BERT Variants

Sang Yup Lee

BERT Variants

- BERT를 변형, 발전시킨 모형들
  - 사용된 방법은 목적에 따라 두 가지 종류로 구분
    - BERT의 파라미터수를 줄여 필요한 컴퓨팅 파워를 감소시키기 위한 것
    - BERT의 성능 향상
  - 주요 모형들
    - ALBERT, RoBERTa, ELECTRA
    - 지식 증류 (Knowledge distillation) 기반 방법들
      - DistilBERT, TinyBERT 등

ALBERT

ALBERT

- 소개
  - Lan et al. (2019)
  - A Lite version of BERT
  - 기존 BERT를 경량화한 버전
- 기존 BERT의 주요 문제
  - 파라미터 수가 너무 많다.
  - $BERT_{BASE}$ = 110M, $BERT_{LARGE}$ = 340M
- 파라미터의 수를 줄이기 위한 주요 방법 두 가지
  - Factorized embedding layer parameterization (FEP)
  - Cross-layer parameter sharing (CPS)

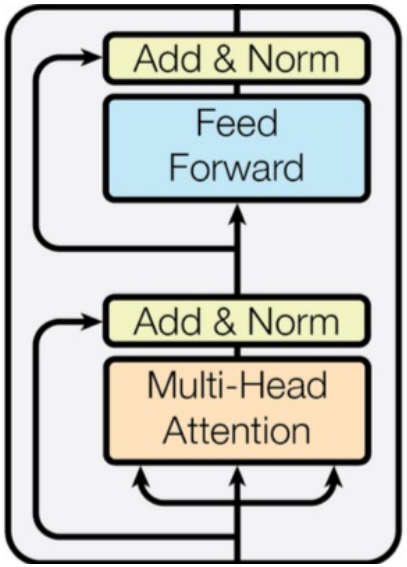
Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R.(2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

ALBERT

- Factorized embedding layer parameterization (FEP)
  - 기존 BERT의 파라미터 수가 많은 주요 이유 중 하나
    - 임베딩 벡터의 크기(E)가 너무 크다.
    - 은닉 상태 벡터의 크기(H=768)와 동일
    - WordPiece tokenizer의 경우 30K개의 토큰 존재 => 파라미터의 수 = 768*30K = 23M
  - FEP
    - 직접적으로 E = H로 설정하지 않고, 일단 저차원(D, D<H)의 벡터로 표현한 뒤, 선형 변환 (linear projection)을 통해 H 차원의 벡터로 확장
    - 예) D = 64, H = 768
      - 30000*64+64*768 (=1,969,152)

ALBERT

- Cross-layer parameter sharing (CPS)
  - 모든 인코더 블록의 파라미터들을 학습하는 것이 아니라, 첫 번째 인코더 블록의 파라미터만을 학습하고 해당 파라미터의 값을 나머지 인코더 블록의 파라미터들이 공유 => 학습을 해야하는 파라미터의 수 감소 뿐만 아니라 학습이 안정적으로 되는 효과 존재
  - 방법
    - All-shared
    - Shared feedforward network
    - Shared attention
- 효과
  - 파라미터의 수 1/18배, 학습의 속도 1.7배



<Encoder block>

ALBERT

- ALBERT의 학습 방법
  - 마스크 언어 모형 (Masked Language Model, MLM)
  - 문장 순서 예측 (Sentence Order Prediction, SOP)
    - 입력된 두 개의 문장들에 대해서 두 문장의 순서가 제대로 되었는지 아니면 바뀌었는지를 예측
  - NSP는 사용하지 않음
    - 모형 성능 개선 효과 미흡 => NSP가 MLM 보다 많이 쉬운 작업이기 때문

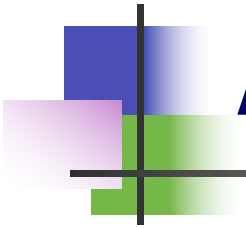
ALBERT

- SOP의 예
  - 예제 1 (다음과 같이 입력)
    - 문장 1: I watched an action movie yesterday.
    - 문장 2: It was fun.
    - 순서 맞음 => 종속변수 = 긍정
  - 예제 2
    - 문장 1: It was fun.
    - 문장 2: I watched an action movie yesterday.
    - 순서 잘못 => 종속변수 = 부정

ALBERT

- BERT와의 비교

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



ALBERT

- 파이썬 코딩 (감성분석)
  - pip install sentencepiece
  - 두 가지 방법
    - Feature-based
    - Fine-tuning
  - Feature-based
    - ALBERT_En_movie_review_sentiment_feature_based.ipynb
  - Fine-tuning
    - ALBERT_En_movie_reviews_sentiment_fine_tuning.ipynb
    - ALBERT_Kor_movie_reviews_sentiment_fine_tuning.ipynb

RoBERTa

RoBERTa

- 소개
  - Robustly Optimized BERT-Pretraining Approach
  - Liu et al. (2019)
- 저자들이 생각하는 BERT의 주요 단점
  - 학습이 충분하게 되지 않았다.
- 새로운 방법들 적용
  - 정적 마스킹 (static masking)이 아닌 동적 마스킹 (dynamic masking) 방법 사용
  - 학습시, NSP 작업을 사용하지 않음
  - 더 많은 데이터를 사용하고, 학습시 더 큰 크기의 미니 배치를 사용
  - WordPiece 토크나이저가 아닌 바이트 단위 바이트 페어 인코딩 (Byte-Level Byte-Pair Encoding, BBPE) 방법 사용

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V.(2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

