NLP Trend

현청천

cchyun@gmail.com

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

목차

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

GPT(Generative Pre-Training)

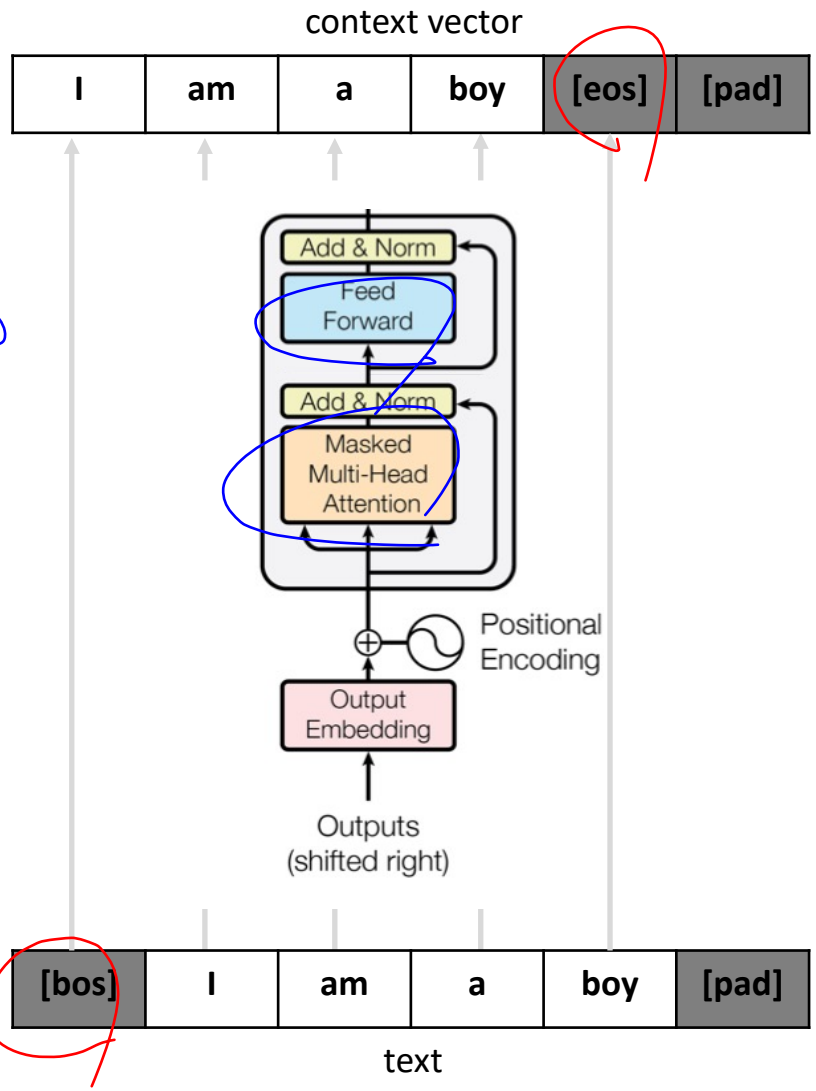
@2021 cchyun@gmail.com., Ltd All Rights Reserved.

GPT

2018년 OpenAI에서 발표
Transformer Decoder를 사용함

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

이전 단어들을 보고
다음 단어를 예측하는 방법으로 LM을 학습 함



@2021 cchyun@gmail.com., Ltd All Rights Reserved.

GPT

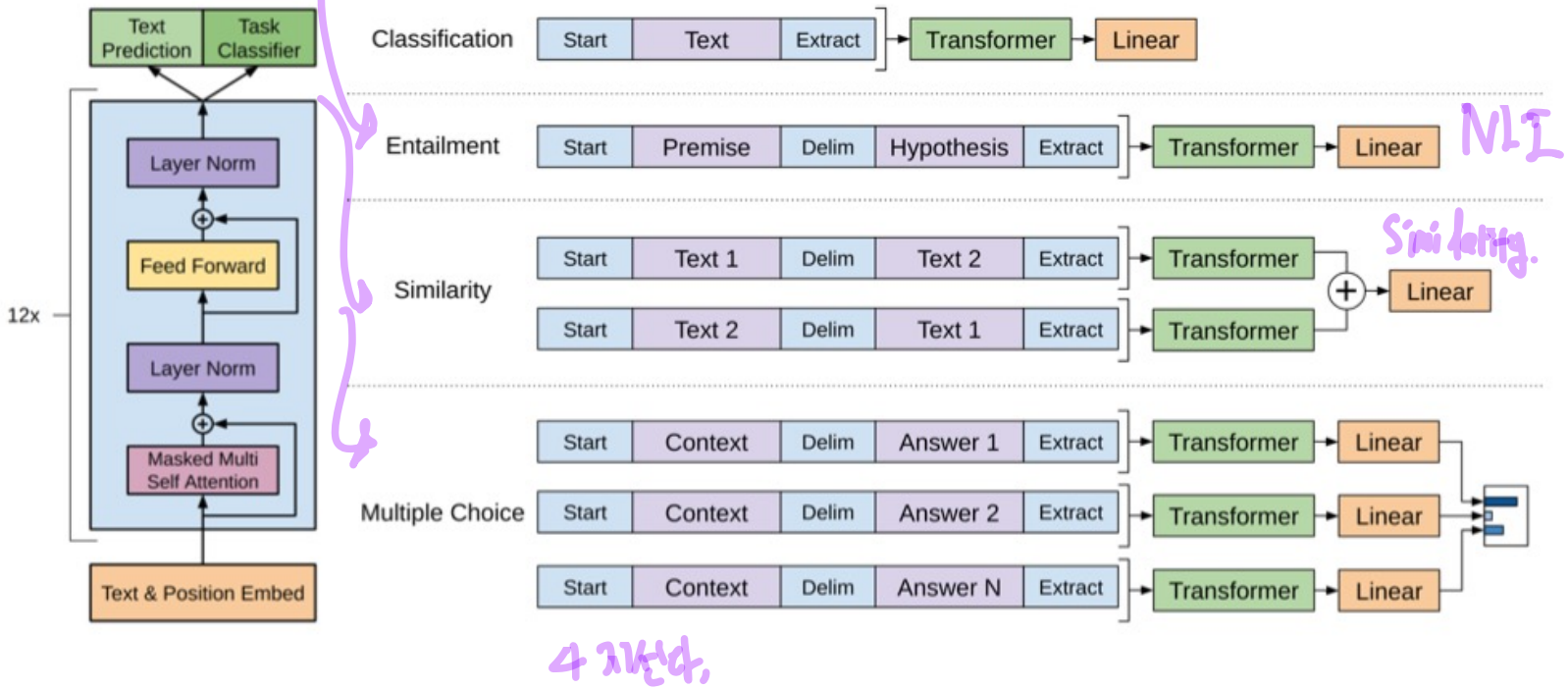
학습된 LM을 여러 Downstream Task에 적용

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y | x^1, \ldots, x^m)$$

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

Task에 적용할 때 목적함수 L_2와 L_1을 동시에 학습할 수도 있음



@2021 cchyun@gmail.com., Ltd All Rights Reserved.

BERT

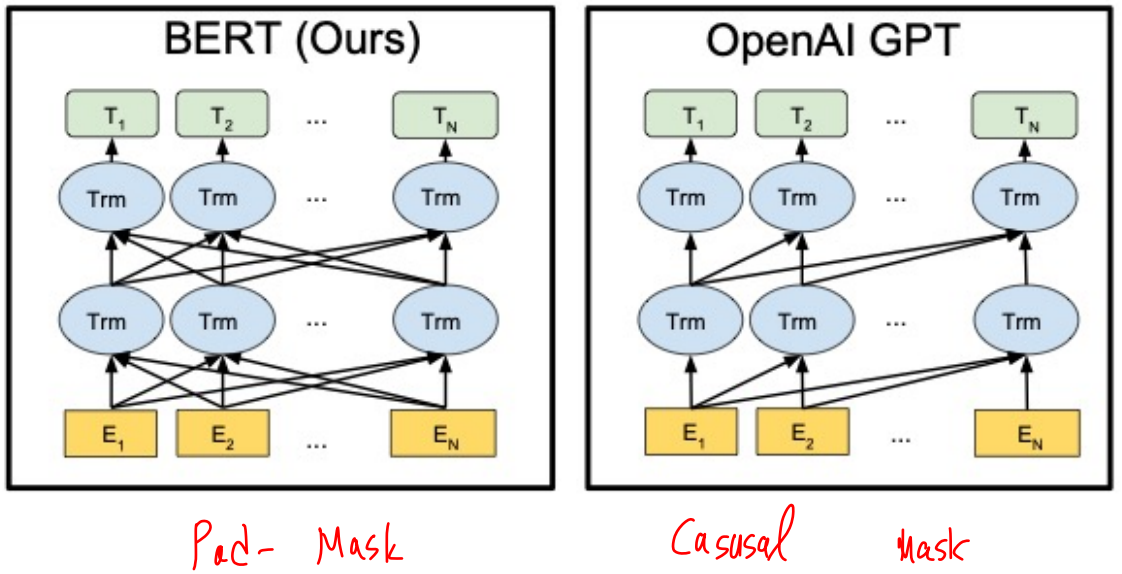
(Bidirectional Encoder Representations from Transformers)

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

BERT vs. GPT

GPT
- Decoder를 사용
- Decoder mask 때문에 이전 단어들(1…n-1)과 현재 단어(n)의 관계를 학습함
  (이전단어들을 이용해 현재 단어를 예측, 단방향)

BERT
- Encoder를 사용
- 전체단어들과 현재 단어의 관계를 학습함
  (전체단어들을 이용해 현재 단어를 예측, 양방향)



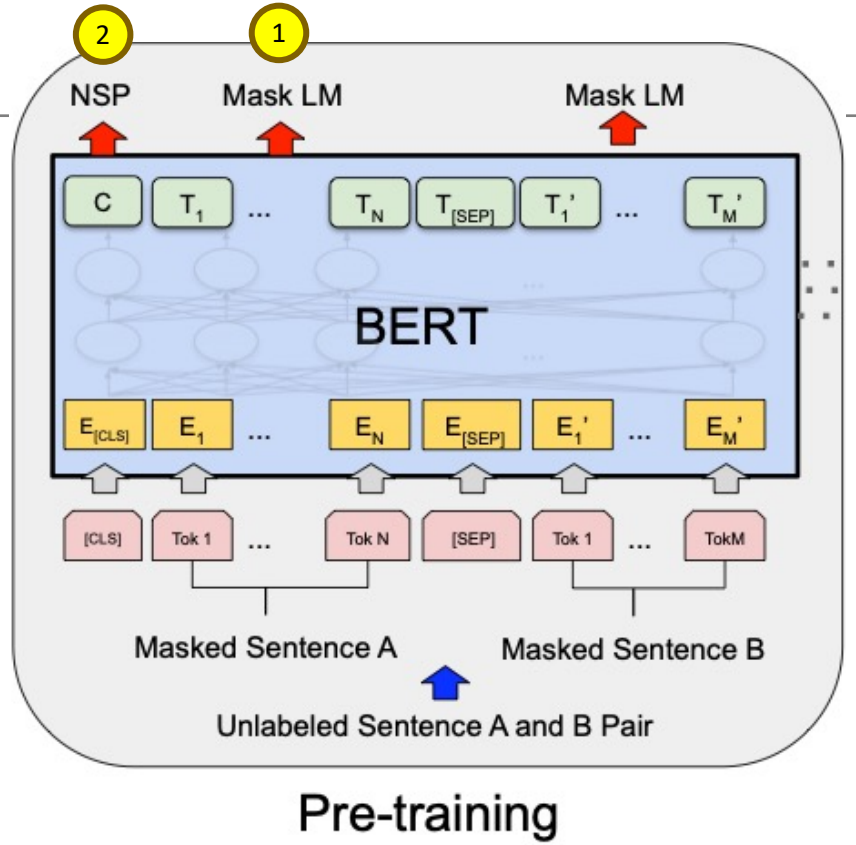
@2021 cchyun@gmail.com., Ltd All Rights Reserved.

BERT

2018년 Google에서 발표
Transformer Encoder를 사용함

MLM (Mask LM) 1
전체 단어의 15%를 [MASK] token으로 바꾼 후 이것을 예측하는 훈련 방식

NSP 2
A 문장과 B 문장이 동일한 단락에 속한 것인지 여부를 훈련하는 방식



Pre-training

Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

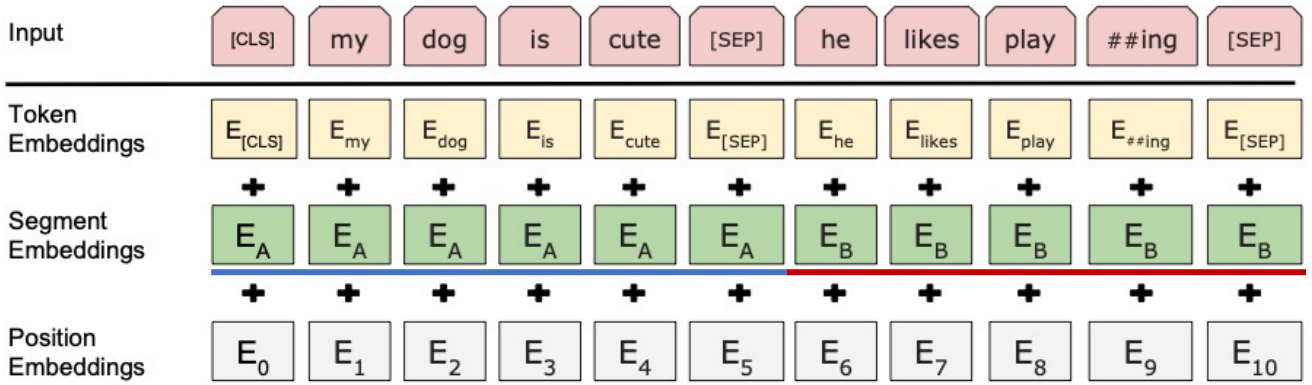
@2021 cchyun@gmail.com., Ltd All Rights Reserved.

