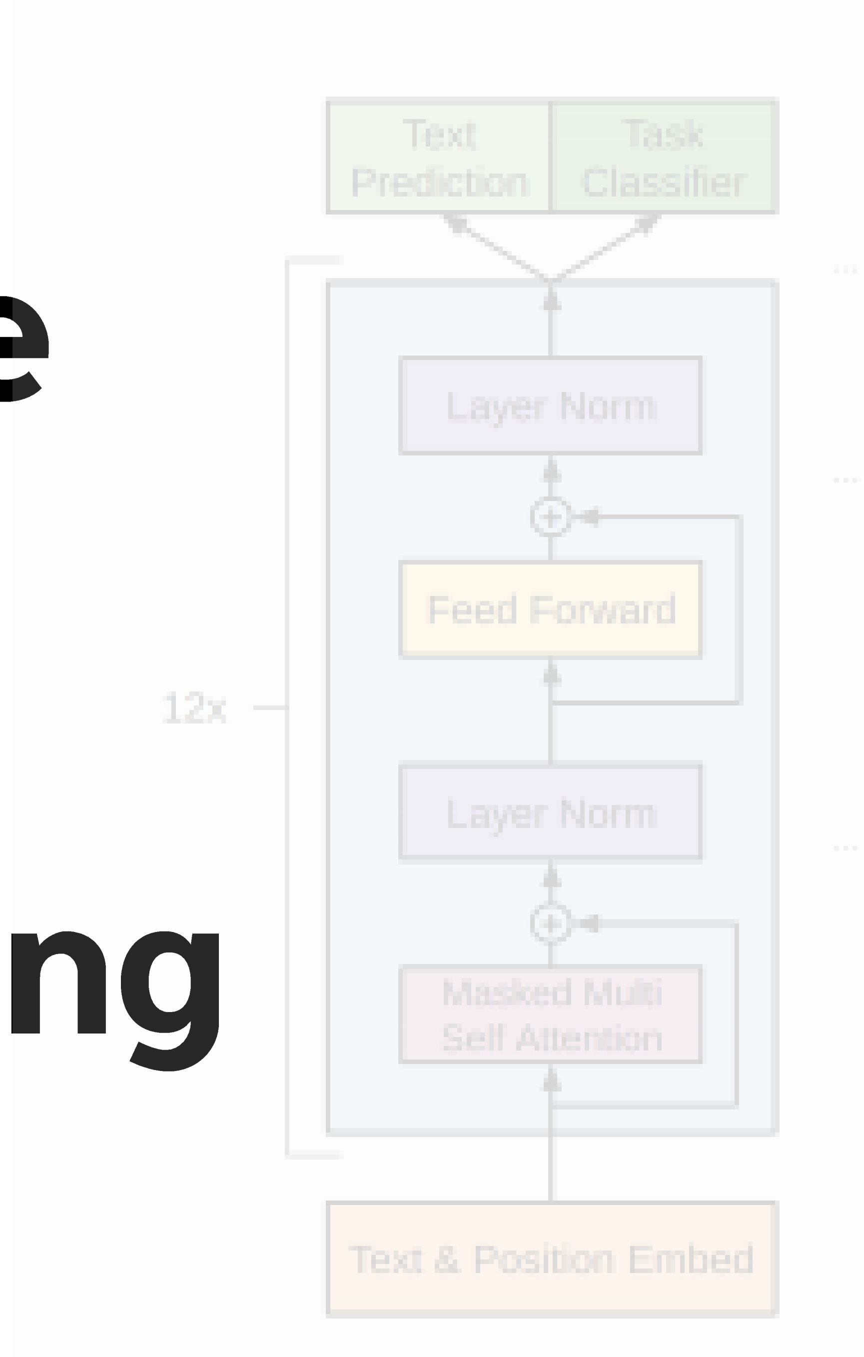
gpt-1

Improving Language Understanding by Generative Pre-Training

2018



목 차

등장 배경

문제제기

1

지도 학습 방식의 문제

→ Labeled data를 충분히 확보하기 쉽지 않음

2

Unlabeled data를

효과적으로 활용할 수 있는

방법 부재

3

서로 다른 task에 대해서

새로운 모델을 학습시켜야함

→ 기계번역, 질의응답, 문장 분류, 상식 추론 등

제안 방법

unsupervised pre-training + **supervised fine-tuning**

전이

제안 방법

unsupervised pre-training

- 대규모 unlabeled text data 사용
- Language modeling task (다음 단어 예측) 언어 모델 학습

