

Transformer Evolution

현 청 천

헬로엔엠에스 / 모두의연구소 / DeepNLP

발표자 소개

- 2018년 9월 모두의연구소 DeepNLP 합류
- 2018년 12월 네이버 NLP Challenge NER 부분 장려상 수상
- 2019년 7월 AI Starthon
 - Intent 부분 2위
 - Query 부분 3위
 - Movie 부분 4위
- 2019년 12월 성남시 빅데이터센터 오픈 이노베이션 챌린지 – 아이디어톤 우승
- 연구 중
 - 음성을 통한 사용자 감정 분석
 - 경량화된 Pretrained Language Model
 - 강화학습을 이용한 자연어 학습
- cchyun@gmail.com
- <https://github.com/paul-hyun>



DeepNLP

2019 Study

- 매주 1~2개의 최신 hot trend 논문 리뷰 (소스가 있는 경우는 소스 리뷰 포함)
- Yandex School NLP Course (https://github.com/yandexdataschool/nlp_course)
- edwith 데이터 사이언스 (<https://www.edwith.org/datascience>)
- soynlp (https://github.com/lovit/fastcampus_textml_blogs)

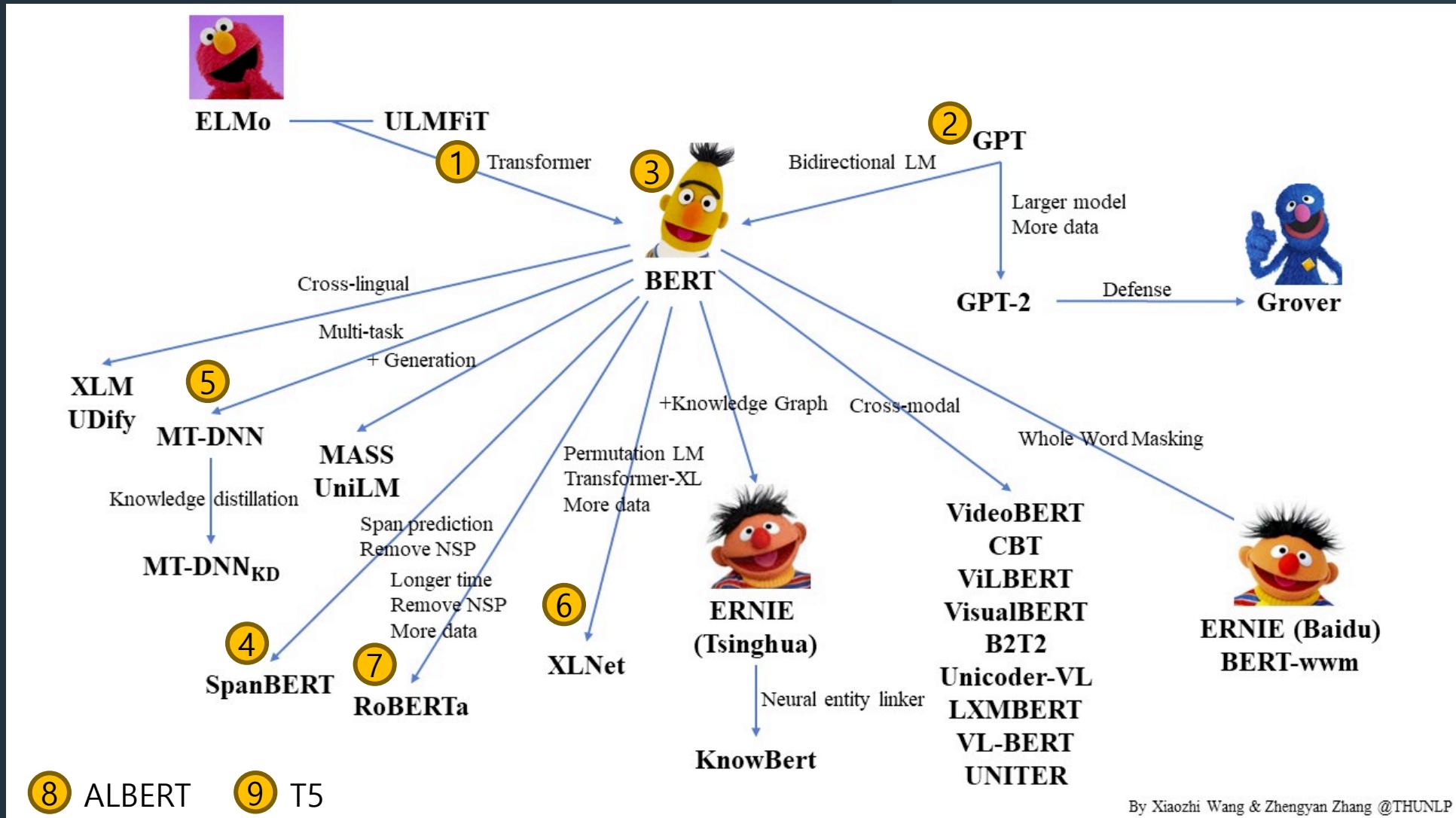
Group

- 논문구현: 3주에 논문 1개씩 리뷰 및 직접 구현에 도전하는 팀
- KorQuAD: 'ChangWook Jun'님 현재 9위 (개인 1등)
- NLP Zero to All: '텐서플로우와 머신러닝으로 시작하는 자연어처리' 스터디
- AI Starthon 2019: Intent 2위, query 3위, movie 4위
- 금융문자 분석 경진대회: 대회 준비 중
- TensorBoy: 인공지능을 알려주는 챗봇

