Paper Review: GPT Understands, too

Pre-trained LM의 Input으로 continuous prompt embedding을 넣을 수 있는 P-tuning 기법 소개

2021.06.09

발표자 조소영

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]

- Xiao Liu, Yanan Zheng et al.
- Tsinghua University etc
- arXiv preprint

Abstract

- **문제점**
  - GPT 모델들은 NLU에서 fine-tuning으로 좋은 성능을 내기 어려움

- **해결방안**
  - P-tuning이란?
    - 학습 가능한 continuous prompt embedding 사용하는 것
  - 이를 이용하면 NLU 태스크에서 비슷한 사이즈의 BERT에 필적하거나 보다 좋은 성능을 낼 수 있음
  - few-shot SuperGlue 벤치마크 데이터셋에서 State-of-the-art 달성
  - GPT 뿐 만 아니라 BERT 성능도 개선함

  - 핵심 : Big model은 fine tuning 하기 어렵고, 또 이미 좋은 representation을 가지고 있으니 이를 유지하고,
    대신 LSTM + MLP를 이용하여 Prompt를 더욱 좋게 개선해보자!

Introduction

- Language model pre-training은 **contextualized text representation** 뿐만 아니라 **grammar, syntactic, commonsense, world knowledge까지** 학습이 된다고 주장하는 연구 결과들이 있음.

- Language model pre-training 종류
  - **1. uni-directional language models** (GPT)
  - **2. bidirectional language models** (BERT)
  - **3. hybrid language models** (XLNet)

- 오래동안 GPT 스타일은 NLU 태스크에 적합하지 않다고 여겨져 왔음

- GPT-3는 적절한 prompt를 이용하여 NLU 태스크를 풀 수 있지만, 매번 좋은 prompt를 바로 찾는 것은 현실적으로 어려움!
  - automatic prompt searching (retrieval 등) 같이 discrete prompts를 찾는 방법들이 등장했지만, discrete prompt를 찾는 것은 차선책일 뿐임

➔ 따라서 본 논문은 continuous space에서 prompt를 찾는 *P-tuning*을 제안
  - *P-tuning*은 여러 NLU 태스크에서 상당한 성능 향상을 이룸.

Motivation

- **Giant Models의 문제점**
  - Big-model은 많은 문제를 해결해 주었으나 **poor-transferability** 문제가 있음
  - 즉, fine-tuning을 제대로 하기가 어려움.
  - Handcraft Prompt Search로 prompt를 찾아 적용할 수 있으나, prompt가 조금만 바뀌어도 큰 성능 차이가 있음

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.

→ 미분하여 최적화 할 수 있는 (학습 가능한) Continuous Prompt 를 이용하여 해결해보자!

