Improving Language Understanding by Generative Pre-Training

Alec Radford et al. (2018, OpenAI)

김유리

Index

Introduction

Introduction

Labeled text data의 부족과

충분한 labeled text data가 있어도 **“unlabeled text data로 더 나은 성능을 이룰 수 있지 않을까?”**라는 기대

Introduction

하지만, unlabeled text data로 word-level이상의 정보를 활용하는 것에는 어려움 존재

- Transfer하기 위한 text representation을 학습하기에 **어떤 optimization objective가 효과적인지 불분명**
- 이렇게 학습된 representation을 target task로 transfer할 때 **어떤 방법이 가장 효과적인지 의견일치가 되지 않음**

→ 이러한 불확실성이 **semi-supervised learning을 어렵게 함**

Introduction

본 논문에서는 **unsupervised pre-training + supervised fine-tuning** 을 통해 NLU task에 대한
semi-supervised approach를 연구
→ **GPT** 제안

Introduction

GPT의 목표

→ **다양한 task에 적용 가능한 universal representation 학습**

Background

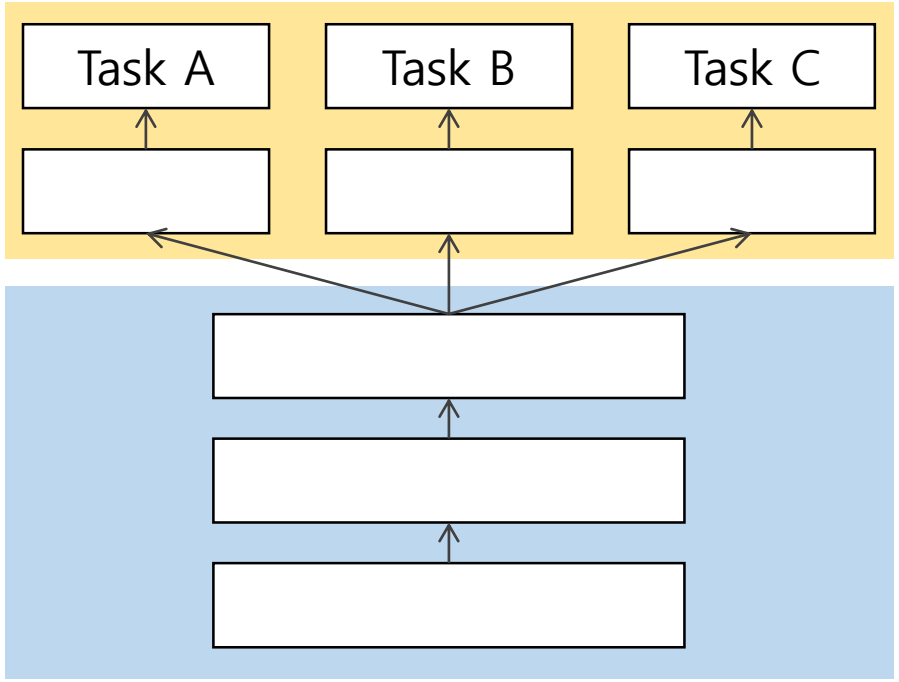
Background
Unsupervised pre-training

Word-level embedding에서 **Context-level embedding으로 발전**

Traditional word vectors	Word Embeddings	More than word-level semantics
Bag of Words TF-IDF Distributional Embeddings …	Word2Vec GloVe FastText	ELMo CoVe

Background
Auxiliary training objectives

Language modeling object 등 **unsupervised training objective를 두면 성능 향상에 효과**가 있음



$L_2(C)$: some task-specific objective

$$L_2(C) = \sum_{(x,y)} logP(y|x^1, x^2, \dots, x^m)$$

$L_1(C)$: unsupervised training objective

$$L_1(C) = \sum_{i} logP(u_i|u_{i-k}, \dots, u_{i-1}; \theta)$$

Model Architecture

Model Architecture

GPT의 목표는 **다양한 task에 적용 가능한 universal representation 학습**

어떻게?

1. 어떤 Architecture? : **Transformer 구조 사용**
2. Representation학습 방식 : **Pre-train Language model with unsupervised learning**
3. Representation을 task-specific하게 transfer하는 방식 : **Very small modification to pre-trained network**

Model Architectures
GPT

1. 어떤 Architecture? : **Transformer 구조 사용**

   → **Transformer의 decoder사용**

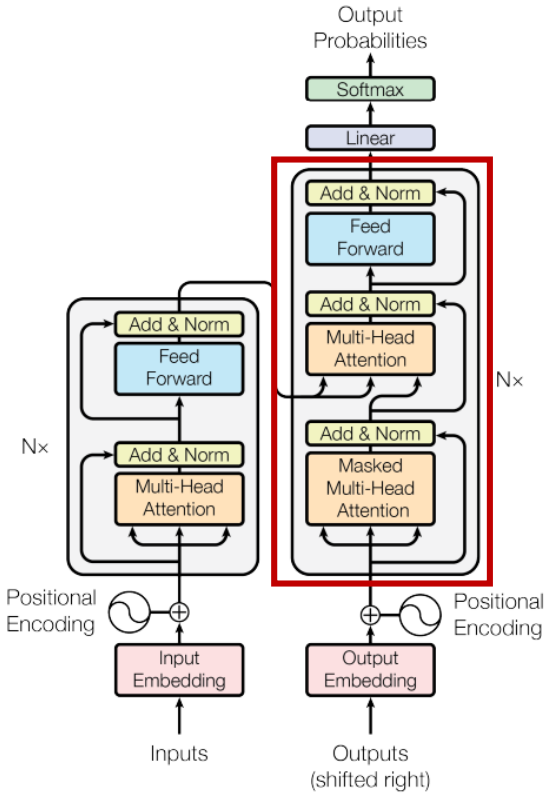
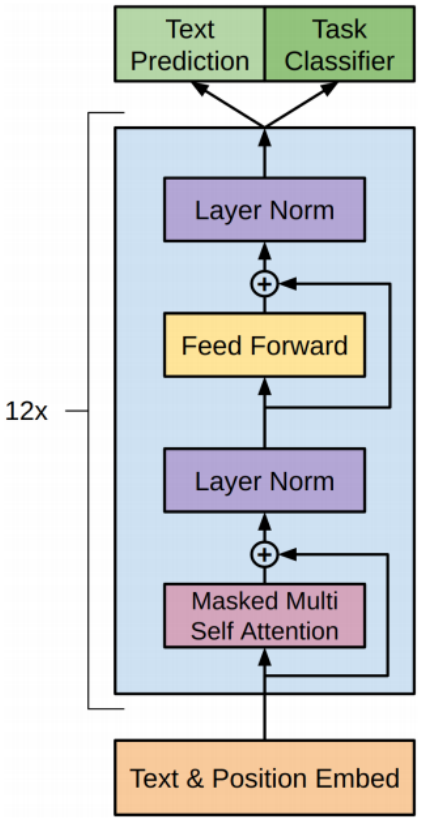


Figure 1: The Transformer - model architecture.



- 기존의 Transformer : Encoder/Decoder 6쌍으로 구성
- GPT : Decoder만 12개(Multi-head)로 구성

Model Architectures
GPT

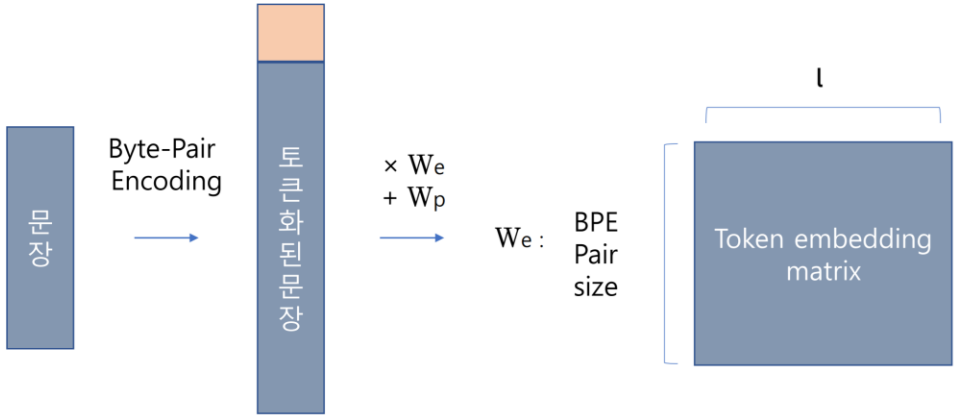
2. Representation학습 방식 : **Pre-train Language model with unsupervised learning**

→ GPT는 **unsupervised pre-training → supervised fine-tuning** 두 단계로 나뉨

Model Architectures
Unsupervised pre-training

unsupervised pre-training

이 단계에서는, 문장단위로 Encoding하고 Transformer decode를 거쳐 Context-level Embedding을 하는 과정을 통해 Unsupervised Learning을 사용한 LM 학습



Model Architectures
Unsupervised pre-training

Unsupervised 코퍼스의 token

$$u = \{u_i, \dots, u_n\}$$

Likelihood를 최대화하는 standard language modeling objective사용

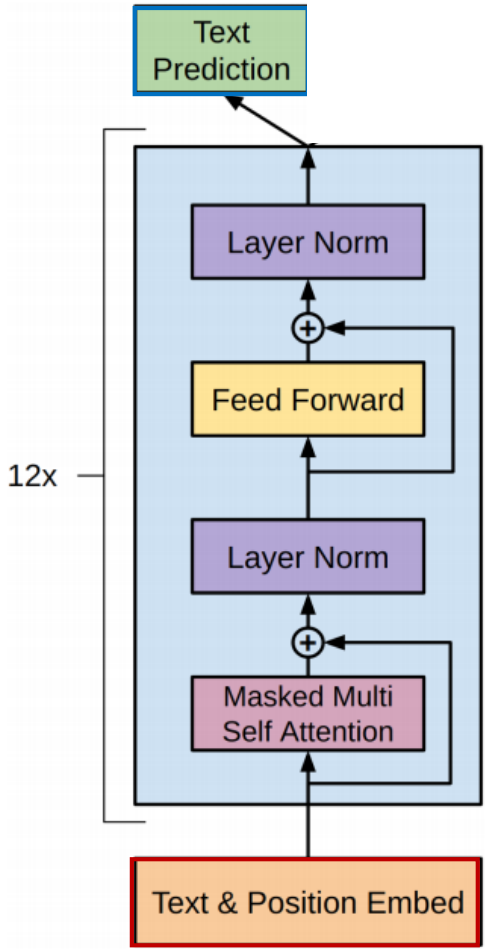
$$L_1(u) = \sum_i logP(u_i|u_{i-k}, \dots, u_{i-1}; \theta)$$

$$U = \{u_{-k}, \dots, u_{-1}\}$$

$$h_0 = UW_e + W_p$$

$$h_1 = \mathrm{t}ransformer\_block(h_{l-1}) \forall i \in [1, n]$$

$$P(u) = \mathrm{softmax}(h_n W_e^T)$$



k : context window 크기
θ : Neural network의 parameters
(parameter들은 stochastic gradient descent이용하여 학습)
U : token의 context vector
n : layer 개수
W_e : token embedding matrix
W_p : position embedding matrix

Model Architectures
Supervised fine-tuning

supervised fine-tuning

Supervised Learning부분은 크게 두 부분으로 나눠 짐

1. Text/Position Embedding부터 12개의 Decoder가 있는 부분인 **Pretrained Model**
2. 그리고 **Task Prediction/Classification** 부분

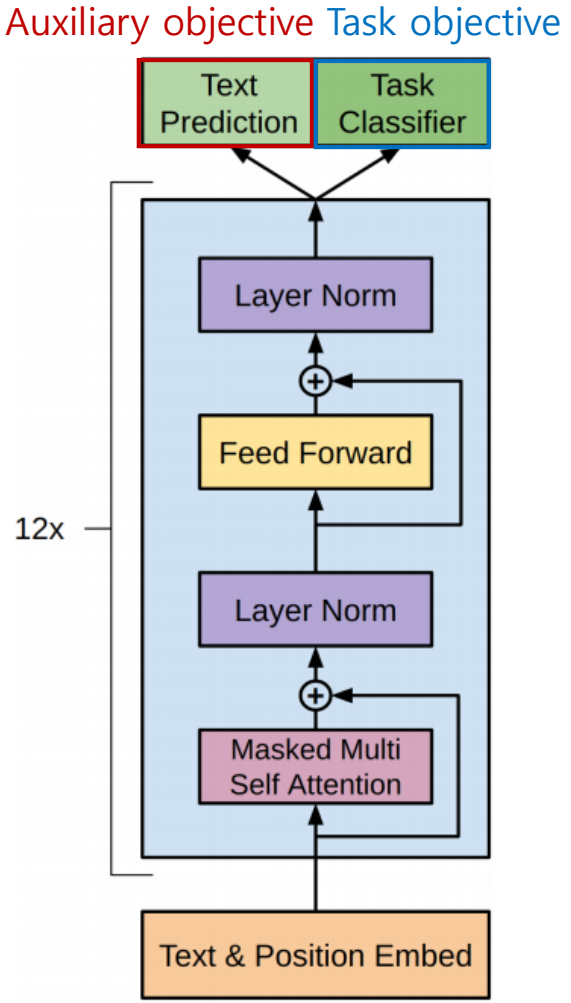
Model Architectures
Supervised fine-tuning

supervised fine-tuning

1. Text/Position Embedding부터 12개의 Decoder가 있는 부분인

Pretrained Model

- Global한 NLP feature를 학습하도록 구성
- 각 Embed는 Unsupervised에서도 언급했듯이 BPE로 구성
- 이렇게 학습된 representation은 Context에 대해 소실되는 정보가 거의 없이 학습된다고 가정하고, 이를 Decoder를 통해 Task에 맞는 정답 Feature를 추출

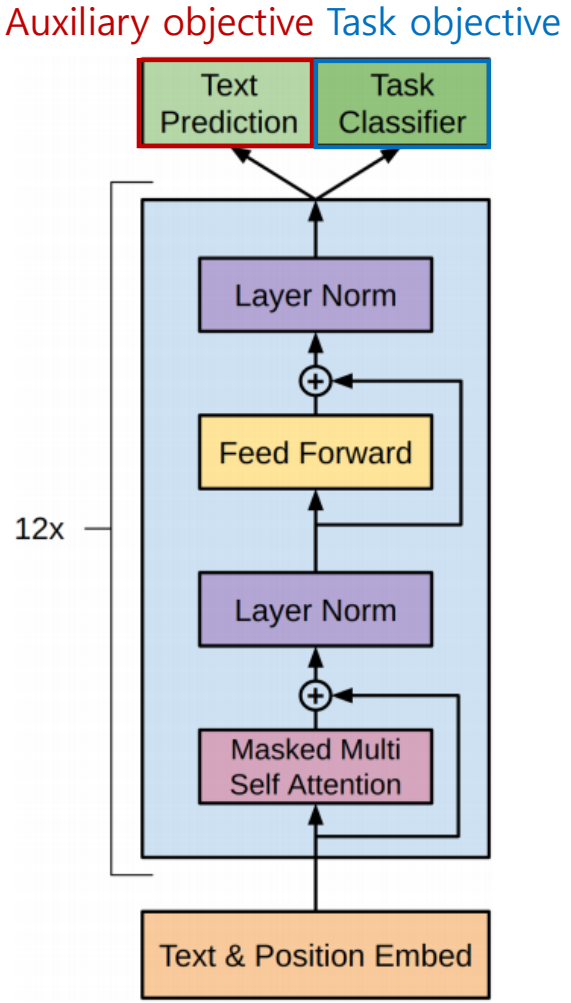


Model Architectures
Supervised fine-tuning

supervised fine-tuning

2. 그리고 **Task Prediction/Classification** 부분

- Task Classifier 또는 Task Prediction처럼 하나의 예상만 출력하지 않음
- 논문에서도 이러한 구조를 Auxiliary Task라는 용어로 표현
- (쉽게 풀어 말하면, 하나의 Task objective에 대해서만 학습하는 것 보다 Auxiliary objective(sub-task)를 같이 학습하는 것이 주요 task에 대한 정확도를 높여 줌)



Model Architectures
Supervised fine-tuning

하나의 layer에 대한 linear layer추가

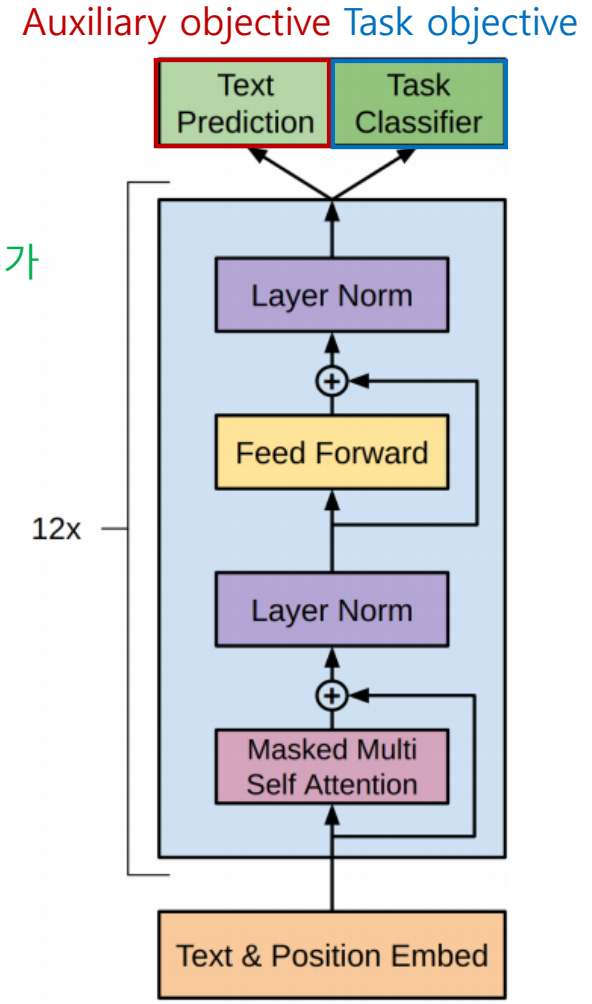
$$P(y|x^1, x^2, \dots, x^m) = softmax(h_l^m W_y)$$

아래 layer가 최대가 되도록 학습

$$L_2(C) = \sum_{(x,y)} logP(y|x^1, x^2, \dots, x^m)$$

즉, 아래의 objective를 최적화

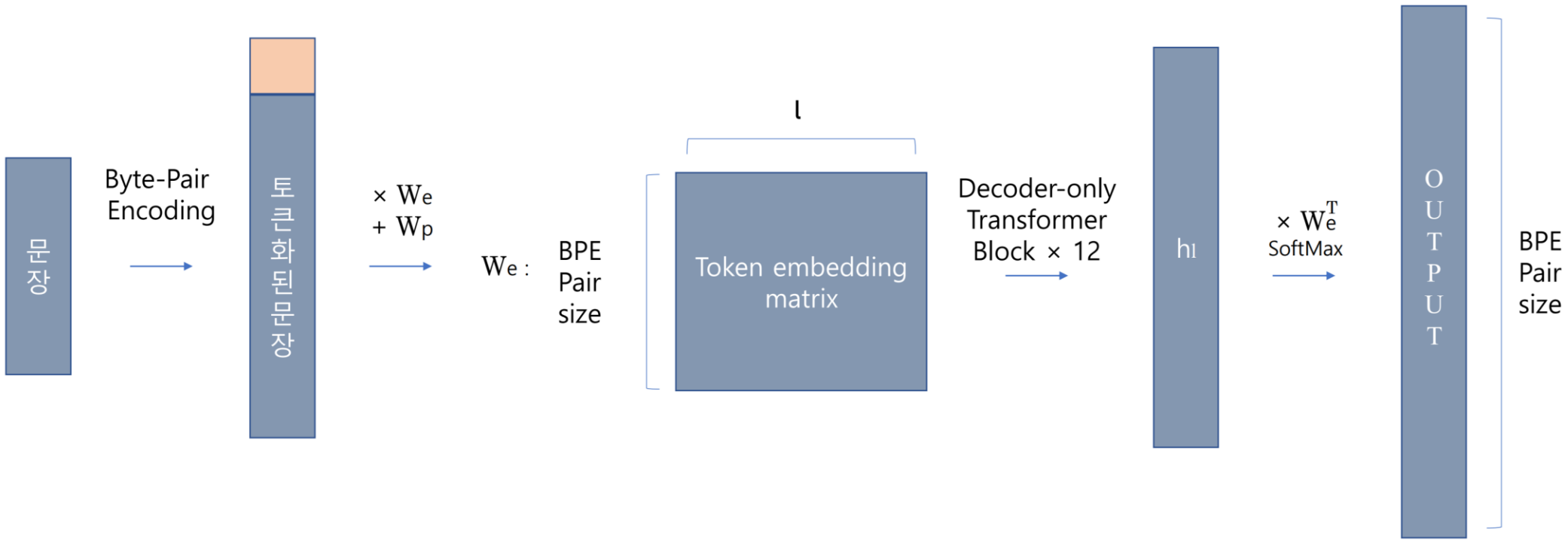
$$L_3(C) = L_2(C) + \lambda L_1(C)$$



C : Labeled dataset
$x^1, x^2, \dots, x^m$: input token
(parameter들은 stochastic gradient descent이용하여 학습)
y: input token에 해당하는 label
h_l^m: 마지막 transformer 블록의 activation
W_y : h_l^m을 input으로 하는 linear layer
$L_2(C)$: some task-specific objective
$L_1(C)$: unsupervised training objective

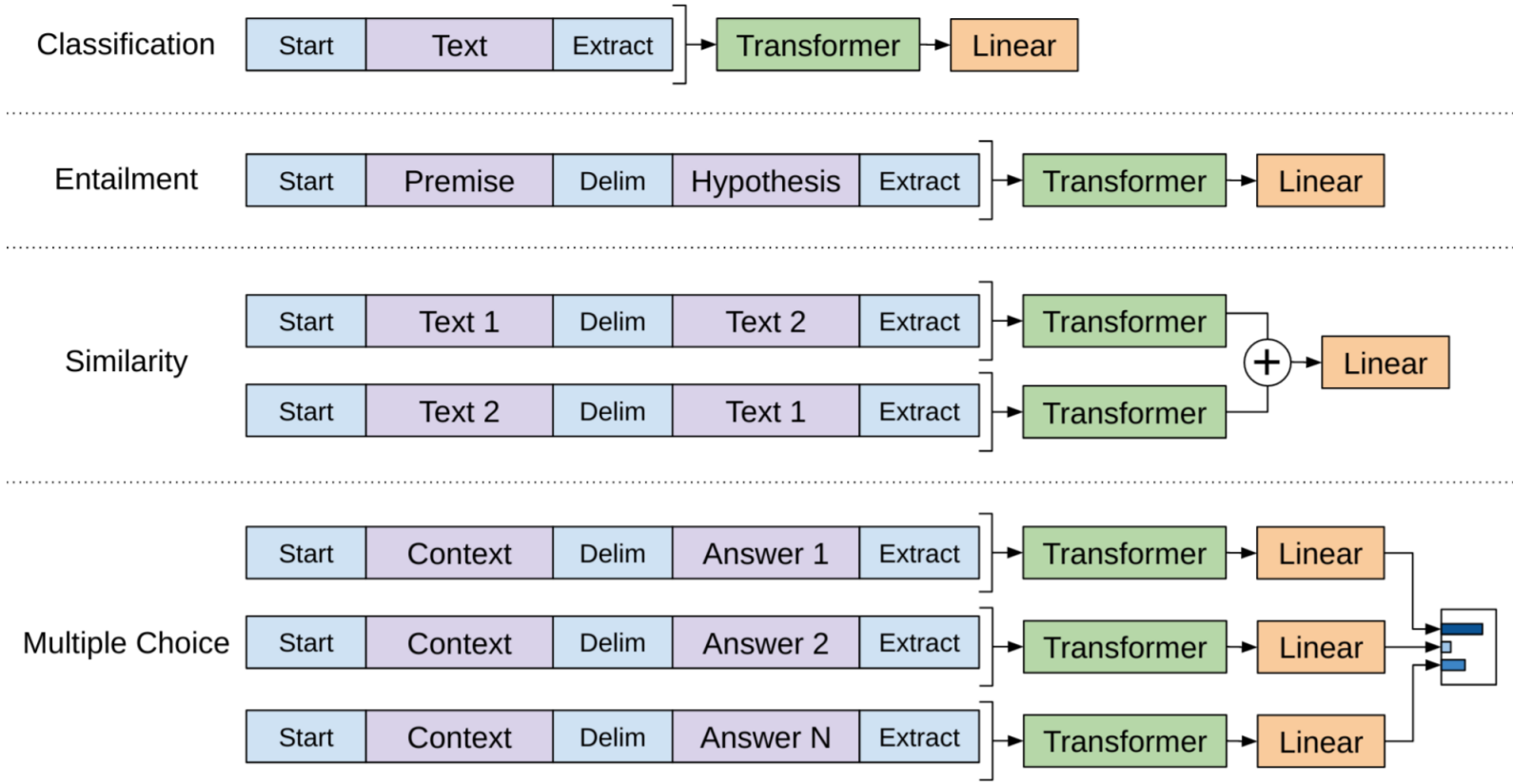
Model Architecture

GPT-1의 전체 구조



Model Architecture

다른 task에서 어떻게 사용?



- Classification은 True/False 또는 category를 예측하기 위해 하나의 구조만 가짐
- Similarity나 Multiple Choice의 경우 Context/Text를 비교하기 위해 각 부분마다 모델을 적용시킨 후, 이를 취합하는 구조

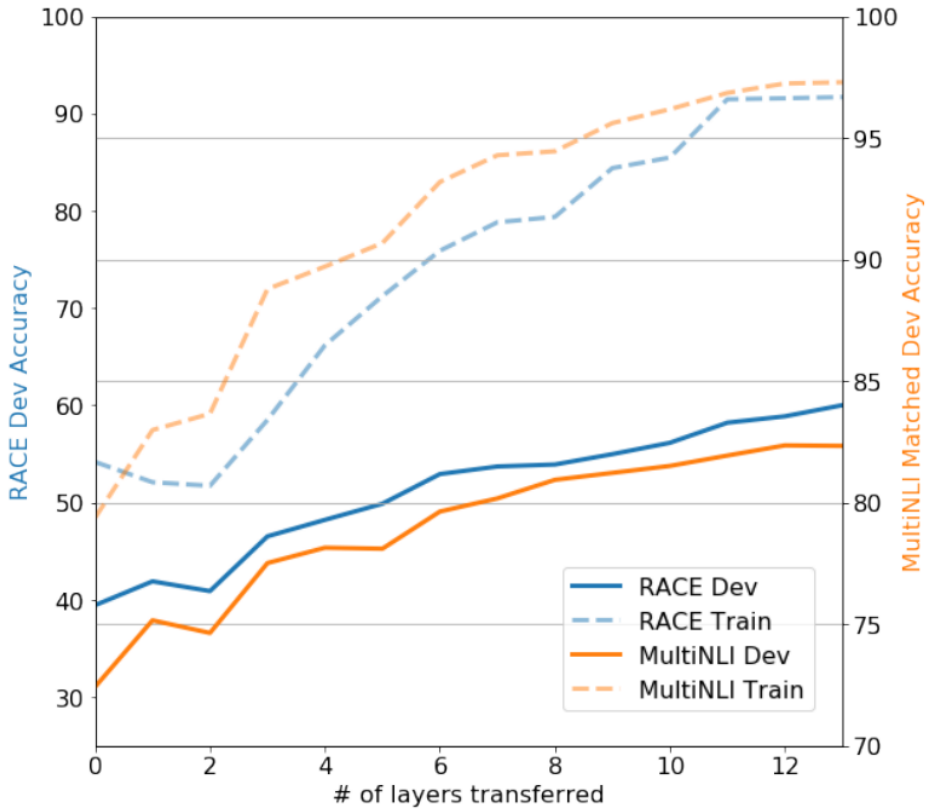
→ GPT 모델을 사용할 때, 하고자 하는 **Task에 맞춰 모델을 알맞게 구성**해 줄 필요가 있음

Experiments

Experiments
Model specifications

- 12 decoder-only transformer
- Adam optimization
- Cosine annealing : learning rate schedules with restart
- Input : Contiguous sequences of 512 tokens
- Weight initialization of N(0, 0.02)
- BPE with 40,000 merges