2020년

- 한국어 데이터셋 모아서 Benchmark 찍기 (klue, korean glue)
- 논문 1편 / 강의 1편
- 워크샵

목차



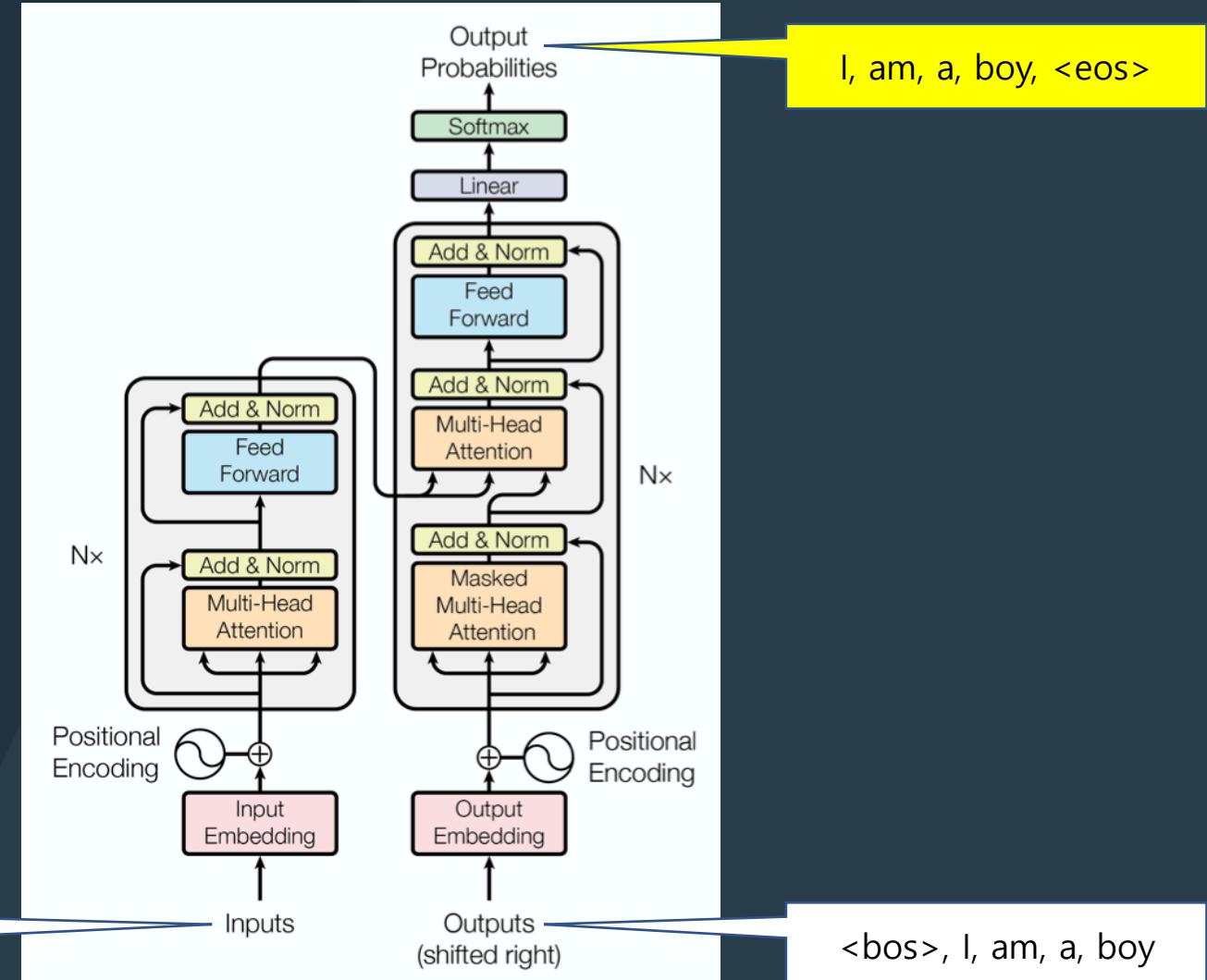
Transformer

Transformer

2017년 Google에서 발표

CNN, RNN 등을 사용하지 않고 Attention 만 사용
번역을 위한 모델로 발표 됨

나는, 소년, 입니다



Transformer

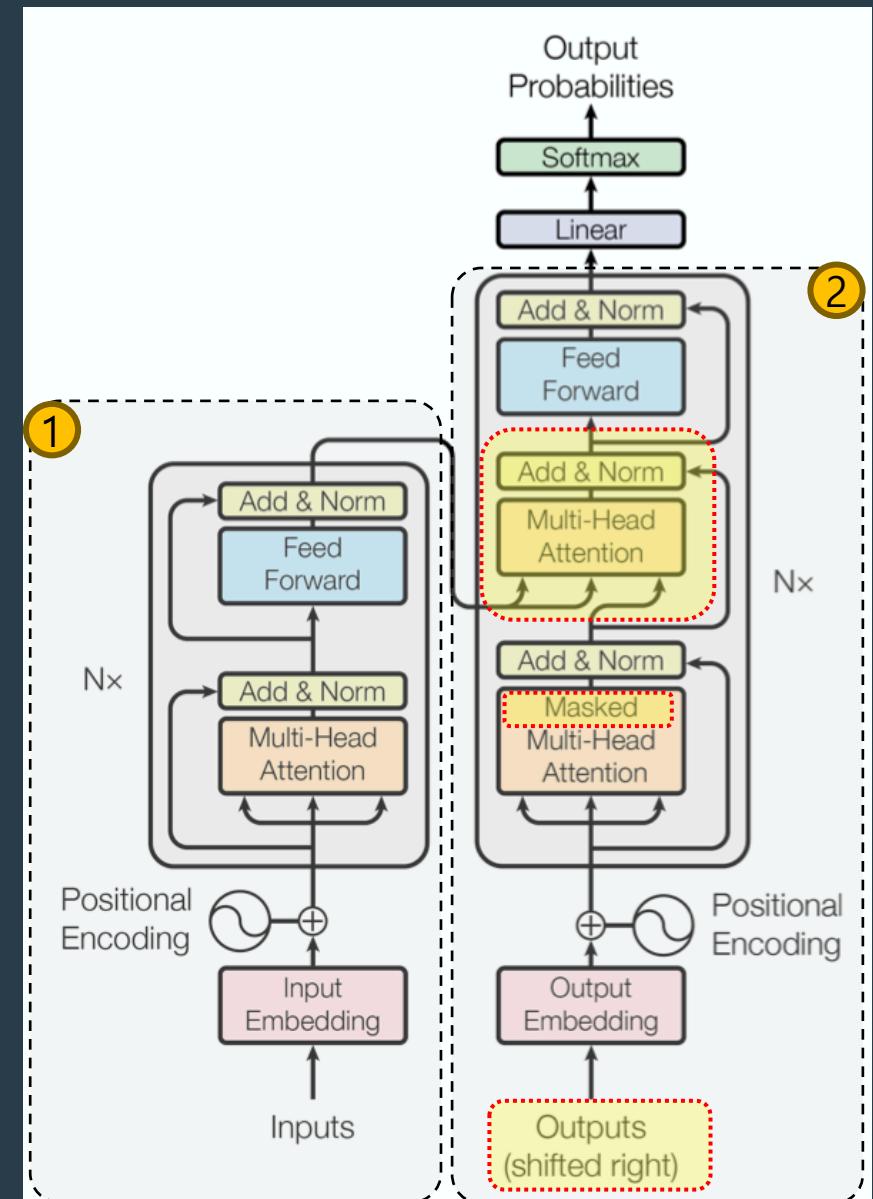
① Encoder

- multi-head self-attention
- feed forward
- residual + layer normalization
- n-layer

② Decoder

- masked multi-head self-attention
- encoder & decoder multi-head attention
- feed forward
- residual + layer normalization
- n-layer

