ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators

발표자: 박채원

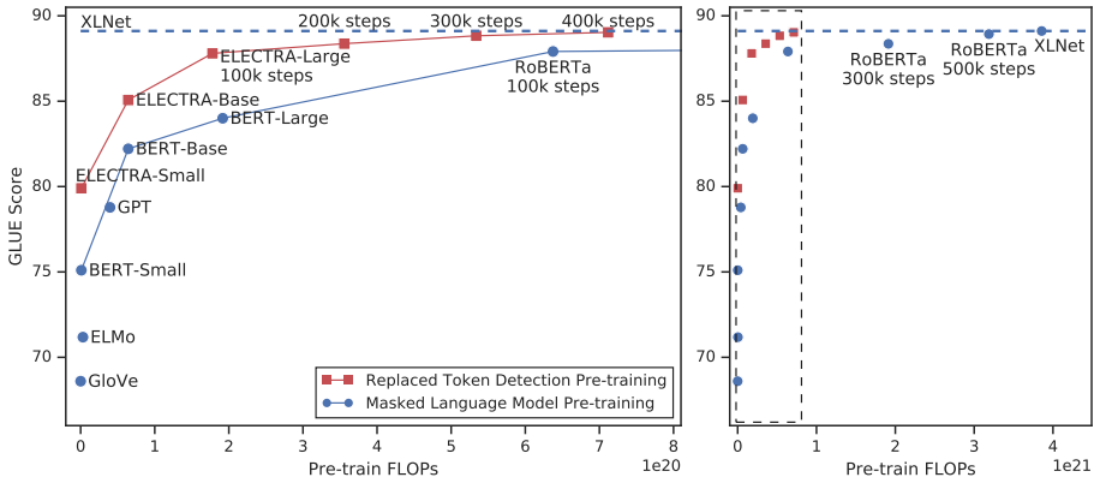
2022-05-13

0. Abstract

- Replaced Token Detection(RTD): Input을 마스킹 하는 대신 generator를 통해 token을 적절한 토큰으로 치환하고, Discriminator가 각 토큰이 generator에 의해 치환 된 토큰인지 예측(이진 분류)함으로써 representation을 학습하는 pre-training 방식
- 결론적으로 기존 Masked Language Modeling(MLM)기반 모델(BERT, XLNet) 보다 동일한 model size, data, compute 조건에서 우수한 성능을 보임