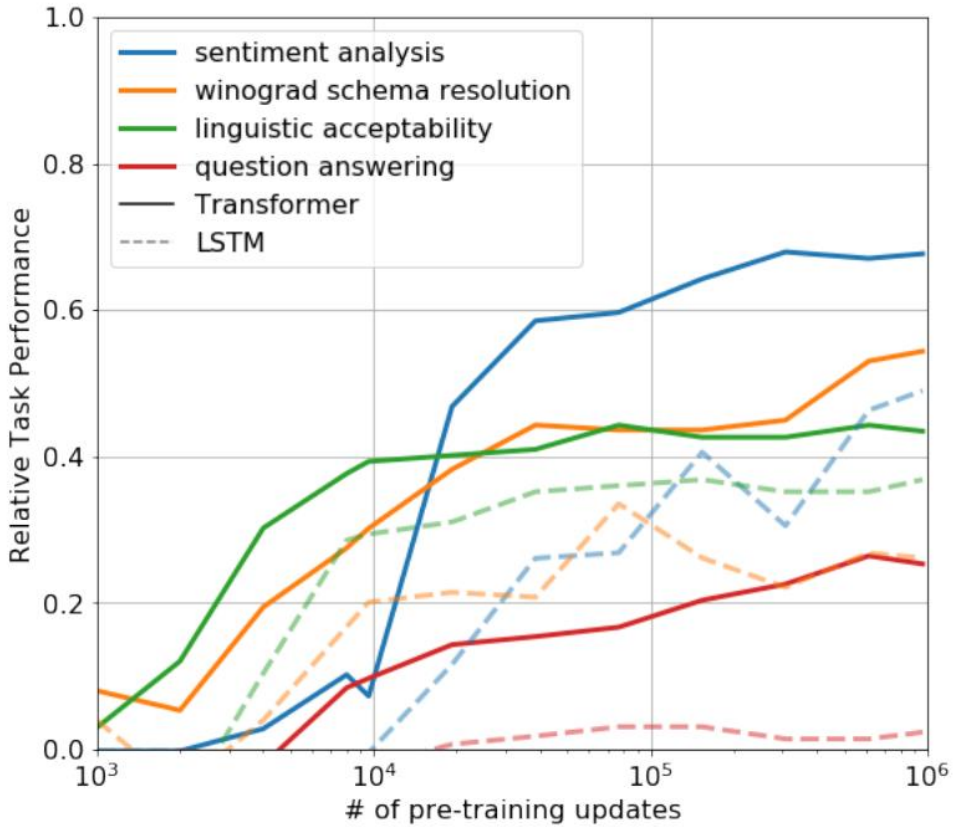
Experiments
Analysis



Transformer Decoder 개수에 따른 accuracy 변화

- RACE, MultiNLI : dataset
- RACE : Question Answering(QA)를 목적
- MultiNLI : textual entailment 또는 Natural Language Inference(NLI)를 목적으로 함
- 두 데이터 셋 모두 Layer 수가 많아질수록 정확도가 비약적으로 상승 (12개정도에서 정확도가 Converge하는듯 함)

Experiments
Analysis



Transformer의 사용 유무에 따른 차이

- 점선(LSTM) / 실선(Transformer)
- 각 색상은 task를 나타냄
- task별로 증가율의 차이는 있지만, 모두 상대적으로 performance가 증가
- LSTM은 higher variance를 보이는 반면에 Transformer는 transfer에 도움이 됨

Experiments
Analysis

Method	Avg. Score	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	MNLI (acc)	QNLI (acc)	RTE (acc)
Transformer w/ aux LM (full)	74.7	45.4	91.3	82.3	82.0	**70.3**	**81.8**	**88.1**	**56.0**
Transformer w/o pre-training	59.9	18.9	84.0	79.4	30.9	65.5	75.7	71.2	53.8
Transformer w/o aux LM	**75.0**	**47.9**	**92.0**	**84.9**	**83.2**	69.8	81.1	86.9	54.4
LSTM w/ aux LM	69.1	30.3	90.5	83.2	71.8	68.1	73.7	81.1	54.6

Auxiliary Objective(sub-task) 유무에 따른 성능 그리고 pre-training이 없을 때의 성능을 보여줌

- 왼쪽 4가지와 오른쪽 4가지 task 결과가 다른데 이것은 dataset size가 다르기 때문
  - 즉, Dataset이 클수록 (QQP, MNLI, QNLI, RTE) auxiliary task가 성능 개선에 여향이 더 크고
  - 작을수록 (CoLA, SST2, MRPC, STSB) auxiliary task없이 학습하는 것이 오히려 성능에 도움이 됨
- Transformer사용 여부에 대한 성능평가도 측정, 모든 경우에 LSTM대신 Transformer를 사용하는 것이 성능 개선에 도움이 됨
- pre-training 유무에 대한 성능평가에서는 full 모델에 비해 pre-training이 없을 때 전체적으로 성능이 매우 감소
  - (pre-training을 안 한다? → unsupervised pre-training 구조를 모두 넘겨버리고 supervised 부분만 사용)

Conclusion

Conclusion

generative pre-training과 discriminative fine-tuning을 통해
task-agnostic model로 강력한 NLU를 성취할 수 있음을 보임

Conclusion

하지만, GPT-1은 BERT에 비해 주목 받지 못함

∵ BERT가 더 범용적으로 쓰이기 용이하며, 성능도 BERT에 비해 좋다는 소식이 안 들림 (SQuAD 1.1, 2.0을 보면 GPT에 대한 성능결과가 없음)

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or *span*, from the corresponding reading passage, or the question might be unanswerable.

Conclusion

그럼에도 불구하고 이 논문을 살펴봐야 할 이유

Decoder로서 Transformer를 Pre-trained Language Model생성에 어떻게 사용

하고 있는가에 대한 좋은 예시가 GPT이기 때문

(BERT와 GPT-1은 많은 부분이 유사)

BERT	GPT
Masked LM 사용	일반적인 LM 사용
Transformer encoder사용	Transformer decoder사용

Thank you