BERT-Input

Token Embedding
Word 또는 Sub-word의 embedding 값

Segment Embedding
Sentence A, Sentence B를 구분하기 위한 embedding 값

Position Embedding
단어의 순서를 표현하기 위한 embedding 값



Position Encoding(x)
" Embedding(o)

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

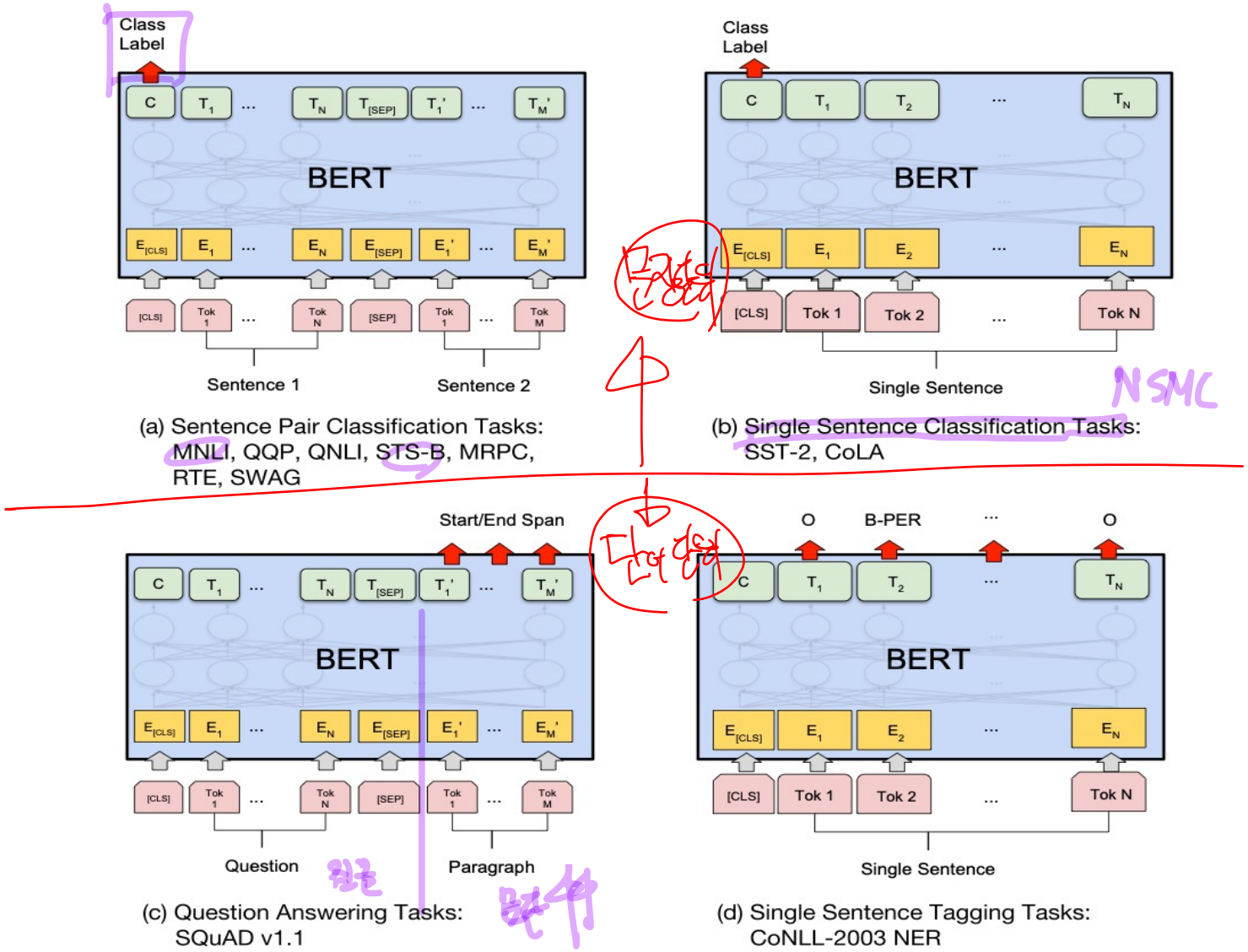
BERT-Downstream Task

Sentence Pair Classification
- input: sentence1, sentence2
- output: [CLS] token 사용해서 예측

Single Sentence Classification
- input: Single sentence
- output: [CLS] token 사용해서 예측

Question Answering
- input: question, paragraph
- output: paragraph내에서 Start/End Span 예측

Single Sentence Tagging Task
- input: Single sentence
- output: Token 별로 Tag 예측



@2021 cchyun@gmail.com., Ltd All Rights Reserved.

GLUE Benchmark

(General Language Understanding Evaluation)

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

GLUE Benchmark

- BERT이후 Transfer Learning을 하는 다양한 모델의 성능을 평가하기 위해 만들어지 dataset

- 11개의 single-sentence 또는 sentence-pair language understanding task
- 기존에 존재하던 dataset 중에서 다양한 dataset 크기, 장르, 난이도를 포함 하도록 선택
- 자연어에 관련해서 모델의 다양한 language understanding 성능을 평가 분석하도록 설계된 데이터
- Leaderboard를 통해 dataset에 대한 모델의 성능을 시각화 해서 표현

(참고) https://huffon.github.io/2019/11/16/glue/

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

GLUE Benchmark

	Rank	Name	Model	URL	Score
	1	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.4
+	2	Alibaba DAMO NLP	StructBERT		90.3
	3	T5 Team - Google	T5		90.3
	4	ERNIE Team - Baidu	ERNIE		90.1
	5	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART		89.9
+	6	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4
	10	Facebook AI	RoBERTa		88.1
+	11	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6
	12	GLUE Human Baselines	GLUE Human Baselines		87.1
	15	Zhuosheng Zhang	SemBERT		82.9
	16	Danqi Chen	SpanBERT (single-task training)		82.8
		XLNet			

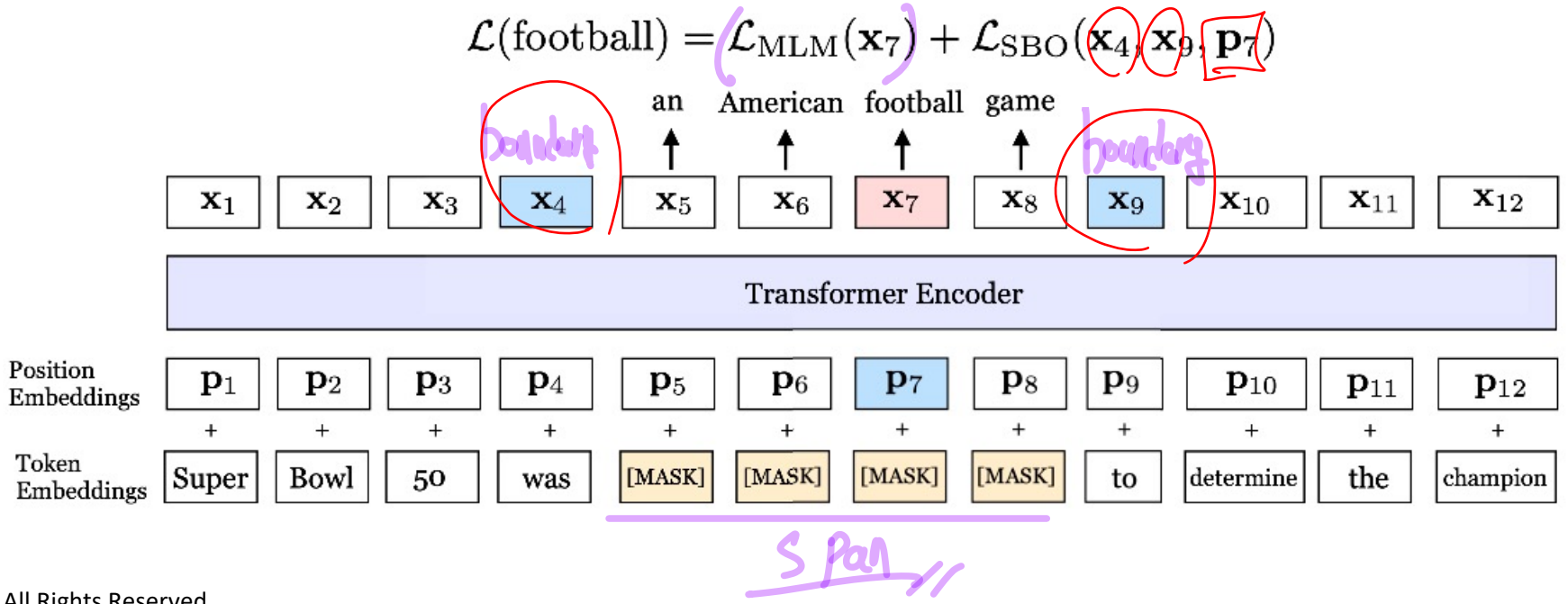
@2021 cchyun@gmail.com., Ltd All Rights Reserved.

SpanBERT

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

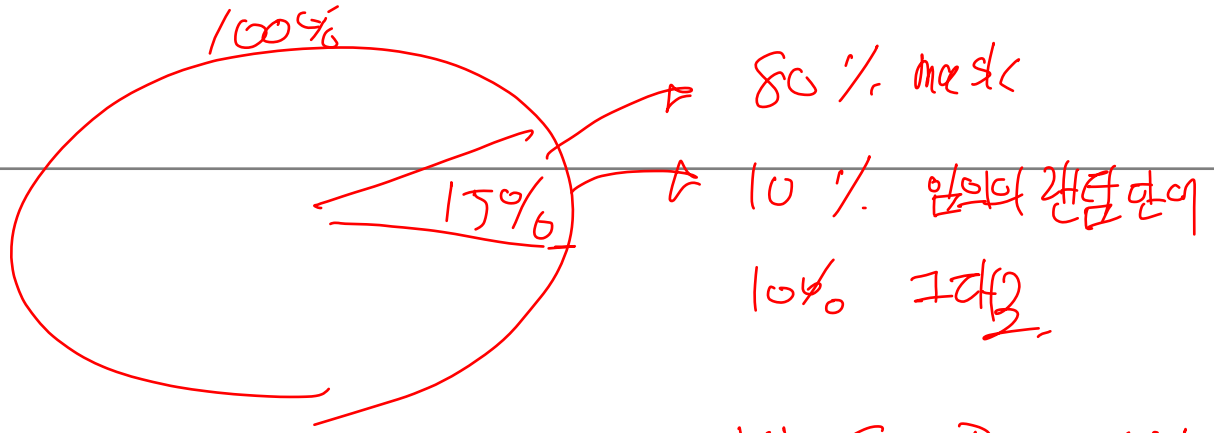
SpanBERT

- 2019년 발표
  - Pretrain시 단일 토큰 마스크가 아닌 **일정 영역(Span)을 Mask한 후 예측**하도록 학습 함
  - MLM 과 **SBO(Span Boundary Object)** 두가지 로스를 줄이도록 학습함
  - 다음문장과의 관계를 예측하는 **NSP를 사용하지 않음**
  - 특히 Span을 예측하는 **Question Answer** 분야에서 좋은 성능을 냄



@2021 cchyun@gmail.com., Ltd All Rights Reserved.

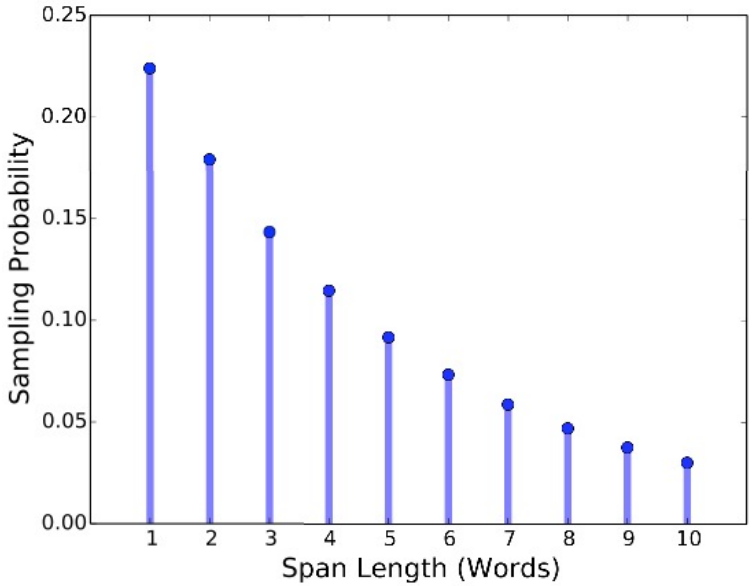
SpanBERT



Span Masking

- BERT와 동일하게 전체 token의 15%만 masking
  - 80% Mask
  - 10% Replace
  - 10% Original
- Span의 길이는 짧은 것은 확률이 높고 긴 것은 확률이 낮은 geometric distribution 사용
- 최대 span길이는 10
- 평균 span 길이는 3.8

ex) 나는 [Mask]를 먹었다
나는 책을 먹었다
나는 밥을 먹었다.

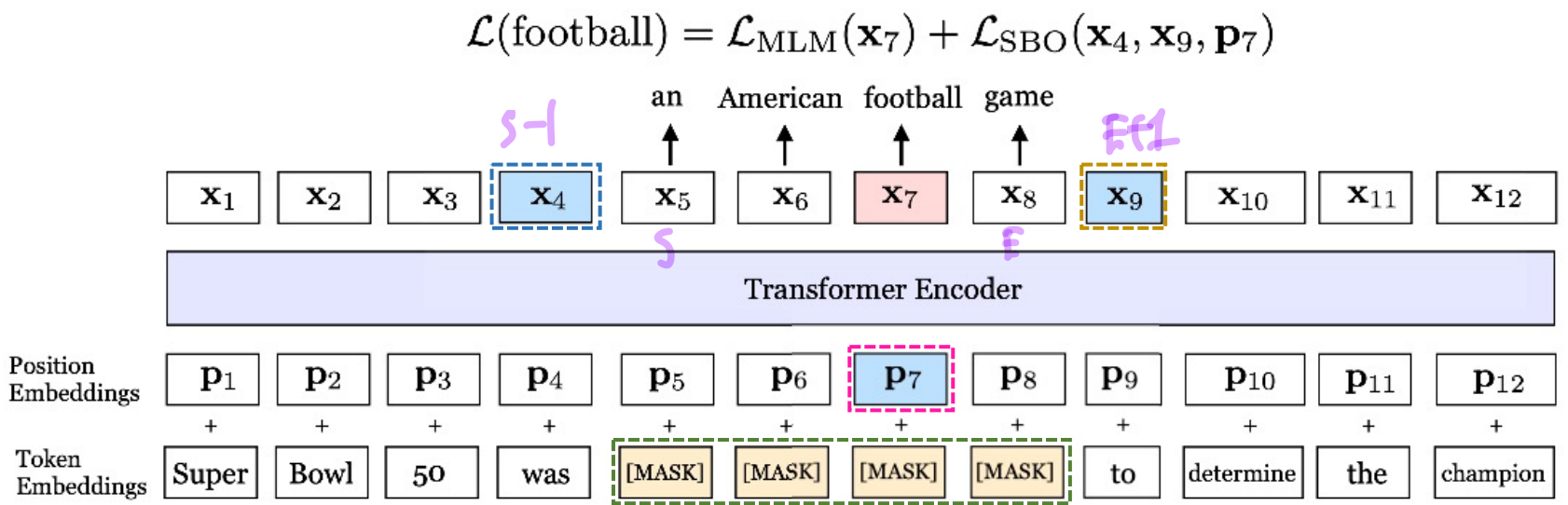


@2021 cchyun@gmail.com., Ltd All Rights Reserved.