→ 문장의 일반적인 구조, 문맥, 어휘 정보를 먼저 학습

supervised fine-tuning

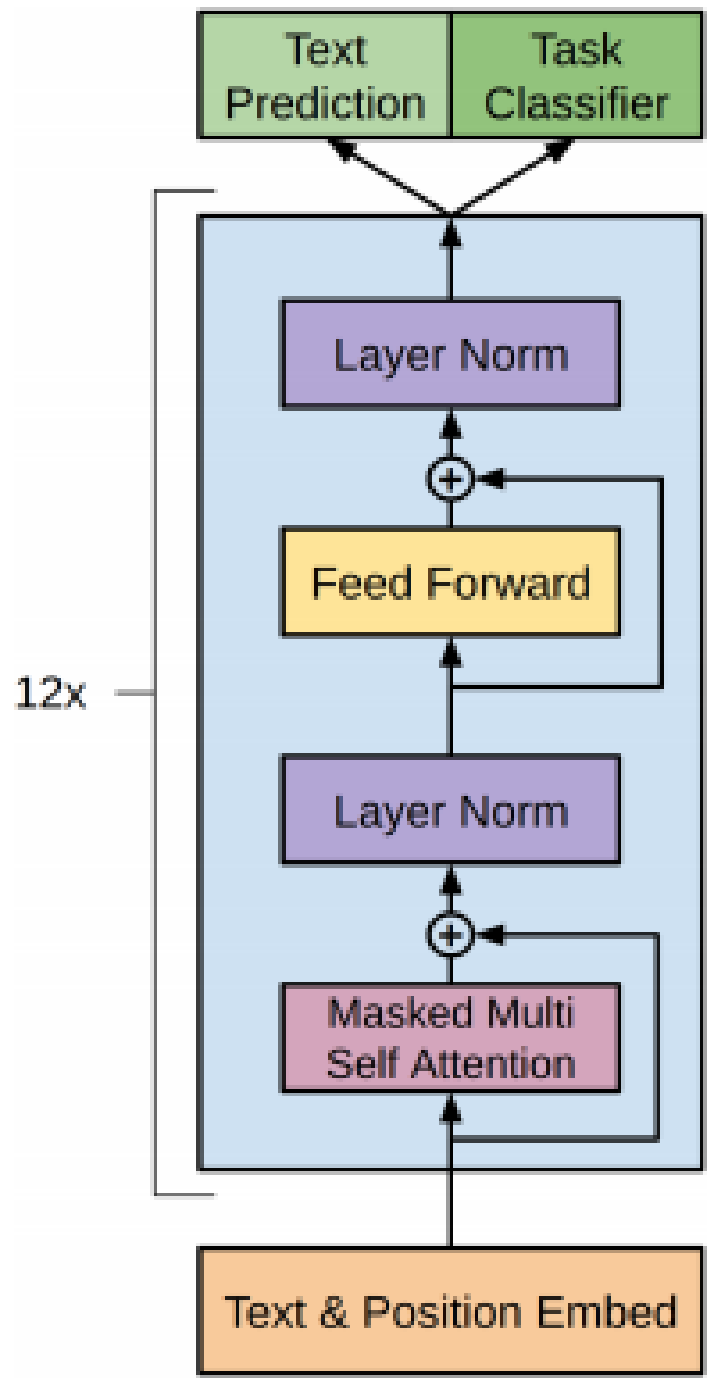
- labeled text data 사용

  → 사전 학습에 사용된 데이터와 같은 도메인일 필요 없음

- 구체적인 task에 맞춰 추가 학습

Framework

Transformer



Transformer에서 decoder 구조를 가져와 사용

Pre-Training

Objective = standard language modeling objective

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

U = {u_1 , . . . , u_n} : unsupervised corpus of tokens
k : size of the context window
Θ : neural network parameters

