-1

DOI: <https://doi.org/10.1016/j.aiopen.2023.08.012> Reference: AIOPEN 62

To appear in: *AI Open*

Received date : 19 August 2023

Accepted date : 22 August 2023

Please cite this article as: X. Liu, Y. Zheng, Z. Du et al., GPT understands, too. *AI Open* (2023), doi: <https://doi.org/10.1016/j.aiopen.2023.08.012>.

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2023 The Authors. Publishing services by Elsevier B.V. on behalf of KeAi Communications Co. Ltd. This is an open access article under the CC BY license ([http://creativecommons.org/licenses/](http://creativecommons.org/licenses/by/4.0/) [by/4.0/](http://creativecommons.org/licenses/by/4.0/)).

Manuscript Click here to view linked References 

1 **GPT Understands, Too**

2

3

## 4 Xiao Liu1*∗*, Yanan Zheng1*∗*, Zhengxiao Du1, Ming Ding1, Yujie Qian2,

5 **Zhilin Yang**1*†***, Jie Tang**1*†*

6 1Tsinghua University 2Massachusetts Institute of Technology

7

8

9

10

11

12 **Abstract**

13

1. Prompting a pretrained language model with
2. natural language patterns has been proved effec-
3. tive for natural language understanding (NLU).

17

1. However, our preliminary study reveals that
2. manual discrete prompts often lead to unsta-
3. ble performance—e.g., changing a single word
4. in the prompt might result in substantial per-
5. formance drop. We propose a novel method
6. P-Tuning that employs trainable continuous
7. prompt embeddings in concatenation with dis-
8. crete prompts. Empirically, P-Tuning not only
9. stabilizes training by minimizing the gap be-
10. tween various discrete prompts, but also im-
11. proves performance by a sizeable margin on
12. a wide range of NLU tasks including LAMA
13. and SuperGLUE. P-Tuning is generally effec-
14. tive for both frozen and tuned language models,
15. under both the fully-supervised and few-shot
16. settings.

34

35

## 36 1 Introduction

37

1. Pretrained language models (PLMs; Brown et al.,
2. 2020) have significantly advanced the performance
3. of natural language understanding (NLU). PLMs
4. are trained with different pretraining objectives,
5. such as masked language modeling (Devlin et al.,

43

1. 2018), autoregressive language modeling (Radford
2. et al., 2019), seq2seq (Raffel et al., 2019), and per-
3. mutation language modeling (Yang et al., 2019).
4. PLMs can be further enhanced with prompting
5. (Brown et al., 2020; Schick and Schütze, 2020),

49

1. which employs manually written prompt patterns as
2. additional input to a language model. With prompt-
3. ing while PLMs are either finetuned on a small la-
4. beled dataset or frozen for direct inference on down-
5. stream tasks. Prompting has significantly improved

55

1. the performance of many NLU tasks (Brown et al.,
2. 2020; Schick and Schütze, 2020).
4. *†* corresponding to: Zhilin Yang [(zhiliny@tsinghua.edu.cn)](mailto:(zhiliny@tsinghua.edu.cn)
5. and Jie Tang [(jietang@tsinghua.edu.cn)](mailto:(jietang@tsinghua.edu.cn)

*∗* indicates equal contribution.

61

Prompt P@1 P@1

w/o PT w/ PT

[X] is located in [Y]. *(original)* 31.3 57.8

[X] is located in which country or state? [Y]. 19.8 57.8

[X] is located in which country? [Y]. 31.4 58.1

1. is located in which country? In [Y]. 51.1 58.1

Table 1: Discrete prompts suffer from instability (high variance), while P-Tuning stabilizes and improves per- formance. Results are precision@1 on LAMA-TREx P17 with BERT-base-cased. “PT” refers to P-Tuning, which trains additional continuous prompts in concate- nation with discrete prompts.

However, we observe that manual discrete prompts suffer from a large degree of instability. As shown in Table 1, with a frozen language model, changing a single word in the prompt might result in substantial performance drop. As we will show in Section 3, when the language model is tuned, the instability problem is alleviated but the perfor- mance difference between different prompts is still sizeable, especially in the few-shot setting. Such an instability issue of discrete prompts poses a crit- ical challenge in practice. Recent approaches of automatic prompting have attempted to search for a better-performing prompt given a task (Shin et al., 2020; Gao et al., 2020; Jiang et al., 2020b), but these methods do not change the unstable nature of discrete prompts.

To reduce the instability of discrete prompts, we propose a novel method P-Tuning that em- ploys trainable continuous prompt embeddings in concatenation with discrete prompts. Specifically, given a discrete prompt as the input, P-Tuning con- catenates continuous prompt embeddings with the discrete prompt tokens and feeds them as the input to the language model. The continuous prompts are updated by backpropagation to optimize the task objective. The intuition is that continuous prompts incorporate a certain degree of learnability into the input, which may learn to offset the effects of mi-

nor changes in discrete prompts to improve training

* 1. stability. To further improve performance, we em-
  2. ploy a prompt encoder using LSTMs or MLPs to
  3. model the dependency between continuous prompt
  4. embeddings.

5

1. We experiment with two NLU benchmarks: the
2. LAMA (Petroni et al., 2019) knowledge probing
3. and SuperGLUE (Wang et al., 2019a). On LAMA,
4. with the language model frozen, P-Tuning out-
5. performs manual discrete prompts and searched

11

1. prompts by 20+ points and 9 points respectively
2. with the same pretrained models. On SuperGLUE,
3. with the language model finetuned, P-Tuning out-
4. performs PET (Schick and Schütze, 2020) with
5. the best discrete prompts under both the fully-

17

18 supervised and few-shot settings. In addition to im-

19 proving performance, our results show that across

20 a wide range of tasks and settings, P-Tuning sub-

21 stantially reduces the performance gap between dif-

22 ferent discrete prompts, which results in improved

23

24 stability for language model adaptation.