Pretrained LM 에서는 Encoder 또는 Decoder 한쪽만 사용
최근 발표된 T5는 Transformer를 그대로 사용



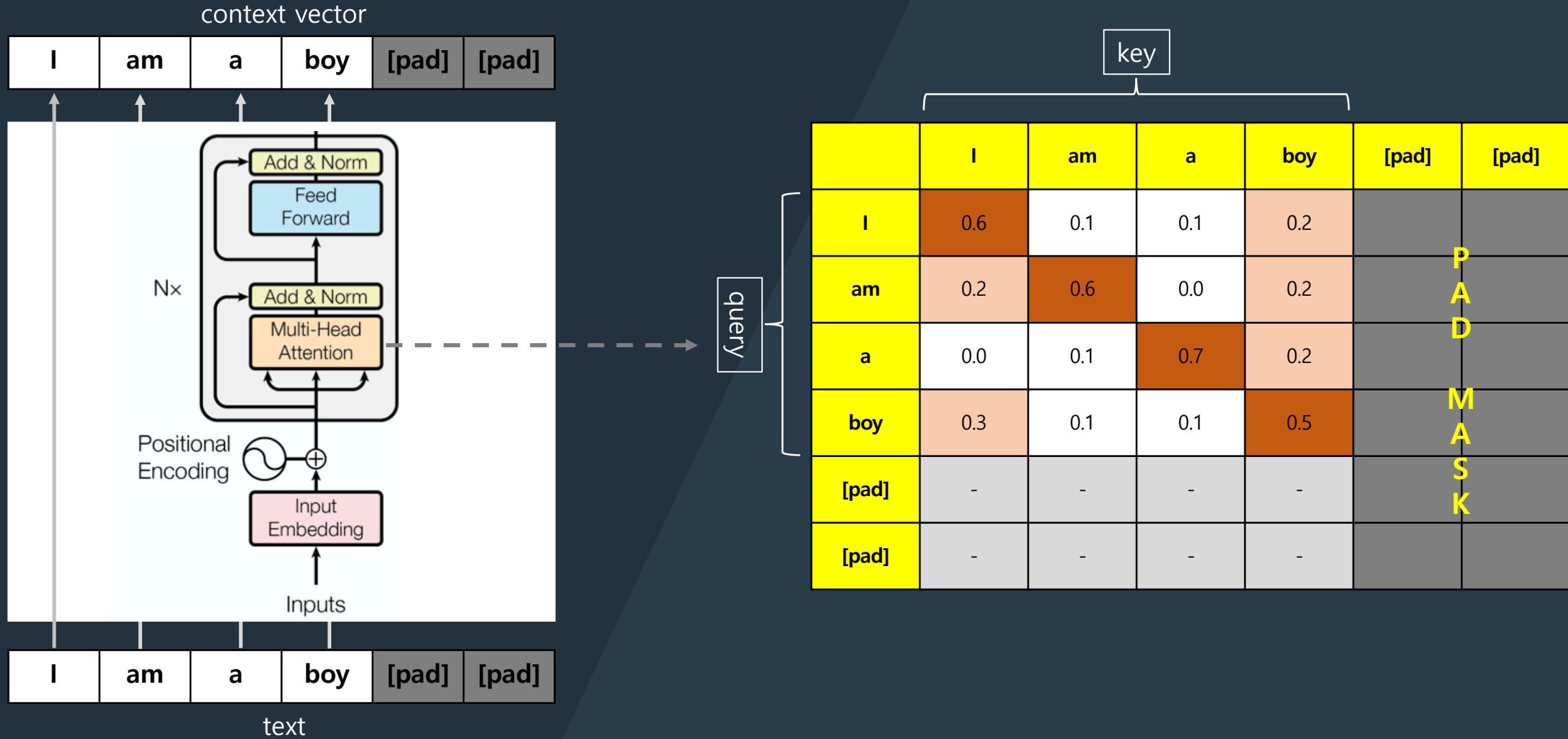
* Token Embedding

Word 또는 Sub-word의 embedding 값

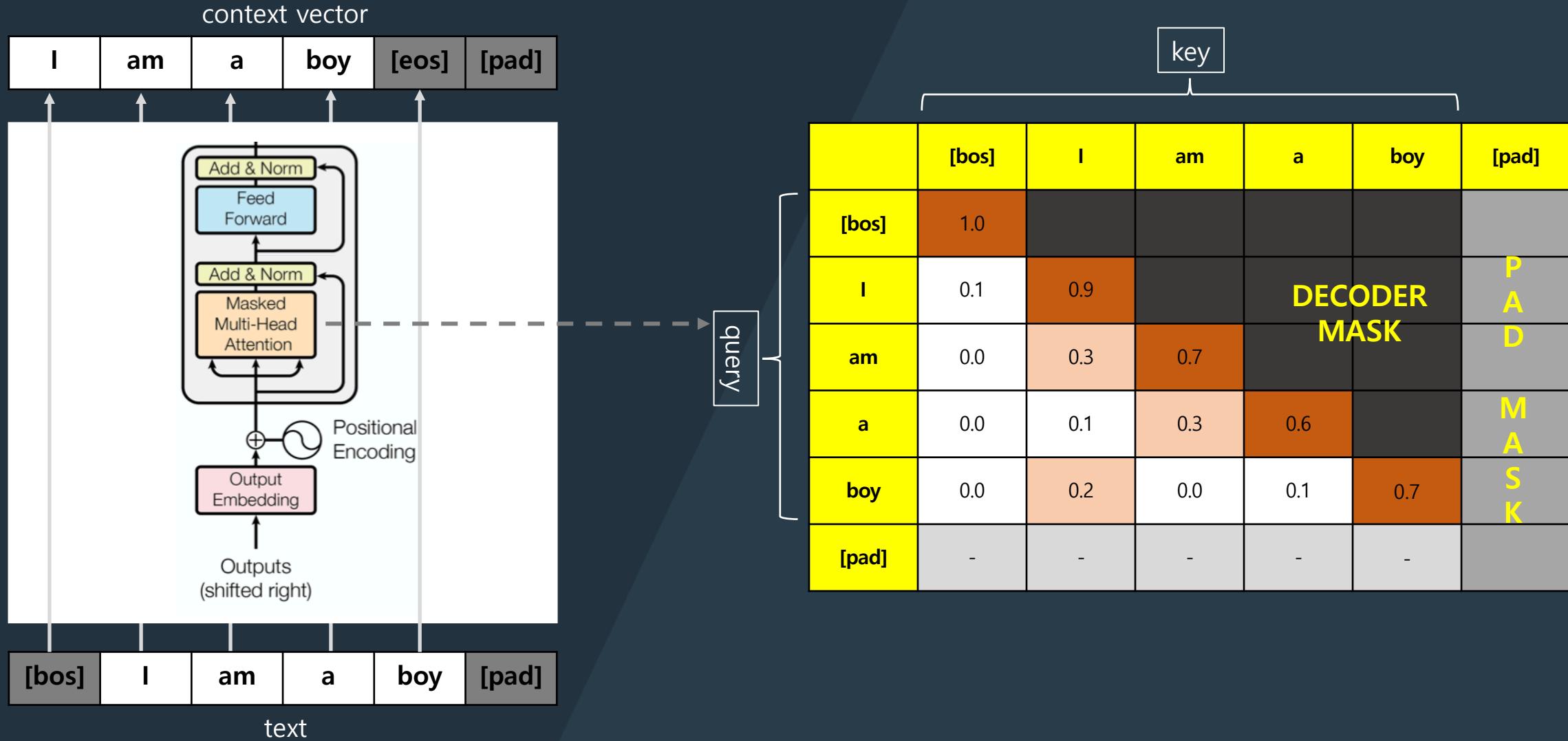
* Position Embedding

단어의 순서를 표현하기 위한 embedding 값

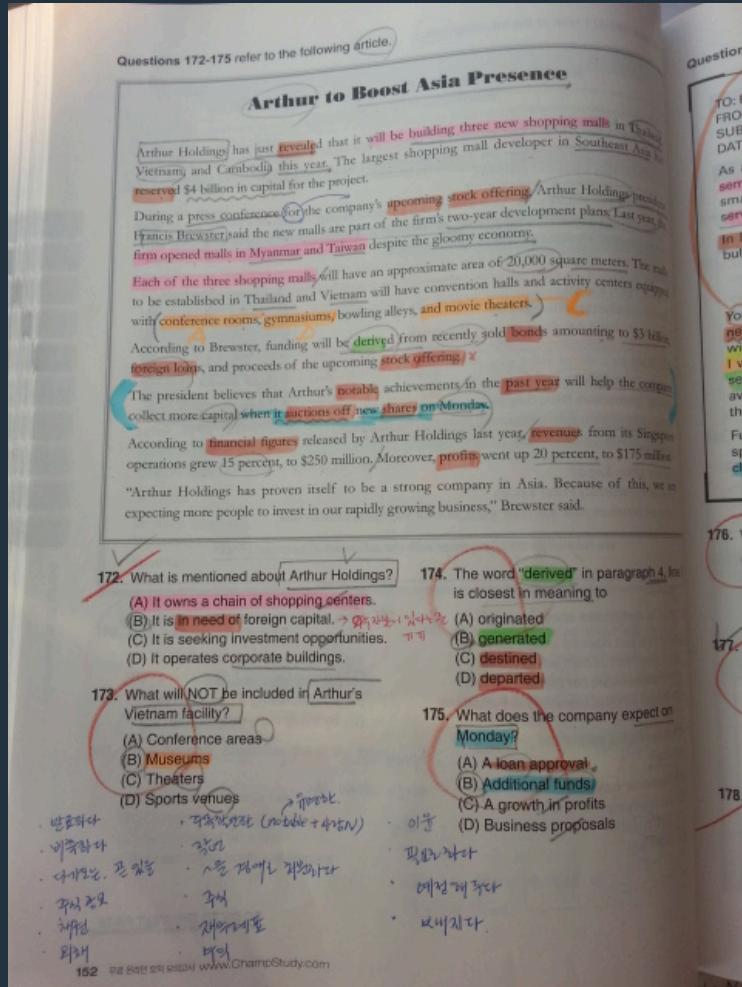
Transformer - Encoder



Transformer - Decoder



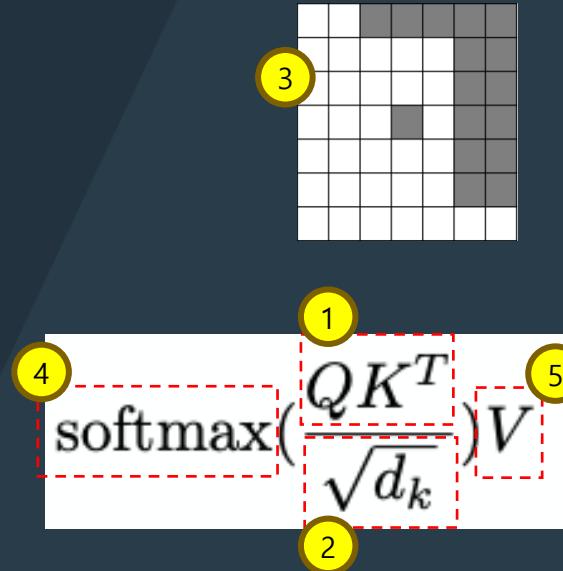
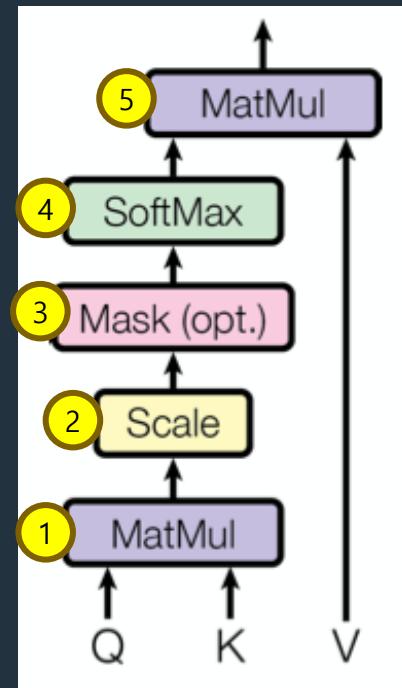
Transformer - Attention



It is in this spirit that a majority of American governments have passed new laws since 2009, making the registration process more difficult.

- 단어와 연관이 있는 부분을 더 집중해서 보게 함
 - 확률 분포로 Attention의 총합은 1이 됨

Transformer - Attention



값의 편차를 줄여 주는 효과
(학습이 더 잘되게 함)

[Scaled-Dot Product Attention]

Transformer - Attention

K^T [d_hidn, k_seq]

		I am a boy [pad] [pad]					
		160	80	8	120	16	80
-		80	160	8	64	0	80
Q		16	32	160	120	64	0
[q_seq, d_hidn]		120	64	40	160	40	32
Q		80	0	0	0	160	80
[q_seq, d_hidn]		0	0	0	160	80	160

1

$[q_seq, d_hidn] \times [d_hidn, k_seq] = [q_seq, k_seq]$

2

$$d_k = 64$$

K^T

		I am a boy [pad] [pad]					
		20	10	1	15	2	10
-		10	20	1	8	0	10
Q		2	4	20	15	8	0
[q_seq, d_hidn]		15	8	5	20	5	4
Q		10	0	0	0	20	10
[q_seq, d_hidn]		0	0	0	20	10	20

1

4

5

2

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Transformer - Attention

K^T

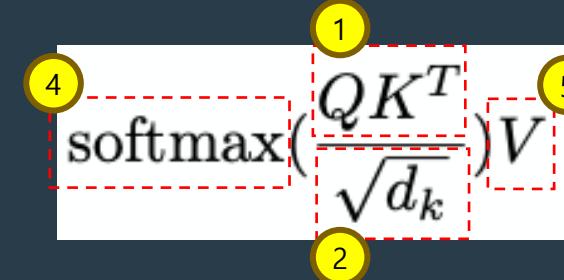
		K^T					
		I	am	a	boy	[pad]	[pad]
Q	-	20	10	1	15	2	10
	am	10	20	1	8	0	10
	a	2	4	20	15	8	0
	boy	15	8	5	20	5	4
	[pad]	10	0	0	0	20	10
	[pad]	0	0	0	20	10	20

0	0	0	0	1	1
0	0	0	0	1	1
0	0	0	0	1	1
0	0	0	0	1	1
0	0	0	0	1	1
0	0	0	0	1	1



K^T

		K^T					
		I	am	a	boy	[pad]	[pad]
Q	-	20	10	1	15	-inf	-inf
	am	10	20	1	8	-inf	-inf
	a	2	4	20	15	-inf	-inf
	boy	15	8	5	20	-inf	-inf
	[pad]	10	0	0	0	-inf	-inf
	[pad]	0	0	0	20	-inf	-inf


$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Transformer - Attention

4
softmax

Q

	K^T					
	I	am	a	boy	[pad]	[pad]
-	0.6	0.1	0.1	0.2	0	0
am	0.3	0.5	0.1	0.1	0	0
a	0.1	0.1	0.5	0.3	0	0
boy	0.3	0.1	0.1	0.5	0	0
[pad]	0.7	0.1	0.1	0.1	0	0
[pad]	0.1	0.1	0.1	0.7	0	0

[q_seq, k_seq]

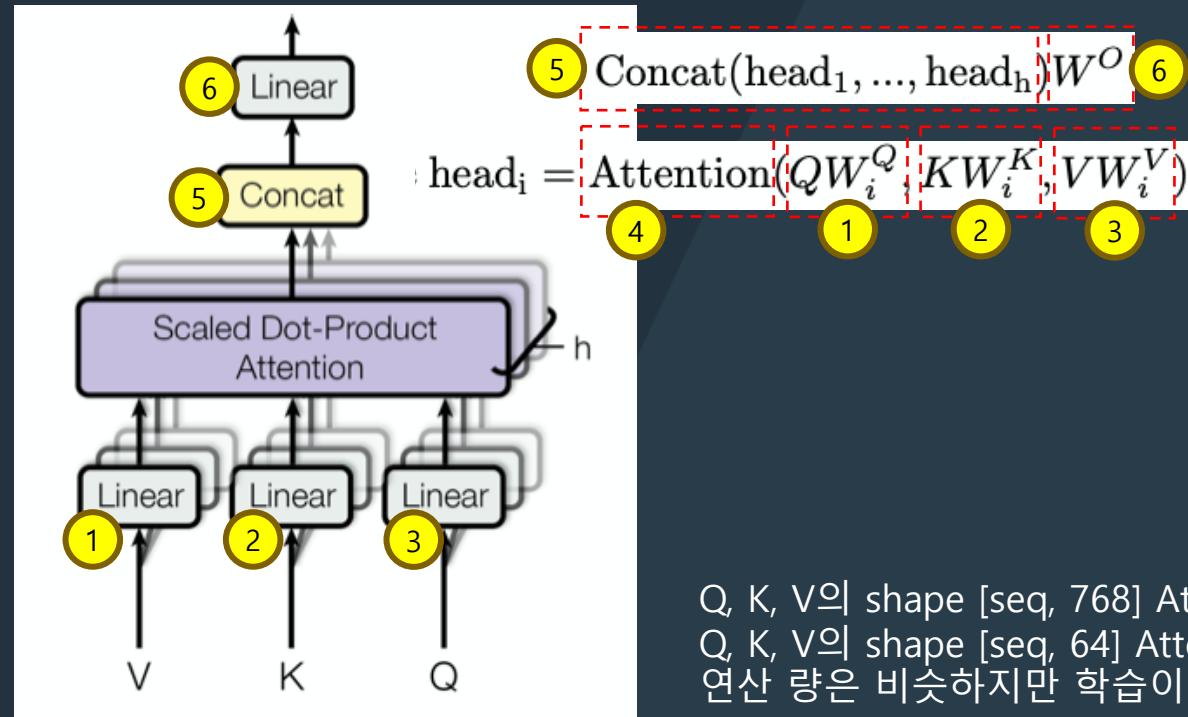
5
 $\times v$

-	am	a	boy	[pad]	[pad]
---	----	---	-----	-------	-------

[q_seq, k_seq] x [k_seq, d_hidn]
= [q_seq, d_hidn]

$$v = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Transformer - Attention



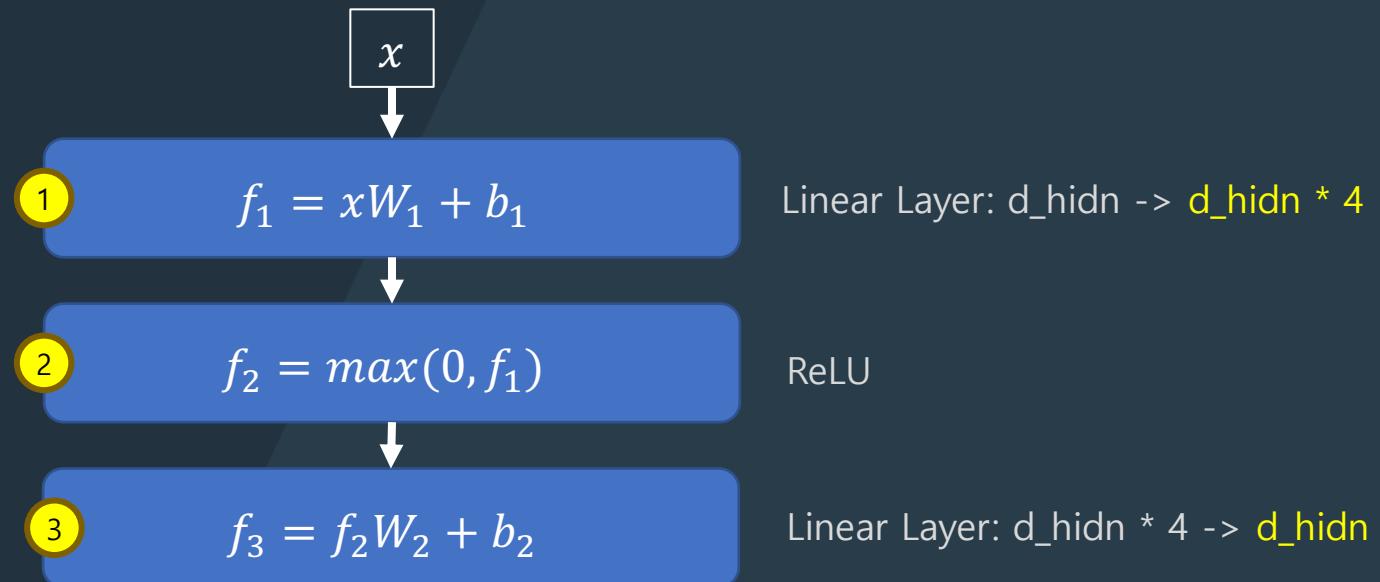
Q, K, V의 shape [seq, 768] Attention 1번보다
Q, K, V의 shape [seq, 64] Attention 12번이
연산 량은 비슷하지만 학습이 더 잘됨

[Multi-Head Attention]

Transformer – Feed Forward

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

2 1 3



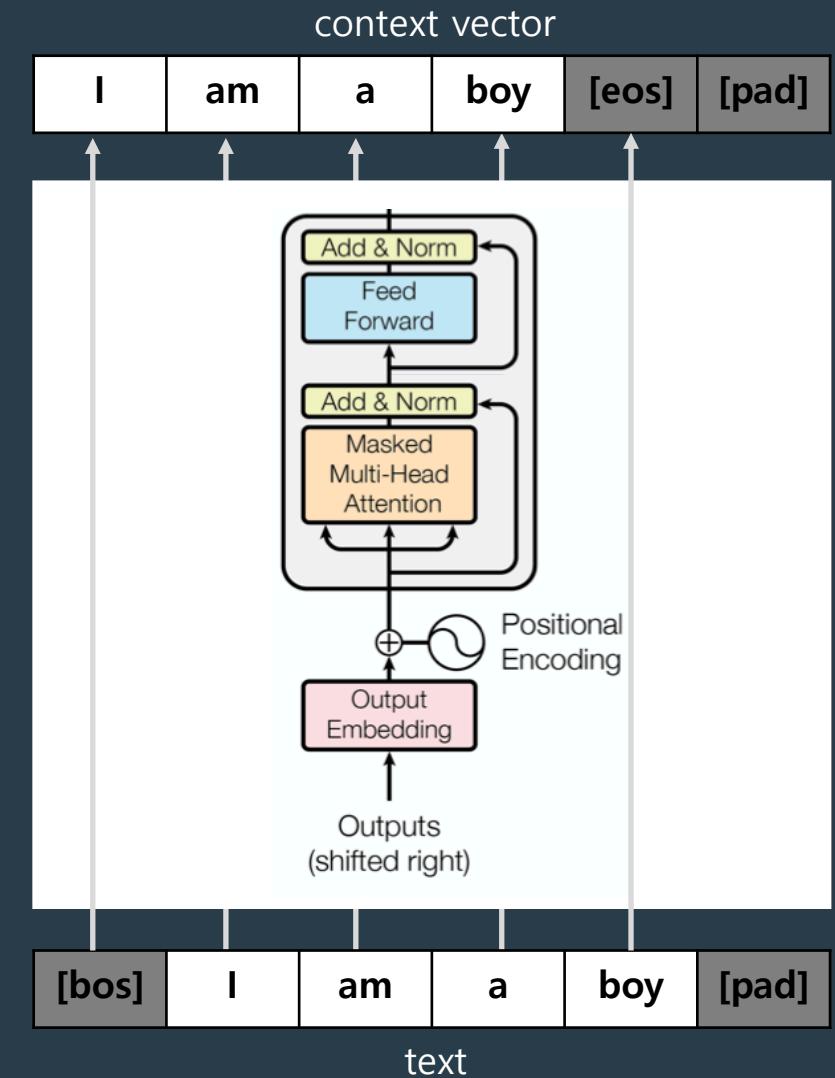
GPT(Generative Pre-Training)

GPT

2018년 OpenAI에서 발표
Transformer Decoder를 사용함

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

이전 단어들 1을 보고
다음 단어 2를 예측하는 방법으로 LM을 학습 함



GPT

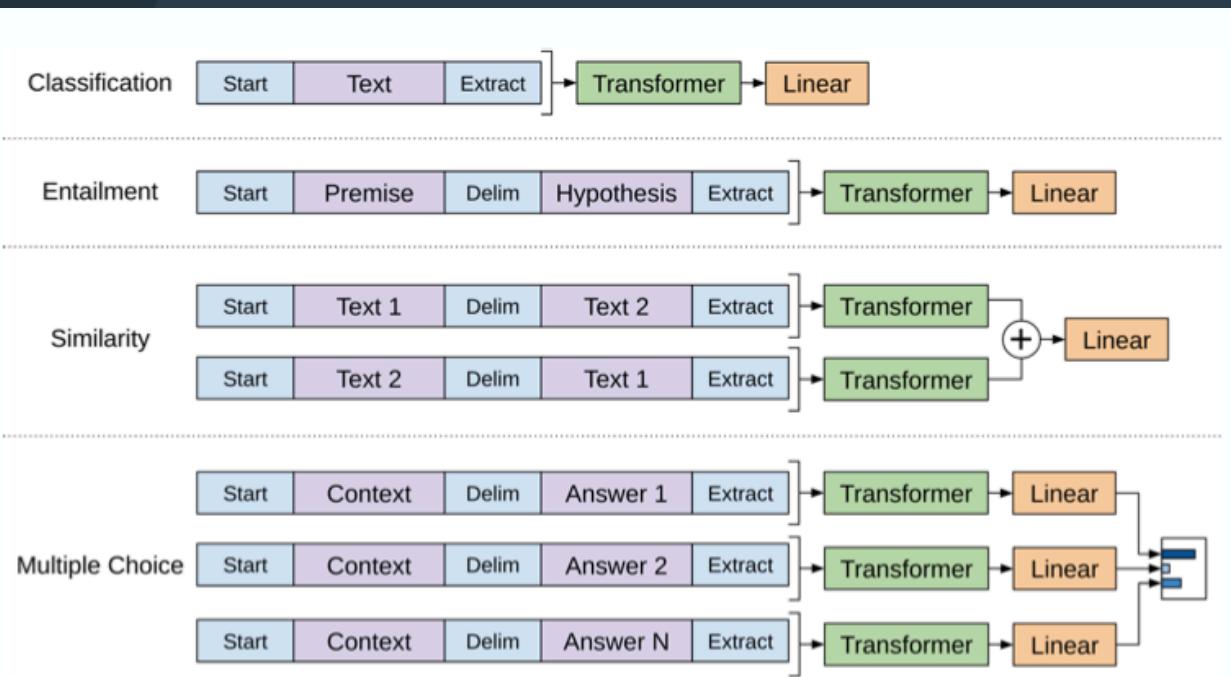
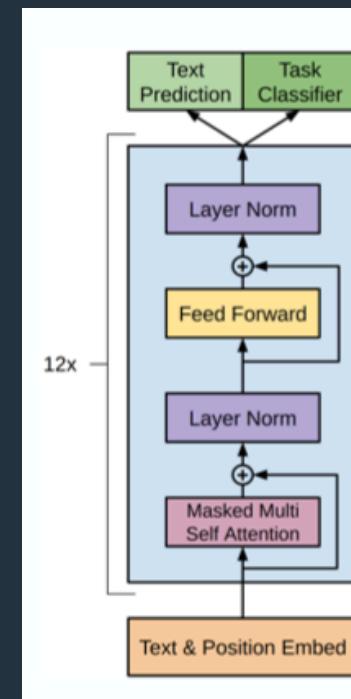
학습된 LM을 여러 Downstream Task에 적용

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

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

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

Task에 적용할 때 목적함수 L_2 와 L_1 을 동시에 학습할 수도 있음
(학습이 더 잘됨)



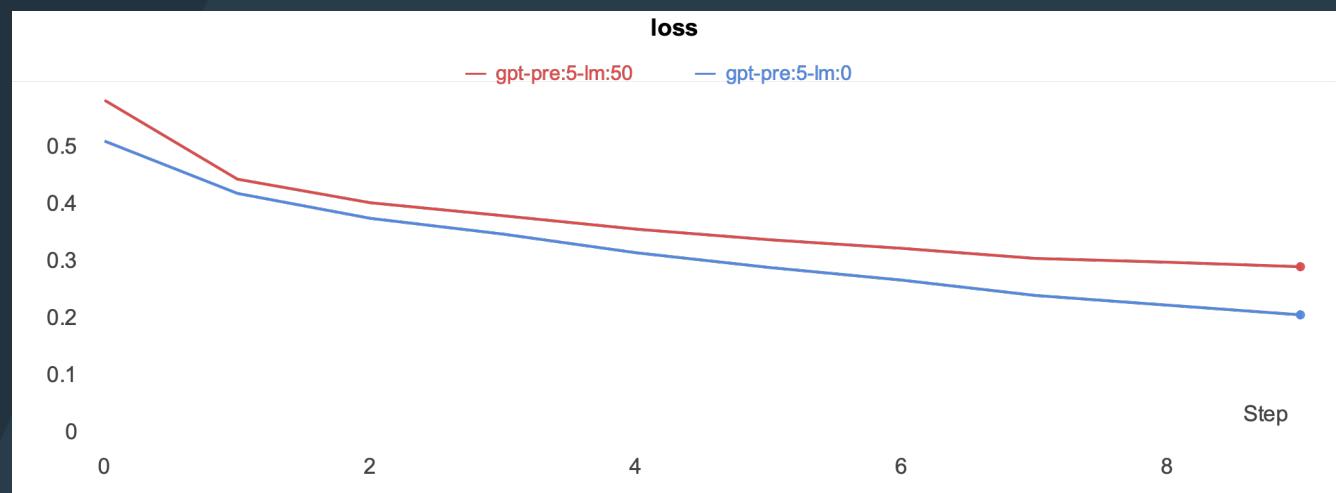
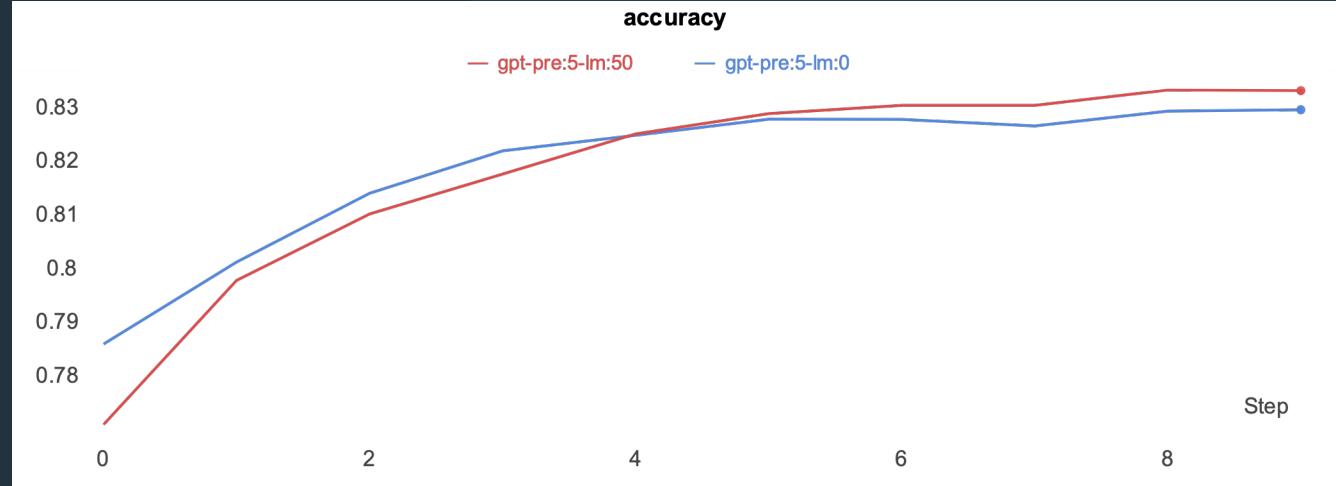
GPT – 네이버 영화 (KoWiki Pretrain 5)

Accuracy

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

- $\lambda = 0$: No LM
- $\lambda = 0.5$: LM

Loss



BERT

(Bidirectional Encoder Representations from Transformers)