SpanBERT

Span Boundary Object

- $y_i = f(x_{s-1}, x_{e+1}, p_i)$
- $h = LayerNorm(GeLU(W_1 \cdot [x_{s-1}; x_{e+1}; p_i]))$
- $f(\cdot) = LayerNorm(GeLU(W_2 \cdot h))$
  - $(x_s, \ldots, x_e)$: masked span, s: start position, e: end position
  - x_{s-1}: start boundary token
  - x_{e+1}: end boundary token
  - p_i: positional embedding
- MLM과 함께 학습



@2021 cchyun@gmail.com., Ltd All Rights Reserved.

SpanBERT

Results

	SQuAD 1.1 EM	SQuAD 1.1 F1	SQuAD 2.0 EM	SQuAD 2.0 F1
Human Perf.	82.3	91.2	86.8	89.4
Google BERT	84.3	91.3	80.0	83.3
Our BERT	86.5	92.6	82.8	85.9
Our BERT-1seq	87.5	93.3	83.8	86.6
SpanBERT	**88.8**	**94.6**	**85.7**	**88.7**

Table 1: Test results on SQuAD 1.1 and SQuAD 2.0.

	NewsQA	TriviaQA	SearchQA	HotpotQA	NaturalQA	(Avg)
Google BERT	68.8	77.5	81.7	78.3	79.9	77.3
Our BERT	71.0	79.0	81.8	80.5	80.5	78.6
Our BERT-1seq	71.9	80.4	84.0	80.3	81.8	79.7
SpanBERT	**73.6**	**83.6**	**84.8**	**83.0**	**82.5**	**81.5**

Table 2: Performance (F1) on the five MRQA extractive question answering tasks.

	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	(Avg)
Google BERT	59.3	**95.2**	88.5/84.3	86.4/88.0	71.2/89.0	86.1/85.7	93.0	71.1	80.4
Our BERT	58.6	93.9	90.1/86.6	88.4/89.1	71.8/89.3	87.2/86.6	93.0	74.7	81.1
Our BERT-1seq	63.5	94.8	**91.2**/87.8	89.0/88.4	**72.1/89.5**	88.0/87.4	93.0	72.1	81.7
SpanBERT	**64.3**	94.8	90.9/**87.9**	**89.9/89.1**	71.9/**89.5**	**88.1/87.7**	**94.3**	**79.0**	**82.8**

Table 4: Test set performance metrics on GLUE tasks. MRPC: F1/accuracy, STS-B: Pearson/Spearmanr correlation, QQP: F1/accuracy, MNLI: matched/mistached accuracies. WNLI (not shown) is always set to majority class (65.1% accuracy) and included in the average.

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

MT-DNN

(Multi-Task Deep Neural Network)

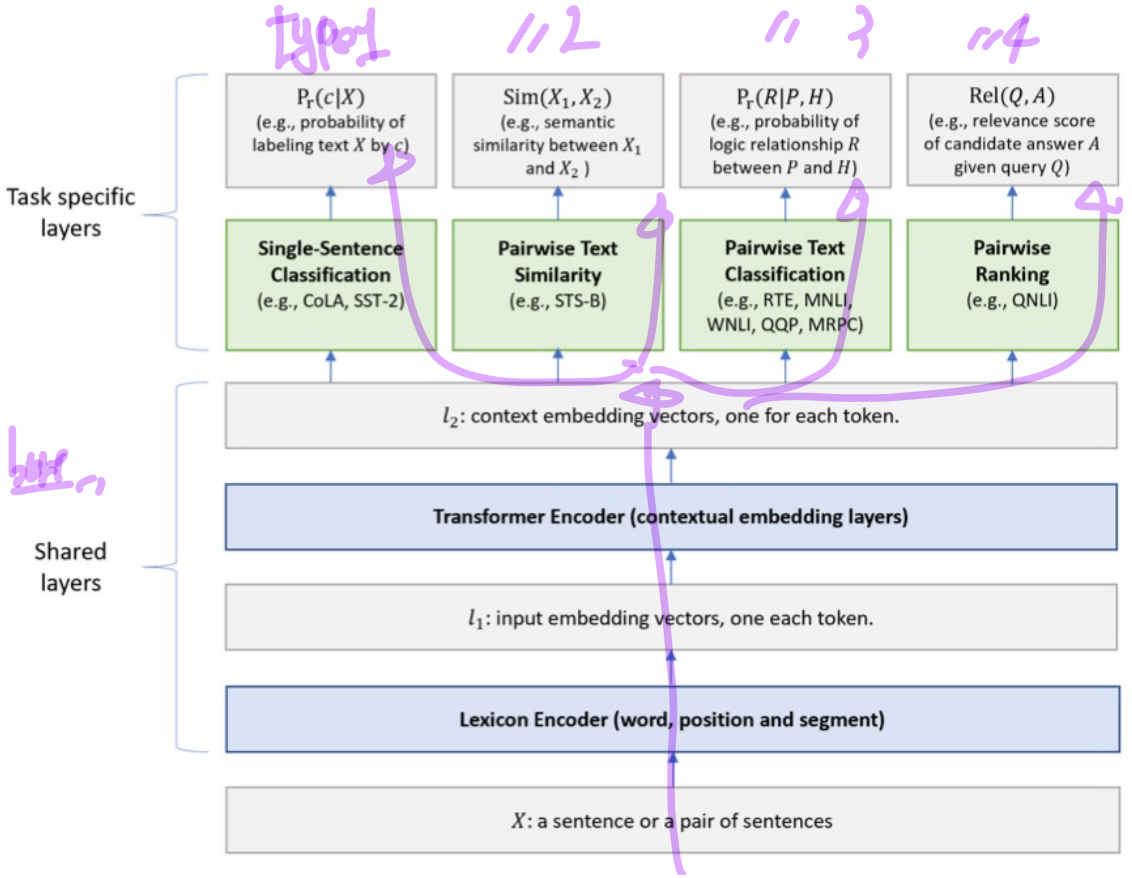
@2021 cchyun@gmail.com., Ltd All Rights Reserved.

MT-DNN

- 2019년 Microsoft에서 발표
- 하나의 BERT 모델이 다양한 Downstream Task를 지원하도록 하여 동시에 학습 함

- **Pre-training stage**
  - Bert 훈련 절차와 동일함
  - 기존 BERT 모델을 사용하여도 됨

- **Multi-task learning stage**
  - GLUE, SNLI, SciTail 등의 데이터를 이용해 추가로 학습 함
  - Downstream task 마다 따로 모델을 만들지 않고 **여러 Task를 한 모델에서 동시에 지원**하도록 함

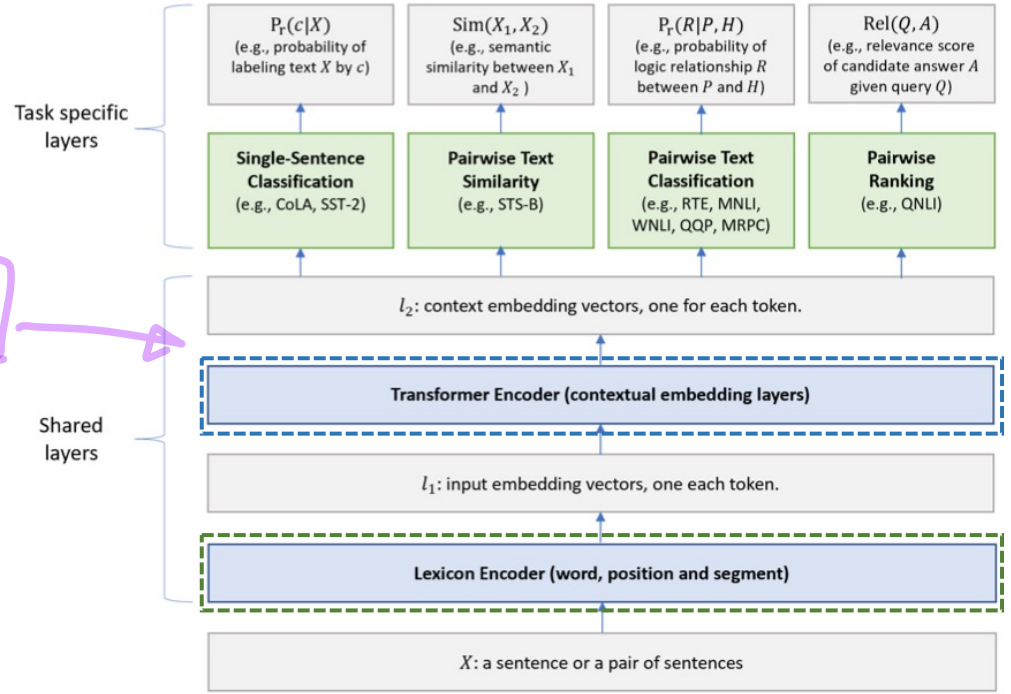


@2021 cchyun@gmail.com., Ltd All Rights Reserved.

MT-DNN

Model

- **Lexicon Encoder (l_1)**
  - $X = \{x_1, \dots, x_m\}$: input
  - x_1: [CLS]
  - Sentence pair인 경우 중간에 [SEP]로 문장 구분
  - Word, position, segment embedding의 합
- **Transformer Encoder (l_2)**
  - Multi-layer bidirectional Transformer encoder
  - $l_1 \in \mathbb{R}^{d \times m}$
  - Multi-task learning 시에 multi-task object와 함께 학습

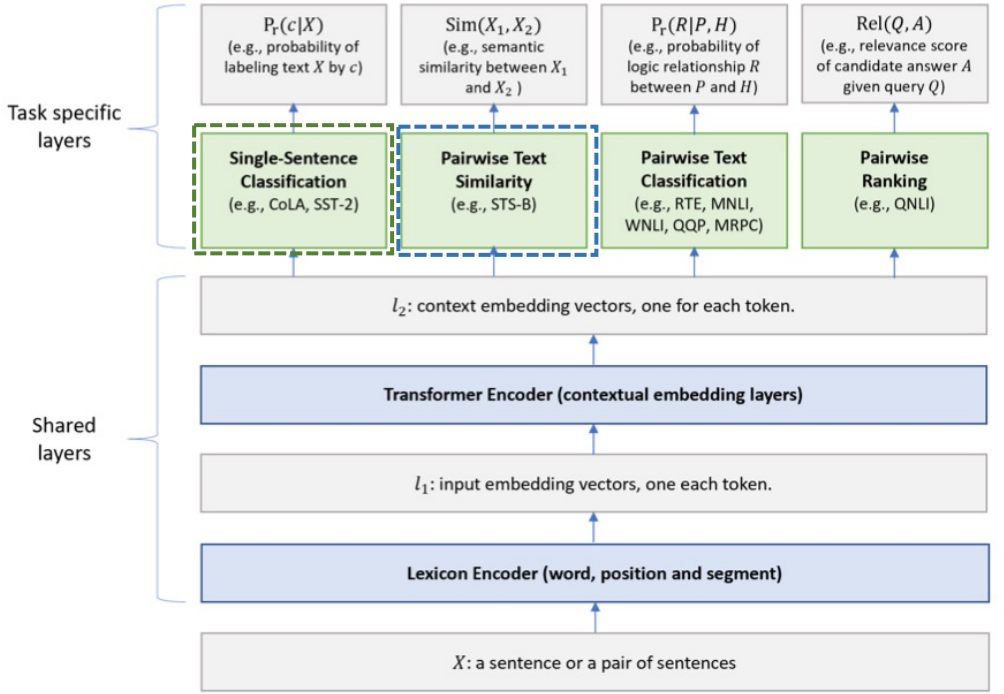


@2021 cchyun@gmail.com., Ltd All Rights Reserved.

MT-DNN

Model

- **Single Sentence Classification Output (SST-2 예)**
  - $P_r(c|X) = softmax(W_{SST}^T \cdot x)$
  - x: l_2의 [CLS] token의 contextual embedding
  - W_{SST}: task specific parameter matrix
- **Text Similarity Output (STS-B 예)**
  - $Sim(X_1, X_2) = W_{STS}^T \cdot x$
  - x: l_2의 [CLS] token의 contextual embedding
  - (X_1, X_2): input sentence pair
  - W_{STS}: task specific parameter matrix
  - $Sim(X_1, X_2)$: range $(-\infty, \infty)$

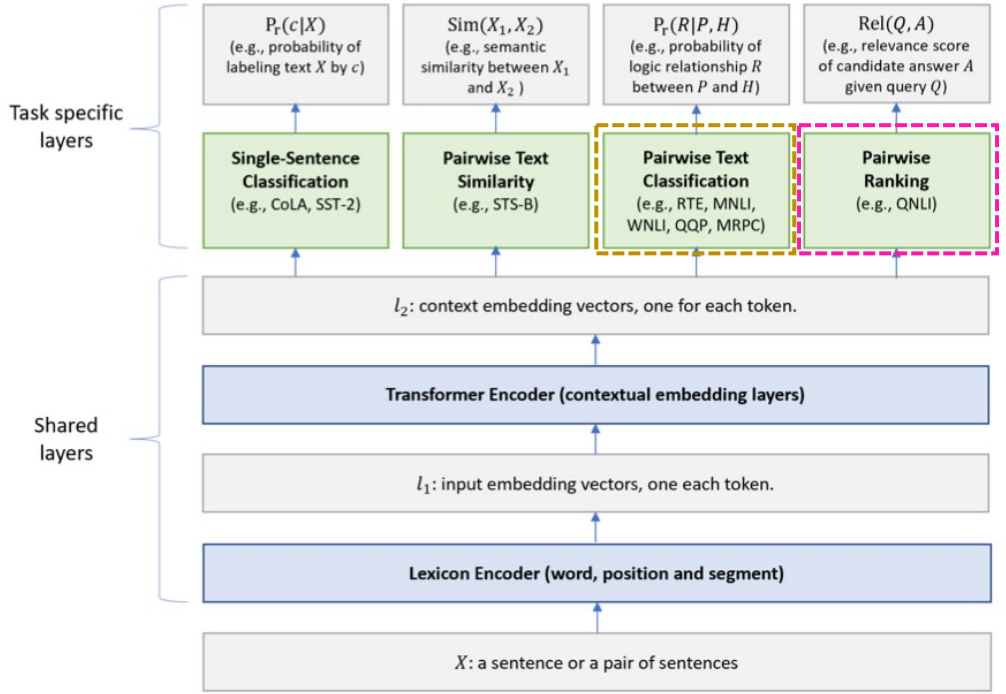


@2021 cchyun@gmail.com., Ltd All Rights Reserved.