25

26 **2 Method**

27

### 2.1 Issues with Discrete Prompts

1. Prompting employs natural language patterns as

30

1. additional inputs to pretrained language models for
2. adaptation to downstream tasks (Brown et al., 2020;
3. Schick and Schütze, 2020). Prior work (Zheng
4. et al., 2021) has pointed out that prompting has
5. achieved consistent and substantial improvements
6. on a number of NLP tasks. However, it still re-

37

1. mains a challenging problem of how to write high-
2. performing discrete prompts.
3. We performed preliminary experiments using
4. different manual prompts on the LAMA knowledge
5. probing task (Petroni et al., 2019), which aims to

43

1. extract triplet knowledge from a language model
2. by predicting the tail entities. Results in Table 1
3. show that manual discrete prompts lead to unstable
4. performance. For example, if we compare the last
5. two prompts in the table, changing a single word

49

1. in prompt causes a drastic decrease of 20 points in
2. performance.
3. In light of the challenge, recent works propose to
4. automate the search procedure of discrete prompts
5. by mining the training corpus (Jiang et al., 2020b),

55

1. gradient-based searching (Shin et al., 2020), and us-
2. ing pretrained generative models (Gao et al., 2020).
3. However, these works aim at searching for better-
4. performing prompts but do not change the nature
5. of instability for discrete prompts. In addition to

the instability issue, searching in the discrete space might not be able to fully leverage the gradients from backpropagation, which will potentially result in suboptimal solutions. To this end, we explore the possibility of training continuous prompts to stabilize and improve the performance of language model adaptation.

### 2.2 P-Tuning

Formally, let *M* be a pretrained language model with a hidden size of *h* and a vocabulary size of

*|V|*. Let *{*(**x***i,* **y***i*))*}i* be a labeled dataset for an NLU task, where **x**0:*n* = *{x*0*, x*1*, ..., xn}* is an

input consisting of a sequence of discrete tokens, and **y** *∈ Y* is a label. Our goal is to estimate the conditional probability for classification *fM*(*x*) = *p*ˆ(*y|x*) with parameters of *M* either finetuned or frozen.

Prompting was proposed in the format of discrete tokens (Schick and Schütze, 2020). Let [D*i*] be a discrete prompt token. Each prompt can be described as a template *T* =

*{*[D0:*i*]*,* **x***,* [D(*i*+1):*j* ]*,* **y***,* [D(*j*+1):*k*]*}*, which could organize the labeled data (including the inputs **x** and the label **y**) into a sequence of text tokens, such

that the task could be reformulated as filling in the blanks of the input text. For example, for the task of predicting a country’s capital (LAMA-TREx P36), a prompt could be “The capital of [INPUT] is [LA- BEL].” With a piece of labeled data “(Britain, Lon- don)”, the reformulated text would be “The capital of Britain is [MASK].”, where “[MASK]" should predict the given label “London”. Both discrete prompts and discrete data are together mapped into input embeddings:

*{***e**(*D*0)*...***e**(*Di*)*,* **e**(*x*0)*, ...,* **e**(*xn*)*, ...,* **e**(*Dk*)*}*

through the pretrained embedding layer, where **e** *∈*

R*|V|×d*.

However, as is discussed in Section 2.1, such discrete prompts tend to be extremely unstable and might not be optimal with back-propagation. Therefore, we propose P-Tuning that uses contin- uous prompt embeddings to improve and stabilize prompting. Let [P*i*] be the *i*th continuous prompt embedding. The prompt template for P-Tuning is as follows:

*T* = *{*[P0:*i*]*,* **x***,* [P(*i*+1):*j* ]*,* **y***,* [P(*j*+1):*k*]*}*

P-Tuning leverages an extra embedding function

*f* : [P*i*] *→ hi* to map the template to

*{h , ..., h ,* **e**(**x**)*, h , ..., h ,* **e**(**y**)*, h , ..., h }*

1. 0 *i*

*i*+1 *j*

*j*+1 *k*

1



Input embedding

**e**( )

**e**([ ])

Input embedding

Pre-trained Language Model (GPT, BERT, …)

Pre-trained Language Model (GPT, BERT, …)

Pseudo Prompts

(b) P-tuning

*h*0 … *hi* **e**( )

**e**( ) **e**( ) **e**( )

**e**( )

[P*i*+1] …[P*m*]

[P0] … [P*i*]

(a) Discrete Prompt Search

Prompt Generator

Discrete rewards

Back Propagation

Prompt Encoder

The capital of Britain is [MASK]

capital Britain

**e**( )

2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | [MASK] | |
| *hi*+1 … *hm* | | | **e**([ ]) |

3

4

5

6

7

8

9

1. Figure 1: An example of prompt search for “The capital of Britain is [MASK]”. Given the context (blue zone,
2. “Britain”) and target (red zone, “[MASK]”), the orange zone refer to the prompt. In (a), the prompt generator only
3. receives discrete rewards; on the contrary, in (b) the continuous prompt embeddings and prompt encoder can be
4. optimized in a differentiable way.

14

15

|  |  |  |  |
| --- | --- | --- | --- |
|  | LAMA | Full SG | Few SG |
|  | frozen | tuned | tuned |
| Improved | ✓ | ✓ | ✓ |
| Stabilized | ✓ | ✗ | ✓ |

1. Finally, we update the embeddings *{Pi}k* to op-

*i*=1

1. timize a task loss function.
2. It is noteworthy that we can also concatenate
3. discrete prompts with continuous prompts, which

20

1. performs better and is adopted throughout our ex-
2. periments. P-Tuning is applicable to both frozen
3. and finetuned language models.

24

### 25 2.3 Prompt Encoder

26

1. In the aforementioned framework, we employ a
2. mapping function *f* to map trainable embeddings
3. *{Pi}* to model inputs *{hi}*. The intuition is that
4. by using a mapping function, it is more conve-
5. nient to model the dependency between different
6. prompt embeddings, compared to using indepen-

33

34 dent learnable embeddings. In our implementation,

35 we use a lightweight neural network to formulate

36 the function *f* . Specifically, we experiment with

37 using long short-term memory (LSTM) networks,

38 multi-layer perceptrons (MLPs), and the identity

39

40 mapping function in Section 3.

