GPT-1

Improving Language Understanding
by Generative Pre-Training (2018)

Transformer의 장단점

- 병렬처리에 용이 (문장이 한꺼번에 들어감)
- Long-term dependency를 잘 해결
- Inductive Bias 가 낮음

  : inductive bias 란? 머신러닝 문제를 풀기 위해 사전에 추가된 정보

  : inductive bias가 높으면 작은 데이터셋으로도 잘 학습됨 (간단한 모델일때 CNN보다 성능이 안 좋다…)

  : 하지만 inductive bias가 낮기 때문에 더 많은 데이터를 다 받아들일 수 있다는 장점이 있음

  : 하지만 그만큼 큰 labeled dataset을 매번 만드는게 쉽지 않음

Transfer learning

Pre-Training
(on large model)

+

Fine-Tuning
(to specific task)

Transfer learning

Pre-Training
(on large model)

- Unlabeled data에 대한 unsupervised learning이 이루어짐
- 데이터 라벨링이 필요없으므로 대량의 데이터수집이 가능해짐
- 큰 모델에 대한 대량의 데이터 학습 => 언어에 대한 전반적인 이해

Fine-Tuning
(to specific task)

- 수행하고자 하는 특정 task에 맞는 labeled 데이터셋으로 추가학습
- 몇번의 학습 (epoch 3~4) 만으로도 optimization 가능
- 경제적이다 (한가지로 우려먹기 가능, 학습 비용 절감)

Abstract

- Unlabeled data는 많은데, 라벨링이 힘들기 때문에 각 task마다 큰 데이터셋을 만들기가 쉽지 않음

- Unlabeled text corpus로 학습시키는 generative pre-training과 specific task에 맞춘 fine-tuning 구조를 제안

- Task에 맞춰 개발된 아키텍처들보다 뛰어난 성능을 보임 (state-of-the-art와 비교했을 때)

선행연구에서는…

1. pre-train에 사용할 optimization objective가 clear하지 않음
   - Language modeling objective를 사용

