P-Tuning 기법 리뷰
(GPT Understands, Too)

에이아이스쿨(AISchool) 대표
양진호 (솔라리스)
http://aischool.ai
http://solarisailab.com

P-Tuning Paper

- Liu, Xiao, et al. "GPT understands, too." AI Open (2023).
- https://browse.arxiv.org/pdf/2103.10385.pdf
- https://github.com/THUDM/P-tuning

GPT Understands, Too

Xiao Liu [*1 2] Yanan Zheng [*1 2] Zhengxiao Du [1 2] Ming Ding [1 2] Yujie Qian [3] Zhilin Yang [4 2] Jie Tang [1 2]

Abstract

While GPTs with traditional fine-tuning fail to achieve strong results on natural language understanding (NLU), we show that GPTs can be better than or comparable to similar-sized BERTs on NLU tasks with a novel method *P-tuning*—which employs trainable continuous prompt embeddings. On the knowledge probing (LAMA) benchmark, the best GPT recovers 64% (P@1) of world knowledge without any additional text provided during test time, which substantially improves the previous best by 20+ percentage points. On the SuperGlue benchmark, GPTs achieve comparable and sometimes better performance to similar-sized BERTs in supervised

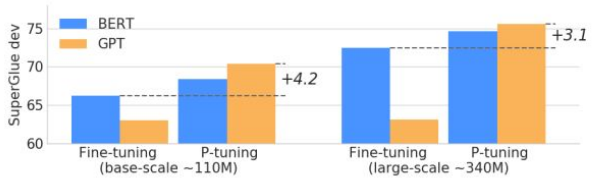

Figure 1. Average scores on 7 dev datasets of SuperGlue. GPTs can be better than similar-sized BERTs on NLU with P-tuning.

guage understanding (NLU) and **hybrid language models** (e.g., XLNet (Yang et al., 2019), UniLM (Dong et al., 2019)) for combining the first two paradigms. For long, researchers have observed that GPT-style models perform poorly for NLU tasks with fine-tuning, and thus assumed that they are not suitable for language understanding in nature.

Overview

- P-Tuning의 핵심 idea : fine-tuning 과정시 pre-train이 끝난 파라미터 w_0를 고정하고 Template 생성을 위한 Prompt Encoder만을 학습시킴

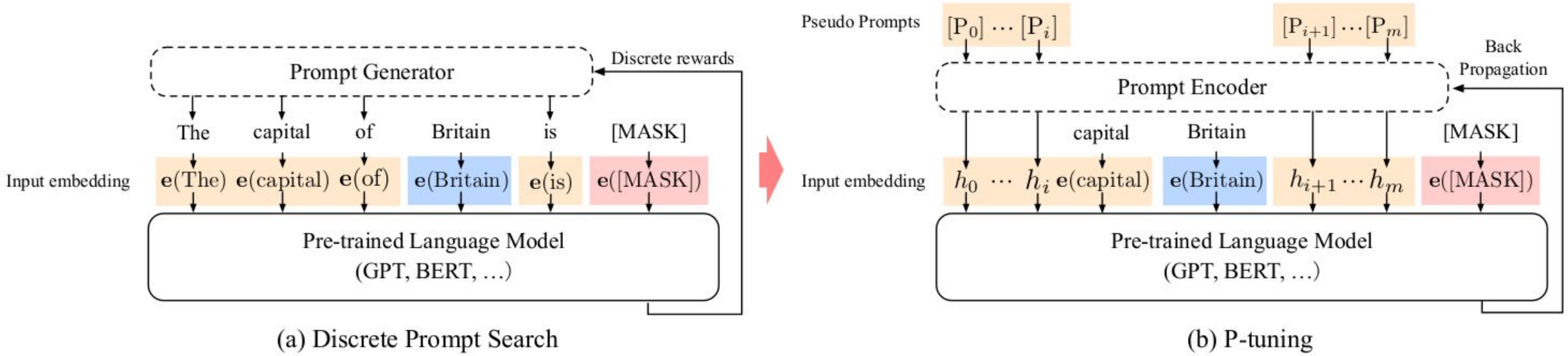

Figure 2. An example of prompt search for "The capital of Britain is [MASK]". Given the context (blue zone, "Britain") and target (red zone, "[MASK]"), the orange zone refer to the prompt tokens. In (a), the prompt generator only receives discrete rewards; on the contrary, in (b) the pseudo prompts and prompt encoder can be optimized in a differentiable way. Sometimes, adding few task-related anchor tokens (such as "capital" in (b)) will bring further improvement.