Pre-Training

$$h_0 = UW_e + W_p$$

$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$

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

W_e: token embedding matrix
W_p: position embedding matrix

n = decoder layer(stack) 개수

Fine-Tuning

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

{x_1 , . . . , x_m} : input tokens / y : label
W_y : embeddings for delimiter tokens
m: input token 개수

h_ml: 최종 디코더 레이어 h_l에서 나온 마지막
토큰 x_m의 hidden state

Fine-Tuning

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

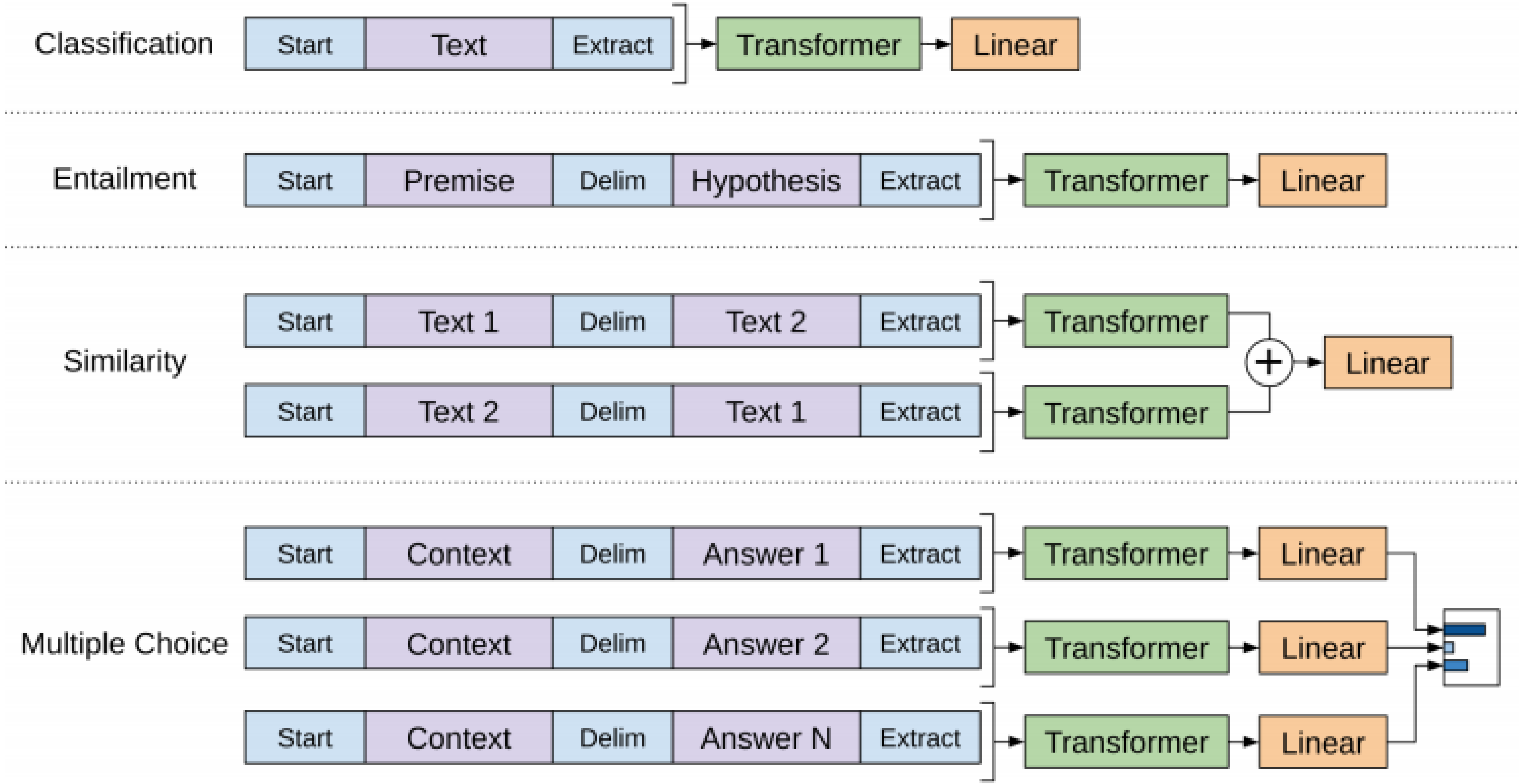
Objective = auxiliary objective

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

($L_1(\mathcal{U}) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)$)

C: labeled dataset
λ : Pre-Training Loss L_1의 가중치

Fine-Tuning


Classification
Start
Text
Extract
Transformer
Linear
Entailment
Start
Premise
Delim
Hypothesis
Extract
Transformer
Linear
Similarity
Start
Text 1
Delim
Text 2
Extract
Transformer
Start
Text 2
Delim
Text 1
Extract
Transformer
+
Linear
Multiple Choice
Start
Context
Delim
Answer 1
Extract
Transformer
Linear
Start
Context
Delim
Answer 2
Extract
Transformer
Linear
Start
Context
Delim
Answer N
Extract
Transformer
Linear

Experiments
&
Analysis

Setup

Unsupervised pre-training

- BooksCorpus dataset : 다양한 장르의 7,000개 이상의 책 데이터셋
-모델이 long-range 정보를 학습하도록 만들 수 있는 긴 텍스트가 포함

Model specifications

- 12-layer decoder-only transformer with masked self-attention heads
- 12개의 attention heads

Fine-tuning details

- reuse the hyperparameter settings from unsupervised pre-training
- add dropout with a rate of 0.1
- learning rate = 6.25e-5
- Batch size = 32 / epochs = 3
- λ = 0.5

Results

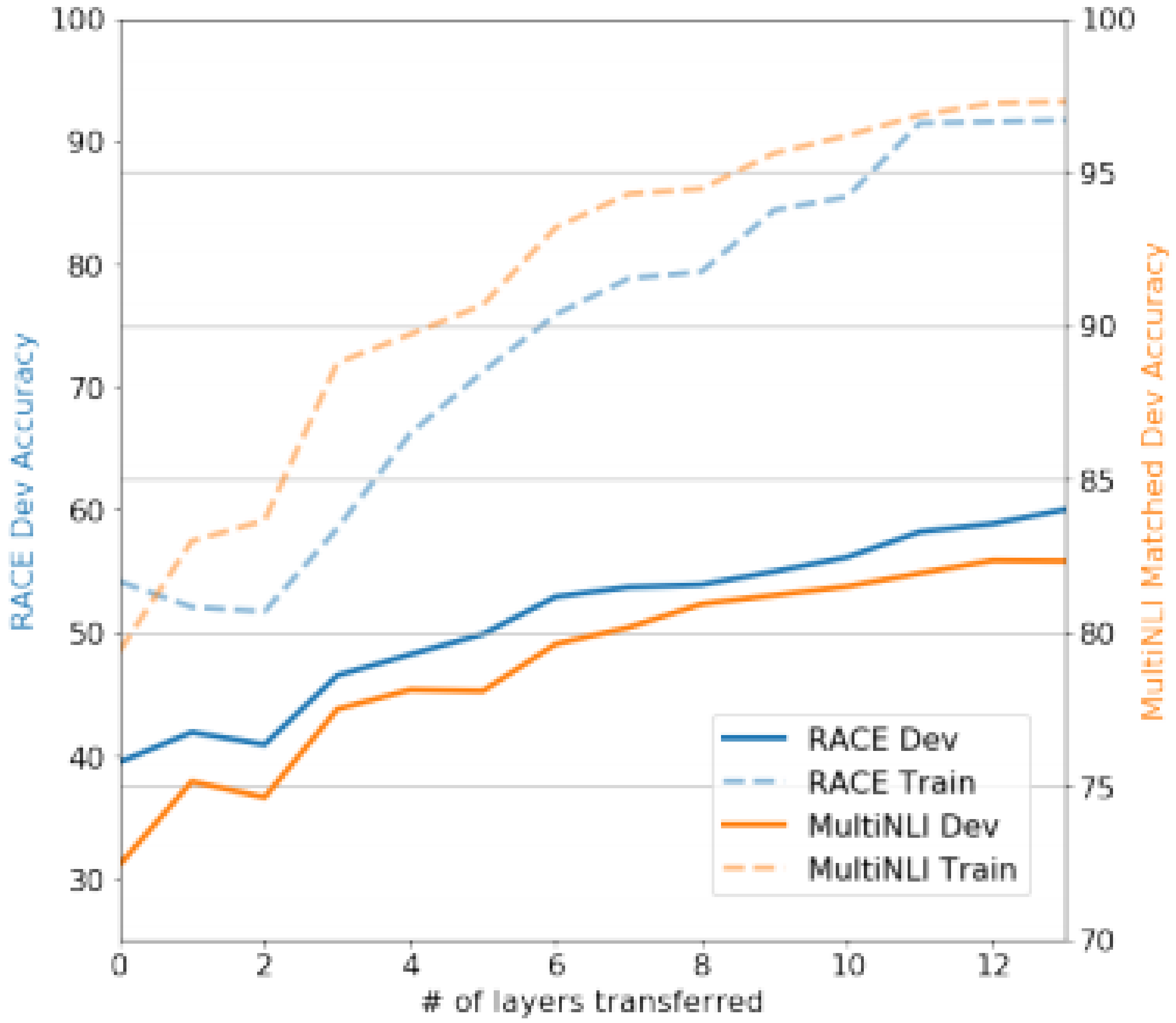
Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	<u>89.3</u>	-	-	-
CAFE [58] (5x)	80.2	79.0	<u>89.3</u>	-	-	-
Stochastic Answer Network [35] (3x)	<u>80.6</u>	<u>80.1</u>	-	-	-	-
CAFE [58]	78.7	77.9	88.5	<u>83.3</u>		
GenSen [64]	71.4	71.3	-	-	<u>82.3</u>	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

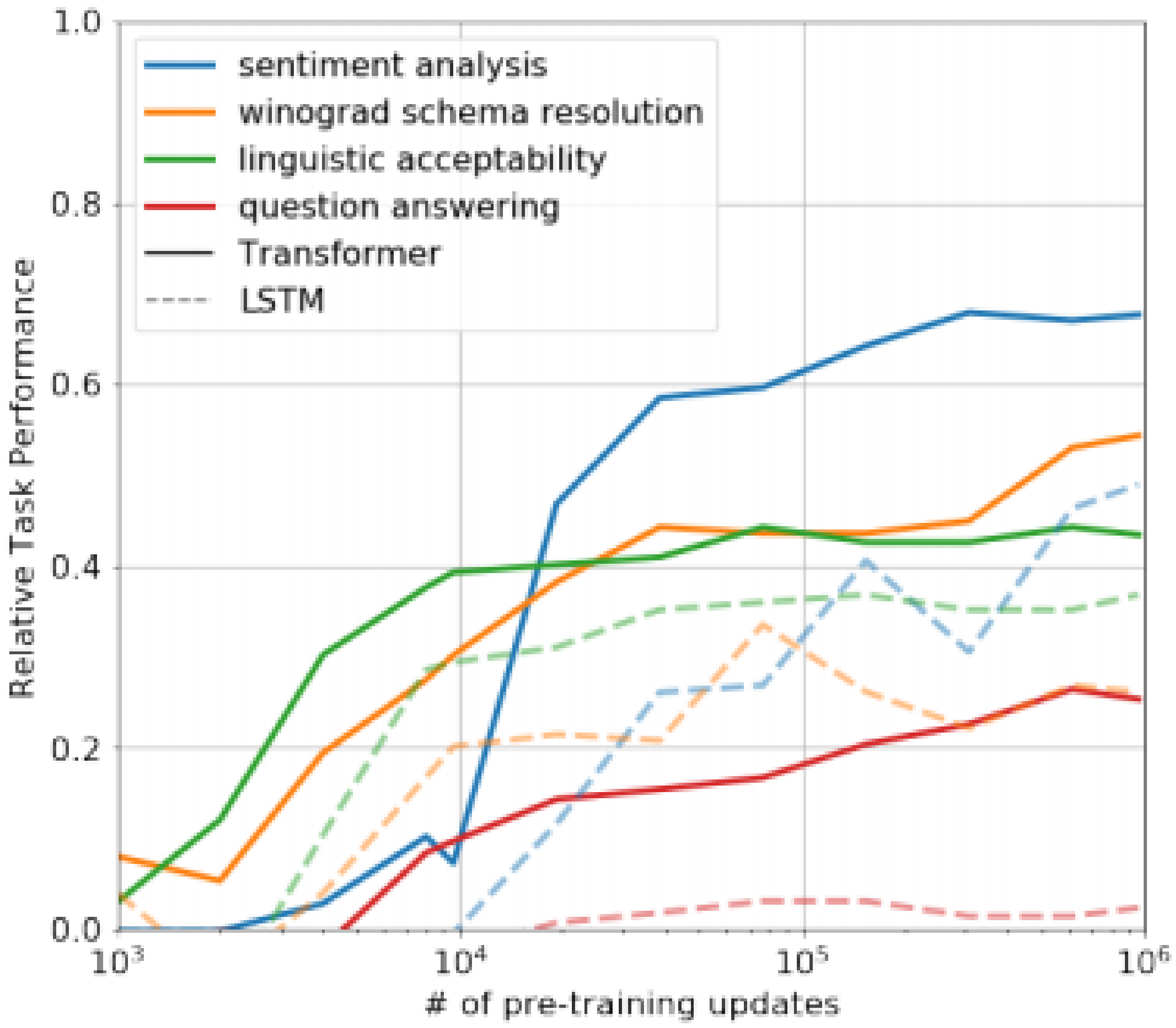
Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	<u>77.6</u>	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	<u>60.2</u>	<u>50.3</u>	<u>53.3</u>
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