2. Fine-tuning을 진행할 transfer 방법에 대한 의견일치 X
   - 구조 변화를 최소화하기 위해 입력값과 학습 object만을 변화시키는 방법 사용

structure

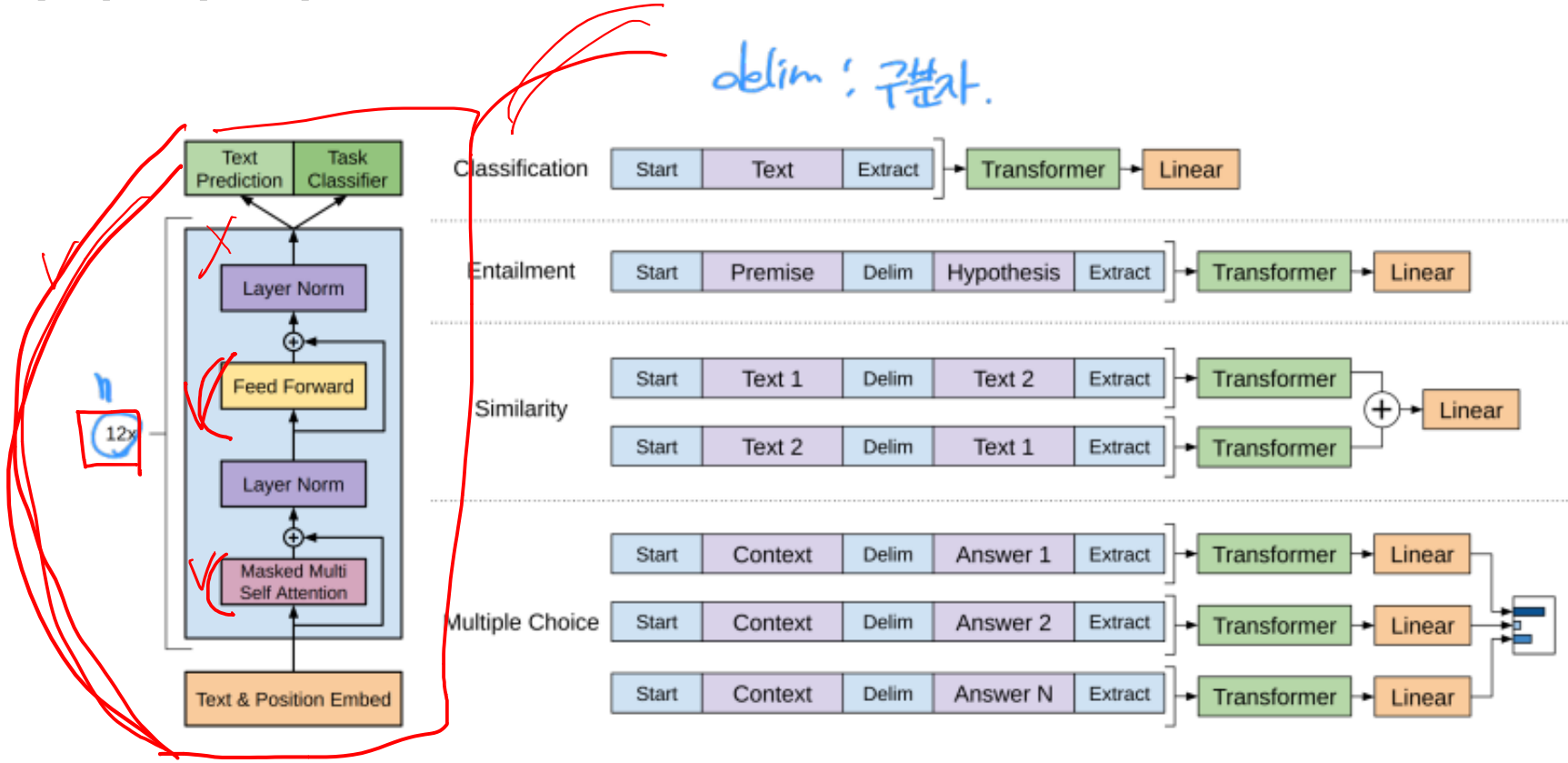


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

1. Unsupervised pre-training

unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$

Standard language modeling objective $L_1(\mathcal{U}) = \sum \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$

k : context window
Θ : parameters.

문맥창

$h_0 = UW_e + W$

$h_l = \texttt{transfor}$

Mult

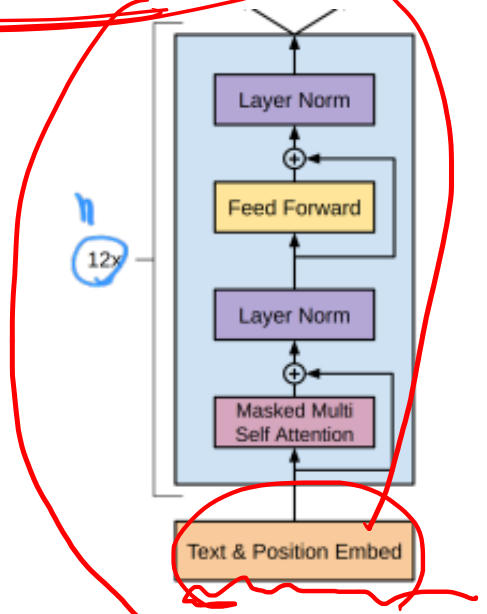
$P(u) = \texttt{softmax}($

크기 $2k+1$의 단어열에 대해 2)