Method

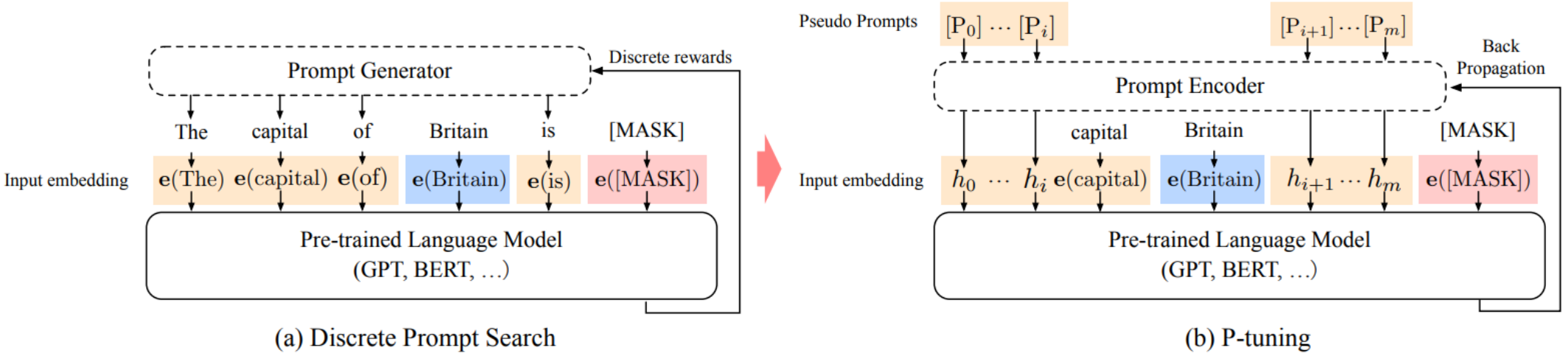
Prompt 개념

- 프롬프트는 context인 x와 타겟 y를 Template로 만들음
- 예시 문제 : 나라의 수도를 맞추는 태스크 (LAMA-TREx P36)
  - Template : The capital of Britain is [MASK]
  - Prompt : The capital of … is … .
  - Context x : Britain
  - Target y: [MASK]

The capital of Britain is [MASK]

Method

Discrete Prompts vs Continous Prompts(P-tuning)



- 기존 Inputs
  - e(token 0), e(token 1), ..., e(token n)
  - e는 embedding

- *P-tuning* Inputs:
  - h(P[0]), ..., h(P[i]), e(x), h(P[i+1]), ..., h(P[m]), e([MASK])
  - h(The), h(capital), h(of), e(Britain), h(is), e([MASK])
  - 이때 h는 학습 가능한 임베딩 텐서
  - 학습이 가능하기 때문에 원래 V가 표현할 수 있는 것보다 더 나은 continuous prompt를 찾을 수 있음

Method

Loss Function

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

- 다운스트림 loss function $\mathcal{L}$을 이용해 이 continuous prompt를 미분하여 최적화
- 예를 들어, The capital of Britain is [MASK] 태스크에서는 The capital of Britain is까지 주고 다음 [MASK]를 맞췄는지 못 맞췄는지에 대한 loss를 이용해 학습

Method

Optimization

- 비록 이렇게 continuous prompt를 학습시키는 것이 간단해 보이지만 2가지 optimization 문제 존재
  - Discreteness
    - 이미 pre-train 단계를 거치면, 단어 임베딩 e는 이미 discrete함
    - prompt embedding인 h가 random distribution으로 초기화 되고 SGD로 학습하는 경우 local minima에 빠지기 쉬움
  - Association
    - 각 임베딩 h_i는 다른 임베딩과 dependent 관계여야 함(rather than independent)
    - 따라서 이를 표현할 수 있는 메커니즘을 사용해야 할 것
- 즉, h 조건
  - h_i는 가벼운 뉴럴 네트워크로 만든 프롬프트 인코더를 사용해야 함
  - 또한, 연속적이고 토큰간 관계를 포착하는 sequence가 되도록 해야함

Method

Prompt Encoder 구조

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

따라서 논문에서는 Bi-directional LSTM과 MLP 구조 사용

```
PromptEncoder(
  (lstm): LSTM(hidden_size, hidden_size // 2, num_layers=2, bidirectional=True)
  (mlp): Sequential(
    (0): Linear(in_features=hidden_size, out_features=hidden_size)
    (1): ReLU()
    (2): Linear(in_features=hidden_size, out_features=hidden_size)
  )
)
```

Experiment

Knowledge Probing (LAMA Dataset)

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).

Fact Retrieval task (빈칸 맞추기)

Experiment

SuperGLUE

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.

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.

Conclusion

- P-tuning : Discrete 하게 prompt 를 주는 것(찾는 것) 보다 continuous 공간에서 prompt를 자동으로 찾도록 하는 것
- GPT, BERT에 상대적으로 작은 사이즈의 prompt-encoder를 도입해서 큰 성능 향상을 얻음
- 이를 통해 NLU에서도 BERT보다 GPT 성능이 높았음
- 같은 방법으로 GPT-3 같은 모델에도 p-tuning을 도입해서 성능 향상을 기대해 볼 수 있음