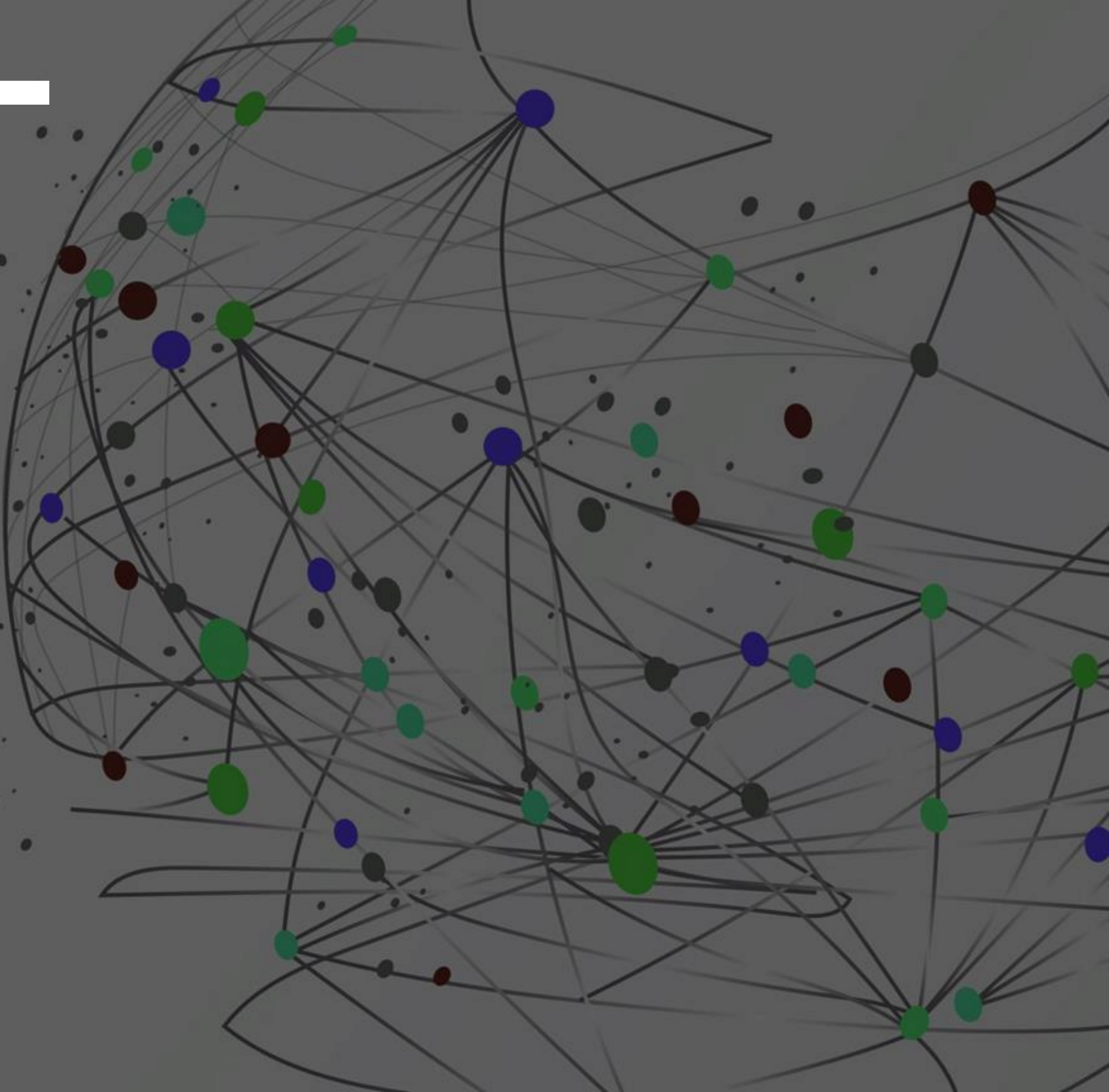
ALBERT

2021.07.28

주세준



ALBERT:
A Lite BERT for self-supervised learning of language representations

2020 ICLR에

Conference에 publish

Zhenzhong Lan

모델이 계속해서 커지는 과정에서
발생하는 문제들에 대한 해결책 제시

2. Training Time

모델이 커질수록 훈련 시키는데 많은 시간이 든다