MT-DNN

Model

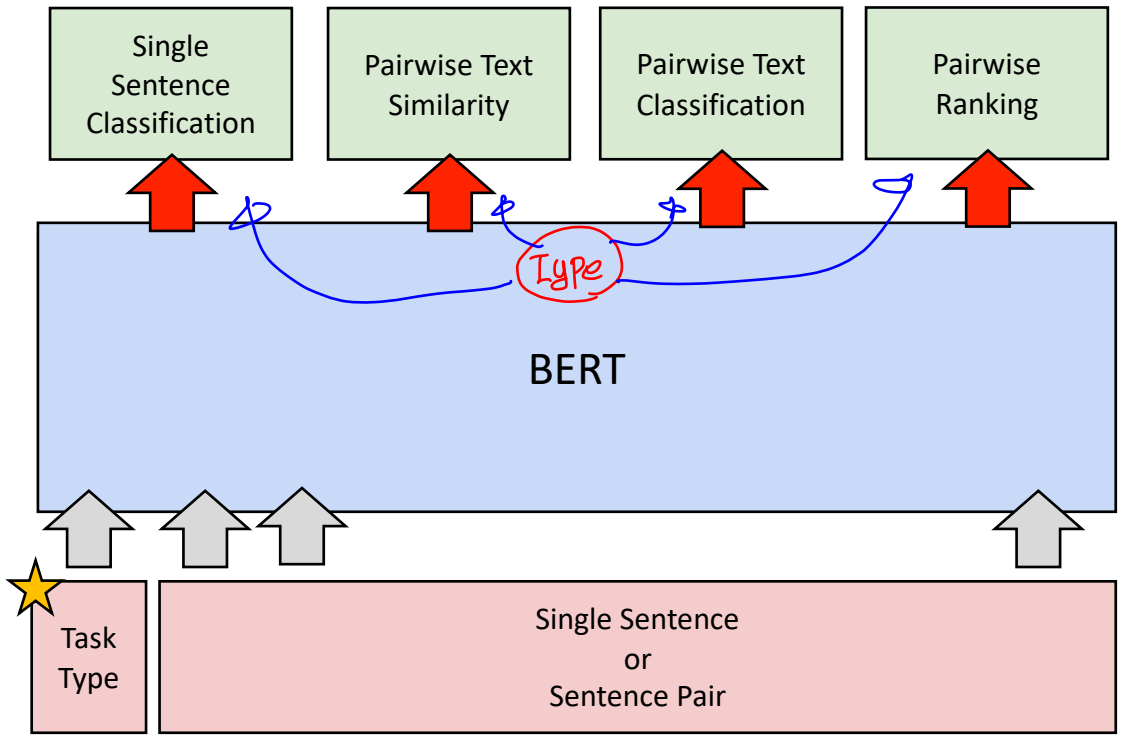
- **Pairwise Text Classification Output (LNI 예)**
  - $P = \{p_1, \dots, p_m\}$: premise
  - $H = \{h_1, \dots, h_m\}$: hypothesis
  - SAN(stochastic answer network)의 계산 방법 사용
- **Relevance Ranking Output (QNLI 예)**
  - $Rel(Q, A) = g(W_{QNLI}^T \cdot x)$
  - x: l_2의 [CLS] token의 contextual embedding
  - Candidate answer의 relevance score를 계산 후 ranking



@2021 cchyun@gmail.com., Ltd All Rights Reserved.

MT-DNN

Algorithm of multi-task learning

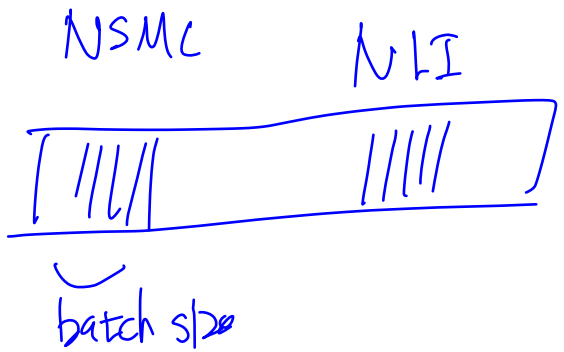


@2021 cchyun@gmail.com., Ltd All Rights Reserved.

MT-DNN

Algorithm of multi-task learning

1. Dataset 별로 data를 mini-batch 단위로 분할: D_t
2. Dataset merge: $D = D_1 \cup D_2 \ldots \cup D_T$
3. Shuffle D
4. D의 Mini-batch b_t 단위로 실행
5. b_t의 task에 따라서 loss 계산: $L(\Theta)$
   - Classification
   - Regression
   - ranking
6. Loss에 따른 gradient 계산: $\nabla(\Theta)$
7. Update model: $\Theta = \Theta - \epsilon\nabla(\Theta)$



Algorithm 1: Training a MT-DNN model.

Initialize model parameters Θ randomly.
Pre-train the shared layers (i.e., the lexicon encoder and the transformer encoder).
Set the max number of epoch: $epoch_{max}$.
//Prepare the data for T tasks.
for t *in* $1, 2, ..., T$ **do**
  Pack the dataset t into mini-batch: D_t.
end
for $epoch$ *in* $1, 2, ..., epoch_{max}$ **do**
  1. Merge all the datasets:
     $D = D_1 \cup D_2 ... \cup D_T$
  2. Shuffle D
  for b_t *in* D **do**
    //b_t is a mini-batch of task t.
    3. Compute loss : $L(\Theta)$
       $L(\Theta)$ = Eq. 6 for classification
       $L(\Theta)$ = Eq. 7 for regression
       $L(\Theta)$ = Eq. 8 for ranking
    4. Compute gradient: $\nabla(\Theta)$
    5. Update model: $\Theta = \Theta - \epsilon\nabla(\Theta)$
  end
end

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

MT-DNN

Results

- 학습 데이터가 적은 경우에 효과가 좋음
  - **RTE** vs MNLI, **MRPC** vs QQP
- Fine-tune 없이도 ($\text{MT-DNN}_{\text{no-fine-tune}}$) $\text{BERT}_{\text{LARGE}}$보다 성능이 좋음 (CoLA 제외)
  - CoLA는 dataset이 작고 다른 task와 형태가 다름
- Fine-tune 후 CoLA의 성능이 62.5로 $\text{BERT}_{\text{LARGE}}$의 60.5에 비해 좋아 짐

Model	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	WNLI 634	AX	Score
BiLSTM+ELMo+Attn [1]	36.0	90.4	84.9/77.9	75.1/73.3	64.8/84.7	76.4/76.1	-	56.8	65.1	26.5	70.5
Singletask Pretrain Transformer [2]	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1/81.4	-	56.0	53.4	29.8	72.8
GPT on STILTs [3]	47.2	93.1	87.7/83.7	85.3/84.8	70.1/88.1	80.8/80.6	-	69.1	65.1	29.4	76.9
$\text{BERT}^{4}_{\text{LARGE}}$	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
$\text{MT-DNN}_{\text{no-fine-tune}}$	58.9	94.6	**90.1/86.4**	89.5/88.8	**72.7/89.6**	86.5/85.8	**93.1**	79.1	65.1	39.4	81.7
MT-DNN	**62.5**	**95.6**	**91.1/88.2**	**89.5/88.8**	**72.7/89.6**	**86.7/86.0**	**93.1**	**81.4**	65.1	**40.3**	**82.7**
Human Performance	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	87.1

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

XLNet

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

XLNet

- 2019년 Google에서 발표
- Transformer Decoder 사용
- AR(auto-regressive, ex:GPT) 과 AE(auto-encoder, ex:BERT)의 장점을 합한 모델

Permutation Language Model

단어의 순서를 섞은 후 섞은 순서대로 단어를 예측하도록 하는 학습 방법

Transformer-XL

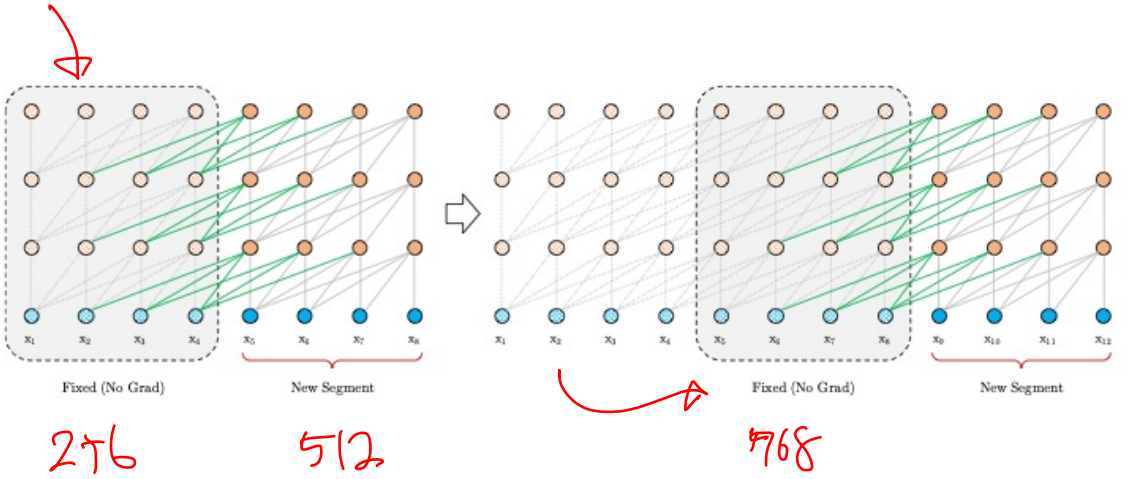
- Segment Recurrence
- Relative position embedding

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

XLNet

Transformer XL

- **Segment Recurrence**
  - 기존 모델은 최대길이 초과하는 문장의 경우는 일부를 버려야 함
  - XLNET은 긴 문장을 여러 Segment로 분리하고 **이전 segment의 hidden state을 저장 후 다음 segment에서 사용하는 방식**으로 recurrence 하게 처리 함



- **Relative Position Embedding**
  - 기존 모델은 입력 시 position embedding을 추가 함
  - XLNET은 입력 position embedding을 제거 하고, **query와 key의 상대적인 거리를 position embedding** 으로 사용
  - 일정 거리 이상은 최대 또는 최소 값으로 사용

	I	am	a	boy
I	0	1	2	3
am	-1	0	1	2
a	-2	-1	0	1
boy	-3	-2	-1	0

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

XLNet

Two-Stream Self Attention

1 **Content Stream**

단어 자신의 정보를 포함

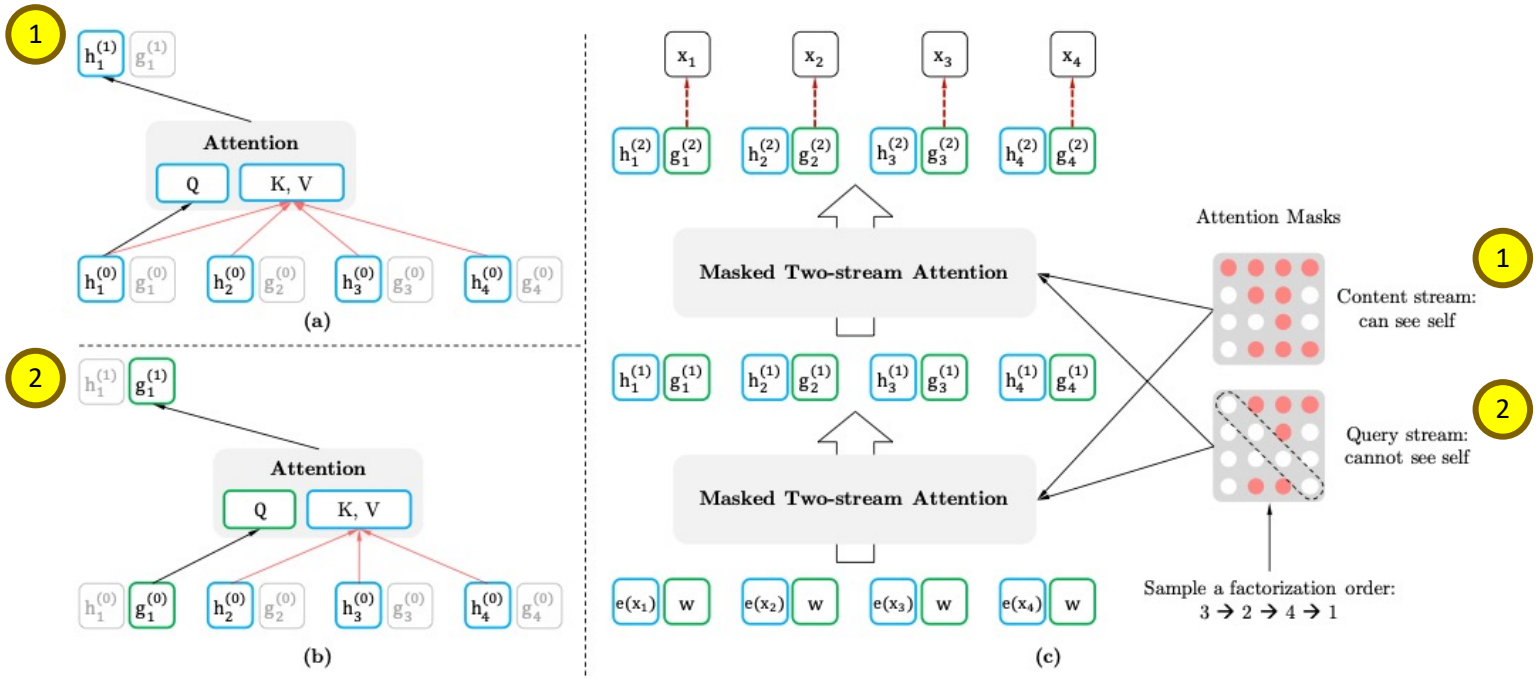
$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = h_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}; \theta)$$

2 **Query Stream**

단어 자신의 정보를 제외 함

단어 예측을 위해 사용

$$g_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = g_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta)$$



단어를 예측하기 위해서 예측할 단어의 정보가 없는 g와
단어의 정보를 포함한 h가
둘다 필요 함

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

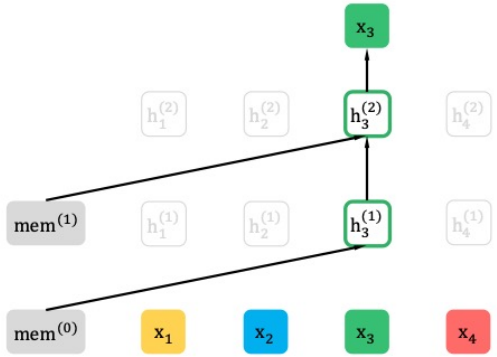
XLNet

Permutation LM

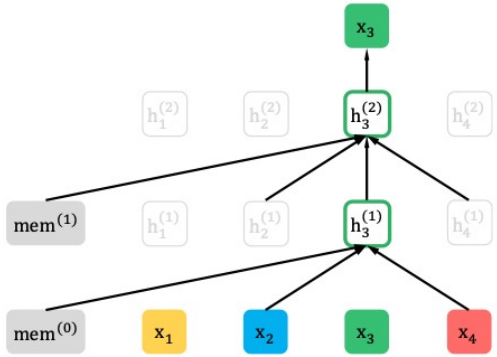
I(1), am(2), a(3), boy(4)

3 → 2 → 4 → 1
입력: mem
출력: a(3)을 예측

mem	I(1)	am(2)	a(3)	boy(4)