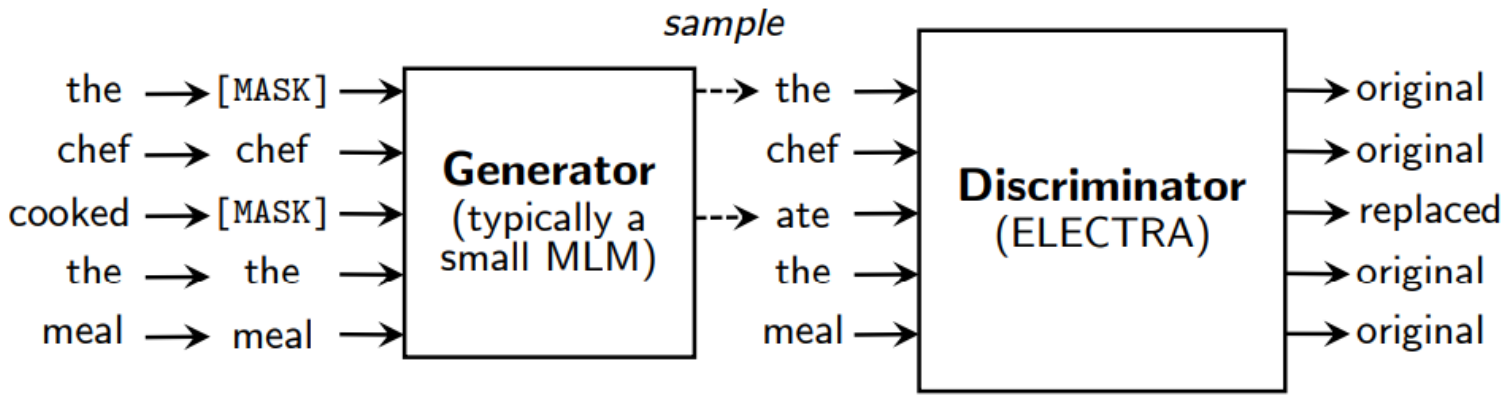
1. Introduction

- 기존 MLM 태스크는 몇가지 문제점들이 존재
  - 하나의 example에서 15%에 대해서만 학습하기 때문에 학습하는 데 비용이 많이 든다
  - 학습 때는 모델이 MASK 토큰을 참고하여 예측하지만 실제 추론시엔 MASK 토큰이 존재하지 않음
- 이러한 문제점을 해결하기 위해 새로운 pre-training 태스크 제안 -> RTD
- RTD: 실제 입력의 일부 토큰을 generator가 생성해낸 가짜 토큰으로 바꾸고, discriminator가 입력 문장의 각 토큰이 진짜 토큰인지 가짜 토큰인지 맞혀 학습하는 사전학습 방법
- BERT보다 훨씬 더 빠르게 학습 가능하다.
- 더 적은 파라미터로 비슷한 성능을 내며,
  더 빠르게 학습 가능하다.
- 즉, 더 효율적인 학습이 가능하다.



<Figure 1>

2. Method



<그림 1.RTD 과정>

- Generator (생성자)
  - BERT의 MLM과 동일

1) 입력 x에 대해서 마스킹할 위치의 집합 m을 결정
2) 결정한 위치에 있는 입력 토큰을 [MASK]로 치환
3) 마스킹 된 입력에 대해서 generator는 원래 토큰이 무엇인지 예측

$$p_G(x_t|\boldsymbol{x}) = \exp\left(e(x_t)^T h_G(\boldsymbol{x})_t\right) / \sum_{x'} \exp\left(e(x')^T h_G(\boldsymbol{x})_t\right)$$

4) 최종적으로 MLM loss로 학습

$$\mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i|\boldsymbol{x}^{\text{masked}})\right)$$

2. Method

- Discriminator (식별자)
  - Discriminator D는 입력 토큰 시퀀스에 대해 각 토큰이 진짜(original)인지 가짜(replaced)인지 이진분류로 학습

1) Generator G를 이용해서 마스킹 된 입력 토큰들을 예측
2) Generator G에서 마스킹할 위치의 집합 m에 해당하는 위치의 토큰을 [MASK]가 아닌, generator의 softmax 분포에 대해 샘플링한 토큰으로 치환

ex) [the, chef, cooked, the, meal] -> [[MASK], chef, [MASK], the, meal] -> [the, chef, ate, the, meal]

3) 치환된 입력에 대해서 discriminator는 각 토큰이 진짜 토큰인지 가짜 토큰인지 예측
   -> 이진 분류 / target classes: original, replaced

$$D(\boldsymbol{x}, t) = \text{sigmoid}(w^T h_D(\boldsymbol{x})_t)$$

4) 최종적으로 아래와 같은 loss로 학습

$$\mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D) = \mathbb{E}\left(\sum_{t=1}^{n} -\mathbb{1}(x_t^{\text{corrupt}} = x_t)\log D(\boldsymbol{x}^{\text{corrupt}}, t) - \mathbb{1}(x_t^{\text{corrupt}} \neq x_t)\log(1 - D(\boldsymbol{x}^{\text{corrupt}}, t))\right)$$

2. Method

- GAN과의 차이점
  - Generator와 Discriminator를 갖는다는 측면에서 GAN과 유사점이 있지만 Electra의 training objective는 GAN과 몇 가지 차이점이 있다.

  1) Generator가 원래 토큰과 동일한 토큰을 생성했을 때, GAN은 negative sample로 간주하지만 ELECTRA는 positive sample로 간주함
  2) Generator가 discriminator를 속이기 위해 adversarial(적대적)하게 학습하는 게 아니고 maximum likelihood로 학습함.
  3) Electra는 Generator의 입력으로 노이즈 벡터를 넣어주지 않음

  - Electra는 대용량 코퍼스 X에 대해서 generator loss와 discriminator loss의 합을 최소화하도록 학습
  - 샘플링 과정이 있기 때문에 discriminator의 loss는 generator로 역전파 되지 않는다. (될 수 없다)
  - 그래서 pre-training을 마친 후 generator는 버리고 discriminator만 취해 downstream task로 fine-tuning을 진행