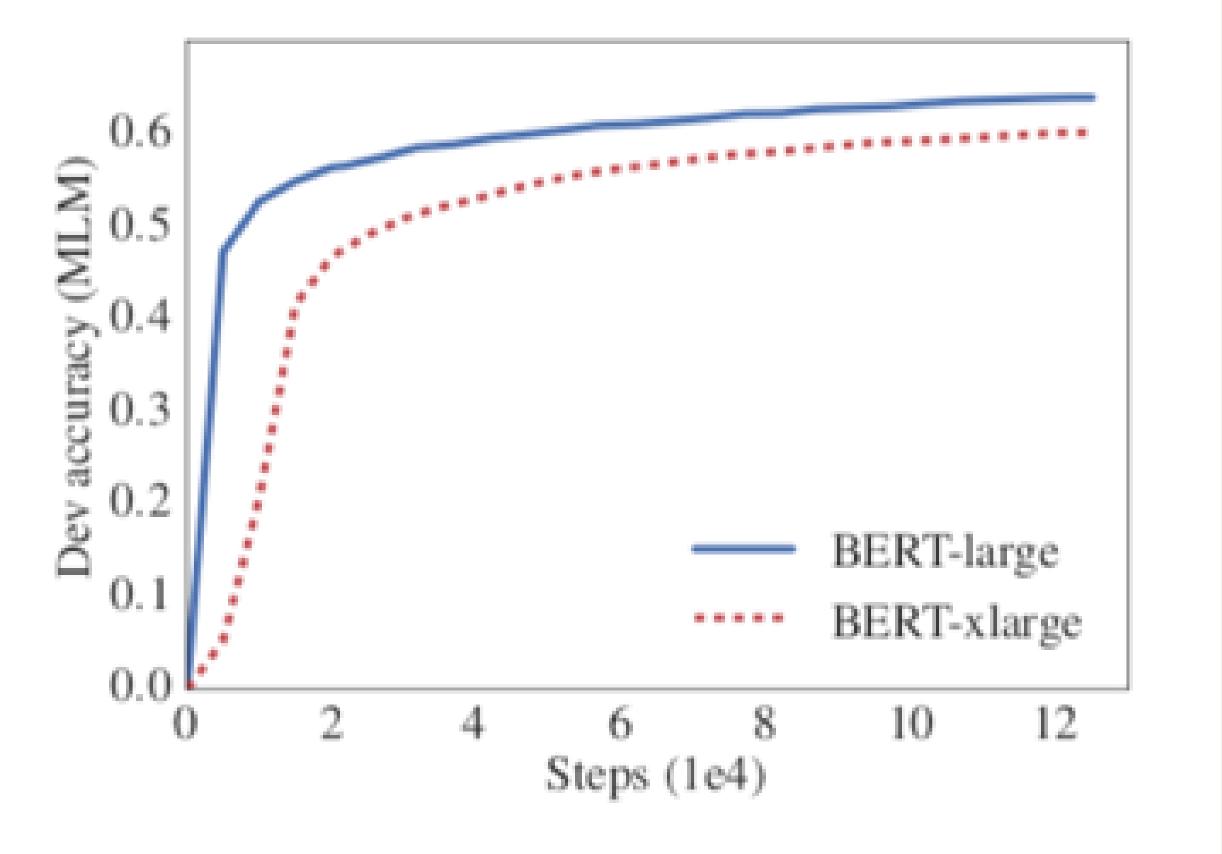
(구글 같은 대기업은 돈으로 해결.
그래도 BERT-base는 16개의 tpu로 4일

Large의 경우 64개의 tpu로 4일
현실적으로 난감하다)

3. Model Degradation

모델의 크기와 성능이 비례하지 않는다

오히려 Bert-xlarge의 경우 large보다 MLM 정확도가 안나온다



1. Factorized embedding parameterization

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.