Results

Method	Classification		Semantic Similarity			GLUE
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	
Sparse byte mLSTM [16]	-	**93.2**	-	-	-	-
TF-KLD [23]	-	-	**86.0**	-	-	-
ECNU (mixed ensemble) [60]	-	-	-	<u>81.0</u>	-	-
Single-task BiLSTM + ELMo + Attn [64]	<u>35.0</u>	90.2	80.2	55.5	<u>66.1</u>	64.8
Multi-task BiLSTM + ELMo + Attn [64]	18.9	91.6	83.5	72.8	63.3	<u>68.9</u>
Finetuned Transformer LM (ours)	**45.4**	91.3	82.3	**82.0**	**70.3**	**72.8**

Analysis


RACE Dev Accuracy
MultiNLI Matched Dev Accuracy
# of layers transferred
RACE Dev
RACE Train
MultiNLI Dev
MultiNLI Train


Relative Task Performance
# of pre-training updates
sentiment analysis
winograd schema resolution
linguistic acceptability
question answering
Transformer
LSTM

Conclusion

결론

- 하나의 task에만 국한되지 않는 강력한 자연어 이해 모델 제안

- 사전학습된 모델은 상당한 지식과 long-range dependencies를 처리할 수 있는 능력을 갖춤으로써 각각의 task를 성공적으로 수행함

- question answering, semantic similarity assessment, entailment determination, text classification 등 NLP task에 대한 12개의 dataset 중 9개에서 새로운 sota를 달성

- unsupervised (pre-)training을 통한 상당한 성능 향상을 실현

- NLP 이외의 다른 영역에서도 unsupervised learning에 대한 새로운 연구를 촉발

```python
class DecoderLayer(nn.Module):
    def __init__(self, d_model, n_heads, d_ff, resid_drop):
        super().__init__()

        self.mha = MHA(d_model, n_heads)
        self.dropout1 = nn.Dropout(resid_drop)
        self.layernorm1 = nn.LayerNorm(d_model, eps=1e-5)

        self.ffn = FFN(d_model, d_ff)
        self.dropout2 = nn.Dropout(resid_drop)
        self.layernorm2 = nn.LayerNorm(d_model, eps=1e-5)

    def forward(self, x, attn_mask):
        # Masked-MHA layer (with residual shortcut connection)
        residual = self.mha(x, x, x, attn_mask)
        residual = self.dropout1(residual)
        x = self.layernorm1(x + residual)

        # FFN layer (with residual shortcut connection)
        residual = self.ffn(x)
        residual = self.dropout2(residual)
        output = self.layernorm2(x + residual)

        return output
```

```python
class GPTLMHead(nn.Module):
    def __init__(self, gpt):
        super().__init__()
        vocab_size, d_model = gpt.decoder.embedding.weight.size()

        self.gpt = gpt
        self.linear = nn.Linear(d_model, vocab_size, bias = False)
        self.linear.weight = gpt.decoder.embedding.weight

    def forward(self, x):
        x = self.gpt(x)

        lm_logits = self.linear(x)

        return lm_logits
```

```python
class GPTClsHead(nn.Module):
    def __init__(self, gpt, n_class, cls_token_id, cls_drop=0.1):
        super().__init__()
        vocab_size, d_model = gpt.decoder.embedding.weight.size()
        self.cls_token_id = cls_token_id

        self.gpt = gpt

        # LM
        self.linear1 = nn.Linear(d_model, vocab_size, bias=False)
        self.linear1.weight = gpt.decoder.embedding.weight
        # Cls
        self.linear2 = nn.Linear(d_model, n_class)
        self.dropout = nn.Dropout(cls_drop)

        nn.init.normal_(self.linear2.weight, std=0.02)
        nn.init.normal_(self.linear2.bias, 0)

    def forward(self, x):
        outputs = self.gpt(x)

        lm_logits = self.linear1(outputs)

        outputs = outputs[x.eq(self.cls_token_id)]
        cls_logits = self.linear2(self.dropout(outputs))

        return lm_logits, cls_logits
```

Q&A

질의 응답