41

## 42 3 Experiments

43

1. We include two NLU benchmarks: LAMA (Petroni
2. et al., 2019) for knowledge probing (§ 3.1) and Su-
3. perGLUE (Wang et al., 2019a) for general natural
4. language understanding. On SuperGLUE, we con-
5. sider both the fully-supervised learning (§ 3.2) and

49

1. few-shot learning (§ 3.3) settings.
2. On LAMA, following Shin et al. (2020); Jiang
3. et al. (2020b), language models are frozen and only
4. the discrete or continious prompts are tuned. For
5. SuperGLUE, following Schick and Schütze (2020);

55

1. Zheng et al. (2021), language models are tuned. In
2. our setting, we jointly optimize the language model
3. parameters and the continuous prompts. This setup
4. not only follows the common, standard settings in
5. prior work, but also allows evaluating P-Tuning

Table 2: Task settings and summary of results in our experiments. P-tuning shows improvement over base- lines on all task settings, and can stabilize performance on LAMA and Few SG. For Full SG, the gap between discrete prompts is not large and training is stable even without P-Tuning. (Full SG: fully-supervised learn- ing on SuperGLUE; Few SG: few-shot SuperGLUE; Improved: overall performance improved; Stabilized: training stabilized by minimizing difference between discrete prompts).

with both tuned and frozen language models.

The overall task setup and a summary of results are shown in Table 2.

### Knowledge Probing

* + 1. **Setup**

Knowledge probing, or referred to as fact retrieval, evaluates how much real-world knowledge has language models gained from pre-training. The LAMA (Petroni et al., 2019) dataset evaluates it with cloze tests created from triples selected in the knowledge bases.

**Datasets and vocabulary.** LAMA enforces all answers in single-token format. We first adopt the original LAMA-TREx dataset, consisting of 41 Wikidata relations and altogether 34,039 test- ing triples (namely LAMA-34k, which covers all BERT vocabularies). Since different pretrained models share distinct vocabularies, to allow direct comparison, we follow previous work (Shin et al., 2020) to adopt a subset that covers the intersection of GPT’s and BERT’s vocabularies. This is caled

1

Prompt type Model

P@1 Model

MP P-tuning

* + - 1. Original
      2. (MP)

BERT-base BERT-large E-BERT

31.1

|  |  |  |
| --- | --- | --- |
| BERT-base (109M) | 31.7 | 52.3 (+20.6) |
| -AutoPrompt (Shin et al., 2020) | - | 45.2 |
| BERT-large (335M) | 33.5 | 54.6 (+21.1) |

32.3

36.2



|  |  |  |
| --- | --- | --- |
| RoBERTa-base (125M) | 18.4 | 49.3 (+30.9) |
| -AutoPrompt (Shin et al., 2020) | - | 40.0 |
| RoBERTa-large (355M) | 22.1 | 53.5 (+31.4) |

5

|  |  |  |
| --- | --- | --- |
|  | LPAQA (BERT-base) | 34.1 |
| Discrete | LPAQA (BERT-large)  AutoPrompt (BERT-base) | 39.4  43.3 |

6

7

8 GPT2-medium (345M) 20.3 46.5 (+26.2)

9 BERT-base 48.3

P-tuning

10 BERT-large **50.6**

11

GPT2-xl (1.5B) 22.8 54.4 (+31.6)

MegatronLM (11B) 23.1 **64.2** (+41.1)

1. Table 3: Knowledge probing Precision@1 on LAMA-34k (left) and LAMA-29k (right). P-tuning outperforms all
2. the discrete prompt searching baselines. (MP: Manual prompt; PT: P-tuning).

14

15

1. LAMA-29k. We again follow Shin et al. (2020) to
2. construct the training, development, and test data
3. to allow for fair comparison.

19

20

1. **Setup.** LAMA has provided a handcraft prompt
2. for each relation, as shown in Table 1, which are
3. effective but likely sub-optimal. For bidirectional
4. masked language models, we only need to replace
5. “[X]” with the subject entity and “[Y]” with the

26

1. [MASK] token; for unidirectional language models
2. such as GPT, following LAMA’s original setting
3. on Transformer-XL (Dai et al., 2019), we use the
4. network output just before the target position.

31

1. The number of prompt tokens and positions are
2. selected based on the development sets, and for
3. simplicity we choose the (3, sub, org\_prompt, 3,
4. obj, 3) template for bidirectional models and (3,

36

1. sub, org\_prompt, 3, obj) for unidirectional models
2. as this configuration performs well for most rela-
3. tions (where the number indicates the number of
4. continuous prompt tokens). Continuous prompts
5. are concatenated with original discrete prompts.

42

1. During the prompt training, we set the learning rate
2. to 1e-5 and use the Adam optimizer.

45

46

### 47 3.1.2 Main results

48

49

1. The results are presented in Table 3. P-tuning sig-
2. nificantly improves the best results of knowledge
3. probing from 43.3% to 50.6% on LAMA-34k and
4. from 45.2% to 64.2% on LAMA-29k. Moreover,
5. P-tuning outperforms previous discrete prompt

55

1. searching approaches such as AutoPrompt (Shin
2. et al., 2020) and LPAQA (Jiang et al., 2020b) on
3. the same-size models. This confirms our intuition
4. in Section 2 that discrete prompts might not be
5. optimal.

