ALBERT: A Lite BERT

2022.02.02(Wed)

Speaker | **Suyeon Nam**

Index

Introduction

일반적으로 모델이 클수록 성능이 좋아지는 Pre-trained Language Model

좋은 환경에서 큰 데이터셋으로 pre-training, 이후 작은 데이터셋으로 fine-tuning

Model	Score
BERT-Tiny	64.2
BERT-Mini	65.8
BERT-Small	71.2
BERT-Medium	73.5

의문: 모델을 무조건 키우는 것이 최선의 방법인가?

모델의 크기를 무작정 키우기엔 메모리의 한계가 있고, 학습에 굉장히 많은 시간이 소요됨

+ Model Degradation 문제: BERT large보다 xlarge의 성능이 오히려 더 떨어짐

모델의 크기를 무작정 키우는 것은 정답이 아니다.

모델의 크기를 결정하는 파라미터 수를 BERT 대비 대폭 줄인 ALBERT의 등장

ALBERT

What is ALBERT?

ALBERT = BERT의 경량화 버전

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	60M	24	2048	128	True
	xxlarge	235M	12	4096	128	True

Solutions: 2가지 **Parameter reduction techniques + Sentence-order prediction**

ALBERT

Parameter Reduction Techniques 1) Factorized Embedding Parameterization

BERT

임베딩 크기 $E \equiv$ 히든 레이어 크기 H

ALBERT

임베딩 크기 $E \ll$ 히든 레이어 크기 H

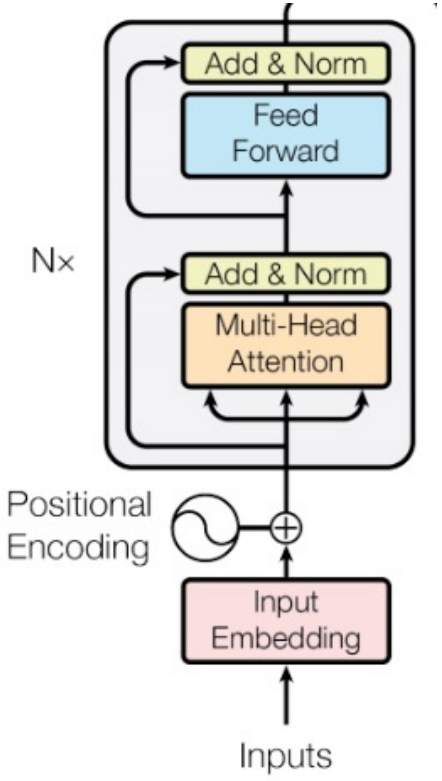
E(WordPiece Embedding)은 문맥과 독립적인 표현

H(Hidden-layer Embedding)은 문장의 문맥 정보를 학습하여 갖고 있음

👉🏻 **중요한 것은 H! E는 H보다 작아도 괜찮다고 판단**

ALBERT

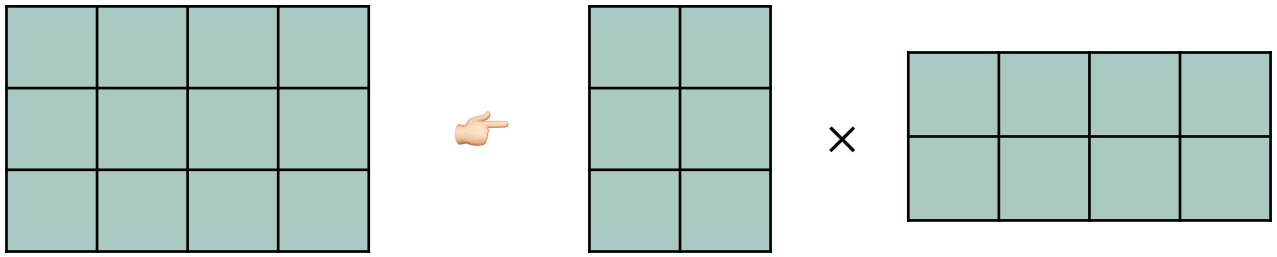
Parameter Reduction Techniques 1) Factorized Embedding Parameterization



E와 H가 달라지면 발생하는 문제:

첫 번째 Transformer layer 입력 시 차원이 맞지 않음

Solution:



임베딩 파라미터의 개수: $O(V \times H)$ ➡ $O(V \times E + E \times H)$

ALBERT

Parameter Reduction Techniques 1) Factorized Embedding Parameterization

성능의 차이는?

Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

Table 3: The effect of vocabulary embedding size on the performance of ALBERT-base.

ALBERT

Parameter Reduction Techniques 2) Cross-layer Parameter Sharing

효율성을 위해 **Transformer 내 각 layer들의 파라미터를 공유**하는 것.

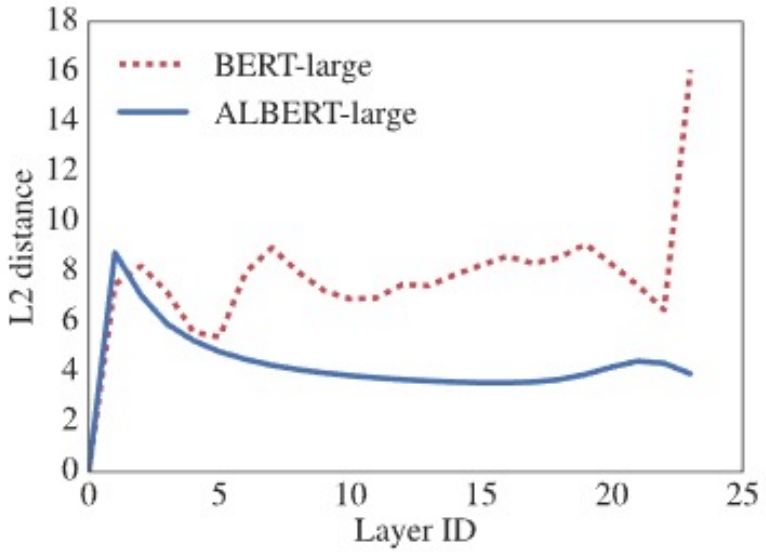
Universal Transformer에서 처음 제안된 아이디어

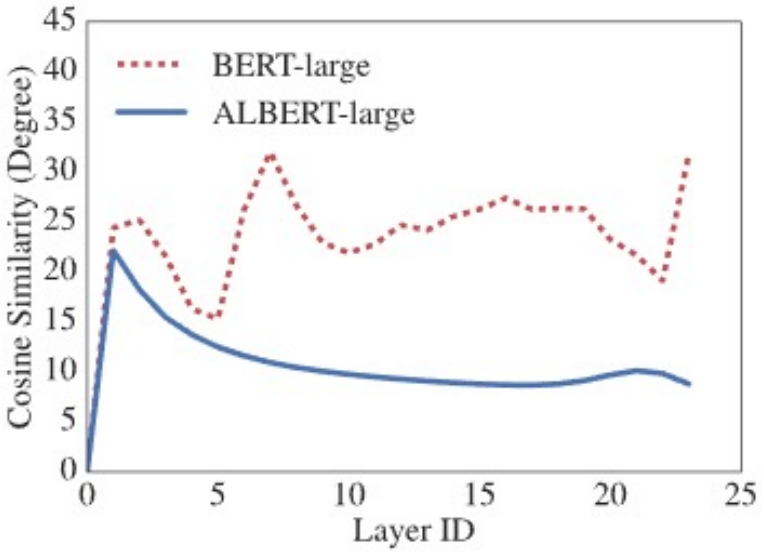
Self-Attention or Feed-Forward Network 파라미터만 공유하거나, 모두 공유하거나(Default)…

ALBERT

Parameter Reduction Techniques 2) Cross-layer Parameter Sharing

파라미터 공유의 의미와 효과





ALBERT

Parameter Reduction Techniques 2) Cross-layer Parameter Sharing

성능의 차이는?

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

Table 4: The effect of cross-layer parameter-sharing strategies, ALBERT-base configuration.

ALBERT

Sentence-order Prediction

BERT

사전학습 전략으로 **NSP** 사용

ALBERT

NSP 대신 **SOP** 사용

💡 NSP는 연관 관계 학습에 효과가 없다.

NSP는 두 번째 문장이 첫 번째 문장의 다음 문장인지 맞추는 문제. 학습 데이터 실제로 연결된 문장(P)이거나 임의로 뽑힌 서로 관련없는 문장(N)이다.
이때 서로 관련없는 문장(N)일 경우 첫 문장과 다른 Topic일 확률이 매우 높은데, 이것 때문에 **두 문장의 연관 관계를 학습한다기 보다는 두 문장이 같은 Topic을 갖는지를 학습하게 된다.**

💡 SOP는 NSP보다 연관 관계 학습 효과가 좋다.

NSP와 다르게 SOP의 학습 데이터는 실제로 연결되어 있는 두 문장(P)과 두 문장의 순서를 바꾼 것(N)으로 구성하고, 문장의 순서가 옳은지 판단하게 한다.
따라서 SOP로 학습 시 두 문장의 연관 관계를 보다 잘 학습할 거라고 기대할 수 있다.

ALBERT

Sentence-order Prediction

		Intrinsic Tasks			Downstream Tasks					
	SP tasks	MLM	NSP	SOP	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
Only MLM	None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
MLM + NSP	NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
MLM + SOP	SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

- MLM + **NSP**의 경우 NSP Task는 잘 하지만(90.5) SOP Task에서 처참한 성능(52.0)
- MLM + **SOP**의 경우 NSP Task에서도 괜찮은 성능을 보이며(78.9) SOP Task도 잘 수행함(86.5)

ALBERT vs. BERT

- Bookcorpus, Wikipedia로 학습
- Batch size = 4,096
- Vocab size = 30,000
- sentencepiece로 tokenize
- LAMB optimizer, lr = 0.00176
- 125,000 steps

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
	xlarge	1270M	24	2048	2048	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	59M	24	2048	128	True
	xxlarge	233M	12	4096	128	True

Table 2: The configurations of the main BERT and ALBERT models analyzed in this paper.

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.5/83.3	80.3/77.3	84.1	91.7	68.3	82.1	17.7x
	large	334M	92.4/85.8	83.9/80.8	85.8	92.2	73.8	85.1	3.8x
	xlarge	1270M	86.3/77.9	73.8/70.5	80.5	87.8	39.7	76.7	1.0
ALBERT	base	12M	89.3/82.1	79.1/76.1	81.9	89.4	63.5	80.1	21.1x
	large	18M	90.9/84.1	82.1/79.0	83.8	90.6	68.4	82.4	6.5x
	xlarge	59M	93.0/86.5	85.9/83.1	85.4	91.9	73.9	85.5	2.4x
	xxlarge	233M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	1.2x

ALBERT vs. BERT

Memory Limitation

같은 layer 수, hidden size를 가질 때
ALBERT가 BERT보다 모델의 크기가 훨씬 작음

Training Time

같은 layer 수, hidden size를 가질 때
ALBERT가 BERT보다 학습 속도가 훨씬 빠름

Model Degradation

ALBERT는 xlarge, BERT xlarge보다 더 큰
xxlarge 모델도 Model degradation
문제가 발생하지 않음

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
	xlarge	1270M	24	2048	2048	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	59M	24	2048	128	True
	xxlarge	233M	12	4096	128	True

Table 2: The configurations of the main BERT and ALBERT models analyzed in this paper.

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.5/83.3	80.3/77.3	84.1	91.7	68.3	82.1	17.7x
	large	334M	92.4/85.8	83.9/80.8	85.8	92.2	73.8	85.1	3.8x
	xlarge	1270M	86.3/77.9	73.8/70.5	80.5	87.8	39.7	76.7	1.0
ALBERT	base	12M	89.3/82.1	79.1/76.1	81.9	89.4	63.5	80.1	21.1x
	large	18M	90.9/84.1	82.1/79.0	83.8	90.6	68.4	82.4	6.5x
	xlarge	59M	93.0/86.5	85.9/83.1	85.4	91.9	73.9	85.5	2.4x
	xxlarge	233M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	1.2x

Conclusion

ALBERT는 BERT보다 파라미터 수를 훨씬 줄여 Memory Limitation 문제를 해결했으며 학습 속도도 빠름

BERT와 비슷한 성능을 가지며(ALBERT xxlarge는 BERT xlarge보다 더 좋은 성능을 보임),

Model Degradation이 BERT xlarge보다도 큰 xxlarge 모델에서도 발생하지 않음