RoBERTa

- 정적 vs. 동적 마스킹
  - BERT의 경우, 정적 마스킹
    - 전처리 단계에서 전체 토큰의 15%를 마스킹 단어로 선택 => 이후 동일한 마스킹 토큰들이 매 에포크 마다 반복적으로 사용됨
  - RoBERTa, 정적 마스킹
    - 동일한 문장을 10개 복사, 각 문장에 대해 랜덤하게 15%를 [MASK] 토큰으로 대체
    - 40 에포크 학습
    - 특정 방식으로 마스킹된 하나의 문장이 4번만 사용

RoBERTa

- 정적 vs. 동적 마스킹
  - RoBERTa, 동적 마스킹
    - 전처리 과정에서 마스킹을 하는 것이 아니라, 문장이 학습에 사용될 때 마다, 문장에서의 15%를 랜덤하게 마스킹 하는 방법

Masking	SQuAD 2.0	MNLI-m	SST-2
$\text{BERT}_{\text{BASE}}$	76.3	84.3	92.8
RoBERTa			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

RoBERTa

- NSP 작업을 사용하지 않음
- 서로 다른 4가지 방식 적용 & 비교
  - NSP
    - SEGMENT-PAIR+NSP, SENTENCE-PAIR+NSP
  - No NSP
    - FULL-SENTENCES, DOC-SENTENCES
- 각 방식 비교
  - SEGMENT-PAIR+NSP
    - 하나의 입력 시퀀스가 두 개의 세그먼트로 구성되며, 하나의 세그먼트는 여러 개의 문장 포함 가능. 단, 전체 길이 <= 512
    - 두 개의 세그먼트가 연속된 것인지 그렇지 않은 것인지를 예측

RoBERTa

- 각 방식 비교 (cont'd)
  - SENTENCE-PAIR+NSP
    - 하나의 입력 시퀀스가 서로 다른 두 개의 문장으로 구성. 단, 전체 길이 <= 512
    - 두 개의 세그먼트가 연속된 것인지 그렇지 않은 것인지를 예측
  - FULL-SENTENCES
    - 연속된 여러 개의 문장을 하나 이상의 서로 다른 문서에서 추출하고 이를 하나의 관측치로 사용
    - 전체 길이 <= 512
    - 관측치를 구성하기 위해서 문장들을 여러 문서에서 추출하는 경우, 하나의 문서의 끝에 도달하는 경우는 그 다음 문서에서 문장들을 추출하고 서로 다른 문서에서 추출된 문장들 사이에는 [SEP] 토큰을 추가

RoBERTa

- 각 방식 비교 (cont'd)
  - DOC-SENTENCES
    - FULL-SENTENCES과 비슷하게 여러 개의 문장으로 하나의 관측치를 구성하지만, 하나의 관측치를 구성하는 문장들은 서로 다른 문서에서 추출될 수 없음
  - 성능 비교

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
RoBERTa (NSP 사용의 경우)				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
RoBERTa (NSP를 사용하지 않는 경우)				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3

RoBERTa

- 더 많은 학습 데이터 사용
  - BERT에서는 16GB의 학습 데이터를 사용한 반면, RoBERTa에서는 160GB의 학습 데이터를 사용
  - BERT에서 사용한 Toronto BookCorpus와 English Wikipedia 데이터셋 (16GB) 이외에 추가적으로 CC-News (Common Crawl-News, 76GB), Open WebText (38GB), Stories (31GB)라고 하는 데이터셋을 사용

RoBERTa

- 더 큰 미니배치 크기를 이용
  - BERT의 경우는 미니배치의 크기 = 256
  - RoBERTa의 경우 미니배치의 크기 = 8000

Data Size	Batch Size	Updates	SQuAD (v1.1/2.0)	MNLI-m	SST-2
16GB	8K	100K	93.6/87.3	89.0	95.3
160GB	8K	100K	94.0/87.7	89.3	95.6
160GB	8K	300K	94.4/88.7	90.0	96.1
160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**

RoBERTa

- 바이트 단위 바이트 페어 인코딩 (BBPE) 방법 사용
  - 이와 관련해서는 이전 'Tokenization_methods.pptx' 파일 참고

- 파이썬 코드
  - RoBERTa_Kor_movie_reviews_sentiment_fine_tuning_KLUE.ipynb

ELECTRA

ELECTRA

- 소개
  - Efficiently Learning an Encoder that Classifies Token Replacements Accurately
  - Clark et al. (2020)
- BERT와의 주요 차이
  - MLM 대신 Replaced Token Detection (RTD)
    - MLM의 주요 단점
      - 미세 조정에서는 [MASK] 토큰이 출현하지 않는 미스 매치 발생
      - 전체 학습 데이터에서 마스킹된 15% 토큰에 대해서만 학습을 진행 => 데이터의 15%만 사용
  - NSP 수행하지 않음