### Fully-supervised Learning

* + 1. **Setup**

**Dataset.** To evaluate P-tuning on fully-supervised learning tasks, we adopt the SuperGLUE bench- mark (Wang et al., 2019b), consisting of 8 challeng- ing natural language understanding (NLU) tasks. We focus on 7 of them since the ReCoRD (Zhang et al., 2018) task adopts no discrete prompts, thus P-tuning is not directly applicable. The tasks in- clude question answering (BoolQ (Clark et al., 2019a) & MultiRC (Khashabi et al., 2018)), tex- tual entailment (CB (De Marneffe et al., 2019) & RTE (Dagan et al., 2005)), co-reference resolution (WiC (Pilehvar and Camacho-Collados, 2018)), causal reasoning (COPA (Roemmele et al., 2011)), and word sense disambiguation (WSC (Levesque et al., 2012)).

**Comparison methods.** We experiment with P- tuning on both unidirectional and bidirectional pretrained models, i.e., GPT and BERT. We include four variants BERT-Base, BERT-Large, GPT2-Base, and GPT-medium. For each model, we compare standard classification finetuning, PET (Schick and Schütze, 2020) (a typical fine- tuning method based on manual discrete prompts) and our P-tuning.

**Configuration.** We use the same metrics as in (Wang et al., 2019b). For fully-supervised learn- ing, we use a large training set to finetune pre- trained models and use a development set for hyper- parameter and model selection. Specifically, the AdamW optimizer with a linearly decayed learn- ing rate is used for training. We use a learning rate of *{*1*e −* 5*,* 2*e −* 5*,* 3*e −* 5*}*, a batch size of

*{*16*,* 32*}*, and a warm-up ratio of *{*0*.*0*,* 0*.*05*,* 0*.*1*}*. For small datasets (i.e., COPA, WSC, CB, RTE),

(a) Fully-supervised performance with base-scale models.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Method | BoolQ  (Acc.) | CB  (Acc.) | (F1) | WiC  (Acc.) | RTE  (Acc.) | MultiRC  (EM) (F1a) | WSC  (Acc.) | COPA  (Acc.) | Avg. |

1

2

3

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BERT-Base (109M) | CLS-FT  PET-FT  P-tuning | 72.9  73.7  73.9 | 85.1  87.5  89.2 | 73.9  90.8  92.1 | 71.1  67.9  68.8 | 68.4  70.4  71.1 | 16.2  13.7  14.8 | 66.3  62.5  63.3 | 63.5  60.6  63.5 | 67.0  70.0  72.0 | 66.2  67.1  68.4 |

4

5

6

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GPT2-Base (117M) | CLS-FT  PET-FT  P-tuning | 71.2  74.8  75.0 | 78.6  87.5  91.1 | 55.8  88.1  93.2 | 65.5  68.0  68.3 | 67.8  70.0  70.8 | 17.4  23.5  23.5 | 65.8  69.7  69.8 | 63.0  66.3  63.5 | 64.4  78.0  76.0 | 63.0  70.2  70.4 |

7

8

9

10 (b) Fully-supervised performance with large-scale models.

11

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Method | BoolQ  (Acc.) | CB  (Acc.) | (F1) | WiC  (Acc.) | RTE  (Acc.) | MultiRC  (EM) (F1a) | WSC  (Acc.) | COPA  (Acc.) | Avg. |

12

13

14

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BERT-Large (335M) | CLS-FT1 PET-FT  P-tuning | 77.7  77.2  77.8 | 94.6  91.1  96.4 | 93.7  93.5  97.4 | 74.9  70.5  72.7 | 75.8  73.6  75.5 | 24.7  17.7  17.1 | 70.5  67.0  65.6 | 68.3  80.8  81.7 | 69.0  75.0  76.0 | 72.5  73.1  74.6 |

15

16

17

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GPT2-Med. (345M) | CLS-FT  PET-FT  P-tuning | 71.0  78.3  78.9 | 73.2  96.4  98.2 | 51.2  97.4  98.7 | 65.2  70.4  69.4 | 72.2  72.6  75.5 | 19.2  32.1  29.3 | 65.8  74.4  74.2 | 62.5  73.0  74.0 | 66.0  80.0  81.0 | 63.1  74.9  75.6 |

18

19

20 1 We report the same results taken from SuperGLUE (Wang et al., 2019a).

21

22 Table 4: Fully-supervised performance on SuperGLUE development set.

23

24

25 we fine-tune pretrained models for 20 epochs. For

26

1. larger datasets (i.e., WiC, BoolQ, MultiRC), we
2. reduce the number of training epochs to be 10 as
3. the model converges earlier. Early stopping is used
4. to avoid over-fitting the training data.

31

### 32 3.2.2 Main Results

33

1. The main results of fully-supervised learning are
2. shown in Table 4. We observe that P-tuning can
3. improve fully-supervised learning performance on
4. both BERTs and GPTs. (1) Specifically, on the

38

1. BERT-Base model, P-tuning achieves best perfor-
2. mance on 5/7 tasks, while with BERT-Large, P-
3. tuning outperforms other methods on 4/7 tasks.
4. The exceptions are WiC and MultiRC, both of
5. which have relatively large training sets. We find

44

1. that P-tuning might not have large gains over CLS-
2. FT on such high-resource tasks, while benefits
3. more on low-resource tasks. On average, P-tuning
4. improves over the considered baselines. (2) On
5. GPT2-Base and GPT2-Medium models, P-tuning

50

1. consistently achieves the best performance on all
2. tasks.

53

### 54 3.3 Few-Shot Learning

55

1. While GPT-3 has shown decent few-shot learning
2. potential with handcrafted prompts, it still struggles
3. on some of the challenging tasks (e.g., natural lan-
4. guage inference) (Brown et al., 2020). We are mo-
5. tivated to study whether P-tuning can also improve

the few-shot learning performance of pretrained models on challenging tasks.

### Setup

**Few-shot Evaluation.** The few-shot performance is sensitive to lots of factors (e.g., the order of train- ing examples, random seed, and prompt patterns), and thus suffers from high variance (Zhao et al., 2021a; Lu et al., 2021; Zhang et al., 2020). There- fore, the few-shot evaluation strategy should make sure that the improvements are indeed from an im- proved method instead of variance. To this end, we follow the FewNLU evaluation procedure (Zheng et al., 2021) that has addressed and handled the issue. Specifically, we use random data splits to perform model selection only on a small labeled set to prevent overfitting a large dev set.