Factorization order: 3 → 2 → 4 → 1



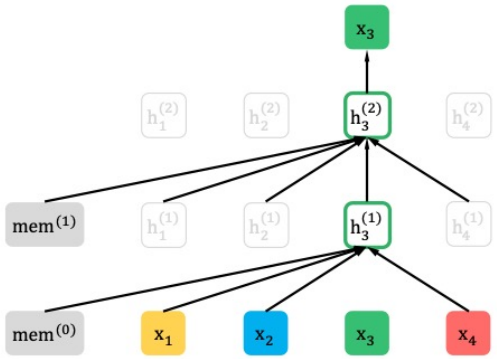
Factorization order: 2 → 4 → 3 → 1

2 → 4 → 3 → 1
입력: mem, am(2), boy(4)
출력: a(3)을 예측

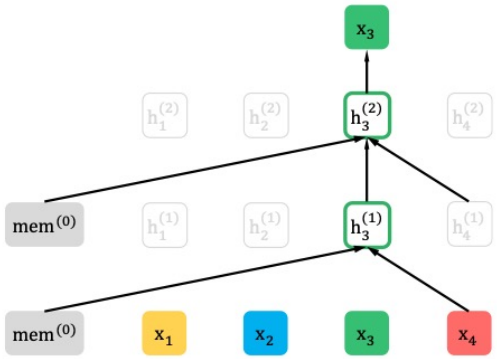
mem	I(1)	am(2)	a(3)	boy(4)

1 → 4 → 2 → 3
입력: mem, boy(4), am(2), I(1)
출력: a(3)을 예측

mem	I(1)	am(2)	a(3)	boy(4)



Factorization order: 1 → 4 → 2 → 3



Factorization order: 4 → 3 → 1 → 2

4 → 3 → 1 → 2
입력: mem, boy(4)
출력: a(3)을 예측

mem	I(1)	am(2)	a(3)	boy(4)

더 다양한 단어 들의 관계를 볼 수 있음

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

XLNet

Pre-train Dataset

- Book Corpus, English Wikipedia (13GB): BERT pre-train dataset
- Giga5 (16GB): news text
- 짧거나 낮은 품질의 데이터는 필터링
  - ClueWeb 201-B (19GB)
  - Common Crawl (78GB)
- SentencePiece Tokenizer (23.89B tokens)
  - Wikipedia (2.78B tokens)
  - BooksCorpus (1.09B tokens)
  - Giga5 (4.75B tokens)
  - ClueWeb (4.30B tokens)
  - Common Crawl (19.97B tokens)

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

XLNet

Results

- 9개중 7개에서 SOTA 기록
- Benchmark에서 가장 많이 사용되는 MNLI-m (2.0), MNLI-mm (1.8)에서 성능 개선 됨
- Single-task에서 BERT와 비교해서 RTE (13.4), MNLI (3.2), CoLA (3.0), SST-2 (2.4), STS-B (1.8) 등의 성능 개선이 됨

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	WNLI
Single-task single models on dev									
BERT [2]	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-
XLNet	**89.8/-**	**93.9**	**91.8**	**83.8**	**95.6**	**89.2**	**63.6**	**91.8**	-
Single-task single models on test									
BERT [10]	86.7/85.9	91.1	89.3	70.1	94.9	89.3	60.5	87.6	65.1
Multi-task ensembles on test (from leaderboard as of June 19, 2019)									
Snorkel* [29]	87.6/87.2	93.9	89.9	80.9	96.2	91.5	63.8	90.1	65.1
ALICE*	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8
MT-DNN* [18]	87.9/87.4	96.0	89.9	**86.3**	96.5	92.7	68.4	91.1	89.0
XLNet*	**90.2/89.7**†	**98.6**†	90.3†	**86.3**	**96.8**†	**93.0**	67.8	**91.6**	**90.4**

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

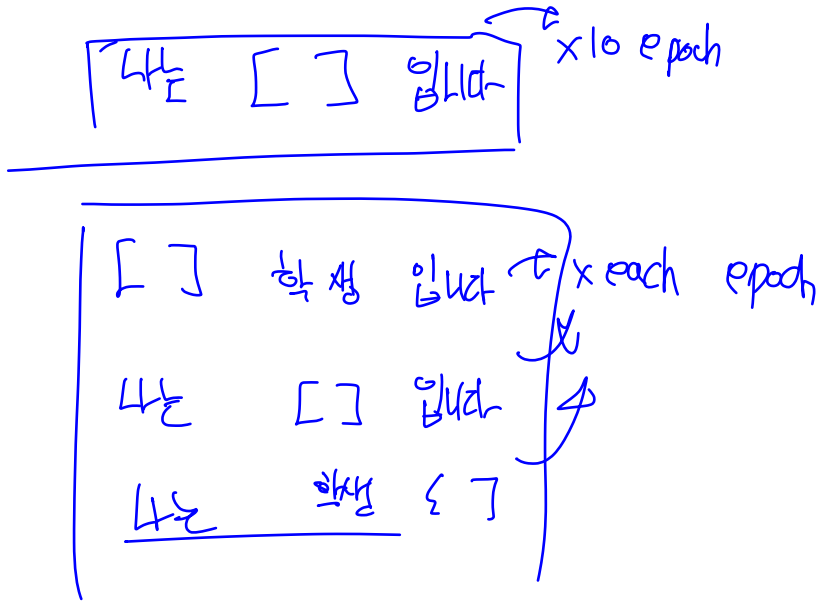
RoBERTa

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

RoBERTa

- 2019년 Facebook에서 발표

- BERT에 비해 10배 정도 많은 데이터를 사용해서 학습 함 (160G)
- Dynamic Mask를 사용함 (BERT: 최초에 Mask 한 후 계속 사용)
- BERT에 비해 8배 큰 배치를 사용함 (2K)
- NSP를 사용하지 않음
- 기타 학습 Parameter를 수정해서 더욱 최적화 함



@2021 cchyun@gmail.com., Ltd All Rights Reserved.

RoBERTa

Experimental Setup

- **Implementation**
  - 다음 3개를 제외한 hyperparameters는 BERT와 동일함
    - Learning rate: 1e-4 → **setting에 따라 수정**
    - Warmup steps: 10,000 → **setting에 따라 수정**
    - Adam optimizer: $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-6, weight\_decay = 0.01$ → $\boldsymbol{\beta_2 = 0.98}$
  - 짧은 문장을 학습하지 않고 **full-length(~~152~~ 512) 문장**만 학습 함 (BERT는 최초 90%는 128 length로 학습)

- **Data (160GB)**
  - Book Corpus & English Wikipedia (16GB): BERT와 동일
  - CC-NEWS (76GB): Common Crawled News Dataset
  - Open Web Text (38GB): 소셜 뉴스 웹 Reddit의 text (좋아요 3개 이상)
  - Stories (31BG): Common Crawled Story Dataset

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

RoBERTa

Training Procedure Analysis

- **Static vs. Dynamic Masking**
  - Static Masking (BERT)
    - Preprocessing시에 랜덤 masking파일을 10개 생성
    - epoch마다 차례대로 학습
    - 40 epoch 학습 시 4번 동일 파일 학습
  - **Dynamic Masking (RoBERTa)**
    - epoch마다 매번 masking 함
    - 약간 성능이 좋음

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

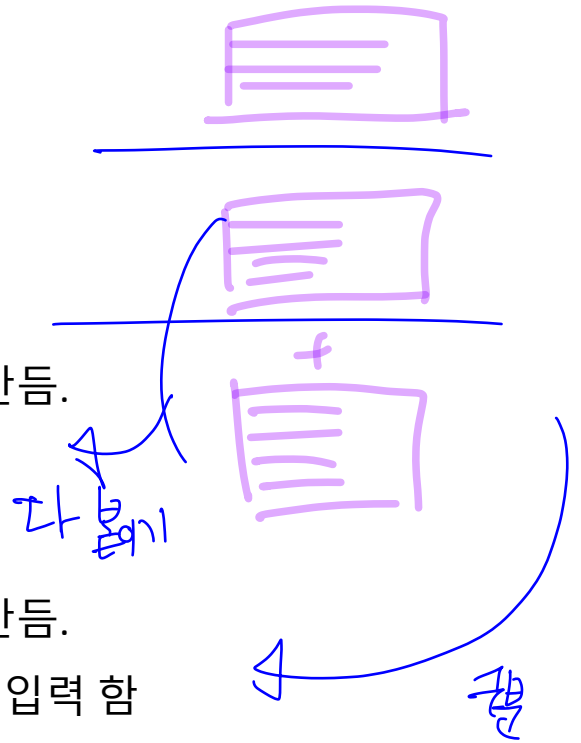
@2021 cchyun@gmail.com., Ltd All Rights Reserved.

RoBERTa

Training Procedure Analysis

- **Model Input Format and Next Sentence Prediction**
  - SEGMENT-PAIR + NSP
    - BERT의 학습방법
    - 두개의 segment를 입력. (Segment는 여러 sentence로 구성 됨)
  - SENTENCE-PAIR + NSP
    - 두개의 sentence를 입력.
    - Batch_size를 키움 (입력 크기가 512 보다 많이 작음)
  - FULL-SENTENCES
    - 입력 token 개수가 512보다 클 때 까지 sentence를 추가해서 만듬.
    - Document가 끝나면 다음 document에서 가져 옴.
  - **DOC-SENTENCES (가장 성능이 좋음)**
    - 입력 token 개수가 512보다 클 때 까지 sentence를 추가해서 만듬.
    - Document가 끝나면 다음 document에서 가져 오지 않고 그냥 입력 함

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
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
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7



@2021 cchyun@gmail.com., Ltd All Rights Reserved.

RoBERTa

Training Procedure Analysis

- **Training with large batches**
  - **256 batch x 1M step**
    - 총 학습 256,000,000 (256 X 1,000,000)
  - **2K batch x 125K step**
    - 총 학습 256,000,000 (2,048 X 125,000)
    - **가장 성능이 좋음**
  - **8K batch x 31K step**
    - 총 학습 256,000,000 (8,192 X 31,250)

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

- **Text Encoding (BPE)**
  - BERT: 30K vocab
  - **RoBERTa: 50K vocab**

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

RoBERTa

Result

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

- 학습 데이터크기의 중요성을 보여 줌
- 500K 학습을 진행해도 overfit 되지 않음

즉, 데이터가 많을수록 좋다.

- RTE, SST, MRPC는 pre-trained RoBERTa보다 MNLI를 학습한 초기 값으로 initialize한 후 학습하면 성능이 개선됨

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

ALBERT

(A LITE BERT)

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ALBERT

- 2019년 google에서 발표

- BERT와 비교해서 parameter 수를 엄청나게 줄이면서도 성능은 더 좋아 짐

  ($BERT_{LARGE}$ (334M) / $ALBERT_{LARGE}$(18M) = 18 / 1)

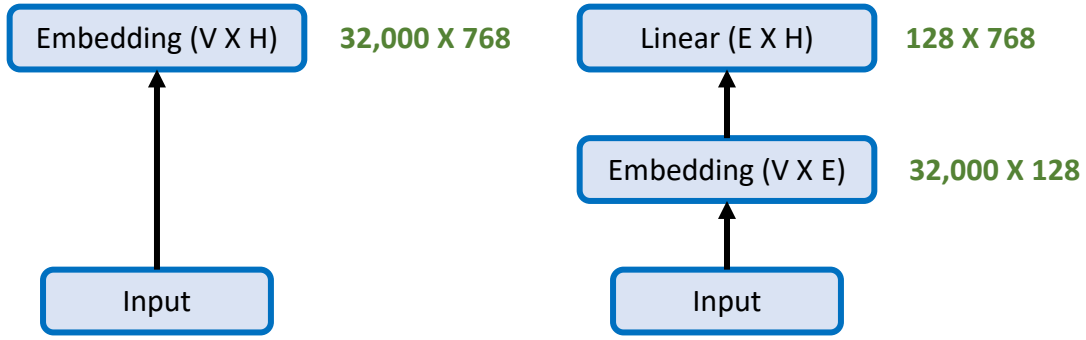
Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
	xlarge	1270M	24	2048	2048	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	59M	24	2048	128	True
	xxlarge	233M	12	4096	128	True

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ALBERT

Model Architecture Choice

- **Factorized embedding parameterization**
  - $BERT_{BASE}$
    - Vocab: 32,000, Hidden: 768
    - Parameter: 24,576,000 (32,000 * 768)
  - $ALBERT_{BASE}$
    - Vocab: 32,000, Embedding: 128, Hidden: 768
    - Parameter: 4,194,304 (32,000 * 128 + 128 * 768)
  - All-shared 중에서 128이 가장 성능이 좋음



BERT

ALBERT

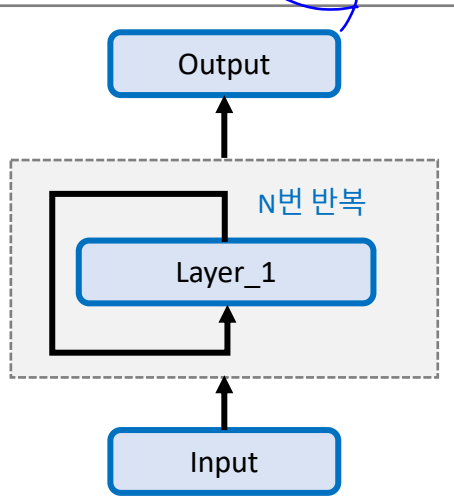
Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	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

@2021 cchyun@gmail.com., Ltd All Rig

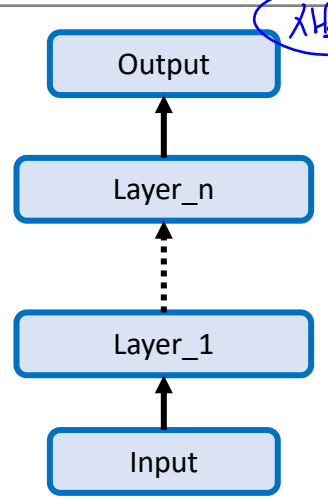
ALBERT

Model Architecture Choice

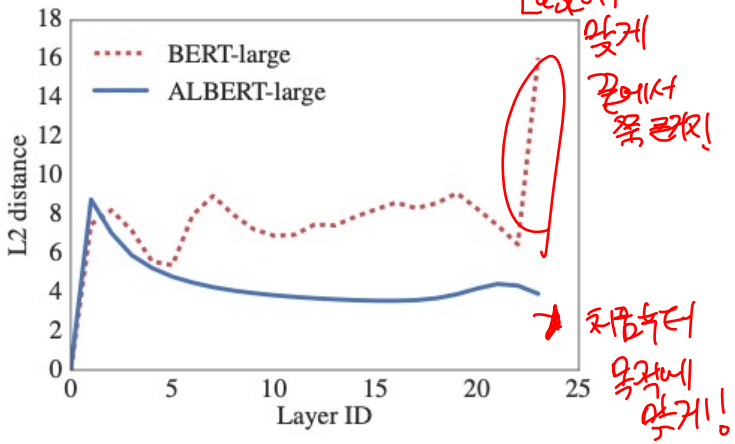
- **Cross-layer parameter sharing**
  - Layer간에 Parameter 공유
  - Layer 1개를 n번 반복 실행
  - 각 layer의 입력과 출력을 비교 했을 때 BERT는 layer간에 편차가 많고 ALBERT는 편차가 적음