3. Experiments

- Experimental Setup
  - 데이터셋
    - GLUE 벤치마크와 SQuAD 데이터셋을 사용
    - 대부분의 실험은 BERT와 동일하게 Wikipedia와 BooksCorpus를 사용해서 pre-training 함
    - Large 모델은 XLNet에서 사용한 ClueWeb, CommonCrawl, Gigaword를 사용해서 pre-training
  - 모델 세팅
    - 모델의 구조와 대부분의 하이퍼 파라미터 모두 BERT와 동일하게 세팅
    - 몇몇 데이터셋은 크기가 작아 랜덤 시드의 영향을 많이 받을 수 있어 10번의 fine-tuning 결과의 중간값을 최종 성능으로 사용

3. Experiments

- Model Extension: 가중치 공유
  - Generator와 Discriminator는 모두 transformer 인코더 구조이기 대문에 두 네트워크의 가중치를 공유하여 학습하는 weight sharing 기법을 써볼 수 있음
    1) 임베딩의 가중치만 공유하고 그 외의 가중치는 따로 학습
    2) 모든 가중치를 서로 공유하는 방법
      - 공유하지 않는 경우: 83.5
      - 임베딩만 공유: 84.3
      - 모든 가중치를 공유: 84.4

Discriminator는 입력으로 들어온 토큰만 학습하는 반면, generator는 출력 레이어에서 softmax를 통해 사전에 있는 모든 토큰에 대해서 밀도있게 학습할 수 있다. 그렇기 때문에 generator와 discriminator가 임베딩을 공유해서 학습한 경우 discriminator는 훨씬 효과적으로 학습할 것이다.

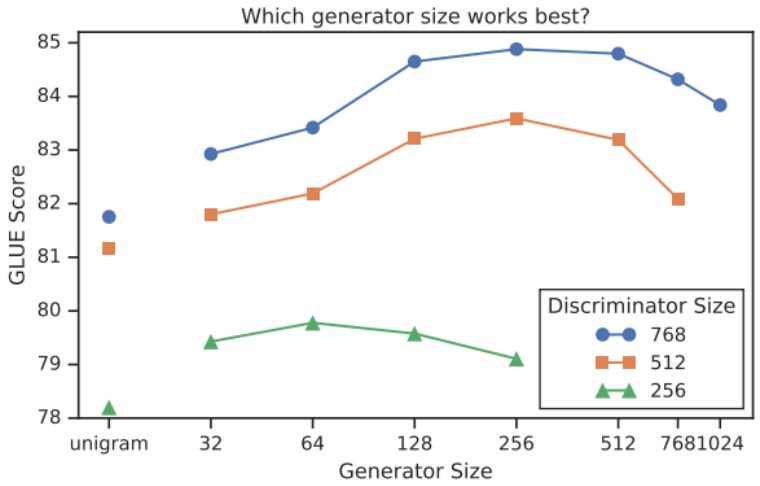
하지만 모든 가중치를 공유하려면 둘의 크기를 동일하게 맞춰야 한다는 제약이 있음

결국엔 generator는 버리고 discriminator만 취하는데 동일한 크기로 설정하는 것은 학습의 효율을 떨어뜨릴 수 있다 -> 모두 임베딩만 공유하는 세팅으로 진행

3. Experiments

- Model Extension: Smaller Generators
  - Generator와 discriminator의 크기를 동일하게 가져가면 일반 MLM 모델에 비해 거의 두 배의 계산량이 필요
  - 이 문제를 완화하기 위해 generator의 (레이어) 크기를 줄여보는 실험 진행
    * 레이어 크기 – 히든 레이어 크기, FFN 크기, 어텐션 헤드의 수

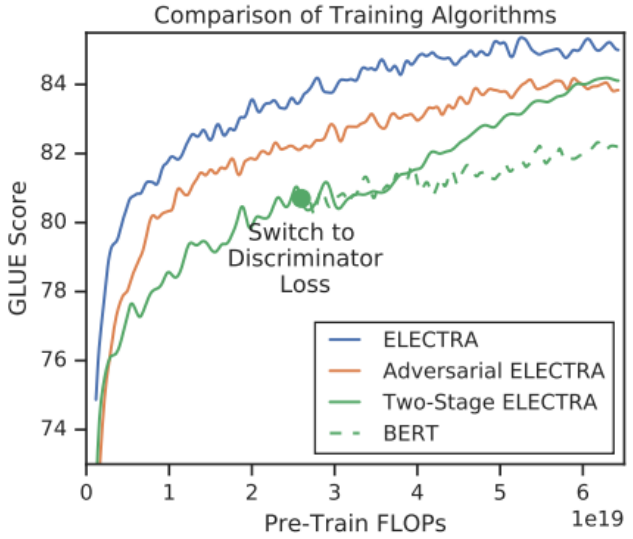
  - discriminator의 크기 대비 1/4 - 1/2 크기의 generator를 사용했을 때 가장 좋은 성능을 보임
  - Generator가 너무 강력해지면 discriminator의 태스크(이진분류)가 어려워져서 이런 현상이 나타날 수 있음
    심지어 discriminator의 파라미터를 실제 데이터 분포가 아닌 generator를 모델링하는 데 사용할 수 있다.