**Dataset.** We use the few-shot SuperGLUE (also known as FewGLUE) benchmark (Schick and Schütze, 2020) and follow the setting in prior work (Zheng et al., 2021) in terms of data split construc- tion.

**Baseline and Hyper-parameter.** In few-shot learn- ing, we again compare P-tuning with PET (Schick and Schütze, 2020), which was shown to out- perform GPT-3 on some of the tasks. Similar to (Schick and Schütze, 2020), we use ALBERT- xxLarge as the base model. For hyper-parameters that are shared by PET and P-tuning (e.g., learn-

ing rate, maximum training step, evaluation fre-

* + - 1. quency), we use the same search space for fair
      2. comparison. Specifically, we search the learning
      3. rate in *{*1*e −* 5*,* 2*e −* 5*}*, the maximum training
      4. step in *{*250*,* 500*}*, and the evaluation frequency in

5

6 *{*0*.*02*,* 0*.*04*}*.

7 **Construction of Prompt Patterns.** For PET, we

8

1. use the same manual prompts reported by Schick
2. and Schütze (2020). When constructing prompt
3. patterns for P-tuning, based on the same manual
4. prompts as PET, we insert different numbers of
5. continuous prompt tokens into different positions,
6. thus formulating a number of pattern candidates.

15

1. We then select the best pattern for P-tuning using
2. the validation strategy of FewNLU (Zheng et al.,
3. 2021). We also conduct further analysis of the num-
4. ber and the position of continuous prompt tokens
5. in §3.3.3.

21

22

### 23 3.3.2 Main Results

24 **Few-Shot Performance.** Table 5 shows the main

25

1. results of few-shot learning. We find that, on AL-
2. BERT, P-tuning consistently outperform PET on
3. average by more than 1 points. It outperforms
4. PromptTuning by more than 13 points. It proves
5. that by automatically learning continuous prompt

31

1. tokens, the pretrained models can achieve better
2. few-shot performance on NLU tasks.

34

### 35 3.3.3 Ablation Study

36

1. **Type of Prompt Encoder** Prior work (Shin et al.,
2. 2020) proposes to simply use an MLP as the prompt
3. encoder, we perform further ablation analysis for
4. prompt encoder selection, and results are shown

41

1. in Table 8. We consider LSTM, MLP, and EMB
2. (i.e., we directly optimize the word embeddings
3. without using additional parameters). From the
4. results, we can see that LSTM, MLP, and EMB
5. all work as a prompt encoder. Results show that

47

1. both LSTM and MLP generally work well on these
2. tasks, while EMB is unstable and can substantially
3. under-perform the other two on some tasks (e.g,.
4. WiC and CB). To sum up, both LSTM and MLP
5. could be taken into account when working on new

53

1. tasks.
2. **Location of Prompt Tokens** To study at which

56

1. location to insert continuous prompt tokens, we
2. perform experiments as Table 7 shows. From the
3. results, we have the following findings.
4. 1. By comparing #1 (or #2) with #3 (or #4), we find

that it would be better if we insert continuous prompt tokens at the location where it does not segment the sentences. For example, in case#1, “[P]” breaks the completeness of sentence “[Hy- pothesis]?” while in case#3, “[P]” is located between sentences.

1. By comparing #2 (or #3) with #4, we find that there’s no special preference for placing on the edge or in the middle of the inputs.
2. It is suggested to write a number of pattern can- didates and then search over them for the best for each task.

**Number of Prompt Tokens** We also study the in- fluence of the number of prompt tokens and show the results in Table 7. By comparing #3, #6, #7, and #8, we can conclude that the number of prompt tokens has a great impact on the few-shot perfor- mance. However, it is not that a larger number of prompt tokens would always be better. We conjec- ture that it could be that due to the limited training data, it becomes difficult to learn the parameters when excessively increasing the number of contin- uous prompt tokens. In practice, it is suggested to search for the best number of prompt tokens through model selection.

### 3.3.4 Comparison with Discrete Prompt Search

Prior work (Gao et al., 2020) proposed to automati- cally search discrete prompts and achieved better results than those of manual prompts. We now proceed to compare P-Tuning with auto-searched discrete prompts. For fair comparison, we follow the setting of LM-BFF (Gao et al., 2020) to also conduct experiments on some of the GLUE tasks (Wang et al., 2018) with RoBERTa-Large model (Liu et al., 2019). Since the the evaluation proto- cols have large impacts on few-shot performance, we use the top-3 discrete prompts searched by LM- BFF and experiment with using only the discrete prompts and additionally applying P-Tuning. For P-Tuning, the prompt patterns are constructed by concatenating the same discrete prompts as well as continuous prompts. Results in Table 9 show that additionally incorporating continuous prompts can further improve few-shot performance. P-Tuning is easy to be combined with existing discrete prompts, while further improving stability as discussed in Section 3.4.