기준단어(April)주변의 단어가 문맥

$k=4$의 예

문맥



2. Supervised fine-tuning

a labeled dataset C: sequence of input tokens $x^1, \ldots, x^m$ & label y

→ task에 맞게 변형된 input

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y)$$

last Transformer activation

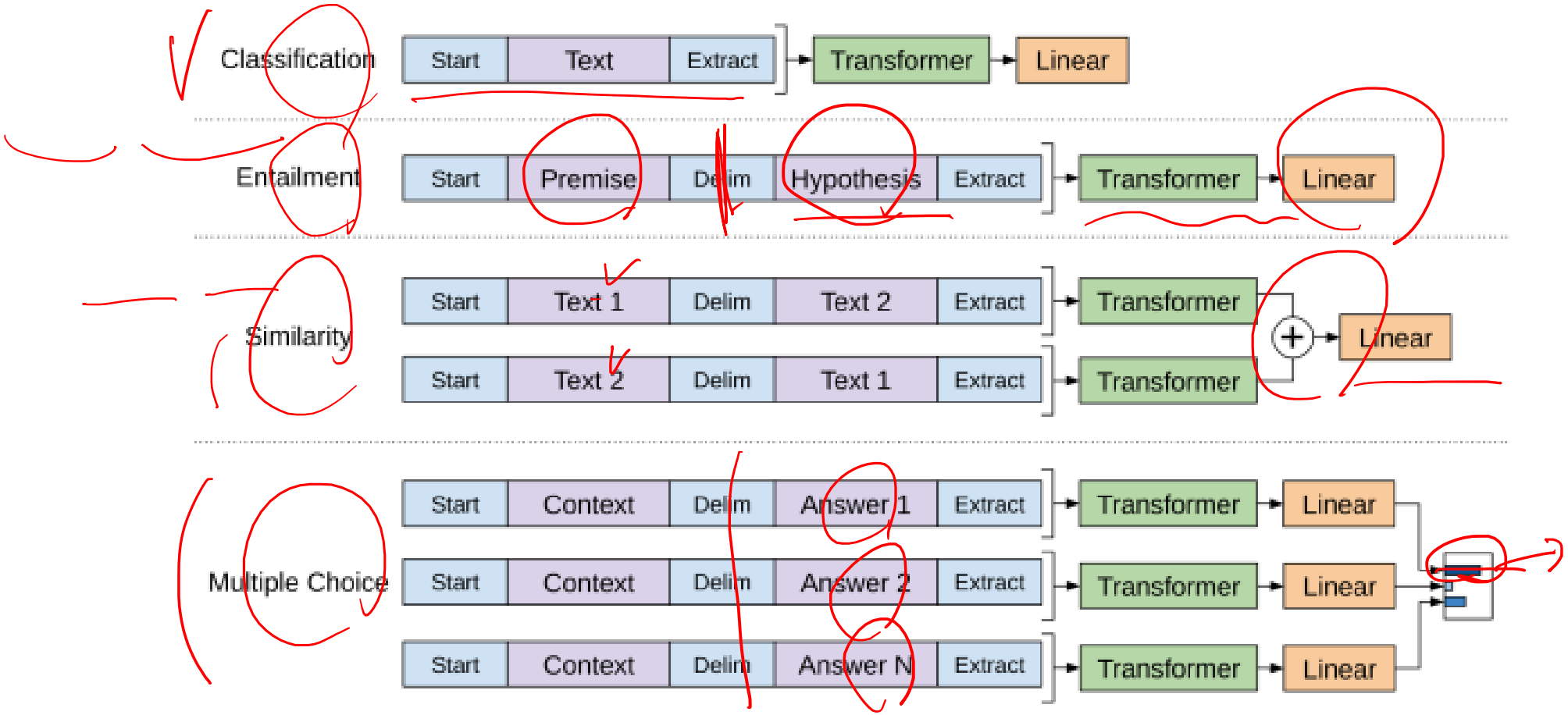
$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m)$$

