**ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations**

**Introduction**

BERT와 같은 모델은 크기가 커지면 성능이 향상됨. 하지만 커짐에 따라 문제가 발생

1. Memory Limitation
2. Training Time
3. Memory Degradation – Layer 수 or Hidden Size가 너무 커지면 모델 성능 오히려 감소

다음 사항을 개선해서 모델 크기 줄이며 GLUE, SQuAD, RACE Task에 대해 더 높은 성능을 얻음

1. Factorized Embedding Parameterization – input layer의 parameter 수를 줄임 -> 모델 크기를 줄였다.
2. Cross-layer parameter sharing – 트랜스포머의 각 layer 간 같은 parameter 공유하여 사용 -> 모델 크기 줄임
3. Sentence order prediction – BERT의 NSP 대신 두 문장 간 순서를 맞추는 방식으로 학습 -> NSP 보다 SQUAD, MNLI, RACE task에서 성능 향상

**Background**

일반적으로 모델 크기가 클수록(layer 수 & hidden size) Language Representation 모델은 더 높은 성능

그렇다면 모델 크기가 커짐에 따라 문제가 없는지 & 작은 모델은 항상 안 좋은지? 에 중점해서 BERT를 개선 = 더 작은 모델로 성능을 높임

(이전에 Knowledge Distillation(증류, 축소) 등의 논문에서 큰 모델로 학습 후(일반적으로 큰 모델이 성능은 좋으니까) model compression을 하곤 했다.) + 심지어 ALBERT는 모델이 커질수록 성능이 안 좋아질 수 있음을 비판

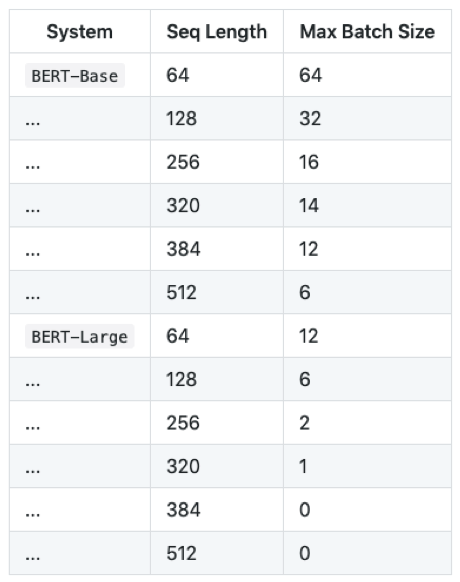
**1.1 Memory Limitation**

Out-Of Memory 발생 가능성. 메모리 128GB의 TPUv3를 사용할 수 있다면 문제 없지만..

12GB의 titan X GPU 사용한다 해보자(얘도 비싼거임 우리한테는)

이 GPU에서 각 seq-length 에 따른 max 배치 사이즈는 다음과 같다.

즉 이 GPU로는 BERT-Large: 384 seq-length로는 한 문장도 infer 불가



**1.2 Training Time**

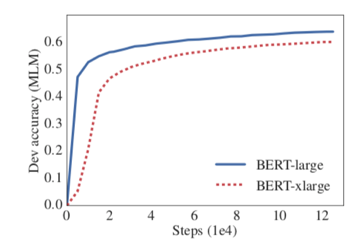
BERT 모델은 상당한 계산량 필요 Google은 학습 시간 줄이려고 여러 장의 TPU(텐서 프로세스 유닛) 활용, BERT base – 16개 TPU로 4일 정도, Large – 64개로 4일 정도

V100 GPU로는 5일 & 8일 이상

**1.3 Model degradation**

모델의 크기가 크다고 무조건 성능 높지 않음

XLarge가 Large보다 hidden size가 2배, 레이어 수는 같음(그렇다면 레이어 수도 같이 늘려준다면?), 그 이유는 설명 안하고 결과만 보여줌.



그렇다면 여기서 목표로 하는 것은 모델의 크기는 줄이고 성능은 높이는 것이다.

**2.1 Factorized Embedding Parameterization**

보통 BERT는 트랜스포머를 이용하기 때문에 layer의 input embedding size = output embedding size = hidden size로 같다. 하지만 이 input embedding size를 hidden size & output embedding size 보다 작게 설정하도록 한다.

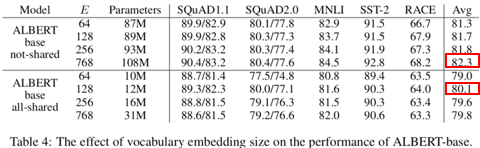
그 이유는 hidden size & output embedding size는 레이어를 거치면서 해당 token과 주변 token의 관계를 반영한 contextualized representation으로 많은 정보를 담기 때문에 줄이기가 좀 그렇다.

하지만 input embedding은 비교적 정보량이 적기 때문에 줄여도 괜찮다.

기존 BERT에서는 token embedding을 한 뒤에 V\*H를 곱해 트랜스포머 입력 차원에 맞게 조절했다. 그런데 여기서 input embedding size를 E로 줄였으니 V\*E -> E\*H로 해줘야 한다.

V는 크고 E는 상대적으로 작은 값이니 여기서 파라미터 수를 줄일 수 있다.

여기서 2개로 나누어 임베딩 = factorized 하다 볼 수 있다.