1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Method | BoolQ RTE WiC CB MultiRC WSC COPA Avg (Acc.) (Acc.) (Acc.) (Acc.) (F1.) (F1a.) (EM.) (Acc.) (Acc.) | | | | | | | | | | | | | | | | | | |
| Prompt Tuning | 58.47 | *±*1.00 | 54.42 | *±*3.05 | 52.74 | *±*2.36 | 75.45 | *±*2.25 | 67.73 | *±*5.70 | 59.28 | *±*4.73 | 15.03 | *±*4.11 | 74.04 | *±*2.99 | 61.50 | *±*4.36 | 58.56 |
| PET-FT | 76.70 | *±*1.85 | 72.83 | *±*1.30 | 53.87 | *±*4.47 | 84.38 | *±*4.47 | 62.56 | *±*7.66 | 76.51 | *±*1.52 | 36.46 | *±*2.13 | 80.05 | *±*2.53 | 81.75 | *±*4.03 | 70.74 |
| P-tuning | 76.55 | *±*2.68 | 63.27 | *±*3.63 | 55.49 | *±*1.21 | 88.39 | *±*3.72 | 84.24 | *±*5.15 | 75.91 | *±*1.74 | 38.01 | *±*0.78 | 78.85 | *±*1.76 | 85.25 | *±*3.30 | 71.81 |

2

3

4

* + - 1. Table 5: The few-shot performance of PET (Schick and Schütze, 2020), Prompt Tuning (Lester et al., 2021) and our
      2. P-tuning over seven tasks based on ALBERT. Each result is averaged over 4 runs with different data splits. Results
      3. show that P-tuning consistently improves average few-shot performance by more than 1 point compared to PET and
      4. by more than 13 points compared to Prompt Tuning.

9

10

11

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Method | P#0 | P#1 | P#2 | P#3 | P#4 | P#5 | STD |

12

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FSL | PET-FT | 77.10  *±*2.21 | 67.96  *±*2.69 | 74.14  *±*1.38 | 72.48  *±*4.31 | 71.77  *±*2.56 | 60.86  *±*3.99 | 5.68 |
| (BoolQ) | P-tuning | 75.41  *±*3.09 | 75.11  *±*1.61 | 73.43  *±*2.60 | 71.35  *±*4.57 | 71.31  *±*8.58 | 65.86  *±*3.80 | 3.52 |

13

14

15

16

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LAMA | MP | 31.3 | 19.8 | 31.4 | 51.1 | 34.0 | 32.7 | 10.1 |
| (P17) | P-tuning | 57.8 | 57.8 | 58.1 | 58.1 | 58.9 | 58.7 | 0.46 |

17

18

1. Table 6: Upper table: Few-shot learning (FSL) of PET and P-tuning in terms of each pattern on SuperGLUE with
2. ALBERT; Lower table: Manual prompt (MP) and P-tuning performance on LAMA-P17 with BERT-base-cased.
3. For each column, P-tuning and compared methods share the same manual prompts, while P-tuning additionally
4. concatenates continuous prompt tokens. We report the standard deviation over multiple results of different patterns.

23

1. Results show that P-tuning achieves smaller standard deviation, proving that P-tuning can improve stability w.r.t.
2. the choice of discrete patterns.

26

27

### 3.4 Stabilizing Language Model Adaptation

1. In the above sections, we have shown that P-Tuning
2. improves over performance across multiple set-

31

1. tings. Now we present results to demonstrate that
2. P-Tuning also stabilizes language model adapta-
3. tion; i.e., reducing the differences between differ-
4. ent prompts. As we have shown in Table 1, manual
5. prompts have a large impact on the performance.

37

1. When it comes to few-shot learning, the perfor-
2. mance gap of different prompts is prominent due
3. to the sensitivity of few-shot learning (Zheng et al.,
4. 2021). Results in Table 6 show that P-tuning im-
5. proves the performance of the worst-performing

43

1. patterns (e.g., P#5), and achieves a smaller stan-
2. dard deviation over multiple patterns. Compared to
3. PET-FT, P-tuning increases the stability w.r.t. the
4. choice of patterns.
5. On LAMA, we observe similar a phenomenon

49

1. that while manual prompts often yield quite volatile
2. results, appending trainable continuous prompts on
3. top of the manual prompts can stabilize their per-
4. formances, reducing the standard deviation from
5. 10.1 to 0.46.

55

56

## 57 4 Related work

58

1. **Language Model Prompting.** GPT-3 (Brown
2. et al., 2020) uses in-context examples (Liu et al.,

2021; Zhao et al., 2021b) as a way of prompting to transfer knowledge from pretraining to downstream tasks. Schick and Schütze (2020) proposed to use cloze patterns, which removes the constraint that the masked token is the last token of the sentence. This further minimizes the gap between pretrain- ing and downstream tasks. To improve prompting for NLU, recent works have proposed methods to automatically search for high-performing prompts by mining the training corpus (Jiang et al., 2020b), gradient-based search (Shin et al., 2020), or using pretrained generative models (Gao et al., 2020). Our approach is different from these prior works in that we resort to using continuous prompt em- beddings, which are found to be complementary to discrete prompts in our experiments.

Recently, some concurrent works also proposed the use of continuous prompts. Prefix-tuning (Li and Liang, 2021) adds continuous prompts at the beginning of the sequence for each layer. In con- trast to our work, prefix-tuning targets natural lan- guage generation tasks.

In the area of NLU, a few concurrent methods were proposed based on continuous prompts, fo- cusing on improving knowledge probing (Qin and Eisner, 2021; Zhong et al., 2021). Lester et al. (2021) showed that with large pretrained models, only tuning continuous prompts with a frozen lan-

1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Prompt Patterns of P-tuning | Seg. | Pos. | #[P] | Acc. | F1. | Avg. |
| 1 | [Premise] Question: [Hypothesis] [P] ? Answer: [M]. | Yes | Mid | 1 | 87.95 | 76.70 | 82.33 |
| 2 | [Premise] Question [P]: [Hypothesis] ? Answer: [M]. | Yes | Mid | 1 | 88.39 | 78.57 | 83.48 |
| 3 | [Premise] Question: [Hypothesis] ? [P] Answer: [M]. | No | Mid | 1 | 89.29 | 79.86 | 84.58 |
| 4 | [Premise] [P] Question: [Hypothesis] ? Answer: [M]. | No | Miid | 1 | 89.73 | 82.15 | 85.94 |
| 5 | [Premise] Question: [Hypothesis] ? Answer: [M]. [P] | No | Edge | 1 | 87.50 | 83.39 | 85.45 |
| 6 | [Premise] Question: [Hypothesis] ? [P][P] Answer: [M]. | No | Mid | 2 | 88.39 | 84.74 | 86.57 |
| 7 | [Premise] Question: [Hypothesis] ? [P][P][P][P] Answer: [M]. | No | Mid | 4 | 88.39 | 85.14 | 86.76 |
| 8 | [Premise] Question: [Hypothesis] ? [P][P][P][P][P][P][P][P] Answer: [M]. | No | Mid | 8 | 83.48 | 73.32 | 78.40 |

2

3

4

5

6

7

8

9

1. Table 7: The few-shot performance of P-tuning on the CB task on ALBERT with different prompt patterns. “Seg.”
2. means whether the inserted prompt tokens segment complete sentences. “Pos.” indicates inserting the prompt tokens
3. at the edge or in the middle of the inputs. “[P]” is continuous prompt token. “[M]” is the mask token.

