3.3.1. ALBERT : A Lite BERT

논문 : https://arxiv.org/abs/1909.11942

ICLR 2020 (2020. 01)에 나온 신작 (아카이브에는 2019.09에 등장)

논문의 목적

- 모델의 파라미터 수 감소 (그와 동시에 훈련 속도도 감소)
- Next Sentence Prediction 대신 사용할 training objective 선정

BERT의 문제점

1. 모델이 너무 크다 => 학습이 오래걸림
2. 워드 임베딩과 히든 사이즈가 동일하다 => 둘중 하나를 올릴 수 없음 (모델이 비대해짐).
3. 자연어 처리 모델의 히든 사이즈를 키워도 성능이 하락된다 (그림 1)

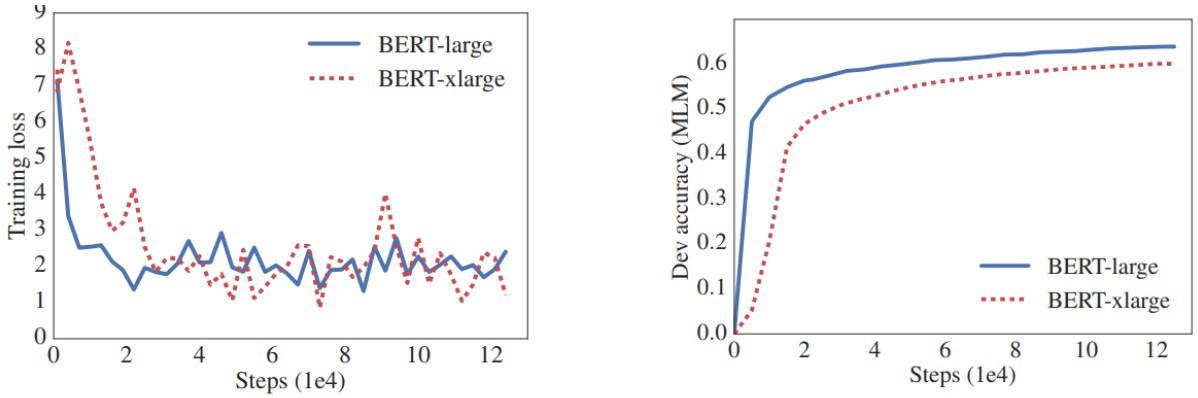

Figure 1: Training loss (left) and dev masked LM accuracy (right) of BERT-large and BERT-xlarge (2x larger than BERT-large in terms of hidden size). The larger model has lower masked LM accuracy while showing no obvious sign of over-fitting.

논문의 제안 기법

Factorized embedding parameterization

문제

- BERT, XLNet, RoBERTa, 등의 NLP 모델들은 Word Embedding size E와 Hidden Layer Size H가 동일함 => E = H
- 모델링 관점
  - Word Embedding은 Context-independent representation을 배우기 위함, Hidden Layer Embedding은 context-dependent representation을 배우기 위함
  - BERT같은 representation의 힘은 Context-dependent representation을 배우는게 중요함 => H의 크기를 늘릴 필요가 있음
  - E는 context-independent이므로 E < H 여도 모델의 성능이 떨어지지 않을 것으로 저자가 판단함
- 실제적인 관점
  - NLP는 대개 매우 큰 사이즈인 Vocabulary V를 요구
  - E=H이면, H가 증가했을 때 임베딩 행렬의 크기도 증가함 V × E
  - 때문에 모델은 100 Million ~ 2 Billion 단위의 파라미터 개수를 가지게되고 학습을 해도 가끔 업데이트가 됨

제안

- ALBERT는 Word Embedding size E와 Hidden Layer size H를 분리하기 위해 Input Layer뒤에 projection layer를 추가함
- Projection Layer : E가 입력됐을 때 H 크기로 바꿔주는 Weight(Bias도 추가될 수 있음)를 가진 Layer
- 이를 통해 기존의 임베딩 파라미터 O(V × H)에서 O(V × E + E × H)로 감소함

Cross-layer parameter sharing

문제

- 각 레이어마다 존재하는 Attention, Feed Forward Network(이하 FFN)의 파라미터의 개수가 너무 많음
- 여러 논문들이 계층 간의 FFN 파라미터 공유나 Attention 파라미터를 공유하는 것으로 모델 크기를 줄이고자함