3. Experiments

- Model Extension: Training Algorithms
  - Electra를 효과적으로 학습시킬 수 있는 알고리즘
    1) Jointly 학습generator와 discriminator를 jointly 학습시키는 방식
    2) Two-stage 학습: generator만 n스텝동안 학습시키고, discriminator를 generator의 학습된 가중치로 초기화하고 discriminator만 n스텝 동안 학습시키는 방식
    3) Adversarial 학습: GAN처럼 adversarial training을 모사해서 학습시키는 방식

  - 실험 결과로 jointly 학습시키는 방식이 가장 좋음을 확인
  - Adversarial 학습이 maximum likelihood 기반의 학습보다 성능이 낮음



3. Experiments

- Small Models
  - 이 연구의 큰 향상점 중 하나는 pre-training의 효율성 향상
  - 이를 검증하기 위해 하나의 GPU로도 빠르게 학습할 수 있는 수준으로 작은 모델을 만들어보는 실험 진행
  - ELECTRA-Small 모델의 하이퍼파라미터 개수는 BERT-Base에 비해서 매우 작음
  - BERT와 함께 ELMo와 GPT도 비교

Model	Train / Infer FLOPs	Speedup	Params	Train Time + Hardware	GLUE
ELMo	3.3e18 / 2.6e10	19x / 1.2x	96M	14d on 3 GTX 1080 GPUs	71.2
GPT	4.0e19 / 3.0e10	1.6x / 0.97x	117M	25d on 8 P6000 GPUs	78.8
BERT-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	75.1
BERT-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	82.2
ELECTRA-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	79.9
50% trained	7.1e17 / 3.7e9	90x / 8x	14M	2d on 1 V100 GPU	79.0
25% trained	3.6e17 / 3.7e9	181x / 8x	14M	1d on 1 V100 GPU	77.7
12.5% trained	1.8e17 / 3.7e9	361x / 8x	14M	12h on 1 V100 GPU	76.0
6.25% trained	8.9e16 / 3.7e9	722x / 8x	14M	6h on 1 V100 GPU	74.1
ELECTRA-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	85.1

  - ELECTRA-Small은 BERT-Small보다 약 5포인트 높은 성능을 보였고, ELECTRA보다 훨씬 큰 GPT보다도 좋은 성능을 보이고, 매우 빠른 수렴 속도를 보임.
  - ELECTRA-Base는 BERT-Base를 능가할 뿐만 아니라 BERT-Large보다도 좋은 성능을 기록

3. Experiments

- Large Models
  - BERT-Large에 맞춰 실험 진행하였고, 학습 데이터로는 XLNet에서 사용한 데이터 사용

Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa-500K	3.2e21 (4.5x)	356M	68.0	96.4	90.9	92.1	92.2	90.2	94.7	86.6	88.9
XLNet	3.9e21 (5.4x)	360M	69.0	**97.0**	90.8	92.2	92.3	90.8	94.9	85.9	89.1
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA-400K	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	**92.4**	90.5	94.5	86.8	89.0
ELECTRA-1.75M	3.1e21 (4.4x)	335M	69.1	96.9	90.8	**92.6**	**92.4**	**90.9**	**95.0**	**88.0**	**89.5**

<GLUE dev set>

Model	Train FLOPs	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	WNLI	Avg.*	Score
BERT	1.9e20 (0.06x)	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	65.1	79.8	80.5
RoBERTa	3.2e21 (1.02x)	67.8	96.7	89.8	91.9	90.2	90.8	95.4	88.2	89.0	88.1	88.1
ALBERT	3.1e22 (10x)	69.1	**97.1**	**91.2**	92.0	90.5	**91.3**	–	89.2	91.8	89.0	–
XLNet	3.9e21 (1.26x)	70.2	**97.1**	90.5	**92.6**	90.4	90.9	–	88.5	**92.5**	89.1	–
ELECTRA	3.1e21 (1x)	**71.7**	**97.1**	90.7	92.5	**90.8**	**91.3**	**95.8**	**89.8**	**92.5**	**89.5**	**89.4**