P-Tuning Paper

- P-Tuning Paper를 같이 살펴보면서 P-Tuning의 디테일한 내용들을 살펴봅시다!

Abstract

- 전통적인 미세 조정을 사용한 GPT는 자연어 이해 (NLU)에서 강력한 결과를 얻지 못하지만, 우리는 훈련 가능한 연속 프롬프트 임베딩(trainable continuous prompt embeddings)을 사용하는 새로운 방법 P-tuning을 통해 GPT가 NLU 작업에서 유사한 크기의 BERT와 비교하여 더 좋거나 유사한 성능을 낼 수 있다는 것을 보여줍니다.
- 지식 탐사 (LAMA) 벤치마크에서 최고의 GPT는 테스트 시간 동안 추가 텍스트를 제공하지 않고도 세계 지식의 64% (P@1)를 복구하며, 이는 이전 최고 기록을 20+ 퍼센트 포인트 이상 향상시킵니다.
- SuperGlue 벤치마크에서 GPT는 지도 학습에서 유사한 크기의 BERT와 비교하여 동등하거나 더 나은 성능을 보입니다.
- 중요하게도, P-tuning은 few-shot과 지도학습 설정에서 BERT의 성능도 향상시키면서 대부분 프롬프트 엔지니어링의 필요성을 크게 줄입니다. 따라서 P-tuning은 few-shot SuperGlue 벤치마크에서 state-of-the-art 기법들의 성능을 뛰어넘습니다.

1. Introduction

- 언어 모델의 사전 훈련은 많은 자연어 처리 작업에 성공적인 접근법이었습니다 (Brown 등, 2020). 증거들은 사전 훈련 중에 언어 모델들이 문맥화된 텍스트 표현만 학습하는 것이 아니라, 문법 (Vig, 2019; Clark 등, 2019b), 구문론적 지식 (Hewitt & Manning, 2019), 상식 (Davison 등, 2019) 및 심지어 세계 지식 (Petroni 등, 2019; Wang 등, 2020)까지도 학습한다는 것을 제안합니다.

- 훈련 목표에 따라 사전 훈련된 언어 모델은 세 가지 카테고리로 나눌 수 있습니다:

1. 자연어 생성 (NLG)을 위한 **단방향 언어 모델** (예: GPT (Radford 등, 2019)),
2. 자연어 이해 (NLU)를 위한 **양방향 언어 모델** (예: BERT (Devlin 등, 2018))
3. 첫 번째와 두 번째 패러다임을 결합하기 위한 **하이브리드 언어 모델** (예: XLNet (Yang 등, 2019), UniLM (Dong 등, 2019)).

- 오랜 시간 동안 연구자들은 GPT 스타일의 모델이 미세 조정을 통해 NLU 작업에서 성능이 떨어지는 것을 관찰했고, 따라서 이들이 본질적으로 언어 이해에 적합하지 않다고 가정했습니다.

1. Introduction

- 새롭게 등장한 GPT-3 (Brown 등, 2020)와 그 특별한 성능은 수작업으로 제작된 프롬프트(handcrafted prompts)와 few-shot 및 zero-shot 학습은 기계 학습 커뮤니티를 강타했습니다.
- 그 성공은 거대한 단방향 언어 모델과 적절한 수작업 프롬프트가 자연어 이해를 위해 작동할 수 있음을 제안합니다.
- 그러나 최고의 성능을 발휘하는 프롬프트를 수작업으로 제작하는 것은 비실용적으로 거대한 검증 세트 필요해서 거대한 건초더미에서 바늘을 찾는 것과 같습니다.
- 많은 경우에, 프롬프트 엔지니어링은 테스트 세트에 과적합하는 것을 의미합니다. 게다가, 상당한 성능 감소를 초래하는 적대적인 프롬프트를 만들기 쉽습니다.
- 이러한 문제를 고려하여, 최근의 연구들은 자동으로 이산 프롬프트를 검색하는 것에 집중하였습니다 (Jiang 등, 2020b; Shin 등, 2020; Reynolds & McDonell, 2021; Gao 등, 2020) 그리고 그 효과를 입증했습니다. 그러나, 신경망이 본질적으로 연속적이므로, 이산 프롬프트는 최적이 아닐 수 있습니다.

1. Introduction

- 이 연구에서는 GPT와 NLU 애플리케이션 간의 차이를 해소하기 위해 연속적 공간에서 자동으로 프롬프트를 검색하는 새로운 방법인 P-tuning을 제안합니다.
- P-tuning은 몇 가지 연속적인 자유 매개변수(continuous free parameters)를 활용하여 사전 훈련된 언어 모델의 입력으로 공급되는 프롬프트로 사용됩니다.
- 그 후 우리는 이산 프롬프트 검색의 대안으로 경사 하강법을 사용하여 연속적인 프롬프트를 최적화합니다.