Input Token Embedding Size(E)를 Hidden Size(H)와 같게 설정하던 BERT와 달리 더 작게 설정

E에서는 각 Token의 정보를 담는 vector 생성

H를 통과한 layer의 output은 주변의 관계도 포함한 contextualized

차원이 안 맞는 문제는 VxH matrix를 VxE, ExH 두개의 matrix로 연달아 곱해 해결 (factorized 인 이유)

-> parameter 수 감소

2. Cross-layer parameter sharing

Dehghani et al., (2018)의 Universal Transformers 에서도 제시된 개념 (Recursive transformer)
쌓는 대신 돌리고 돌리고

Transformer layer 간 같은 Parameter를 공유하며 사용하는 것

Parameter를 공유해도 성능이 크게 떨어지지 않음. (FFN를 공유 시에는 큰 차이가 있음)

All-share E=768일 때 훨씬 성능이 떨어진다(FFN 때문일 것) 하지만 부분적으로 공유하는 것은 parameter 수가 극적으로 늘어나게 되어 All-shared 전략 선택

APPENDIX A DETAILED SCHEMA OF THE UNIVERSAL TRANSFORMER

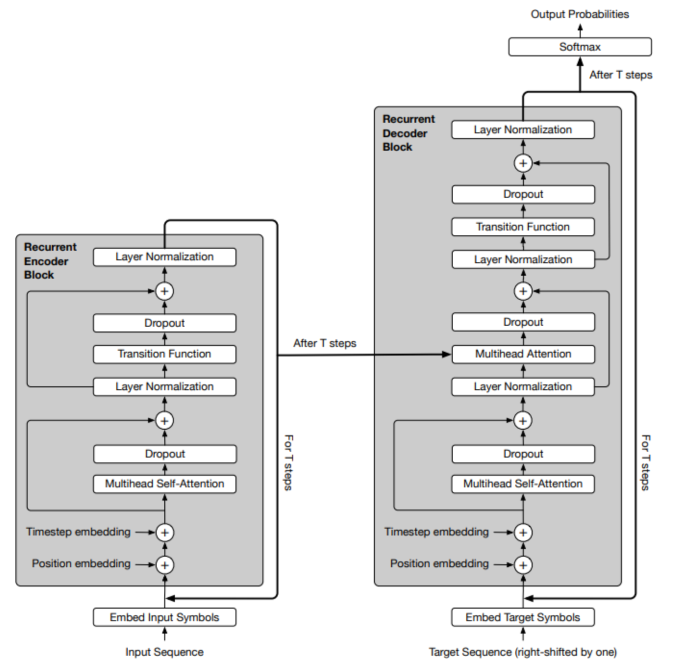


Figure 4: The Universal Transformer with position and step embeddings as well as dropout and layer normalization.

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.

More specifically, we use the scaled dot-product attention which combines queries Q, keys K and values V as follows

$$\textsc{Attention}(Q,K,V) = \textsc{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V, \tag{1}$$

where d is the number of columns of Q, K and V. We use the multi-head version with k heads, as introduced in (Vaswani et al., 2017),

$$\textsc{MultiHeadSelfAttention}(H^t) = \textsc{Concat}(\mathrm{head_1},\ldots,\mathrm{head_k})W^O \tag{2}$$

$$\text{where } \mathrm{head_i} = \textsc{Attention}(H^t W_i^Q, H^t W_i^K, H^t W_i^V) \tag{3}$$

and we map the state H^t to queries, keys and values with affine projections using learned parameter matrices $W^Q \in \mathbb{R}^{d \times d/k}$, $W^K \in \mathbb{R}^{d \times d/k}$, $W^V \in \mathbb{R}^{d \times d/k}$ and $W^O \in \mathbb{R}^{d \times d}$.