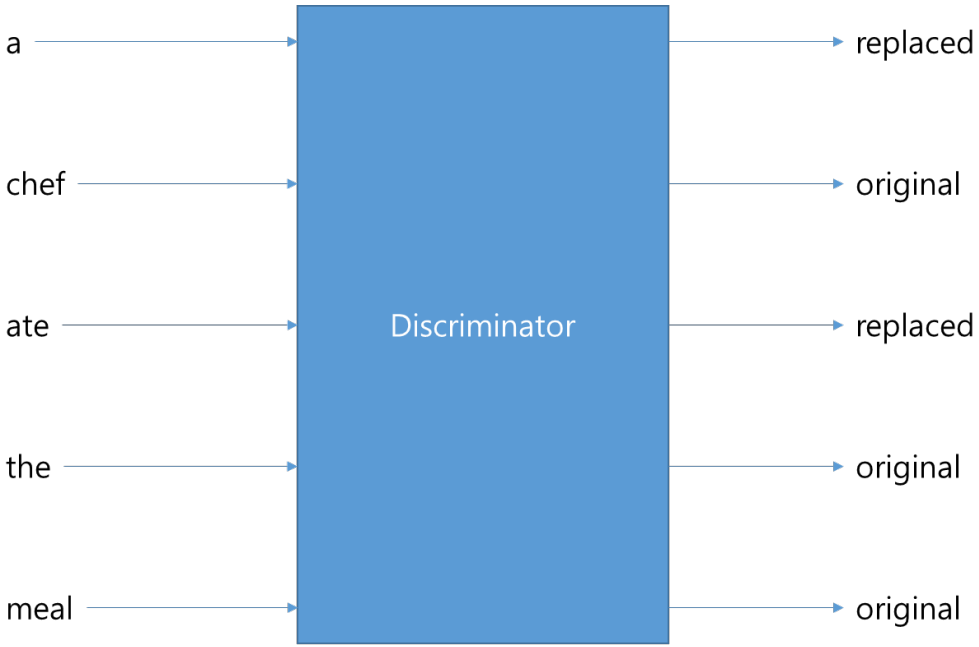
Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D.(2020). Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

ELECTRA

- RTD의 작동 방식
  - 입력 데이터
    - tokens = [the, chef, cooked, the, meal]
  - 일부를 대체
    - 첫 번째 토큰인 the와 세 번째 토큰인 cooked가 각각 a와 ate로 대체
    - tokens = [a, chef, ate, the, meal]
    - 각 토큰에 대해서 해당 토큰이 원래의 토큰인지 아니면 대체된 토큰인지를 예측하는 작업을 수행

ELECTRA

- RTD (cont'd)

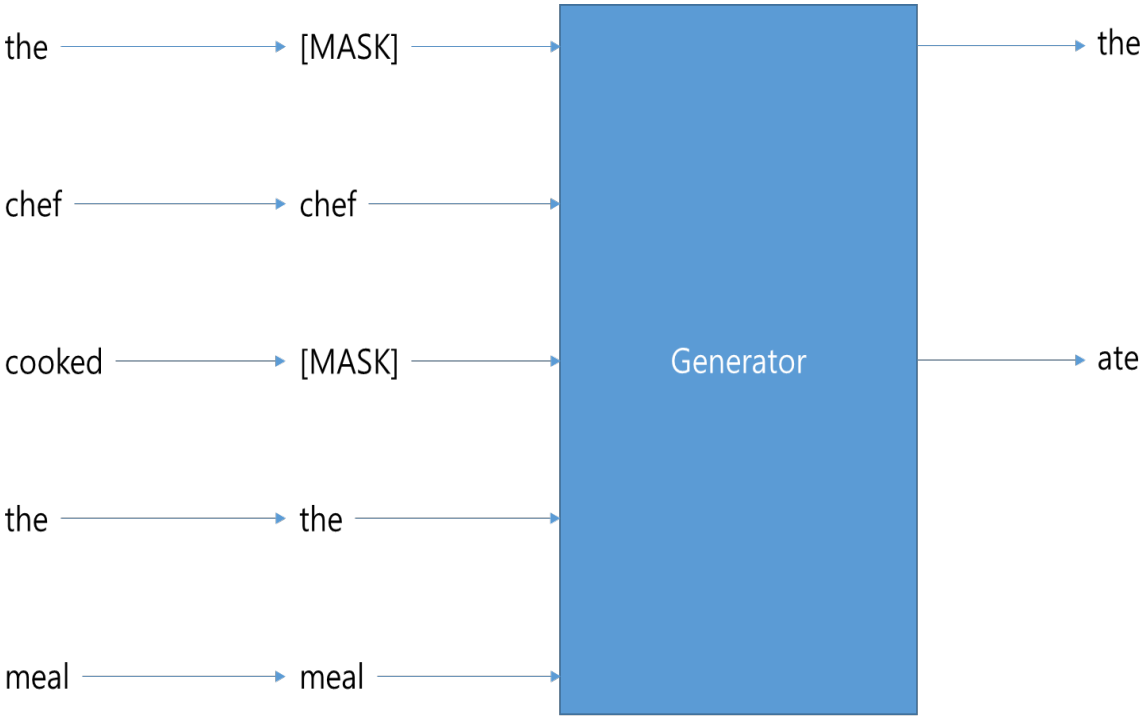


ELECTRA

- RTD (cont'd)
  - 어떠한 토큰들을 어떻게 대체하는가?
    - MLM 방법 사용
      - BERT 논문과 유사하게 전체 토큰 중에서 15%를 [MASK] 토큰으로 대체한 후, [MASK] 토큰에 대한 예측 작업을 수행하여, [MASK] 토큰에 대한 새로운 토큰을 예측하고, 그 결과를 이용해서 RTD 작업을 추가적으로 수행
    - Example
      - 입력 데이터: tokens = [the, chef, cooked, the, meal]
      - 15%를 마스킹: tokens = [[MASK], chef, [MASK], the, meal]
      - 단어 예측
        - the를 대체한 [MASK] 토큰이 the 토큰으로 예측되었고, cooked를 대체한 [MASK] 토큰이 ate라고 하는 토큰으로 예측되었다고 가정