1. Introduction

- 간단한 P-tuning 방법은 GPT에 큰 향상을 가져다줍니다.
- 우리는 LAMA (Petroni 등, 2019) 지식 탐사 및 SuperGLUE (Wang 등, 2019b)의 두 NLU 벤치마크에서 P-tuning 기반의 GPT를 조사합니다.
- 모델 파라미터가 고정된 LAMA 지식 탐사에서, 원래의 수작업 프롬프트와 비교하여 P-tuning을 기반으로 한 GPT는 Precision@1에서 26.2%-41.1%의 절대적인 향상을 보입니다. 최고의 성능을 보이는 모델은 LAMA에서 64.2%를 달성하며, 이는 state-of-the-art 45.2% 프롬프트 검색 접근법을 크게 초과합니다.
- 또 다른 NLU 벤치마크인 SuperGlue에서는 few-shot과 완전한 지도 시나리오에서 P-tuning과 미세 조정을 함께 적용합니다.
- 결과적으로, GPT는 동일한 규모의 BERT 모델과 경쟁력 있는 성능을 보이며, 일부 데이터셋에서는 GPT가 BERT를 능가하기도 합니다. 추가 실험은 BERT 스타일의 모델도 어느 정도 P-tuning으로부터 이점을 얻을 수 있음을 보여줍니다.
- 우리는 P-tuning을 사용한 ALBERT가 이전 접근법들을 크게 능가하며 few-shot 세팅으로 SuperGLUE 벤치마크에서 새로운 state-of-the-art 결과를 달성함을 보여줍니다.

1. Introduction

- 우리의 발견은 GPT는 생성할 수 있지만 이해하지 못한다는 고정관념을 깹니다. 또한 언어 모델이 우리가 이전에 가정했던 것보다 훨씬 더 많은 세계 지식과 이전 작업 지식을 포함하고 있음을 제안합니다. P-tuning은 또한 최적의 하위 작업 성능을 위해 사전 훈련된 언어 모델을 조정하는 일반적인 방법으로도 사용됩니다. 요약하면, 우리는 다음과 같은 기여를 합니다:

1. 우리는 P-tuning을 사용하여 GPT가 자연어 이해에서 BERT와 경쟁력 있게 (때로는 더 나은 성능으로) 작동할 수 있음을 보여줍니다. 이로써, 자연어 이해를 위한 GPT 스타일 아키텍처의 잠재력이 과소평가되었음을 밝혀냈습니다.
2. 우리는 P-tuning이 few-shot과 완전한 지도 설정에서 GPT와 BERT를 모두 향상시키는 일반적인 방법임을 보여줍니다. 특히, P-tuning을 사용하면 우리의 방법은 LAMA 지식 탐사와 few-shot 세팅으로 SuperGlue에서 state-of-the-art 방법들을 능가합니다. 이것은 언어 모델이 우리가 이전에 생각했던 것보다 사전 훈련 중에 더 많은 세계 지식과 이전 작업 지식을 습득했음을 나타냅니다.

2. Motivation

- GPT-3 (Brown 등, 2020)와 DALL-E (Ramesh 등, 2021)의 기적은 거대한 모델들이 기계 지능을 향상시키기 위한 만병통치약에 다름 아니라는 것을 제안하는 것 같습니다. 그러나 번영의 뒷면에는 무시할 수 없는 도전들이 있습니다.
- 치명적인 문제 중 하나는 거대한 모델들이 전송 능력이 부족하다는 것입니다. 하위 작업에 미세 조정하는 것은 그 트릴리언 규모의 모델들에게는 거의 효과가 없습니다. many-shot fine-tuning 설정에도 불구하고, 이러한 모델들은 여전히 미세 조정 샘플들을 (Yue 등, 2020) 빠르게 기억하는 것(memorize)이 너무 큽니다.

2. Motivation

- 대체 방법으로, GPT-3와 DALL-E는 하위 작업 애플리케이션을 위해 모델을 조절하기 위해 수작업으로 제작된 프롬프트(handcrafted prompts)를 활용하는 것으로 보고되었습니다.
- 그러나 수작업 프롬프트 검색은 비실용적으로 큰 검증 세트에 크게 의존하며, 그 성능 역시 변동성이 큽니다.
- LAMA (Petroni 등, 2019) 지식 탐사 (표 1)에서 비슷한 경우를 보여줍니다.
- 여기서 한 단어의 변화가 극적인 차이를 초래할 수 있습니다.