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

L1을 보조적 objective로 사용하는 장점

1. Improving generalization
2. Accelerating convergence

3. Task specific input transformations



모델 스펙

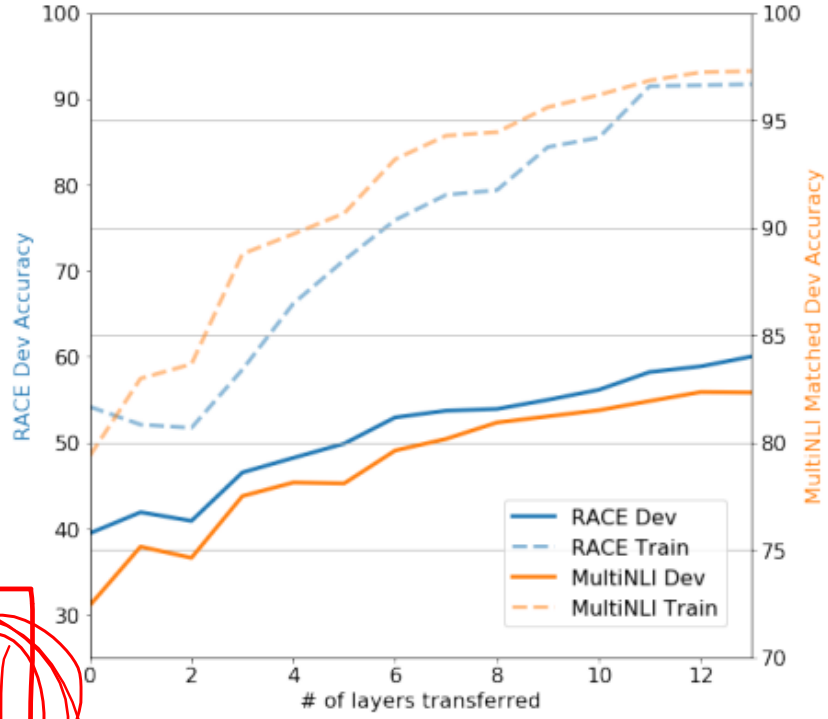
Pre-training

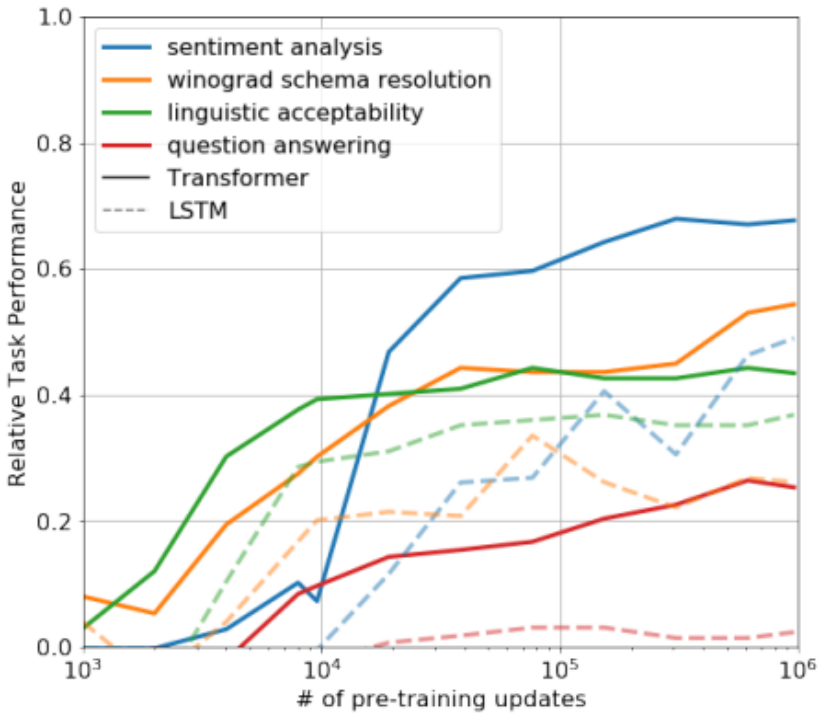
- 12 layer decoder-only transformer (masked self-attention, FFNN)
- Adam optimization
- Scheduled learning rate
- 100 epochs
- 64 minibatches
- Token 길이 512

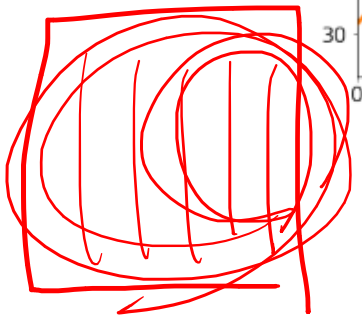
Fine-tuning

- Batch size 32
- 3 epochs
- Learning rate decay

분석





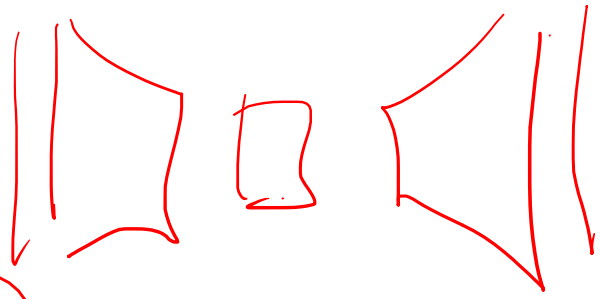


- 레이어를 많이 transfer 할수록 더 성능이 좋아졌다.

추가 연구

1. fine-tuning에서 보조적 LM objective를 사용 유무
   - 큰 데이터셋은 보조적 LM을 사용했을 때 이득!
2. pre-training 모델로 Transformer와 LSTM 비교
   - 대부분의 데이터셋에서 transformer가 더 좋았음!
3. Pre-training 유무 비교
   - pre-training 없이 처음부터 labeled data로 학습시켰을 때 전반적인 성능이 크게 떨어졌음!

GPT vs BERT



GPT: transformer의 decoder 사용
⇒ Natural Language Generation (NLG) task에 강하다!

BERT: transformer의 encoder 사용
⇒ Natural Language Understanding (NLU) task에 강하다!

T5: encoder, decoder 모두 사용
⇒ 두 개의 task에 모두 강하지만, 파라미터가 굉장히 많다.

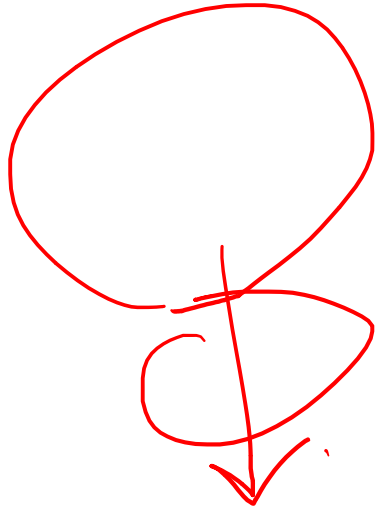
Hugging face 소개

pre-training을 하려면 엄청나게 많은 데이터로 엄청 큰 모델을 학습시켜야 한다.

개인이 학습시키기에는 비용이 많이 든다. (불가능)

기업 등에서 pre-train 완료된 모델들을 많이 공개해놓았다.

Python 라이브러리인 hugging face에서 아주아주 쉽게 사용할 수 있다!



The AI community building the future.

Build, train and deploy state of the art models powered by the reference open source in machine learning.

질문