분석

- Universal Transformer가 기존의 Transformer보다 더 좋은 성능을 보임 (Universal Transformer 논문 : https://arxiv.org/abs/1807.03819)
- Deep Equilibrium Models (DQE) 논문에서는 입력과 출력이 동일하게 유지되는 평형점에 도달하는 것을 보임 (DQE)
    - 실제로 실험해보니까 수렴하기보다는 변화가 있었음!
    - 심지어 기존 BERT보다 층간의 전이가 훨씬 더 부드럽다는 것을 관측함! (그림 2)
    - 물론 BERT에 비해 두가지 지표에 대한 성능 저하는 있음
    - 하지만 파라미터 개수가 5배 가까이 차이남

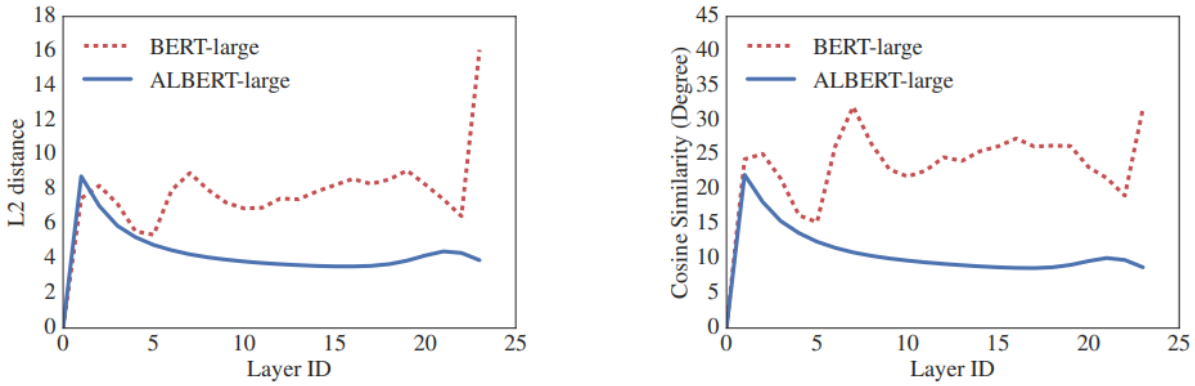

Figure 2: The L2 distances and cosine similarity (in terms of degree) of the input and output embedding of each layer for BERT-large and ALBERT-large.

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	60M	24	2048	128	True
	xxlarge	235M	12	4096	128	True

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

Inter-sentence coherence loss

문제

- BERT의 학습은 두가지로 진행
    - Masked Language Modeling (MLM) loss, Next Sentence Prediction (NSP) Loss
    - NSP 목표 : 문장 쌍 간의 관계에 대한 추론을 필요로하는 자연 언어 추론과 같은 세부 task에서 성능향상을 꾀함
    - 현실 : NSP가지고 성능 향상이 됐다고 보기 힘들다는 후속 논문(XLNet, RoBERTa)들이 나옴. MLM에 비해서 어려운 task가 아니기 때문!
        - NSP는 주제 예측과 일관성 예측을 하나의 task로 통합하는 문제가 있음. (다음 문장이 아니라고 같은 주제를 의미하지 않는건 아니기 때문)

제안

- NSP 쓰지말자! 나도안쓴다!
- ALBERT는 문장의 일관성에 기초한 loss를 제안 => NSP 대신 Sentence Order Prediction (SOP) loss를 사용
    - 모델이 담화 수준의 일관성 특성에 대해 더 세밀한 구별을 배우게 됨
    - Positive Example : 학습 코퍼스에서 연속적인 세그먼트를 학습데이터로 사용
    - Negative Example : 학습 코퍼스에서 연속적인 세그먼트를 추출후 순서를 뒤집어서 학습데이터로 사용

- NSP대신 SOP를 쓸 경우 Pritrain된 모델이 NSP을 잘 못하지만 fine-tuning하면 금방 복구되는 것을 실험을 통해 확인

SP tasks	Intrinsic Tasks MLM	NSP	SOP	Downstream Tasks 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**

Table 6: The effect of sentence-prediction loss, NSP vs. SOP, on intrinsic and downstream tasks.

---

n-gram masking

SpanBERT에서 보여준 Span Masking(=N-Gram Masking)을 사용 (SpanBERT 논문 : https://arxiv.org/abs/1907.10529)

ALBERT에서는 n-gram의 최대 길이를 3으로 지정

=> MLM의 타겟을 "White House correspondents" 와같은 단어까지 고려

batch size : 4096

LAMB optimizer에 Learning rate 0.00176 (Reducing BERT pre-training time from 3 days to 76 minutes 논문 : https://arxiv.org/abs/1904.00962)

그외에 모든 실험 환경 설정은 BERT와 동일

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	17.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	3.8x
	xlarge	1270M	86.4/78.1	75.5/72.6	81.6	90.7	54.3	76.6	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	21.1x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	6.5x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	2.4x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	1.2x

Table 3: 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).

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

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