2. Motivation

Prompt	P@1
[X] is located in [Y]. *(original)*	31.29
[X] is located in which country or state? [Y].	19.78
[X] is located in which country? [Y].	31.40
[X] is located in which country? In [Y].	51.08

Table 1. Case study on LAMA-TREx P17 with bert-base-cased. A single-word change in prompts could yield a drastic difference.

2. Motivation

- 이러한 도전을 고려하여, 최근의 일부 연구들은 훈련 코퍼스의 탐색 (Jiang 등, 2020b), 경사 검색 (Shin 등, 2020) 및 별도의 모델 사용 (Gao 등, 2020)을 통해 이산 프롬프트의 자동 검색에 집중하였습니다.
- 그러나 우리는 differentially 최적화될 수 있는 연속적인 프롬프트를 찾는 문제에 깊이 파고들었습니다.

Discrete Prompt Search의 예시 - AutoPrompt

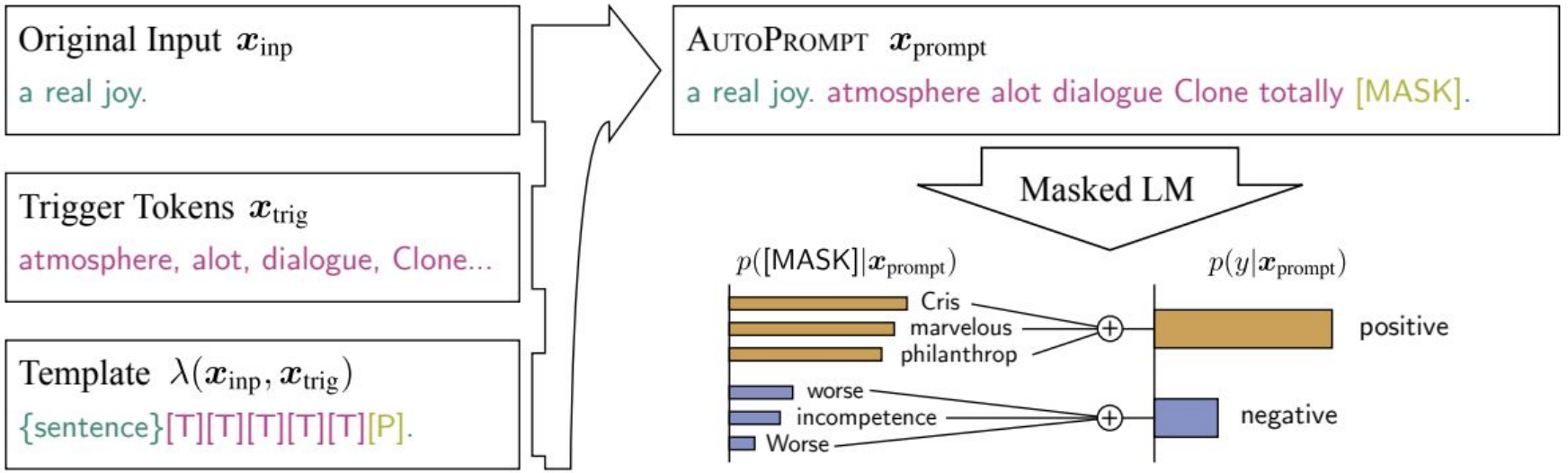

Figure 1: **Illustration of AUTOPROMPT** applied to probe a masked language model's (MLM's) ability to perform sentiment analysis. Each input, x_{inp}, is placed into a natural language prompt, x_{prompt}, which contains a single [MASK] token. The prompt is created using a template, λ, which combines the original input with a set of trigger tokens, x_{trig}. The trigger tokens are shared across all inputs and determined using a gradient-based search (Section 2.2). Probabilities for each class label, y, are then obtained by marginalizing the MLM predictions, $p([MASK]|x_{prompt})$, over sets of automatically detected label tokens (Section 2.3).

3. Method: P-tuning

- 이 섹션에서는 P-tuning의 구현에 대해 소개합니다.
- 이산 프롬프트와 유사하게, P-tuning은 입력에 대한 비침습적 수정만 적용합니다.
- 그럼에도 불구하고, P-tuning은 사전 훈련된 언어 모델의 입력 임베딩을 그것의 미분가능한 출력 임베딩으로 교체합니다.