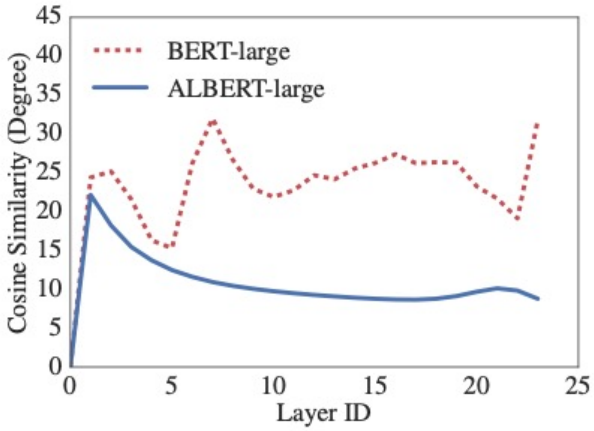


ALBERT



BERT





@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ALBERT

Model Architecture Choice

- **Cross-layer parameter sharing**
  - 128-all-shared (-1.5), 768-all-shared (-2.5)
  - Attention share는 성능에 큰 영향을 주지 않음
  - FFN을 share할 때 성능이 떨어 짐

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

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ALBERT

Model Architecture Choice

- **Sentence Order Prediction**
  - **NSP**
    - NSP는 False의 경우 다른 단락의 두 문장을 비교
    - 두 문장 간의 주제를 비교하는 문제로 MLM에 비해서 문제가 너무 쉬워 학습이 잘 안됨
  - **SOP**
    - SOP는 False일 경우 동일한 단락의 연속된 문장의 순서를 바꿈
    - 두 문장 간의 의미를 비교하는 문제로 NSP에 비해서 어려운 문제 임

NSP

sentence_a = "신용 **대출**을 신청 하고 싶어요"
sentence_b = "**대출** 관련 서류를 준비해 오셨어요?"
Label: IsNext

sentence_a = "신용 **대출**을 신청 하고 싶어요"
sentence_b = "어떤 팀이 올해 **프로야구** 우승 했나요?"
Label: NotNext

SOP

sentence_a = "신용 **대출**을 신청 하고 싶어요"
sentence_b = "**대출** 관련 서류를 준비해 오셨어요?"
Label: IsNext

sentence_a = "**대출** 관련 서류를 준비해 오셨어요?"
sentence_b = "신용 **대출**을 신청 하고 싶어요"
Label: NotNext

- NSP로 학습한 경우 SOP를 전혀 예측 못함
- SOP로 학습한 경우 NSP를 어느정도 예측
- SOP는 SQuAD1.1 (+1%), SQuAD2.0 (2%), RACE (1.7%) 정도 성능 개선 효과가 있음

	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**

@2021 cchyun@gmail.com., Ltc

ALBERT

Result

Number of layers	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
1	18M	31.1/22.9	50.1/50.1	66.4	80.8	40.1	52.9
3	18M	79.8/69.7	64.4/61.7	77.7	86.7	54.0	71.2
6	18M	86.4/78.4	73.8/71.1	81.2	88.9	60.9	77.2
12	18M	89.8/83.3	80.7/77.9	83.3	91.7	66.7	81.5
24	18M	**90.3/83.3**	**81.8/79.0**	83.3	91.5	**68.7**	**82.1**
48	18M	90.0/83.1	**81.8/78.9**	**83.4**	**91.9**	66.9	81.8

$ALBERT_{LARGE}$ Layer 수 별 성능 비교

- 24 layer에서 최대 성능, 48 layer에서 성능 감소

Hidden size	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
1024	18M	79.8/69.7	64.4/61.7	77.7	86.7	54.0	71.2
2048	59M	83.3/74.1	69.1/66.6	79.7	88.6	58.2	74.6
4096	223M	**85.0/76.4**	**71.0/68.1**	**80.3**	**90.4**	**60.4**	**76.3**
6144	497M	84.7/75.8	67.8/65.4	78.1	89.1	56.0	74.0

3-layer $ALBERT_{LARGE}$ hidden 별 성능 비교

- 4096에서 최대 성능 6144에서 성능 감소

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

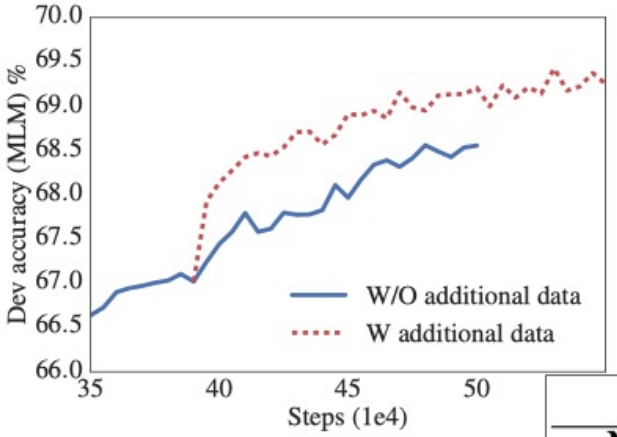
ALBERT

Result

Number of layers	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
12	94.0/88.1	88.3/85.3	87.8	95.4	82.5	88.7
24	94.1/88.3	88.1/85.1	88.0	95.2	82.3	88.7

$ALBERT_{XXLARGE}$ Layer 수 별 성능 비교

- 12 layer, 24 layer 성능차이가 거의 없음



	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
No additional data	**89.3/82.3**	**80.0/77.1**	81.6	90.3	64.0	80.1
With additional data	88.8/81.7	79.1/76.3	**82.4**	**92.8**	**66.0**	**80.8**

Wikipedia & Book Corpus 외에 추가 데이터 학습

- SQuAD를 제외한 나머지 task에서 성능 향상이 됨
- SQuAD가 Wikipedia를 기반으로 작성됨 다른 데이터로부터 negative effect 발생

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ALBERT

Result

	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
With dropout	94.7/89.2	89.6/86.9	90.0	96.3	85.7	90.4
Without dropout	**94.8/89.5**	**89.9/87.2**	**90.4**	**96.5**	**86.1**	**90.7**

Layer 개수 반복했다가.

Dropout 영향 비교

- Dropout을 사용하지 않은 경우가 성능이 좋음

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**

GLUE Benchmark

- 89.4로 SOTA 기록

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

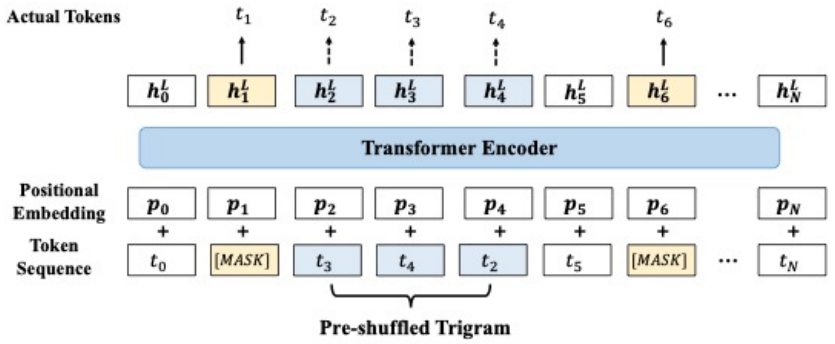
StructBERT

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

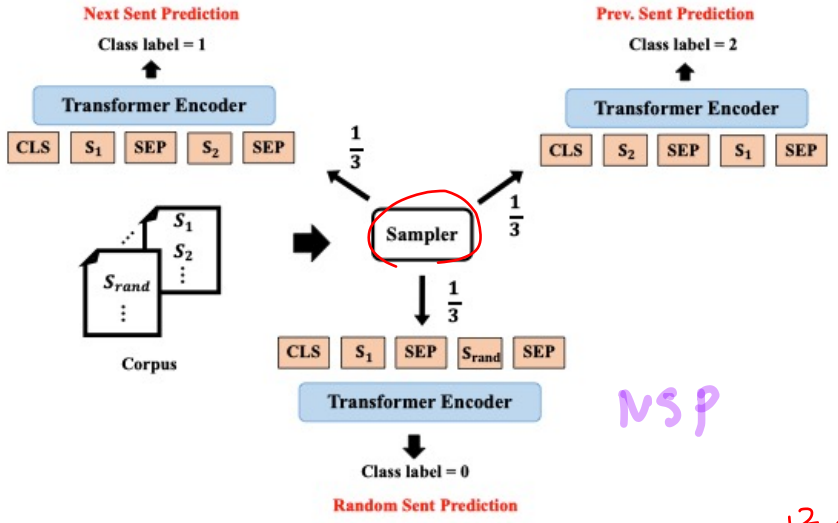
StructBERT

- 2019년 alibaba에서 발표

- MLM 이외에 단어의 순서를 섞은 후 **순서를 복원하는 Word Structure Objective** 추가
- NSP를 사용하지 않고 두개의 문장을 입력 후 **순서 관계를 예측하는 Sentence Structure Objective** 추가



(a) Word Structural Objective

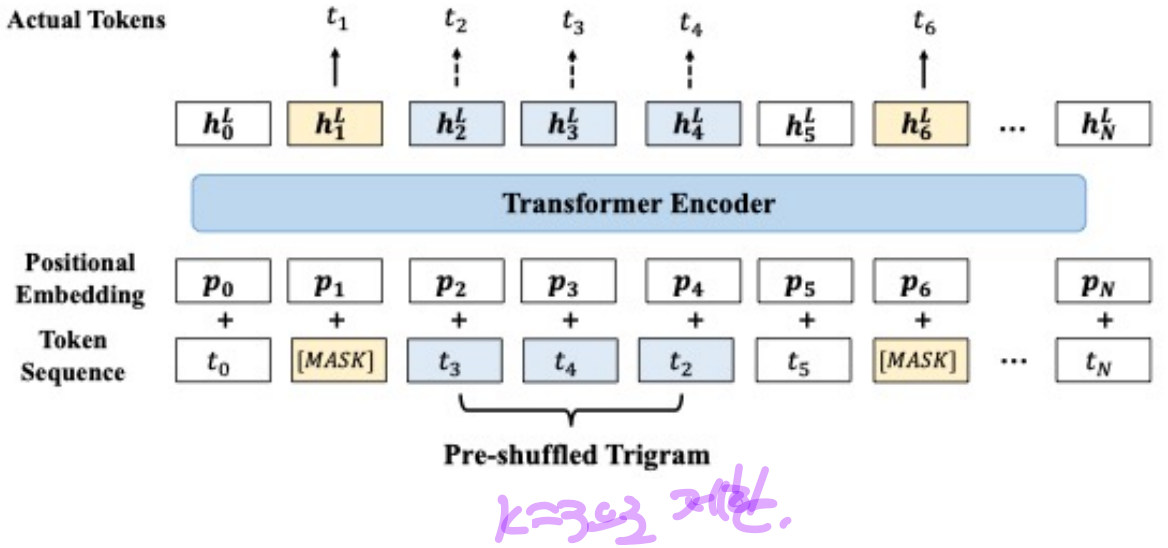


(b) Sentence Structural Objective

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

StructBERT

Word Structure Objective



- Mask 하지 않는 token을 랜덤 하게 선택 한 후 순서를 섞음
- token의 출력을 이용해 원래 위치의 token을 예측
- MLM과 동일한 비중으로 pre-train

$$\arg\max_{\theta} \sum \log P(\text{pos}_1 = t_1, \text{pos}_2 = t_2, \ldots, \text{pos}_K = t_K | t_1, t_2, \ldots, t_K, \theta),$$

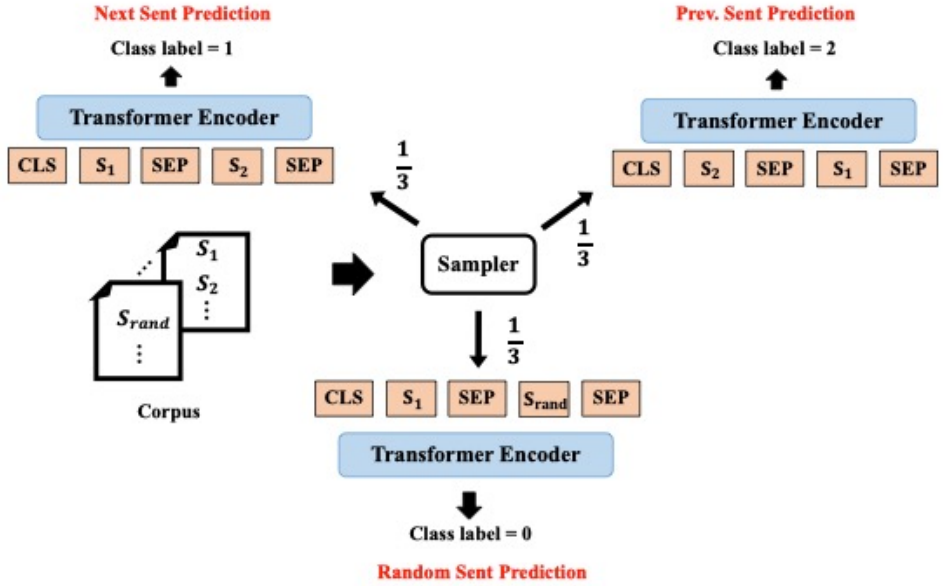
- 순서를 섞을 단어개수 K는 3으로 함
- Word Structure Objective를 제거하면 성능 감소가 발생 함. 특히 CoLA의 경우는 -4.1% 성능 감소

Task	CoLA (Acc)	SST-2 (Acc)	MNLI (Acc)	SNLI (Acc)	QQP (Acc)	SQuAD (F1)
StructBERTBase	**85.8**	**92.9**	**85.4**	91.5	**91.1**	**90.6**
-word structure	81.7	92.7	85.2	**91.6**	90.7	90.3
-sentence structure	84.9	**92.9**	84.1	91.1	90.5	89.1
BERTBase	80.9	92.7	84.1	91.3	90.4	88.5

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

StructBERT

Sentence Structure Objective



- NSP는 쉬운 task로 97%~98%는 쉽게 달성 함 (사용하지 않음)
- Sentences S_1, S_2, S_{rand} 중 두개를 입력으로 함 (S_{rand}는 다른 document)
  - Next sentence: S_1 다음에 S_2가 오는 경우
  - Prev sentence: S_2 다음에 S_1이 오는 경우
  - Random sentence: S_1 다음에 S_{rand}가 오는 경우
  - 각각 1/3 확률로 생성
- Sentence Structure Objective를 제거하면 MNLI, SNLI, QQP, SQuAD 등 **두 문장을 입력하는 task**에서 성능 감소

Task	CoLA (Acc)	SST-2 (Acc)	MNLI (Acc)	SNLI (Acc)	QQP (Acc)	SQuAD (F1)
StructBERTBase	**85.8**	**92.9**	**85.4**	91.5	**91.1**	**90.6**
-word structure	81.7	92.7	85.2	**91.6**	90.7	90.3
-sentence structure	84.9	**92.9**	84.1	91.1	90.5	89.1
BERTBase	80.9	92.7	84.1	91.3	90.4	88.5