13

14

15

Task LSTM MLP EMB WiC-ACC 56.27 *±*1.54 55.25 *±*3.09 53.96 *±*3.23

CB-ACC. 81.70 *±*7.49 88.39 *±*3.72 82.59*±*3.69

CB-F1. 77.41 *±*9.15 84.24*±*5.15 67.27*±*6.78

BoolQ-ACC. 75.41*±*3.09 76.46*±*2.84 76.87*±*1.69

16

17

18

19

20

21

1. Table 8: The few-shot performance on WiC, CB and
2. BoolQ tasks with ALBERT using different prompt en-
3. coders. Results show that both LSTM and MLP gener-
4. ally work well on these tasks, while EMB is unstable
5. and can substantially under-perform the other two on
6. some tasks (e.g,. WiC and CB). “EMB” means using an
7. identity mapping for the prompt encoder.

29

2. Task LM-BFF (Auto) P-Tuning

32

|  |  |  |
| --- | --- | --- |
| SST-2 | 92.89 | 92.78 |
| MNLI | 57.53 | 58.70 |
| MRPC | 68.26 | 69.49 |

33

34

35

36 Table 9: Few-shot performance of automatically

37

1. searched prompts and P-Tuning. We evaluated LM-
2. BFF (Auto) using the reported top-3 searched patterns
3. under our evaluation procedure. P-Tuning also uses
4. the same discrete prompts, in concatenation with con-
5. tinuous prompts. Results show that P-Tuning can be
6. effectively combined with existing discrete patterns and
7. achieve further performance improvement.

45

46

1. guage model achieves comparable performance to
2. full-model tuning.

49

1. Compared to these concurrent works on NLU,
2. P-Tuning reaches a unique conclusion that contin-
3. uous prompts improve performance and stabilize
4. training with either frozen or tuned models under
5. both the few-shot and fully-supervised settings. For

55

1. example, no concurrent works have shown that
2. continuous prompts can improve performance with
3. a tuned language model. Technically, P-Tuning
4. also has a few unique designs such as using hy-
5. brid continuous-discrete prompts and employing a

prompt encoder.

**Knowledge in Language Models.** Self- supervised (Liu et al., 2020) pre-trained language models (Han et al., 2021) including GPT (Rad- ford et al., 2019), BERT (Devlin et al., 2018), XL- Net (Yang et al., 2019), RoBERTa (Liu et al., 2019) have been observed to learn not only contextual- ized text representations but also linguistic and world knowledge. (Hewitt and Manning, 2019) demonstrates that contextualized representations produced by language models can form a parse tree in the embedding space. (Vig, 2019; Clark et al., 2019b) look into the multi-head attention patterns within transformers and discover that certain atten- tion heads may correspond to some grammatical functions, including co-reference and noun modi- fiers. LAMA (Petroni et al., 2019, 2020) propose the LAMA task that leverages cloze tests to pre- dict the fact triples of knowledge bases to examine language model’s ability of memorizing facts with answers in the single-token format. In (Wang et al., 2020), the authors investigate the attention matrices to find evidence about knowledge triples contained in the context. (Jiang et al., 2020a) develops a multi-token fact retrieval dataset based on LAMA.

## 5 Conclusions

In this paper, we present a method P-Tuning that uses continuous prompts in concatenation with dis- crete prompts. P-Tuning improves performance and stabilizes training for pretrained language model adaptation. P-Tuning is effective with both tuned and frozen language models under both the few-shot and fully-supervised setings.

## References

1. Tom B Brown, Benjamin Mann, Nick Ryder, Melanie
2. Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind
3. Neelakantan, Pranav Shyam, Girish Sastry, Amanda
4. Askell, et al. 2020. Language models are few-shot
5. learners. *arXiv preprint arXiv:2005.14165*.

6

1. Christopher Clark, Kenton Lee, Ming-Wei Chang,
2. Tom Kwiatkowski, Michael Collins, and Kristina
3. Toutanova. 2019a. Boolq: Exploring the surprising
4. difficulty of natural yes/no questions. In *Proceedings*
5. *of the 2019 Conference of the North American Chap-*
6. *ter of the Association for Computational Linguistics:*
7. *Human Language Technologies, Volume 1 (Long and*
8. *Short Papers)*, pages 2924–2936.
9. Kevin Clark, Urvashi Khandelwal, Omer Levy, and
10. Christopher D Manning. 2019b. What does bert look
11. at? an analysis of bert’s attention. *arXiv preprint*
12. *arXiv:1906.04341*.

19

1. Ido Dagan, Oren Glickman, and Bernardo Magnini.
2. 2005. The pascal recognising textual entailment chal-
3. lenge. In *Machine Learning Challenges Workshop*,
4. pages 177–190. Springer.

24

1. Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Car-
2. bonell, Quoc V Le, and Ruslan Salakhutdinov.
3. 2019. Transformer-xl: Attentive language mod-
4. els beyond a fixed-length context. *arXiv preprint*
5. *arXiv:1901.02860*.
6. Marie-Catherine De Marneffe, Mandy Simons, and Ju-
7. dith Tonhauser. 2019. The commitmentbank: Inves-
8. tigating projection in naturally occurring discourse.
9. In *proceedings of Sinn und Bedeutung*, volume 23,
10. pages 107–124.