3.1. Architecture

- 사전 훈련된 언어 모델 M이 주어지면, 이산 입력 토큰의 연속 x_1:n = {x_0, x_1,...,x_n}은 사전 훈련된 임베딩 레이어 e ∈ M에 의해 입력 임베딩 {e(x_0), e(x_1),...,e(x_n)}으로 매핑됩니다.
- 특정 시나리오에서, 컨텍스트 x에 조건을 주면, 우리는 종종 하위 처리를 위해 목표 토큰 y의 출력 임베딩을 사용합니다.
- 예를 들어, 사전 훈련에서 x는 마스크되지 않은 토큰을 나타내고 y는 [MASK] 토큰을 나타냅니다;
- 그리고 문장 분류에서 x는 문장 토큰을 나타내고 y는 종종 [CLS]를 나타냅니다.

3.1. Architecture

- 프롬프트 p의 기능은 컨텍스트 x, 대상 y 및 자체를 템플릿 T로 구성하는 것입니다.
- 예를 들어, 나라의 수도를 예측하는 작업에서 (LAMA-TREx P36) 템플릿은 "영국의 수도는 [MASK]입니다."가 될 수 있습니다(그림 2 참조).
- 여기서 " ... 의 수도는 ... 입니다."는 프롬프트, "영국"은 컨텍스트, "[MASK]"는 대상입니다.
- 프롬프트는 매우 유연할 수 있어 컨텍스트나 대상 안에 삽입할 수도 있습니다.

3.1. Architecture

- V는 언어 모델 M의 어휘를 나타내며 [P_i]는 템플릿 T의 i번째 프롬프트 토큰을 나타냅니다. 간단히 말해서, 주어진 템플릿 T = {[P_0:i], x, [P_i+1:m], y}에서, 전통적인 이산 프롬프트는 [P_i] ∈ V를 만족시키며 T를 다음과 같이 매핑합니다.

$$\{\mathbf{e}([\mathrm{P}_{0:i}]), \mathbf{e}(\mathbf{x}), \mathbf{e}([\mathrm{P}_{i+1:m}]), \mathbf{e}(\mathbf{y})\} \qquad (1)$$

- P-tuning은 대신 [P_i]를 의사 토큰(pseudo tokens)으로 간주하고 템플릿을 다음과 같이 매핑합니다.

$$\{h_0, ..., h_i, \mathbf{e}(\mathbf{x}), h_{i+1}, ..., h_m, \mathbf{e}(\mathbf{y})\} \qquad (2)$$

- 여기서 h_i (0 ≤ i < m)는 학습 가능한 임베딩 텐서입니다.
- 이를 통해 우리는 원래의 언어 모델 M의 어휘 V가 표현할 수 있는 것을 넘어 더 나은 연속적인 프롬프트를 찾을 수 있게 됩니다.
- 마지막으로, 하류 손실 함수 L을 사용하여, 우리는 연속적인 프롬프트 h_i (0 ≤ i < m)를 다음과 같이 미분 최적화할 수 있습니다.

$$\hat{h}_{0:m} = \arg\min_{h} \mathcal{L}(\mathcal{M}(\mathbf{x}, \mathbf{y})) \qquad (3)$$

3.1. Architecture

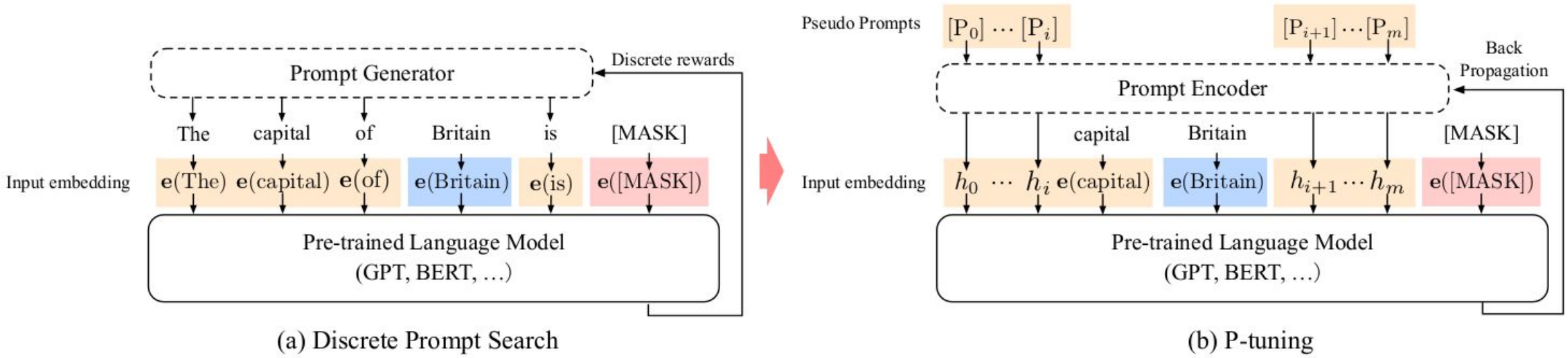

Figure 2. An example of prompt search for "The capital of Britain is [MASK]". Given the context (blue zone, "Britain") and target (red zone, "[MASK]"), the orange zone refer to the prompt tokens. In (a), the prompt generator only receives discrete rewards; on the contrary, in (b) the pseudo prompts and prompt encoder can be optimized in a differentiable way. Sometimes, adding few task-related anchor tokens (such as "capital" in (b)) will bring further improvement.