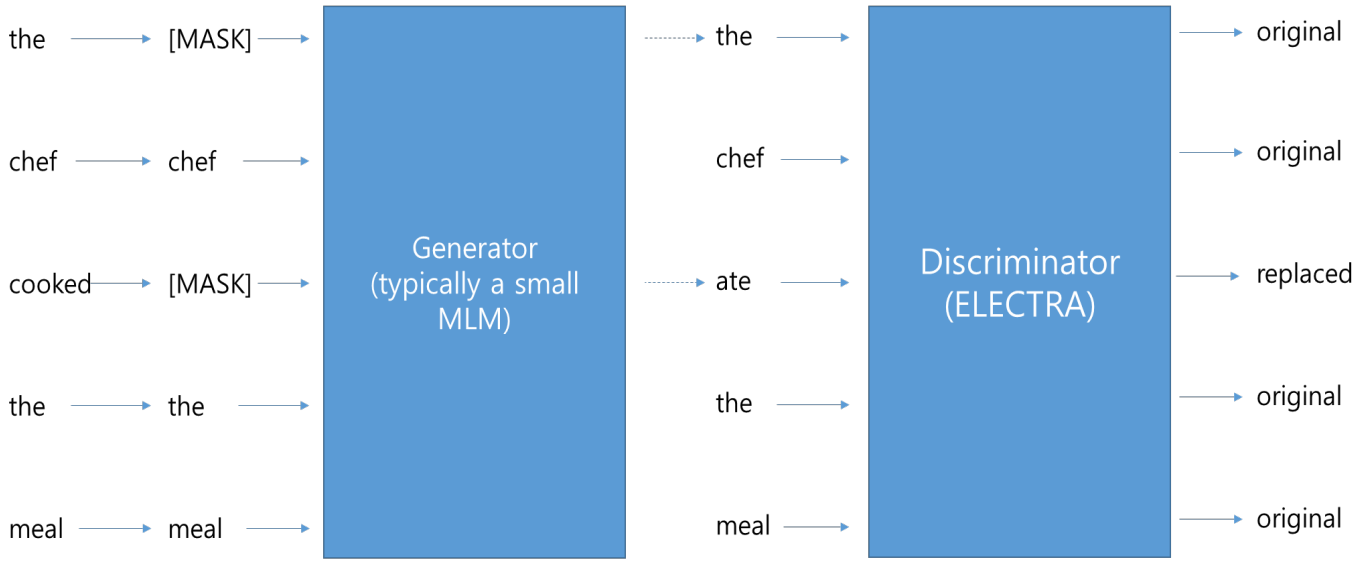
ELECTRA

- RTD (cont'd)



ELECTRA

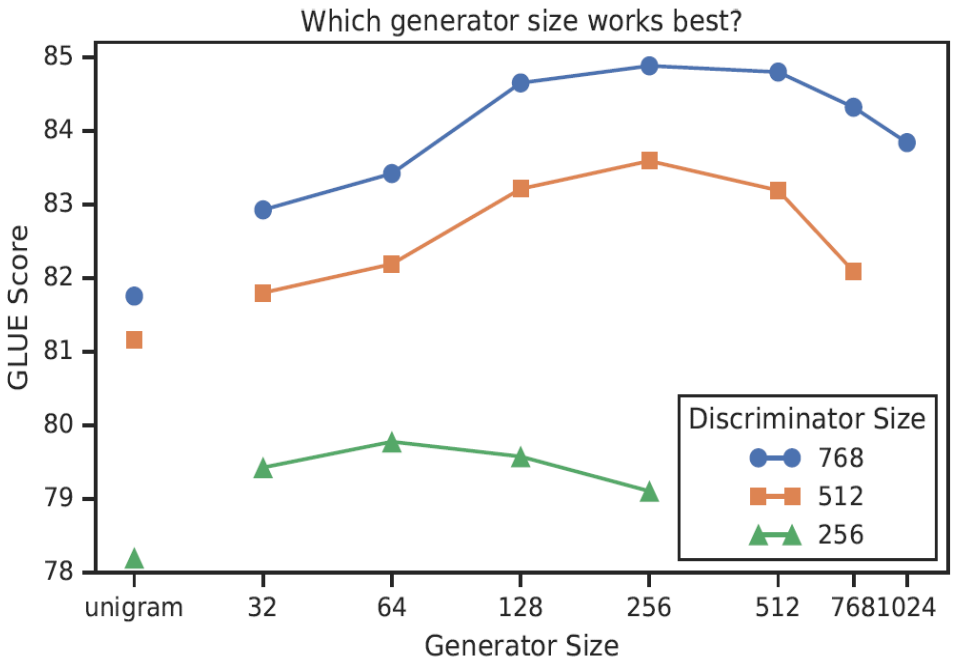
- RTD (cont'd)

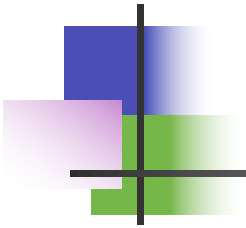


- 전체 비용함수
  - MLM 비용함수 + λ*RTD 비용함수

ELECTRA

- 그외 주요 방법
  - 생성기와 판별기 간 가중치 공유 (weight sharing)
  - 생성기의 크기를 판별기의 크기보다 작게 설정함
    - 동일하게 설정하는 경우, 두 배의 컴퓨팅 파워 필요
    - 인코더 블록이 출력하는 은닉 상태 벡터의 크기만 작게 설정