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

StructBERT

Pre-training Setup

- Data: Wikipedia, Book Corpus (BERT와 동일한 비교적 **적은 데이터**)
- MLM: 15% (80%: Mask, 10%: Replace, 10%: Original)
- Word Shuffle: 5% 3-gram
- Adam: $\beta_1: 0.9, \beta_2: 0.999, L2\ weight\ decay: 0.01$
- Learning rage: 1e-4, warmup: 10% of total step
- Dropout: 0.1
- Activation: gelu

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

StructBERT

Result

- MRPC/RTE/STS-B는 MNLI와 비슷함. MNLI을 학습한 후 MRPC/RTE/STS-B를 학습 함 (two-stage transfer learning)
- RTE, QNLI, QQP, SST-2, CoLA, MNLI 는 자기 자신의 데이터로만 학습
- CoLA의 경우는 $BERT_{LARGE}$에 비해 $StructBERT_{LARGE}$가 4.8% 성능 증가

System	CoLA 8.5k	SST-2 67k	MRPC 3.5k	STS-B 5.7k	QQP 363k	MNLI 392k	QNLI 108k	RTE 2.5k	WNLI 634	AX	Average
Human Baseline	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	
BERTLarge [6]	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
BERT on STILTs [19]	62.1	94.3	90.2/86.6	88.7/88.3	71.9/89.4	86.4/85.6	92.7	80.1	65.1	28.3	82.0
SpanBERT [12]	64.3	94.8	90.9/87.9	89.9/89.1	71.9/89.5	88.1/87.7	94.3	79.0	65.1	45.1	82.8
Snorkel MeTaL [22]	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9	87.6/87.2	93.9	80.9	65.1	39.9	83.2
MT-DNN++ [14]	65.4	95.6	91.1/88.2	89.6/89.0	72.7/89.6	87.9/87.4	95.8	85.1	65.1	41.9	83.8
MT-DNN ensemble [14]	65.4	96.5	92.2/89.5	89.6/89.0	73.7/89.9	87.9/87.4	96.0	85.7	65.1	42.8	84.2
StructBERTBase	57.2	94.7	89.9/86.1	88.5/87.6	72.0/89.6	85.5/84.6	92.6	76.9	65.1	39.0	80.9
StructBERTLarge	65.3	95.2	92.0/89.3	90.3/89.4	74.1/90.5	88.0/87.7	95.7	83.1	65.1	43.6	83.9
StructBERTLarge ensemble	68.6	95.2	92.5/90.1	91.1/90.6	74.4/90.7	88.2/87.9	95.7	83.1	65.1	43.9	84.5
XLNet ensemble [32]	67.8	96.8	93.0/90.7	91.6/91.1	74.2/90.3	90.2/89.8	98.6	86.3	90.4	47.5	88.4
RoBERTa ensemble [15]	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8/90.2	98.9	88.2	89.0	48.7	88.5
Adv-RoBERTa ensemble	68.0	96.8	93.1/90.8	92.4/92.2	**74.8/90.3**	**91.1/90.7**	98.8	**88.7**	89.0	**50.1**	88.8
StructBERTRoBERTa ensemble	**69.2**	**97.1**	**93.6/91.5**	**92.8/92.4**	**74.4/90.7**	90.7/90.3	**99.2**	87.3	**89.7**	47.8	**89.0**

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

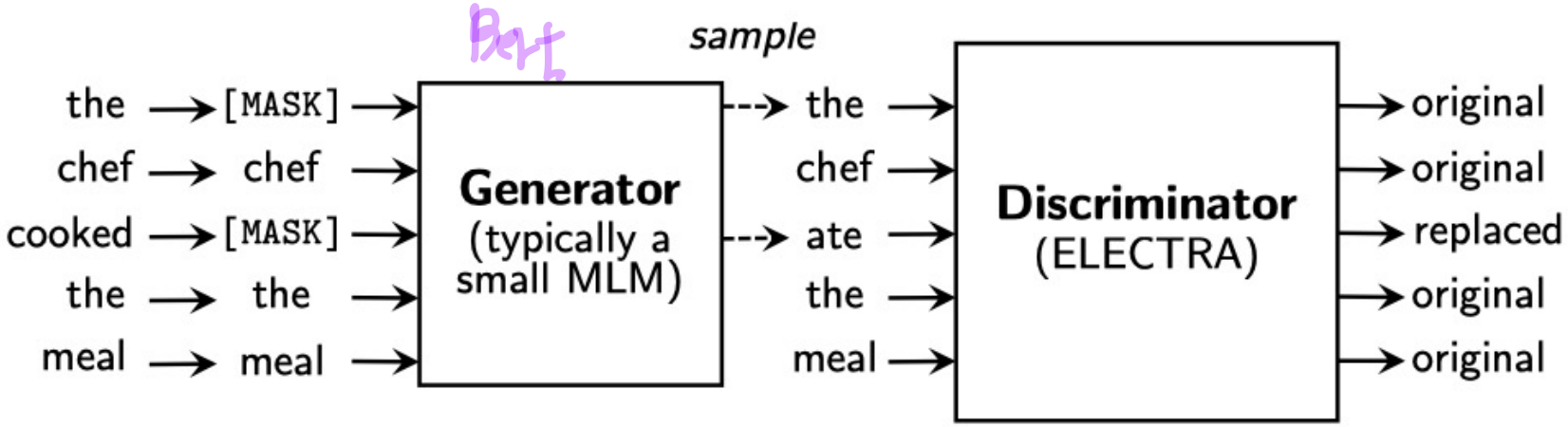
ELECTRA

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ELECTRA

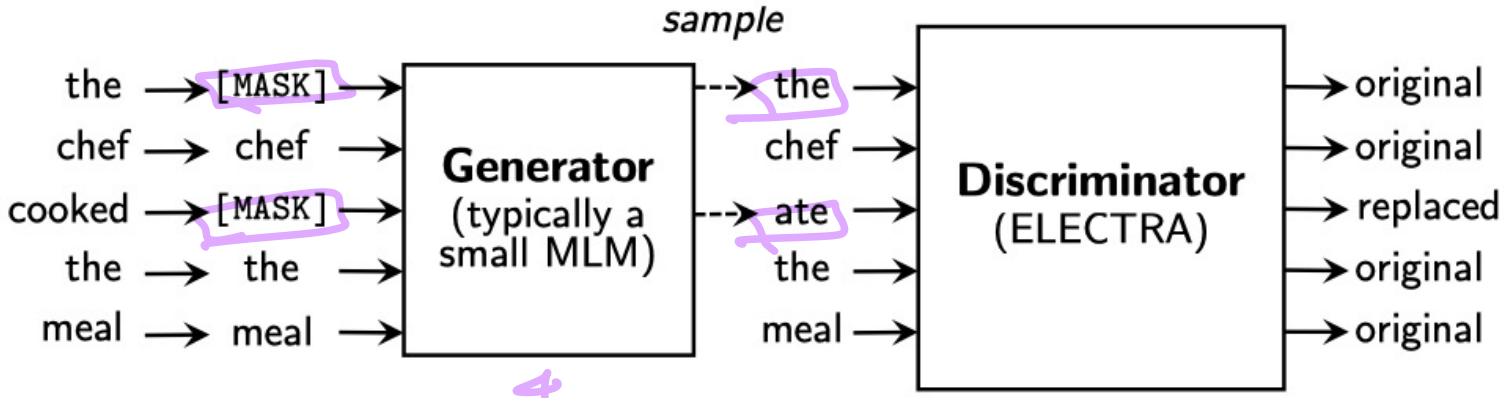
- 2020년 google & Stanford에서 발표

- **MLM 학습하는 Generator**와 입력 token이 **original / replace 여부를 예측하는 Discriminator** 구조



@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ELECTRA



Model

- **Generator**
  - Small BERT를 이용해서 MLM을 학습한
  - Masked된 token의 원래 token을 예측 함
- **Discriminator**
  - Generator의 출력이 원래 Token과 일치 하는 여부를 예측 하는 binary classification
  - $D(\boldsymbol{x}, t) = sigmoid(w^T h_D(\boldsymbol{x})_t)$
    - $\boldsymbol{x} = [x_1, \ldots, x_n]$: input token
    - $h(x) = [h_1, \ldots, h_n]$: vector representation
- Minimize combination loss

$$\min_{\theta_G, \theta_D} \sum_{\boldsymbol{x} \in \mathcal{X}} \mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) + \lambda \mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D)$$

- Fine-tuning시에는 Discriminator만 사용

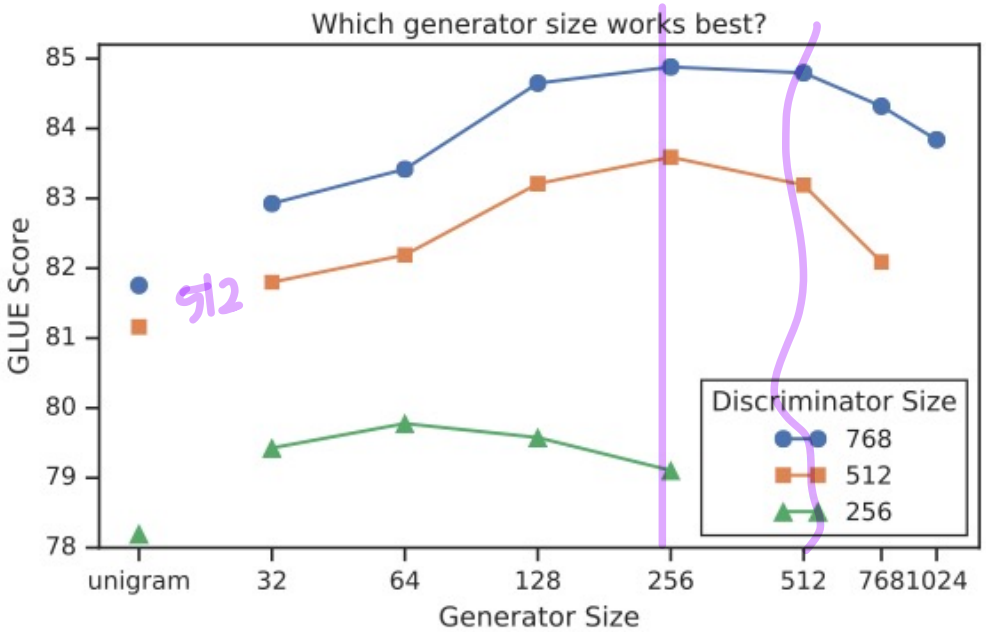
@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ELECTRA

Model Extension

- **Weight Sharing**
  - Pre-train효율을 증가시키기 위해 Generator와 Discriminator간에 **Weights sharing** 함
  - 500K step 학습 후 GLUE score 비교
    - No tying: 83.6, tying embedding: 84.3, tying all: 84.4
  - All weight tie가 가장 좋은 성능을 발휘 하지만 Generator를 줄이기 위해 **Embedding tie** 사용

- **Smaller Generator**
  - Generator는 다른 parameter는 유지하고 layer수만 줄임
  - Generator 크기가 Discriminator의 ¼ ~ ½에서 좋은 성능을 냄



@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ELECTRA

Experimental Setup

- **ELECTRA$_{Base}$**
  - BERT와 동일하게 Wikipedia, BooksCorpus를 이용해 pre-train
- **ELECTRA$_{Large}$**
  - XLNet data 사용 (+, ClueWeb, CommonCrawl, Gigaword)
- 모델 구조와 hyperparameter는 BERT와 동일

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ELECTRA

Result

- **Small Models**
  - Single GUP에서도 빠르게 학습 될 수 있는 모델
  - seq length: 128, batch size: 128, hidden: 256, embedding:128
  - 1.5M step training
  - BERT-Small에 비해서 5점 정도 점수가 좋음, parameter가 많은 GPT에 비해서도 좋음
  - $ELECTRA_{Base}$는 $BERT_{Base}$ 및 $BERT_{Large}$보다 성능이 좋음

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

@2021 cchyun@gmail.com., Ltd All rights reserved.

ELECTRA

Result

- **Large Models**
  - ELECTRA-400K의 성능은 RoBERTa 및 XLNet과 비슷함 (¼ compute)
  - ELECTRA-1.75M은 대부분 성능에서 좋아짐. (적은 compute)

Dev

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**

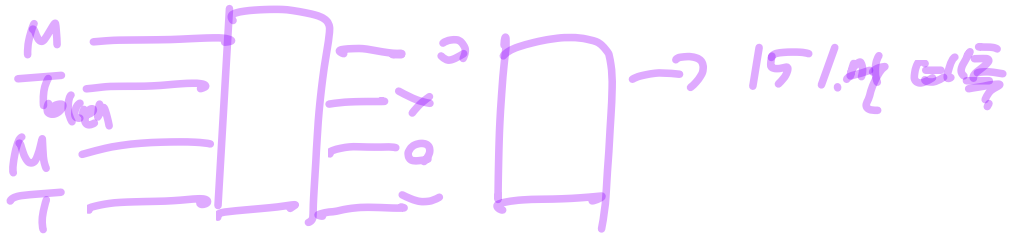
Test

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**

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

ELECTRA

Result



- **Efficiency Analysis**
  - **ELECTRA 15%**
    - Generator 입력 중 MASK만 부분만을 Discriminator가 예측하도록 함
  - **Replace MLM**
    - MASK token 대신에 generator의 token으로 대체. MASK token만 예측 함
  - **All-Tokens MLM**
    - MASK token 대신에 generator의 token으로 대체. 모든 Token을 예측 함
  - 일부만 예측하지 않고 모든 token을 예측하는 것이 가장 큰 효과
  - **All-Tokens MLM**은 ELECTRA에 근접하는 결과를 냄