At step t, the UT then computes revised representations $H^t \in \mathbb{R}^{m \times d}$ for all m input positions as follows

$$H^t = \textsc{LayerNorm}(A^t + \textsc{Transition}(A^t)) \tag{4}$$

$$\text{where } A^t = \textsc{LayerNorm}((H^{t-1} + P^t) + \textsc{MultiHeadSelfAttention}(H^{t-1} + P^t)), \tag{5}$$

$P^t \in \mathbb{R}^{m \times d}$ above are fixed, constant, two-dimensional (position, time) *coordinate embeddings*, obtained by computing the sinusoidal position embedding vectors as defined in (Vaswani et al., 2017) for the positions $1 \leq i \leq m$ and the time-step $1 \leq t \leq T$ separately for each vector-dimension $1 \leq j \leq d$, and summing:

$$P^t_{i,2j} = \sin(i/10000^{2j/d}) + \sin(t/10000^{2j/d}) \tag{6}$$

$$P^t_{i,2j+1} = \cos(i/10000^{2j/d}) + \cos(t/10000^{2j/d}). \tag{7}$$

3. Sentence order prediction

두 개의 text segment의 순서를 예측하는 task를 pre-train에 사용

BERT NSP보다 어려운 task
특정 downstream task 에서 더 효과적이다

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

Better than BERT

BERT에 비해 parameter는 작게
성능은 더 좋게 나온다

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

Table 2: Dev set results for models pretrained over BOOKCORPUS and Wikipedia for 125k steps. Here and everywhere else, the Avg column is computed by averaging the scores of the downstream tasks to its left (the two numbers of F1 and EM for each SQuAD are first averaged).

4. NLU task SOTA

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4	-	-
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7	-	-
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**	-	-
Ensembles on test (from leaderboard as of Sept. 16, 2019)										
ALICE	88.2	95.7	**90.7**	83.5	95.2	92.6	**69.2**	91.1	80.8	87.0
MT-DNN	87.9	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5
Adv-RoBERTa	91.1	98.8	90.3	88.7	96.8	93.1	68.0	92.4	89.0	88.8
ALBERT	**91.3**	**99.2**	90.5	**89.2**	**97.1**	**93.4**	69.1	**92.5**	**91.8**	**89.4**

Table 9: State-of-the-art results on the GLUE benchmark. For single-task single-model results, we report ALBERT at 1M steps (comparable to RoBERTa) and at 1.5M steps. The ALBERT ensemble uses models trained with 1M, 1.5M, and other numbers of steps.

Models	SQuAD1.1 dev	SQuAD2.0 dev	SQuAD2.0 test	RACE test (Middle/High)
Single model (from leaderboard as of Sept. 23, 2019)				
BERT-large	90.9/84.1	81.8/79.0	89.1/86.3	72.0 (76.6/70.1)
XLNet	94.5/89.0	88.8/86.1	89.1/86.3	81.8 (85.5/80.2)
RoBERTa	94.6/88.9	89.4/86.5	89.8/86.8	83.2 (86.5/81.3)
UPM	-	-	89.9/87.2	-
XLNet + SG-Net Verifier++	-	-	90.1/87.2	-
ALBERT (1M)	94.8/89.2	89.9/87.2	-	86.0 (88.2/85.1)
ALBERT (1.5M)	**94.8/89.3**	**90.2/87.4**	**90.9/88.1**	**86.5 (89.0/85.5)**
Ensembles (from leaderboard as of Sept. 23, 2019)				
BERT-large	92.2/86.2	-	-	-
XLNet + SG-Net Verifier	-	-	90.7/88.2	-
UPM	-	-	90.7/88.2	
XLNet + DAAF + Verifier	-	-	90.9/88.6	-
DCMN+	-	-	-	84.1 (88.5/82.3)
ALBERT	**95.5/90.1**	**91.4/88.9**	**92.2/89.7**	**89.4 (91.2/88.6)**

Table 10: State-of-the-art results on the SQuAD and RACE benchmarks.

Discussion

Parameter number smaller than BERT-large nut computationally more expensive due to larger structure (T5)

모델의 성능은 물론 model-size를 상당하게 줄여서 의미가 있다