< GLUE test set>

  - 적은 계산량으로 RoBERTa나 XLNet과 유사한 성능을 보임
  - 더 많이 학습시킨 ELECTRA는 이들을 뛰어넘는 성능을 보이고, 이 계산량 조차도 두 모델보다 작다.
  - GLUE와 SQuAD에서도 마찬가지로 ELECTRA는 가장 좋은 성능을 보임

3. Experiments

- Efficiency Analysis
  - 구성 요소를 일부 제거해 효율성 분석: 일종의 ablation study

  1) ELECTRA 15%: 구조를 유지하고, discriminator loss를 입력 토큰의 15%만으로 만들도록 세팅
  2) Replace MLM: discriminator를 MLM 학습을 하되, [MASK]로 치환하는 게 아니고 generator가 만든 토큰으로 치환
  3) All-Tokens MLM: Replace MLM처럼 하되, 모든 토큰을 generator가 생성한 토큰으로 치환

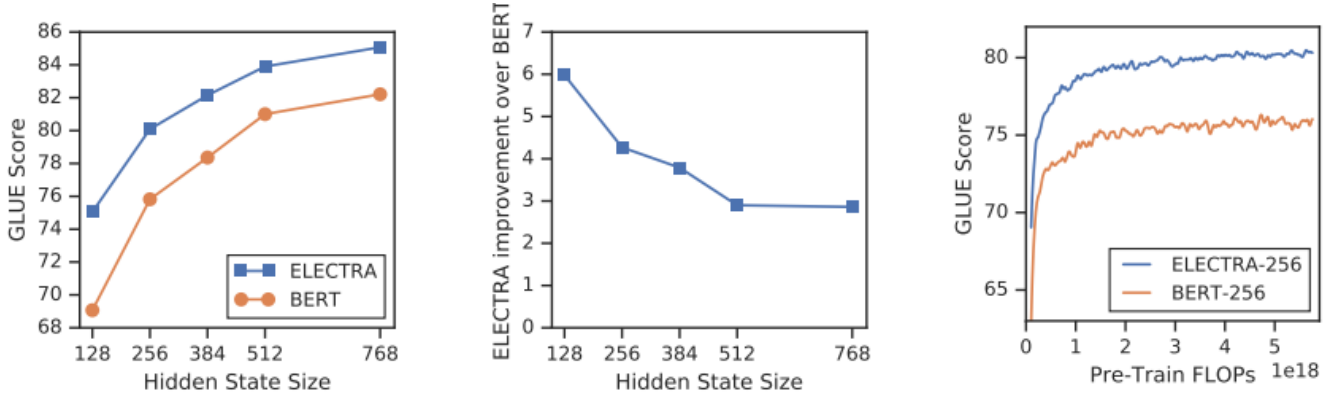
Model	ELECTRA	All-Tokens MLM	Replace MLM	ELECTRA 15%	BERT
GLUE score	85.0	84.3	82.4	82.4	82.2

- ELECTRA 모델이 가장 높은 성능을 보였고, All-Tokens MLM이 그에 가장 근접한 성능을 보임.

- 결론적으로 ELECTRA가 학습 효율도 좋으며, [MASK] 토큰에 대한 pre-training과 fine-tuning간의 불일치 문제도 완화시킨 것을 알 수 있음

3. Experiments

- Efficiency Analysis
  - 히든레이어 크기에 따른 성능 변화 실험



  - 히든 레이어의 크기가 작을수록 BERT와 ELECTRA의 성능 차이가 커짐
  - 모델이 작아도 BERT에 비해 ELECTRA는 매우 빠르게 수렴함

  - 즉, ELECTRA가 BERT보다 효율적으로 학습함

4. Conclusion

- 이 논문은 language representation learning을 위한 새로운 self-supervision 태스크인 Replaced Token Detection을 제안
- 작은 generator가 만들어 낸 질 좋은 negative sample과 입력 토큰을 구별하도록 텍스트 인코더(discriminator)를 학습
- MLM에 비해, 제안하는 RTD는 훨씬 효율적이고 downstream tasks에 대한 결과 역시 좋다는 것을 다양한 실험을 통해 확인할 수 있었음
- 또한 다른 모델들에 비해서 상대적으로 적은 계산량을 사용하는 경우에 더 효과적이었음