3.2. Optimization

- 연속적인 프롬프트를 훈련하는 아이디어는 직관적이지만, 실제로는 두 가지 최적화 문제에 직면합니다:

1. 이산성(Discreteness) : 원래의 단어 임베딩 e는 사전 훈련 후 이미 매우 이산적으로 변해 있습니다. 만약 h가 무작위 분포로 초기화되고 확률적 경사 하강법(SGD)으로 최적화된다면, 이는 파라미터를 작은 영역 내에서만 변경한다는 것이 (Allen-Zhu et al., 2019)에 의해 증명되었기 때문에, 최적화 도구는 쉽게 지역 최소값에 빠지게 됩니다.
2. 연관성(Association) : 또 다른 우려는, 직관적으로 프롬프트 임베딩 h_i의 값들이 서로 독립적이기보다는 서로에게 의존적이어야 한다고 생각합니다. 우리는 프롬프트 임베딩들을 서로 연결하는 어떤 메커니즘이 필요합니다.

3.2. Optimization

- 이러한 도전을 고려하여 P-Tuning에서는 연속성과 연관성 문제를 해결할 수 있는 매우 경량의 신경망을 포함하는 프롬프트 인코더를 사용하여 h_i를 시퀀스로도 모델링하도록 제안합니다.
- 실제로, 우리는 이산성을 장려하기 위해 ReLU 활성화 두 층 다층 퍼셉트론(MLP)과 함께 양방향 장기 단기 메모리 네트워크(LSTM)를 선택합니다.
- 형식적으로 말하면, 언어 모델 M로의 실제 입력 임베딩 h'_i는 다음과 같이 유도됩니다.

$$
\begin{aligned}
h_i &= \mathrm{MLP}([\overrightarrow{h_i} : \overleftarrow{h_i}]) \\
&= \mathrm{MLP}([\mathrm{LSTM}(h_{0:i}) : \mathrm{LSTM}(h_{i:m})])
\end{aligned} \tag{4}
$$

3.2. Optimization

- LSTM 헤드의 사용은 실제로 연속적인 프롬프트의 훈련에 몇몇 파라미터를 추가하게 되지만, LSTM 헤드는 사전 훈련된 모델보다 여러 크기의 차수로 작습니다.
- 더욱이 추론에서는 출력 임베딩 h만 필요로 하며 LSTM 헤드는 폐기할 수 있습니다.
- 또한, SuperGLUE 벤치마크에서 몇몇 NLU 작업에 몇 개의 앵커 토큰을 추가하는 것이 도움이 된다는 것도 발견했습니다.
- 예를 들면, RTE 작업의 경우, 프롬프트 템플릿 "[PRE][prompt tokens][HYP]?[prompt tokens][MASK]" 내의 토큰 "?"은 특별히 앵커 토큰으로 추가되며 성능에 큰 영향을 미칩니다.
- 보통 이러한 앵커 단어들은 각 구성 요소를 특징 짓는데, 이 경우 "?"는 "[HYP]"가 질문 부분으로 작용한다는 것을 나타냅니다.