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

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

T5

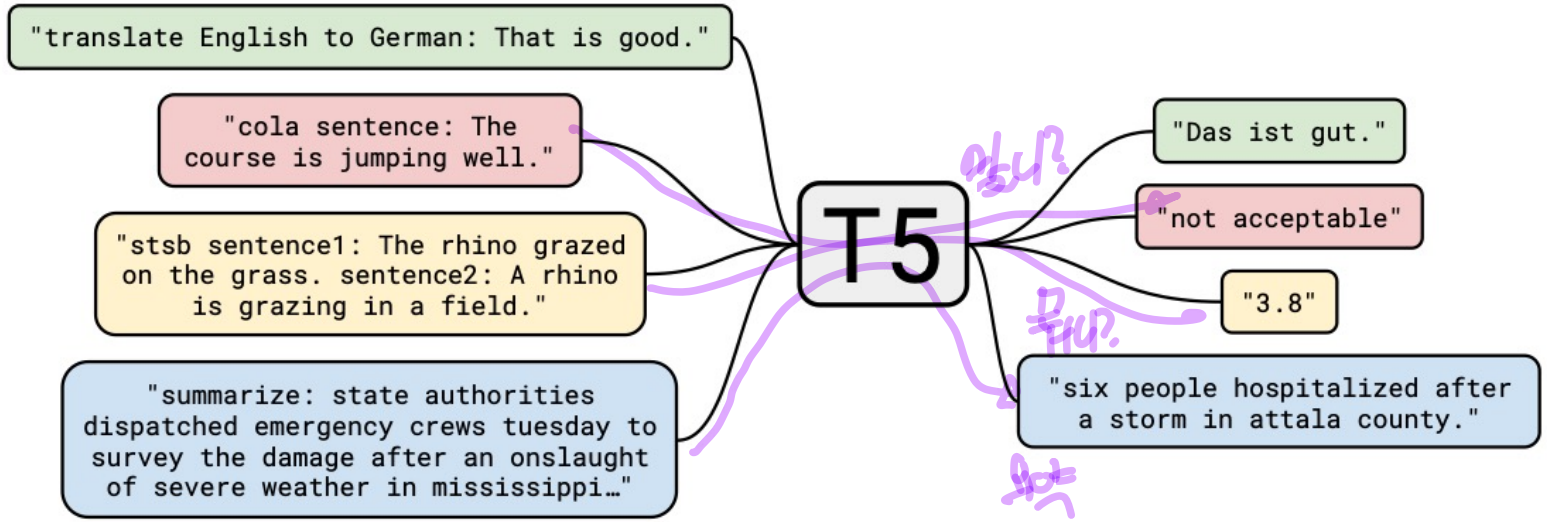
(Text-to-Text Transfer Transformer)

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

T5

- 2019년 google에서 발표

- **Standard Transformer (Encoder-Decoder)** 사용
- 입력과 출력을 모두 Text로 처리하는 간단한 구조



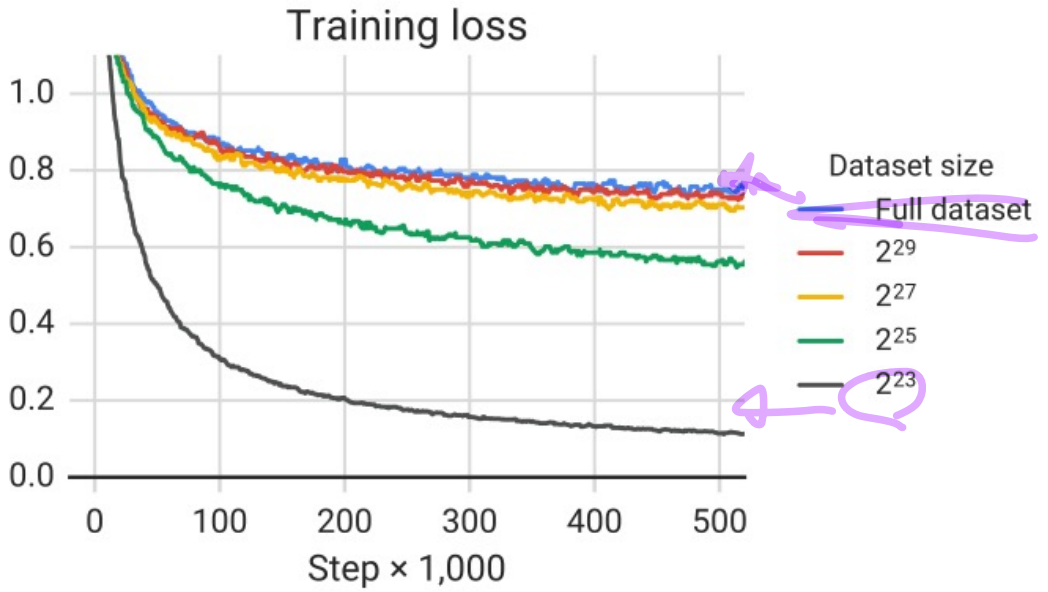
@2021 cchyun@gmail.com., Ltd All Rights Reserved.

T5

The Colossal Clean Crawled Corpus

- C4: 1개월 20TB web text 수집 후 filtering 해서 750 GB 데이터 사용

- 적은 데이터를 여러 번 학습하는 것 보다
  많은 데이터를 적게 학습하는 것이 더 효과적 임



Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Full dataset	0	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
2^{29}	64	**82.87**	**19.19**	**80.97**	**72.03**	**26.83**	**39.74**	**27.63**
2^{27}	256	82.62	**19.20**	79.78	69.97	**27.02**	**39.71**	27.33
2^{25}	1,024	79.55	18.57	76.27	64.76	26.38	39.56	26.80
2^{23}	4,096	76.34	18.33	70.92	59.29	26.37	38.84	25.81

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

T5

Input and output format

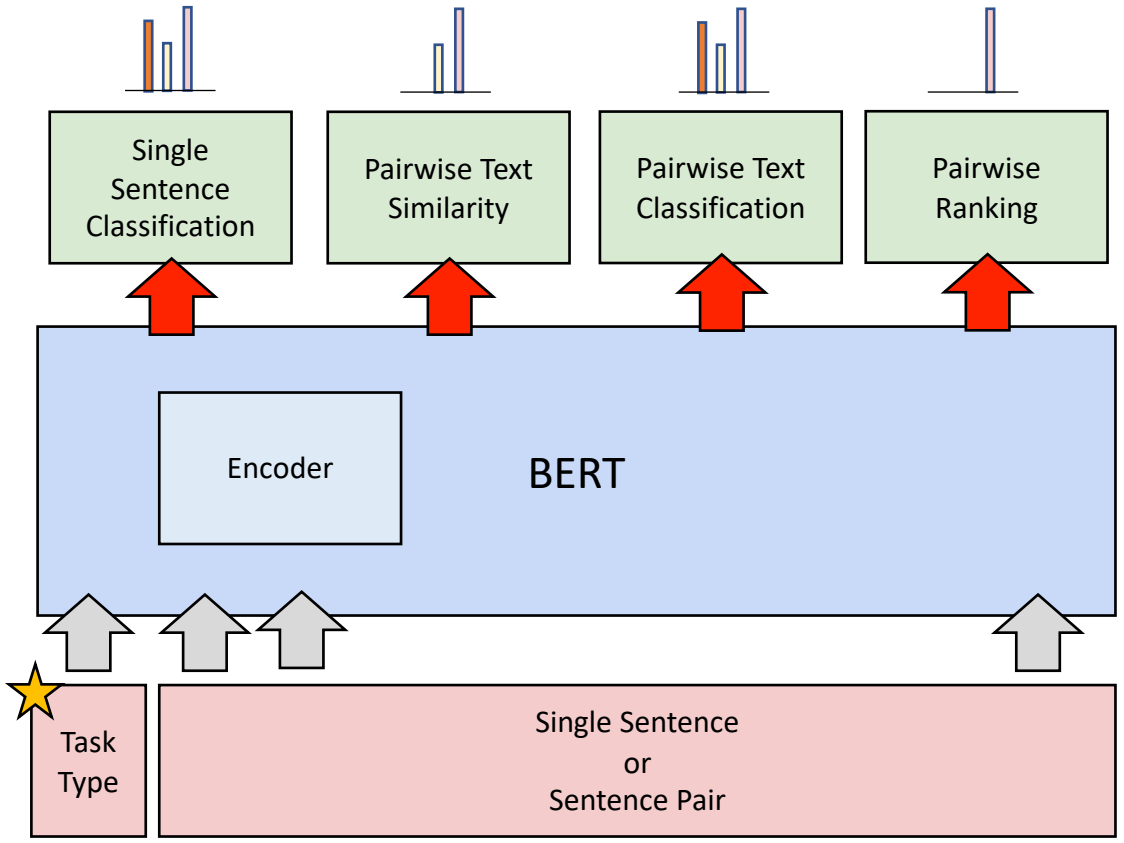
- 입력 출력 모두 text 형식

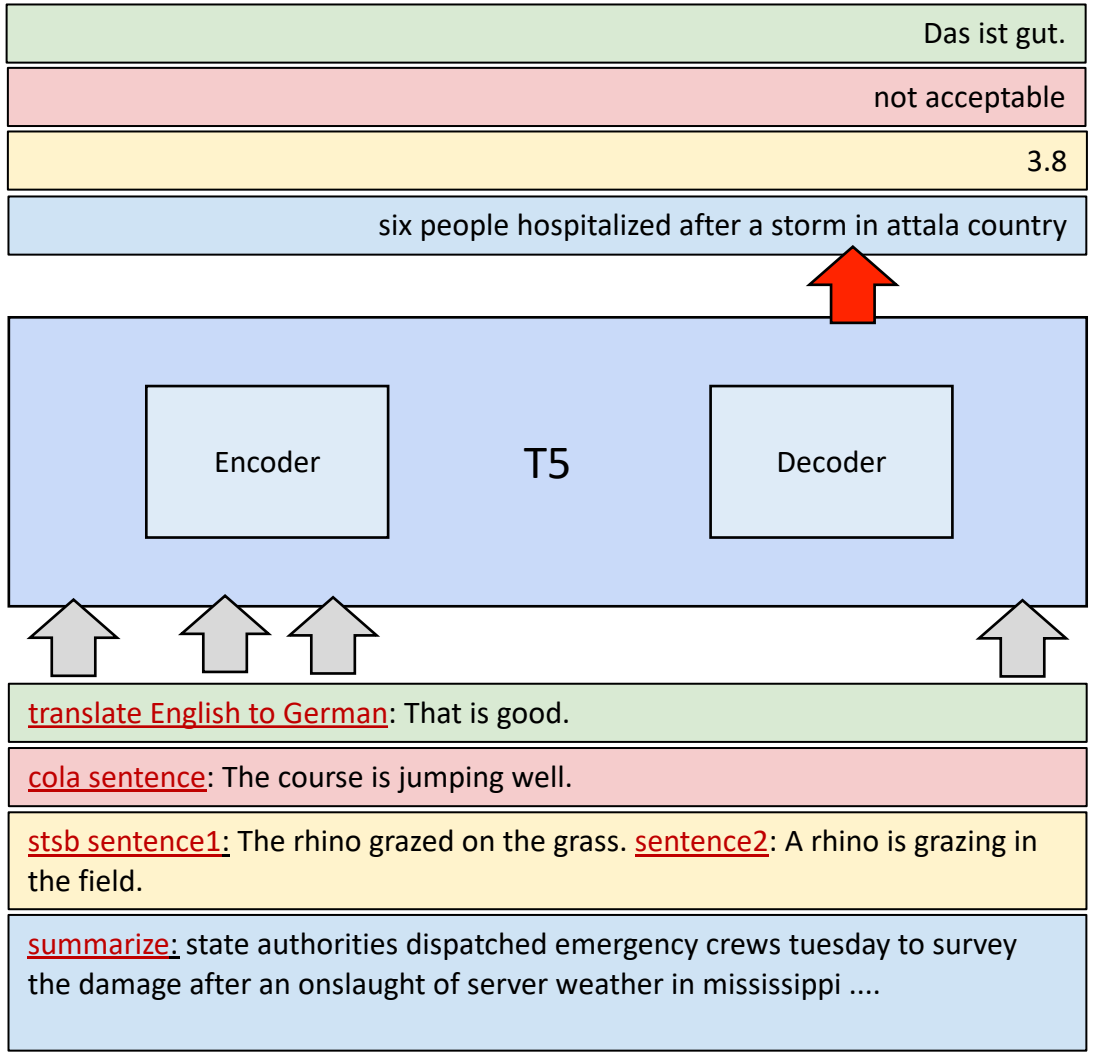
translate English to German: That is good.	⇨	Das ist gut.
mnli premise: I hate pigeons. hypothesis: My feelings towards pigeons are filled with animosity.	⇨	entailment
cola sentence: The course is jumping well.	⇨	not acceptable
stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in the field.	⇨	3.8
summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of server weather in mississippi	⇨	six people hospitalized after a storm in attala country

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

T5

MT-DNN vs T5





MTDNN

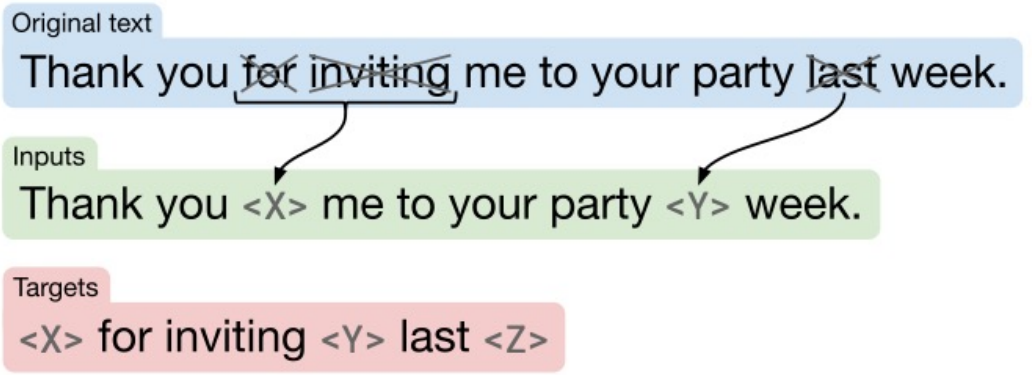
T5

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

T5

Denoising objective

- 일정 영역을 Mask한 후 예측 함



- 전체 token의 10%, 15%, 25%, 50% Mask
  - 15%가 가장 성능이 좋음

Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
10%	**82.82**	19.00	**80.38**	69.55	**26.87**	39.28	**27.44**
★15%	**83.28**	19.24	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
25%	**83.00**	**19.54**	**80.96**	70.48	**27.04**	**39.83**	**27.47**
50%	81.27	19.32	79.80	70.33	**27.01**	**39.90**	**27.49**

- 평균 Mask Span 길이 1, 2, 3, 5, 10
  - 3이 가장 성능이 좋음

Span length	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★Baseline (i.i.d.)	**83.28**	19.24	80.88	71.36	**26.98**	**39.82**	**27.65**
2	**83.54**	19.39	**82.09**	**72.20**	**26.76**	**39.99**	**27.63**
3	**83.49**	**19.62**	**81.84**	**72.53**	**26.86**	39.65	**27.62**
5	**83.40**	19.24	**82.05**	**72.23**	**26.88**	39.40	**27.53**
10	82.85	19.33	**81.84**	70.44	**26.79**	39.49	**27.69**

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

T5

Result

- T5-11B이 가장 좋은 성능을 냄
- T5-3B: d_{model} = 1024, 24 layer, d_{kv} = 128, d_{ff} = 16,384, 32 head
- T5-3B: d_{model} = 1024, 24 layer, d_{kv} = 128, d_{ff} = 65,536, 128 head

Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	**97.1**[a]	**93.6**[b]	**91.5**[b]	**92.7**[b]	**92.3**[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**89.7**	**70.8**	**97.1**	91.9	89.2	92.5	92.1

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	**74.8**[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	74.6	90.4	**92.0**	**91.7**	96.7	**92.5**	**93.2**

@2021 cchyun@gmail.com., Ltd All Rights Reserved.

Thanks

@2021 cchyun@gmail.com., Ltd All Rights Reserved.