ELECTRA

- 파이썬 코드
  - ELECTRA_Kor_movie_reviews_sentiment_fine_tuning.ipynb

Knowledge Distillation 기반 방법들

Knowledge Distillation

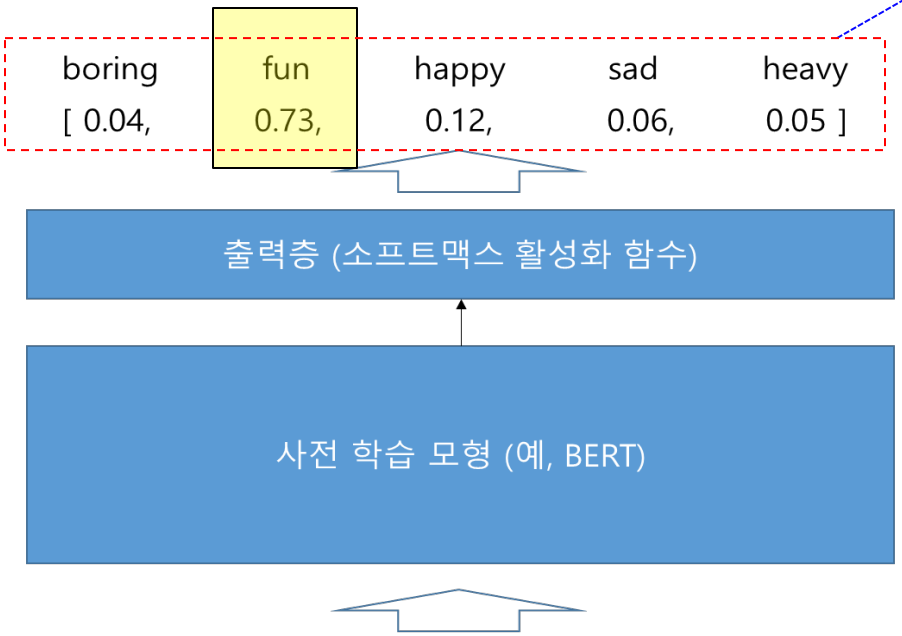
Knowledge Distillation

- What is it?
  - 작은 모형이 사전 학습되어 있는 큰 모형의 행동을 비슷하게 수행할 수 있도록 큰 모형이 학습을 통해 습득한 정보 (이러한 정보를 지식, knowledge라고 표현)를 작은 버전의 모형에 전이하는 것을 의미
  - a.k.a., 교사-학생 학습 (teacher-student learning) 방법
    - 사전 학습되어 있는 큰 모형이 교사가 되고, 작은 버전의 모형이 학생이 되는 것

Knowledge Distillation

- Example
  - 교사 모형: 언어 모형
    - 정답 후보 단어에 다섯 개만 존재한다고 가정

Dark Knowledge



정답 단어 정보 이외 추가적으로 얻을 수 있는 정보는? => 모형이 예측하는 정답 단어가 아닌 다른 단어들이 정답일 확률에 대한 정보 (정답으로 예측된 단어이외의 단어들 중에서 어떠한 다른 단어들이 상대적으로 큰 확률을 갖는지) =>모형의 일반화 정도를 반영