35

1. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and
2. Kristina Toutanova. 2018. Bert: Pre-training of deep
3. bidirectional transformers for language understand-
4. ing. *arXiv preprint arXiv:1810.04805*.

40

1. Tianyu Gao, Adam Fisch, and Danqi Chen. 2020.
2. Making pre-trained language models better few-shot
3. learners. *arXiv preprint arXiv:2012.15723*.
4. Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao
5. Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao
6. Han, Minlie Huang, et al. 2021. Pre-trained models:
7. Past, present and future. *AI Open*.

48

1. John Hewitt and Christopher D. Manning. 2019. A
2. structural probe for finding syntax in word representa-
3. tions. In *North American Chapter of the Association*
4. *for Computational Linguistics: Human Language*
5. *Technologies (NAACL)*. Association for Computa-
6. tional Linguistics.

55

1. Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki,
2. Haibo Ding, and Graham Neubig. 2020a. X-factr:
3. Multilingual factual knowledge retrieval from pre-
4. trained language models. In *Proceedings of the 2020*
5. *Conference on Empirical Methods in Natural Lan-*
6. *guage Processing (EMNLP)*, pages 5943–5959.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020b. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading com- prehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chap- ter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Pa- pers)*, pages 252–262.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *ArXiv*, abs/2104.08691.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thir- teenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

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

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. 2020. Self- supervised learning: Generative or contrastive. *arXiv preprint arXiv:2006.08218*, 1(2).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Man- dar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining ap- proach. *arXiv preprint arXiv:1907.11692*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Over- coming few-shot prompt order sensitivity. *CoRR*, abs/2104.08786.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects lan- guage models’ factual predictions. *arXiv preprint arXiv:2005.04611*.

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

Mohammad Taher Pilehvar and José Camacho-Collados. 2018. Wic: 10, 000 example pairs for eval- uating context-sensitive representations. *CoRR*, abs/1808.09121.

Guanghui Qin and J. Eisner. 2021. Learning how to ask:

1. Querying lms with mixtures of soft prompts. *ArXiv*,
2. abs/2104.06599.

3

1. Alec Radford, Jeffrey Wu, Rewon Child, David Luan,
2. Dario Amodei, and Ilya Sutskever. 2019. Language
3. models are unsupervised multitask learners. *OpenAI*
4. *blog*, 1(8):9.
5. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine
6. Lee, Sharan Narang, Michael Matena, Yanqi Zhou,
7. Wei Li, and Peter J Liu. 2019. Exploring the limits
8. of transfer learning with a unified text-to-text trans-
9. former. *arXiv preprint arXiv:1910.10683*.

13

1. Melissa Roemmele, Cosmin Adrian Bejan, and An-
2. drew S Gordon. 2011. Choice of plausible alter-
3. natives: An evaluation of commonsense causal rea-
4. soning. In *AAAI Spring Symposium: Logical Formal-*
5. *izations of Commonsense Reasoning*, pages 90–95.

19

1. Timo Schick and Hinrich Schütze. 2020. It’s not just
2. size that matters: Small language models are also
3. few-shot learners. *Computing Research Repository*,
4. arXiv:2009.07118.

24

1. Taylor Shin, Yasaman Razeghi, Robert L Logan IV,
2. Eric Wallace, and Sameer Singh. 2020. Autoprompt:
3. Eliciting knowledge from language models with
4. automatically generated prompts. *arXiv preprint*
5. *arXiv:2010.15980*.

30

1. Jesse Vig. 2019. A multiscale visualization of at-
2. tention in the transformer model. *arXiv preprint*
3. *arXiv:1906.05714*.
4. Alex Wang, Yada Pruksachatkun, Nikita Nangia, Aman-
5. preet Singh, Julian Michael, Felix Hill, Omer Levy,
6. and Samuel R Bowman. 2019a. Superglue: A stickier
7. benchmark for general-purpose language understand-
8. ing systems. *arXiv preprint arXiv:1905.00537*.

39

1. Alex Wang, Yada Pruksachatkun, Nikita Nangia, Aman-
2. preet Singh, Julian Michael, Felix Hill, Omer Levy,
3. and Samuel R. Bowman. 2019b. SuperGLUE: A
4. Stickier Benchmark for General-Purpose Language
5. Understanding Systems. In *NeurIPS 2019*, pages
6. 3261–3275.

46

1. Alex Wang, Amanpreet Singh, Julian Michael, Felix
2. Hill, Omer Levy, and Samuel R. Bowman. 2018.
3. Glue: A multi-task benchmark and analysis plat-
4. form for natural language understanding. *ArXiv*,
5. abs/1804.07461.

52

1. Chenguang Wang, Xiao Liu, and Dawn Song. 2020.
2. Language models are open knowledge graphs. *arXiv*
3. *preprint arXiv:2010.11967*.

56

1. Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Car-
2. bonell, Ruslan Salakhutdinov, and Quoc V Le.
3. 2019. Xlnet: Generalized autoregressive pretrain-
4. ing for language understanding. *arXiv preprint*
5. *arXiv:1906.08237*.

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

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Wein- berger, and Yoav Artzi. 2020. Revisiting few-sample BERT fine-tuning. *CoRR*, abs/2006.05987.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021a. Calibrate before use: Im- proving few-shot performance of language models. *CoRR*, abs/2102.09690.

Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021b. Calibrate before use: Improv- ing few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.

Yanan Zheng, Jing Zhou, Yujie Qian, Ming Ding, Jian Li, Ruslan Salakhutdinov, Jie Tang, Sebastian Ruder, and Zhilin Yang. 2021. Fewnlu: Benchmarking state- of-the-art methods for few-shot natural language un- derstanding.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [mask]: Learning vs. learning to recall. *ArXiv*, abs/2104.05240.

### Declaratiio if ioterettt

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

* The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: