Improving Language Understanding by Generative Pre-Training

발표자: 강주영

Introduction

- Unlabeled text가 많으나, 이로부터 단어 수준 이상의 정보를 얻기 어려움.
  1. 어떤 최적화 수식이 전이(transfer)에 유용한 텍스트 표현(representation)을 배우는데 가장 효과적인지 불분명
  2. Target tasks에 있어 가장 효과적인 representation에 대한 일치된 합의의 부재
     → semi-supervised learning 접근이 어려움

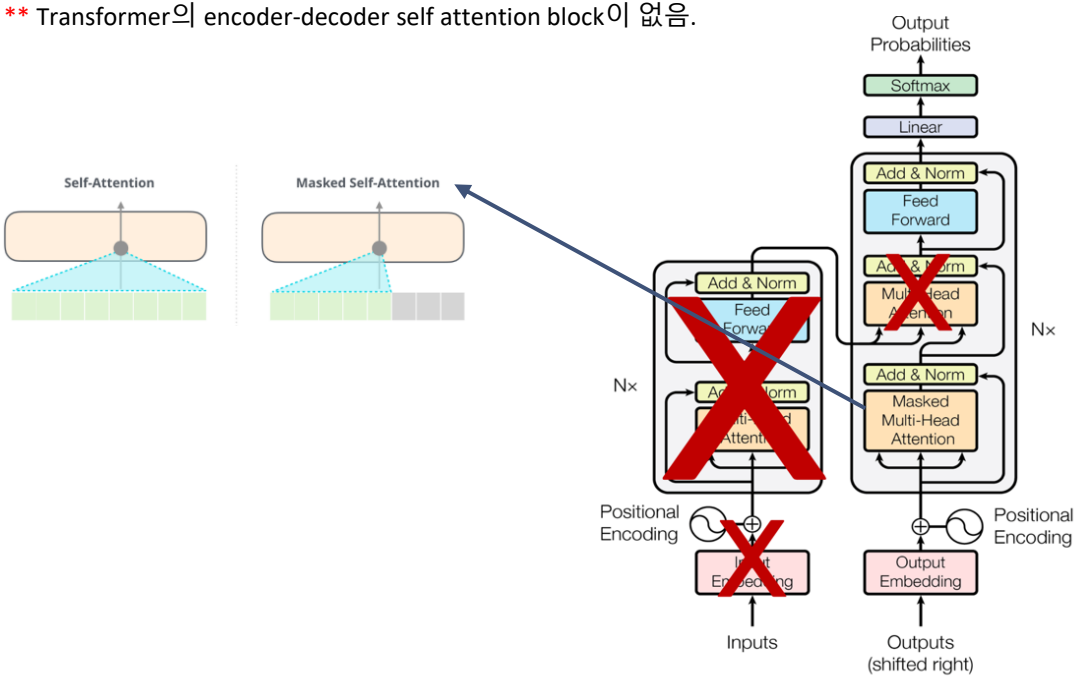
→ Combination of unsupervised pre-training and supervised fine-tunning

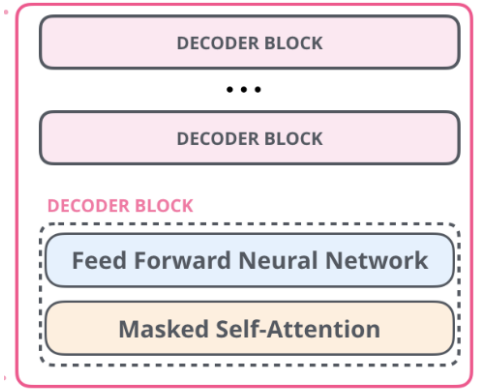
Related Work

- **Semi-supervised learning for NLP**
  - Unlabeled corpus로 word-level embedding학습 → supervised learning의 feature로 사용
  - GPT에서는 더 높은 수준인 phrase-level or sentence-level embedding을 학습하는 것이 목표
- **Unsupervised pre-training**
  - Supervised learning 목적함수를 수정없이 사용할 수 있는 좋은 초기화 지점을 찾는 것이 목표
  - transformer 모델을 통해 더 긴 언어구조를 포착 가능
  - 전이학습 동안, 모델 구조에 있어 최소한의 변화만 필요함.
- **Auxiliary training objectives**
  - 비지도 학습 수식에 보조수식을 넣는 것은 semi-supervised learning의 대안
  - 이 논문에서도 보조수식을 사용하지만, 이미 비지도 학습에서 타겟 작업에 대한 언어정보들을 학습했다는 사실로 보여줌

Model Architecture

** Transformer의 encoder-decoder self attention block이 없음.



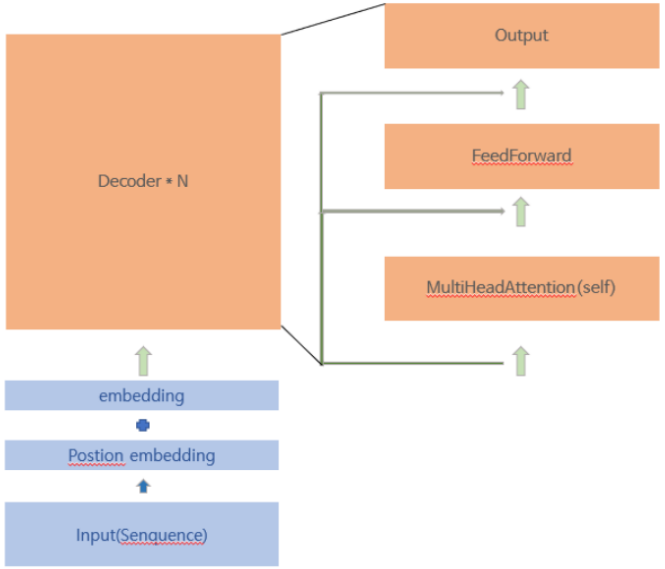


Framework

- **2 states of training procedure**
  1. **Unsupervised pre-training**

$$L_1(\mathcal{U}) = \sum \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)$$
$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

$\mathcal{U}$: Unsupervised corpus of tokens
U : Context vector of tokens
We : Token embedding
Wp : Position embedding



Framework

- **2 states of training procedure**

2. Supervised fine-tunning

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

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

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

1) Improving generalization of the supervised model
2) Accelerating convergence(학습속도 향상)

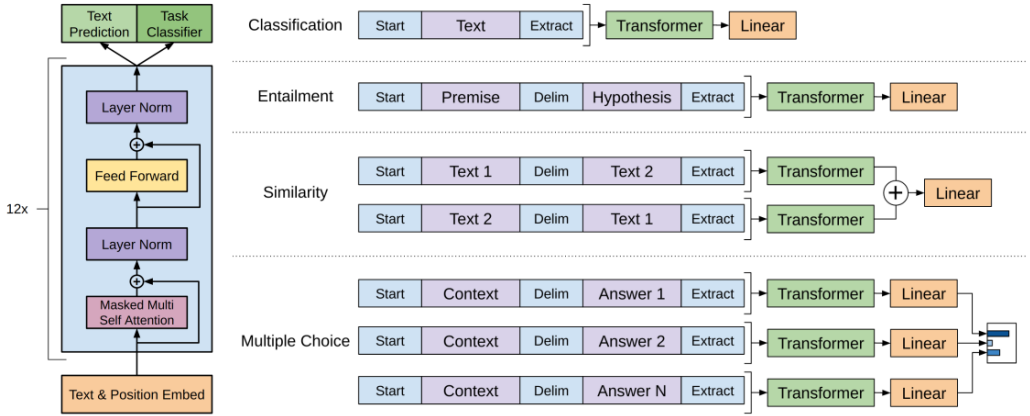
C : Supervised corpus of tokens
h_l^m : The final transformer's block
W_y : Parameters

Task-specific input transformations

- **4 different tasks for fine-tunning**

  - Pre-trained model모델의 형태에 맞추어, ordered sequence로 변환

1. Classification
2. Textual entailment
3. Similarity
4. Question Answering
   & Commonsense Reasoning



Experiments

Unsupervised pre-training

- Transformer에서 decoder만 사용
- 12-layer decoder
- Adam optimizer
- Batch size = 64, epoch = 100
- Gaussian Error Linear Unit activation function
- Encoding: BPE(Byte Pair Encoding)
  - ➢ 단어를 character 단위로 나누어 subword를 생성하는 방법
  - ➢ Character 단위로 나누기 때문에, out of Vocabulary 문제를 극복

Supervised fine-tunning

- Reuse hyperparameter from unsupervised pre-training
- Add dropout (=0.1)
- Batch size = 32, epochs = 3

BPE(Byte Pair Encoding)

sentence : aaabdaaabac

→ vocab : (a, b, c, d) - 초기에는 character 단위로 잘라줌
→ tokenize를 단위로 잘라줌 : (a a a b d a a a b a c)
→ 2글자 단위로 pair를 만든다 : (aa, aa, ab, bd, da, aa, aa, ab, ba, ac)
→ 제일 많이 등장한 token : aa(4번)
→ Iteration이 한번 끝나면서 vocab에 aa가 추가됨 : (a, b, c, d, aa)
→ 추가된 vocab으로 다시 tokenizing : (aa a b d aa a b a c)
→ 2번의 Iteration을 돌면 : (aa ab d aa ab a c) → vocab은 (a, b, c, d, aa, ab)
→ 3번의 Iteration을 돌면 : (aaab d aaab a c) → vocab은 (a, b, c, d, aa, ab, aaab)

https://zsunn.tistory.com/entry/AI-BPE-Byte-Pair-Encoding-%EC%84%A4%EB%AA%85-%EB%B0%8F-%EC%98%88%EC%A0%9C

Results (1)

Natural language Inference (= Textual Entailment)

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	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

Question Answering & Commonsense Reasoning

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

← RTE : the smaller datasets (2490 examples 평가)

- 큰 데이터 셋과 달리, 작은 데이터 셋에서 NLI 성능이 낮게 나온 이유는 규명되지 않음.

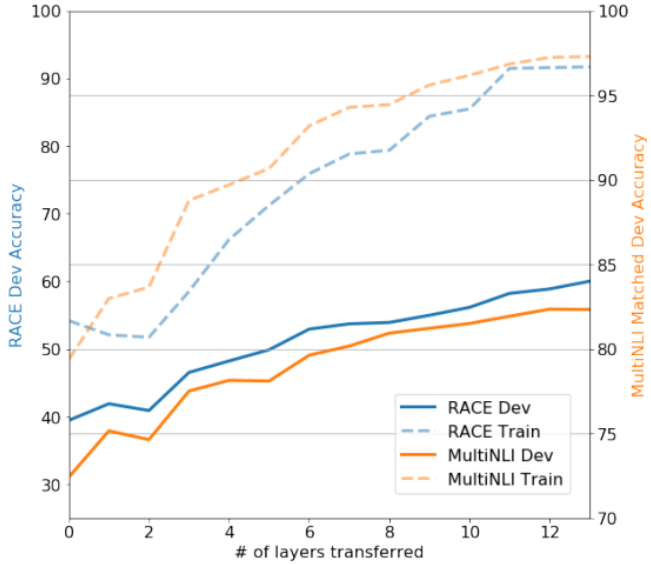
Results (2)

Semantic Similarity & Classification

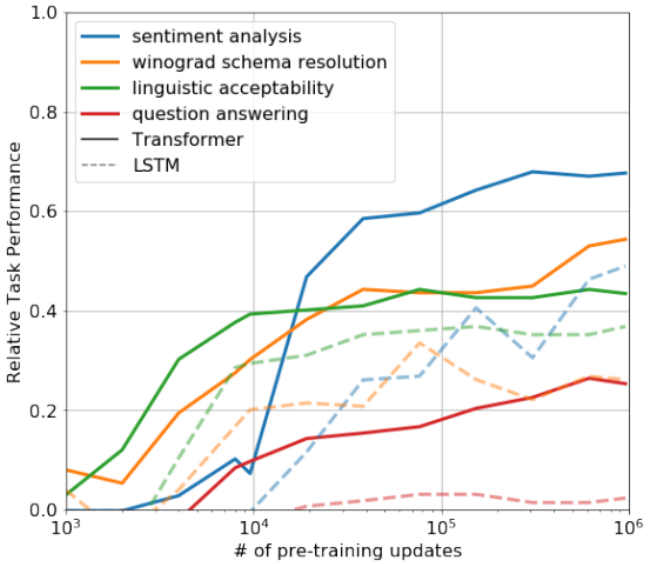
Method	Classification		Semantic Similarity			GLUE
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	
Sparse byte mLSTM [16]	-	**93.2**	-	-	-	-
TF-KLD [23]	-	-	**86.0**	-	-	-
ECNU (mixed ensemble) [60]	-	-	-	81.0	-	-
Single-task BiLSTM + ELMo + Attn [64]	35.0	90.2	80.2	55.5	66.1	64.8
Multi-task BiLSTM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

GPT는 전반적으로, 다른 크기의 데이터 셋 (5,700개 ~ 550,000개)에 대하여 좋은 성능을 보임.

Analysis (1)



Unsupervised에서 supervised로 가는 layer수가 증가할수록, 성능이 더 증가함.
→ 각 layer가 target task를 해결하는데 유용한 정보를 포함하고 있음을 의미함.



Pre-training update가 증가할 수록, 각 테스크에 대한 zero-shot performance가 증가함.

Analysis (2)

Auxiliary object에 대한 효과

Method	Avg. Score	작은 데이터 셋 CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	큰 데이터 셋 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

← LSTM은 여기서만 성능이 조금 좋았음.

- **큰 데이터 셋에서는 auxiliary가 성능 향상에 도움을 주었으나, 작은 데이터 셋에서는 그렇지 않음.**

THANKS!