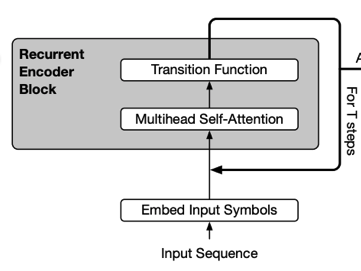
물론 성능은 임베딩 차원이 줄수록 떨어지지만 그만큼 계산량의 이점이 있다. 그리고 그렇게 많이 떨어지지도 않는다. 심지어 all-shared 버전에서는 128로 임베딩 차원이 낮을 때 성능이 높다.

**2.2 cross-layer parameter sharing**

위의 표 => cross-layer parameter 사용 시 파라미터 크게 준다.

사실 이것을 처음 제안한 건 Universal Transformer라는 논문 => 즉 Recurrent Transformer!

아래 그림처럼 Layer의 Output이 다시 Input으로 들어가게 된다.



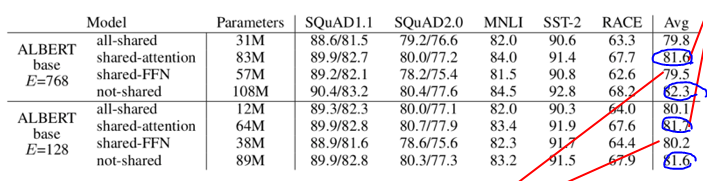
그렇다면 파라미터 공유 시 성능은?

Self-Attention Layer만 공유 했을 때는 괜찮지만 FFN을 공유 시에 크게 떨어짐

논문에서는 해석은 하지 않았고 결과만 뿌림. 설명이 필요함.

즉, 레이어 간에 파라미터 공유한다 해도 크게 성능 안떨어진다..!

따라서 ALBERT는 BERT와 같은 layer 수, hidden size 일지라도  
input embedding size 줄임 & 파라미터 공유로 모델의 크기가 작아지면서 성능이 크게 저하되지는 않는다.

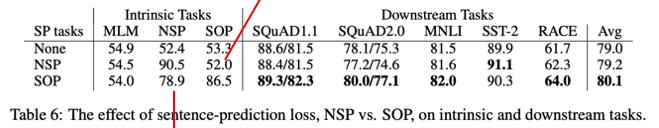


**2.3 Sentence Order Prediction**

또한 위에 2개(모델 크기 줄이기)를 떠나서 학습 방법 자체를 바꿔 보았다.

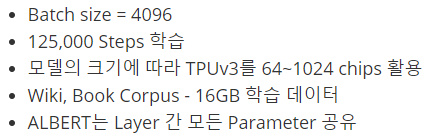
기존 BERT의 negative sample 시 아예 다른 topic(도메인)의 문장일 확률 매우 높다 => 그저 그 두 문장이 같은 토픽에 해당하는지 따지는 짓과 같다.(두 번째 문장이 첫 번째 다음으로 진짜 올까?) => 문장 간 연관 관계를 학습하기 힘들다.

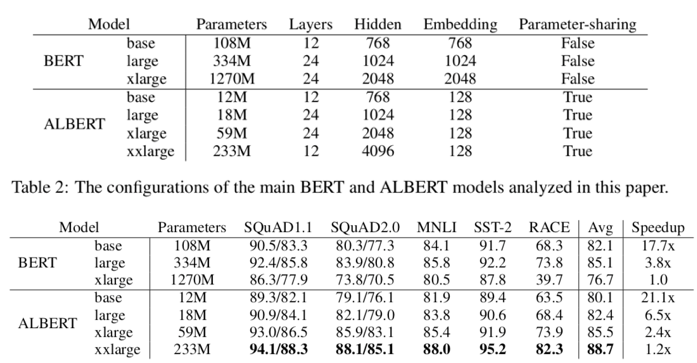
따라서 negative sample을 임의로 가져오는 것이 아니라 연속되게 가져오되 두 순서를 바꾸어 입력한다. 따라서 순서 맞추기 문제로 바뀌며 두 문장의 연관 관계를 더 잘 파악할 것이다.



SOP로 한다면 NSP Task는 어느정도 맞추지만 NSP로 할 시 SOP를 잘 못 맞춤 => NSP가 범용적이지 않음.

Experiment





이거로 3개의 challenge 해결

메모리 줄이고(parameters), 학습 속도 향상(speed up) 하지만 이유는 자세히 설명 파악 못함.

Model degradation에서도

Bert large가 xlarge 보다 성능이 높다(보통은 낮은데.. 모델은 커지는데 데이터 양이 부족해서 그런가 오버핏이라 ??? 아니면 레이어수랑 같이 늘려줘야 하나..?)

하지만 ALBERT의 경우 large 보다 xlarge가 성능 더 높다.

또한 xxlarge가 xlarge보다 모델이 더 큰데 성능 더 높다.

그런데 bert large랑 albert large 에서는 bert large가 역시 더 높다(파라미터 공유 x, 임베딩 차원 높음) 그렇지만 albert 에서는 여유가 있으니 xxlarge까지 높여도 model degradation이 안일어난다.(하지만 xxxlarge(6144개)가 xxlarge보다는 떨어진다.)