Results

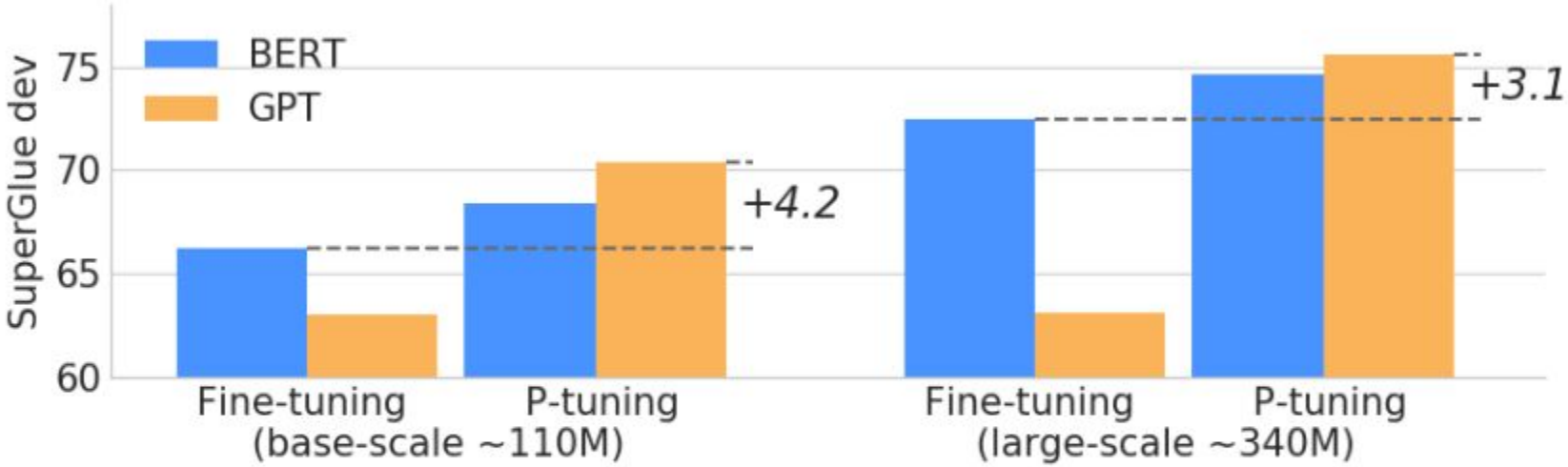

Figure 1. Average scores on 7 dev datasets of SuperGlue. GPTs can be better than similar-sized BERTs on NLU with P-tuning.

Results

GPT Understands, Too

Prompt type	Model	P@1
Original (MP)	BERT-base	31.1
	BERT-large	32.3
	E-BERT	36.2
Discrete	LPAQA (BERT-base)	34.1
	LPAQA (BERT-large)	39.4
	AutoPrompt (BERT-base)	43.3
P-tuning	BERT-base	48.3
	BERT-large	**50.6**

Model	MP	FT	MP+FT	P-tuning
BERT-base (109M)	31.7	51.6	52.1	52.3 (+20.6)
-AutoPrompt (Shin et al., 2020)	-	-	-	45.2
BERT-large (335M)	33.5	54.0	55.0	54.6 (+21.1)
RoBERTa-base (125M)	18.4	49.2	50.0	49.3 (+30.9)
-AutoPrompt (Shin et al., 2020)	-	-	-	40.0
RoBERTa-large (355M)	22.1	52.3	52.4	53.5 (+31.4)
GPT2-medium (345M)	20.3	41.9	38.2	46.5 (+26.2)
GPT2-xl (1.5B)	22.8	44.9	46.5	54.4 (+31.6)
MegatronLM (11B)	23.1	OOM*	OOM*	**64.2** (+41.1)

* MegatronLM (11B) is too large for effective fine-tuning.

Table 2. Knowledge probing Precision@1 on LAMA-34k (left) and LAMA-29k (right). P-tuning outperforms all the discrete prompt searching baselines. And interestingly, despite fixed pre-trained model parameters, P-tuning overwhelms the fine-tuning GPTs in LAMA-29k. (MP: Manual prompt; FT: Fine-tuning; MP+FT: Manual prompt augmented fine-tuning; PT: P-tuning).

Results

GPT Understands, Too

Method	BoolQ (Acc.)	CB (Acc.)	CB (F1)	WiC (Acc.)	RTE (Acc.)	MultiRC (EM)	MultiRC (F1a)	WSC (Acc.)	COPA (Acc.)	Avg.
BERT-base-cased (109M)										
Fine-tuning	72.9	85.1	73.9	71.1	68.4	16.2	66.3	63.5	67.0	66.2
MP zero-shot	59.1	41.1	19.4	49.8	54.5	0.4	0.9	62.5	65.0	46.0
MP fine-tuning	73.7	87.5	90.8	67.9	70.4	13.7	62.5	60.6	70.0	67.1
P-tuning	73.9	89.2	92.1	68.8	71.1	14.8	63.3	63.5	72.0	68.4
GPT2-base (117M)										
Fine-tune	71.2	78.6	55.8	65.5	67.8	17.4	65.8	63.0	64.4	63.0
MP zero-shot	61.3	44.6	33.3	54.1	49.5	2.2	23.8	62.5	58.0	48.2
MP fine-tuning	74.8	87.5	88.1	68.0	70.0	23.5	69.7	66.3	78.0	70.2
P-tuning	75.0 (+1.1)	91.1 (+1.9)	93.2 (+1.1)	68.3 (-2.8)	70.8 (-0.3)	23.5 (+7.3)	69.8 (+3.5)	63.5 (+0.0)	76.0 (+4.0)	70.4 (+2.0)

Table 3. Fully-supervised learning on SuperGLUE dev with base-scale models. MP refers to manual prompt. For a fair comparison, MP zero-shot and MP fine-tuning report results of a single pattern, while anchors for P-tuning are selected from the same prompt. Subscript in red represents advantages of GPT with P-tuning over the best results of BERT.