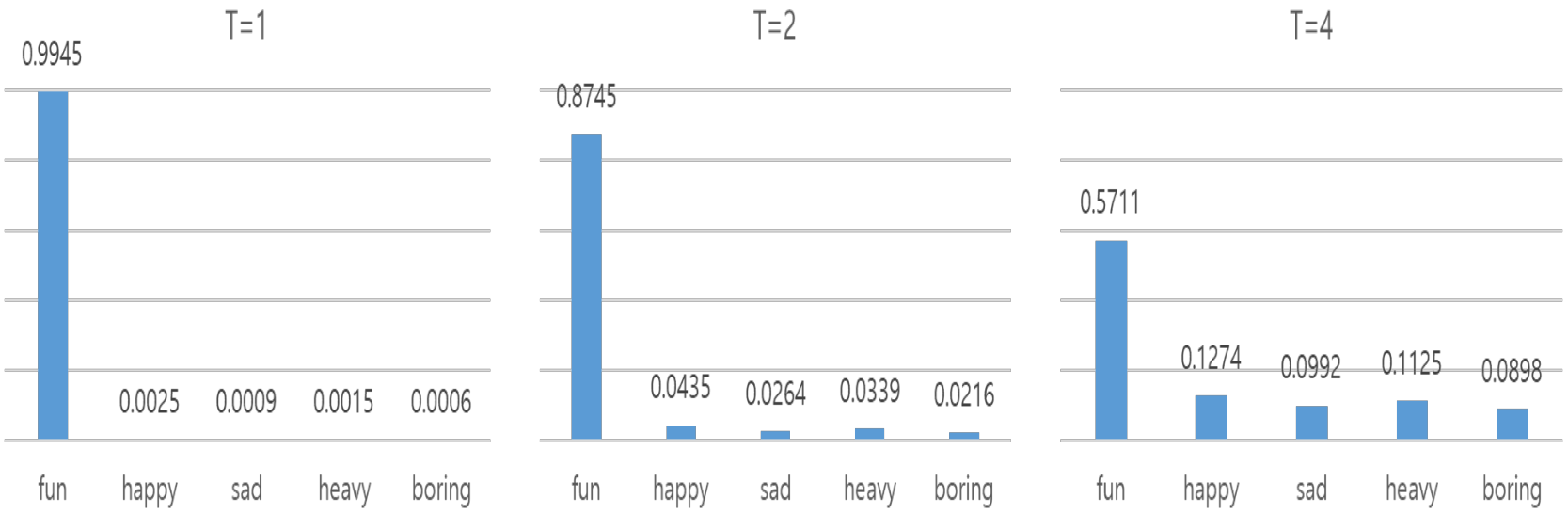
Knowledge Distillation

- 어떻게 지식 전이?
  - 교사 모형이 출력하는 확률 분포를 학생 모형이 사용하는 비용함수의 정답 정보로 사용함으로써
  - 학생 모형의 관점에서 교사 모형이 예측하는 이러한 확률 분포를 소프트 타겟 (soft target), 소프트 타켓에 대한 학생 모형의 예측을 소프트 예측 (soft prediction)이라 함
  - 비용함수: $Loss(\text{soft target}, \text{soft prediction})$
- Softmax Temperature
  - 성능이 좋은 교사 모형은 정답 단어에 대한 확률이 1에 가깝게 그리고 나머지 단어들의 확률은 0에 가깝게 출력 => little dark knowledge
  - 이러한 문제를 해소하기 위해 T 사용

$$S(z_i) = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

Knowledge Distillation

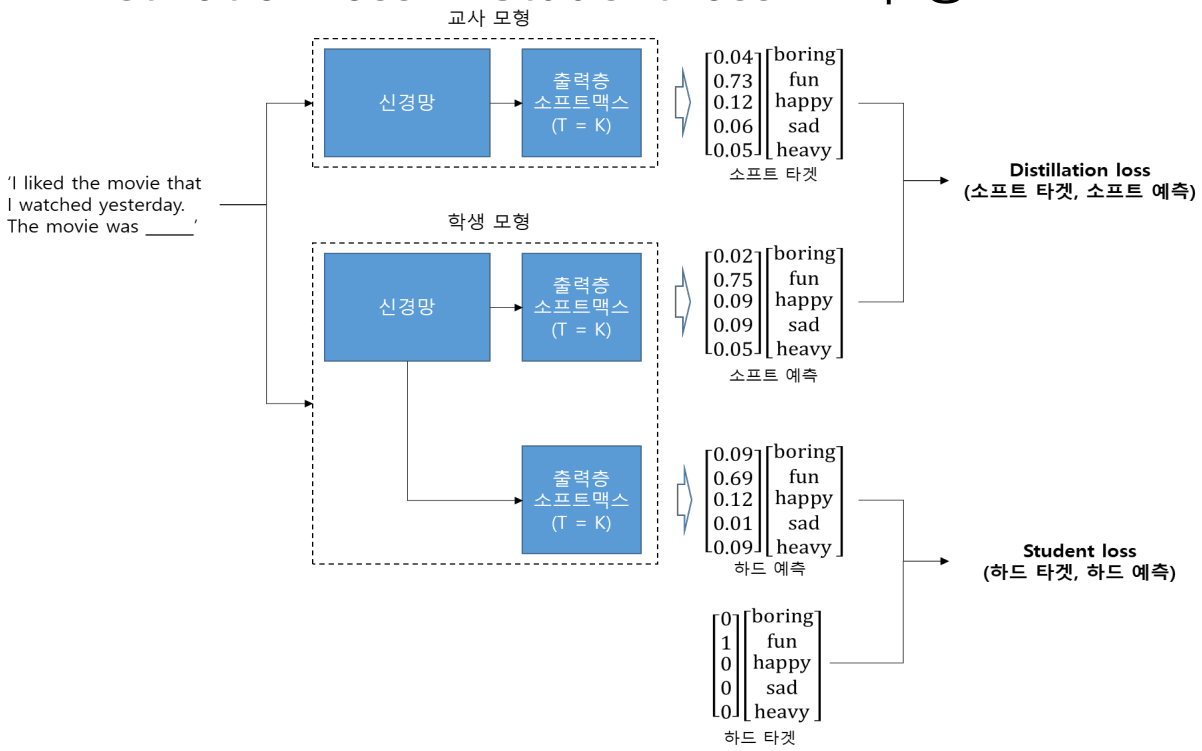
- Softmax Temperature의 예



Knowledge Distillation

학생 모형 자체적으로 원래 정답 정보를 이용해서 학습

- KD에서의 비용함수
  - Distillation loss + Student loss 로 구성



DistilBERT

DistilBERT

- 소개
  - 지식 증류 방법을 이용해서 파라미터의 수를 줄인 BERT
  - Sanh et al. (2019)
  - 교사 모형: $\text{BERT}_{\text{BASE}}$
  - BERT에 비해서 속도가 60% 정도 빠르고, 파라미터의 수도 40% 정도 감소

Sanh, V., Debut, L., Chaumond, J., & Wolf, T.(2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

DistilBERT

- 주요 사용 방법
  - 은닉 상태 벡터의 크기는 그대로 사용
  - 층수를 줄임
  - 데이터셋도 원래와 동일
  - RoBERTa에서 제안한 방법 일부 사용
    - NSP 사용하지 않음, 동적 마스킹 사용, 미니 배치 크기 증가 (4000)
  - 코사인 임베딩 비용함수 추가 사용
    - 교사 모형의 인코더 블록에서 출력되는 은닉 상태 벡터와 학생 모형의 인코더 블록에서 출력되는 은닉 상태 벡터 간의 코사인 거리
  - 최종 비용함수

$$\mathcal{L} = \alpha_d T^2 * \mathcal{L}_{distil} + \alpha_s * \mathcal{L}_{student} + \alpha_{cos} * \mathcal{L}_{cos}$$

$$T = 2, \alpha_d = 5, \alpha_s = 2, \alpha_{cos} = 1$$

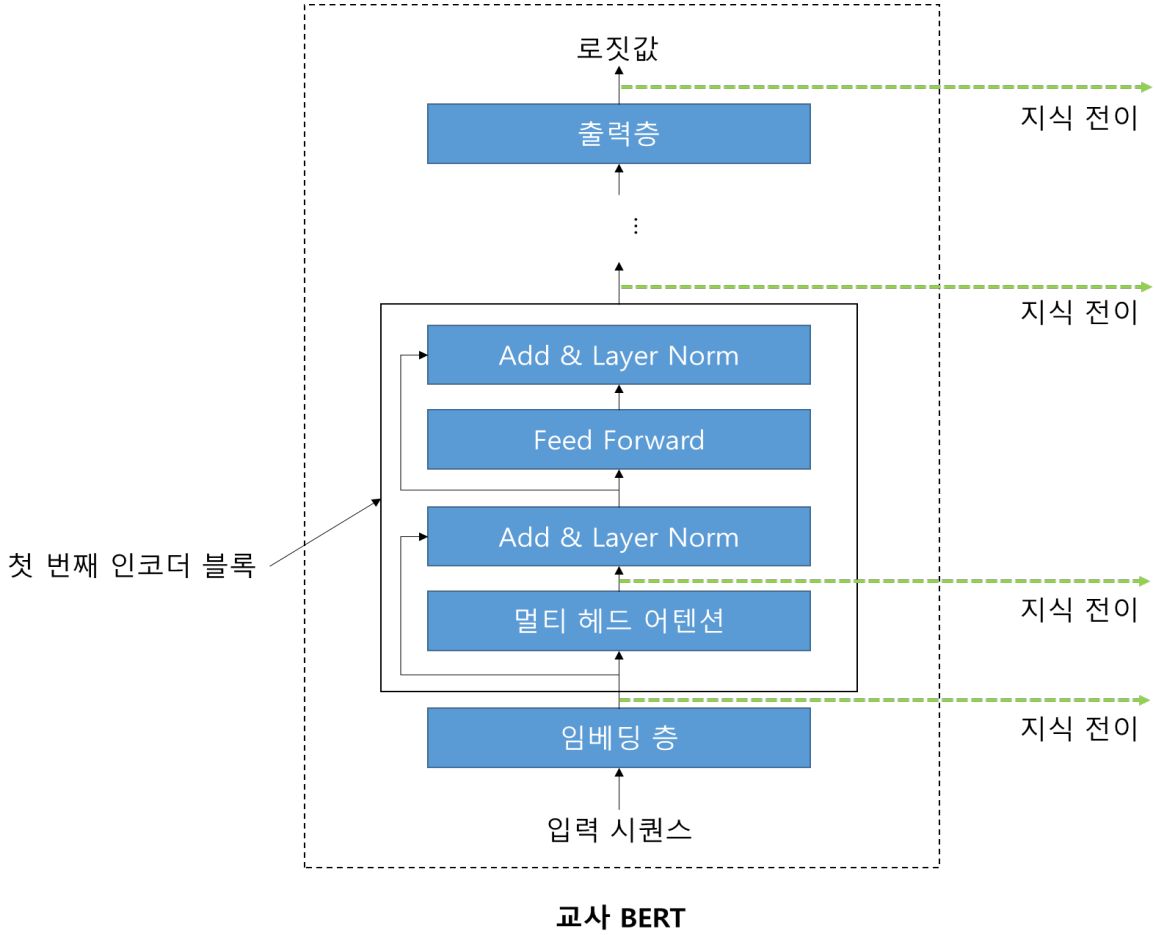
TinyBERT

TinyBERT

- Jiao et al. (2019)
- 지식 증류 방법 사용
- DistilBERT와의 주요 차이 두 가지
  - TinyBERT에서는 마지막 출력층 뿐만 아니라 임베딩 층과 각 인코더 블락에서 출력하는 결과를 이용해서 추가적인 지식 전이 수행
  - 사전 학습 뿐만 아니라 미세 조정 단계, 즉, 다운스트림 작업을 수행하는 과정에서도 지식 증류 수행

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., ... & Liu, Q.(2019). Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

TinyBERT


로짓값
출력층
지식 전이
지식 전이
Add & Layer Norm
Feed Forward
Add & Layer Norm
멀티 헤드 어텐션
첫 번째 인코더 블록
지식 전이
지식 전이
임베딩 층
입력 시퀀스
교사 BERT

TinyBERT

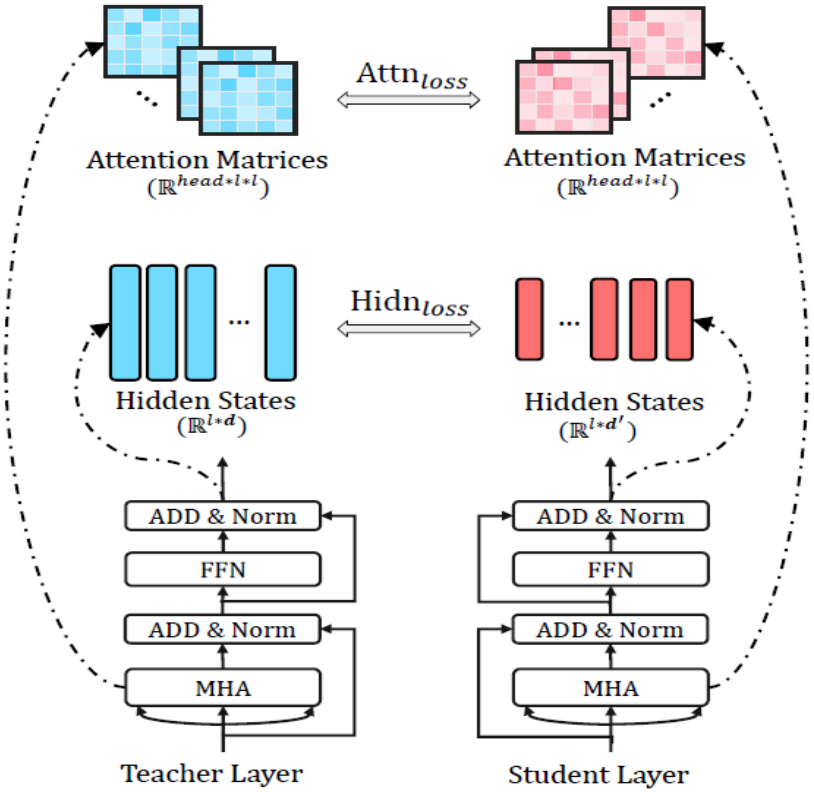
- 인코더 블록의 비용함수
  - 멀티 헤드 어텐션 층

$$L_{attention} = \frac{1}{h}\sum_{i=1}^{h} MSE(A_i^T, A_i^S),$$

$$\text{where A} = \frac{QK^T}{\sqrt{d_k}}$$

  - 은닉 상태 벡터

$$L_{hidden} = MSE(H^S W_h, H^T)$$



TinyBERT

- 임베딩 층의 비용함수

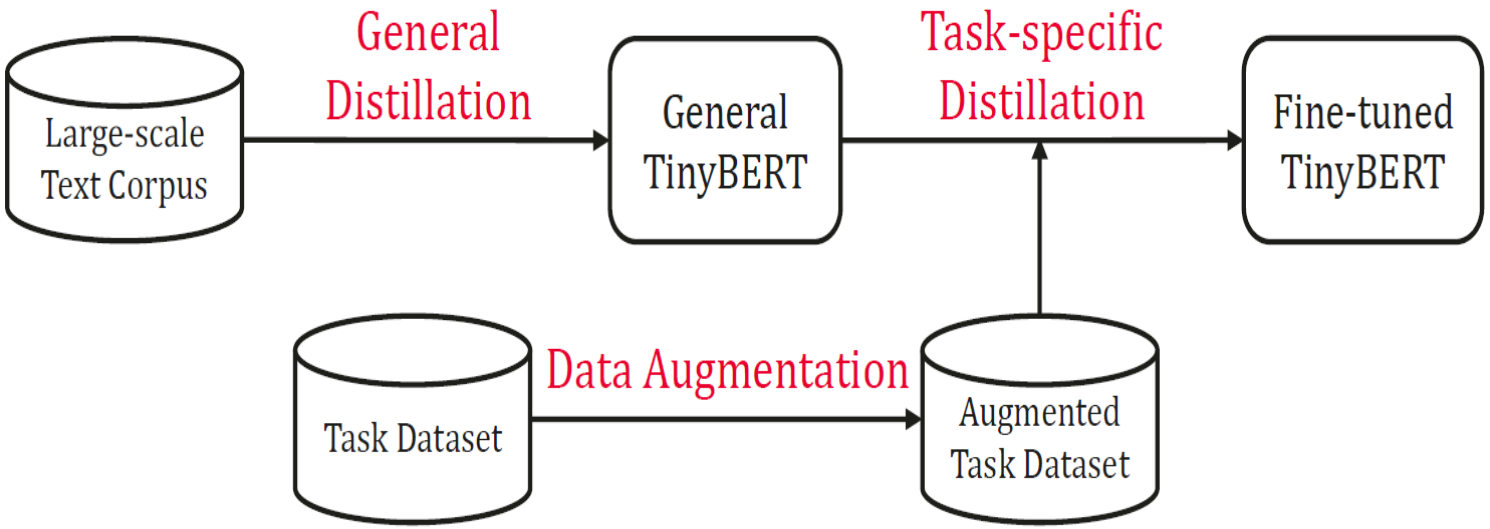
$$L_{embedding} = MSE(E^S W_e, E^T)$$

- 출력층의 비용함수

$$L_{prediction} = CE(z^T/_t, z^S/_t), \qquad t = 1$$

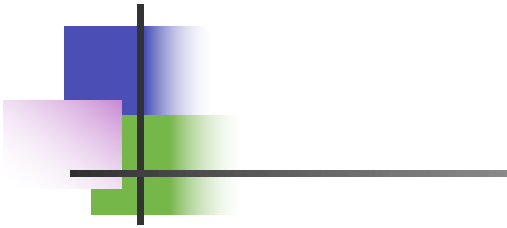
TinyBERT

- 지식 증류를 2 단계에 걸쳐 진행
  - 단계 1: 사전 학습 단계 (General distillation)
  - 단계 2: 다운스트림 작업 단계 (Task-specific distillation)



TinyBERT

- 2 단계 지식 증류
  - 단계 1
    - MLM과 NSP를 통해 사전학습된 BERT 모형을 교사 모형으로 사용하여 학생 모형을 도출 (General TinyBERT)
    - 학습 데이터
      - English Wikipedia와 Toront BookCorpus 데이터셋
  - 단계 2
    - 이 단계에서 사용되는 교사 모형은 특정 다운스트림 태스크에 대해 미세 조정된 BERT
    - 사전 학습 단계에서 얻어진 General TinyBERT를 두 번째 증류 과정에서 사용되는 학생 모형의 시작점으로 사용



Q & A