Results

Method	BoolQ (Acc.)	CB (F1)	CB (Acc.)	WiC (Acc.)	RTE (Acc.)	MultiRC (EM)	MultiRC (F1a)	WSC (Acc.)	COPA (Acc.)	Avg.
BERT-large-cased (335M)										
Fine-tune*	77.7	94.6	93.7	74.9	75.8	24.7	70.5	68.3	69.0	72.5
MP zero-shot	49.7	50.0	34.2	50.0	49.9	0.6	6.5	61.5	58.0	45.0
MP fine-tuning	77.2	91.1	93.5	70.5	73.6	17.7	67.0	80.8	75.0	73.1
P-tuning	77.8	96.4	97.4	72.7	75.5	17.1	65.6	81.7	76.0	74.6
GPT2-medium (345M)										
Fine-tune	71.0	73.2	51.2	65.2	72.2	19.2	65.8	62.5	66.0	63.1
MP zero-shot	56.3	44.6	26.6	54.1	51.3	2.2	32.5	63.5	53.0	47.3
MP fine-tuning	78.3	96.4	97.4	70.4	72.6	32.1	74.4	73.0	80.0	74.9
P-tuning	78.9 (+1.1)	98.2 (+1.8)	98.7 (+1.3)	69.4 (-5.5)	75.5 (-0.3)	29.3 (+4.6)	74.2 (+3.7)	74.0 (-7.7)	81.0 (+5.0)	75.6 (+1.0)

* We report the same results taken from SuperGLUE (Wang et al., 2019b).

Table 4. Fully-supervised learning on SuperGLUE dev with large-scale models. MP refers to manual prompt. For fair comparison, MP zero-shot and MP fine-tuning report results of a single pattern, while anchors for P-tuning are selected from the same prompt. Subscripts in red represents improvements of GPT with P-tuning over the best results of BERT.

Results

GPT Understands, Too

Prompt	$\mathcal{D}_{dev}$ Acc.	$\mathcal{D}_{dev32}$ Acc.
Does [PRE] agree with [HYP]? [MASK].	57.16	53.12
Does [HYP] agree with [PRE]? [MASK].	51.38	50.00
Premise: [PRE] Hypothesis: [HYP] Answer: [MASK].	68.59	55.20
[PRE] question: [HYP]. true or false? answer: [MASK].	70.15	53.12
P-tuning	76.45	56.25

Table 6. Few-shot performance comparison of different manual prompts and tuned prompts on RTE tasks using albert-xxlarge-v2. Experiments use $\mathcal{D}_{dev32}$ for model selection and hyper-parameter tuning and evaluate on $\mathcal{D}_{dev}$. There's no obvious correlations between manual prompts and performance. Besides, $\mathcal{D}_{dev32}$ is not able to select the best manual prompts.

6. Conclusion

- 이 논문에서는 연속 공간(continuous space)에서 더 나은 프롬프트를 자동으로 검색함으로써 사전 훈련된 모델의 자연어 이해 능력을 향상시키는 P-tuning을 제시합니다.
- 우리의 P-tuning 방법은 큰 검증 세트에 덜 의존하며, 적대적인 프롬프트로부터의 피해를 덜 받고, 과적합을 완화합니다.
- 테스트 시간 동안 추가 텍스트를 제공하지 않고 사전 훈련된 언어 모델에서 64% (P@1)의 세계 지식을 회복할 수 있음을 보여줍니다.
- SuperGLUE 벤치마크에서 P-tuning은 GPT 스타일의 모델에 자연어 이해에서 유사한 크기의 BERT와 경쟁력 있는 성능을 제공하며, 이는 과거에는 불가능하다고 생각되었습니다.
- P-tuning은 양방향 모델에도 도움을 주며 결과적으로 몇 번의 시도로 SuperGlue 벤치마크에서 state-of-the-art 방법을 능가합니다.
- 또한 사전 훈련 중에 우리가 생각했던 것보다 언어 모델이 세계 지식과 이전 작업 지식을 효과적으로 포착한다는 것을 증명합니다.