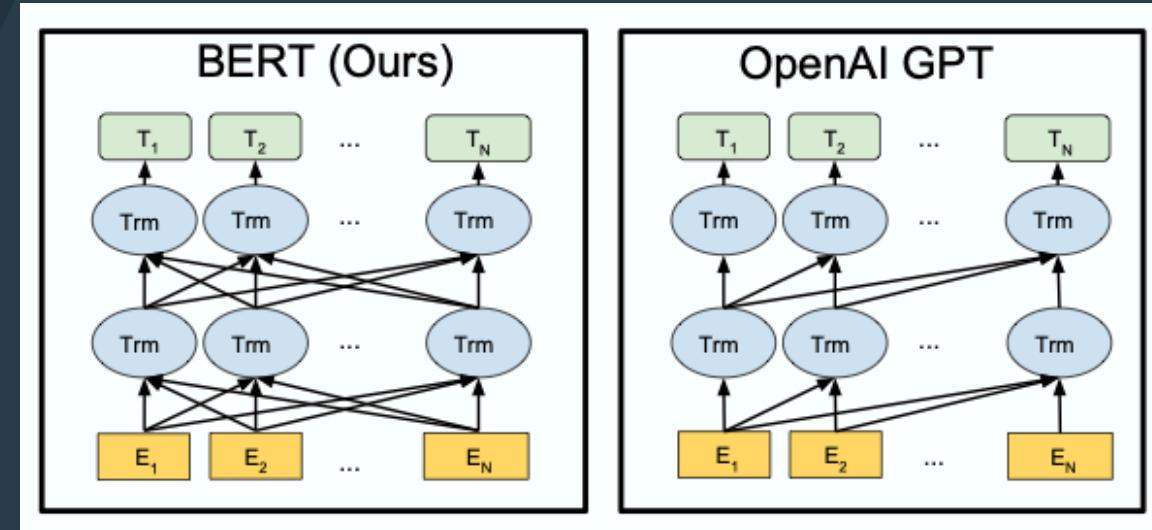
BERT vs. GPT

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)

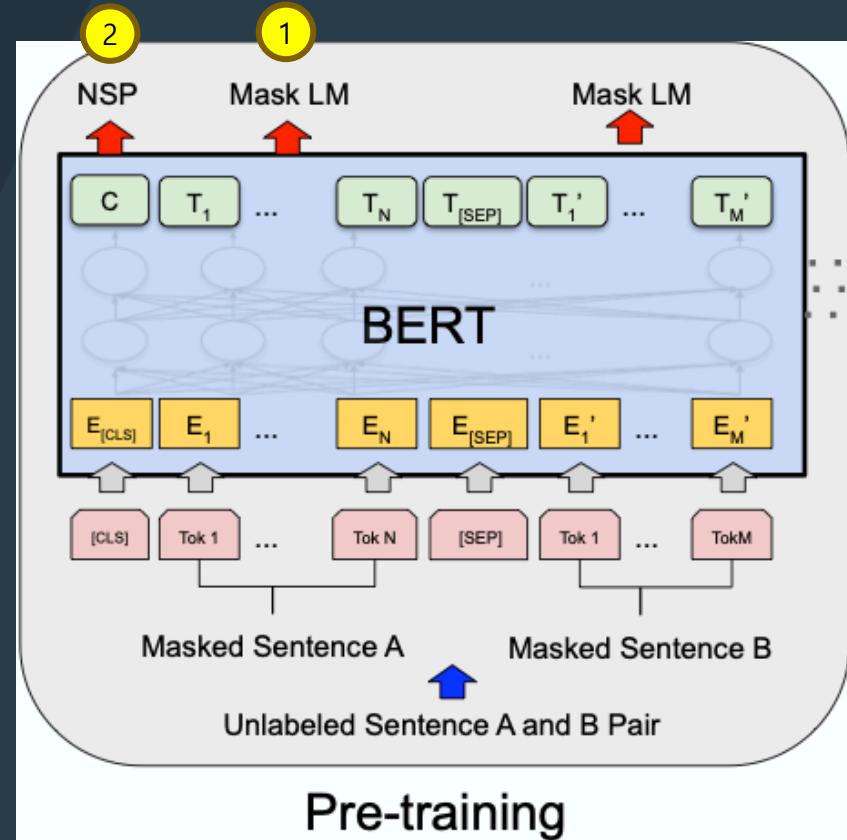


BERT

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



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

BERT-MLM, NSP

1. His eloquent speech made the students_____.

- ①fascinate ②fascinating ③fascinated ④fascinations

2. Not too many years ago, it was an_____experience to travel 50 miles from home.

- ①excited ②excite ③exciting ④excitedly

3. Many employees of the corporation said that the sudden pay raise was_____.

- ①surprising ②to be surprised ③surprised ④surprise

4. Mr. Park sometimes feels_____because his English isn't very good.

- ①disappointed ②to be disappointed ③disappoint ④disappointing

5. Job applicants should be reminded that false information_____in the interview
may result in automatic dismissal.

- ①give ②given ③giving ④was given

MLM (Mask LM)

sentence_a = "신용 대출을 신청 하고 싶어요"

sentence_b = "대출 관련 서류를 준비해 오셨어요?"

Label: IsNext

sentence_a = "신용 대출을 신청 하고 싶어요"

sentence_b = "어떤 팀이 올해 프로야구 우승 했나요?"

Label: NotNext

NSP (Next Sentence Prediction)

BERT-Input

Token Embedding

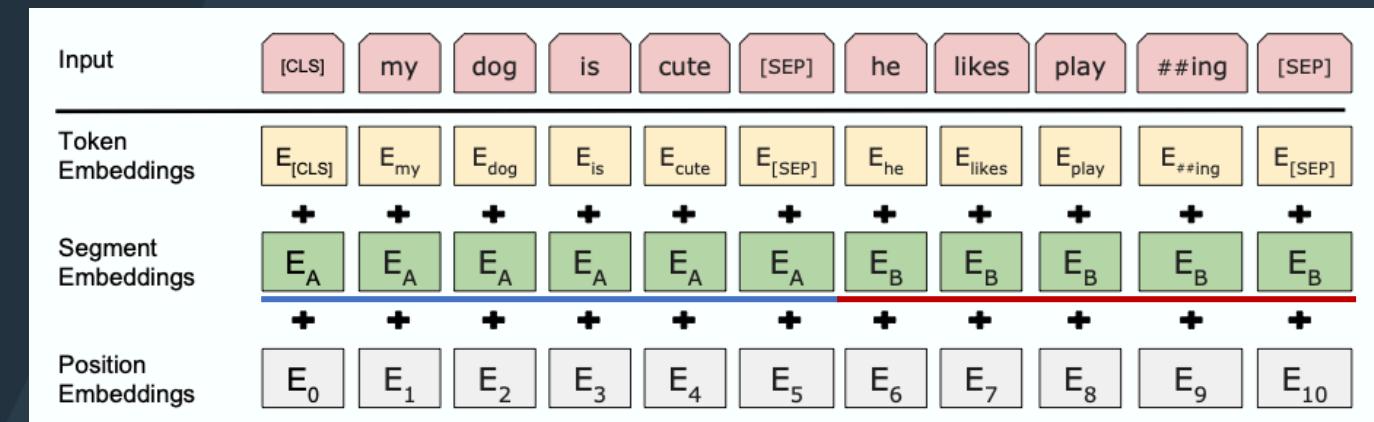
Word 또는 Sub-word의 embedding 값

Segment Embedding

Sentence A, Sentence B를 구분하기 위한 embedding 값

Position Embedding

단어의 순서를 표현하기 위한 embedding 값



BERT-Downstream Task

Sentence Pair Classification

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

Single Sentence Classification

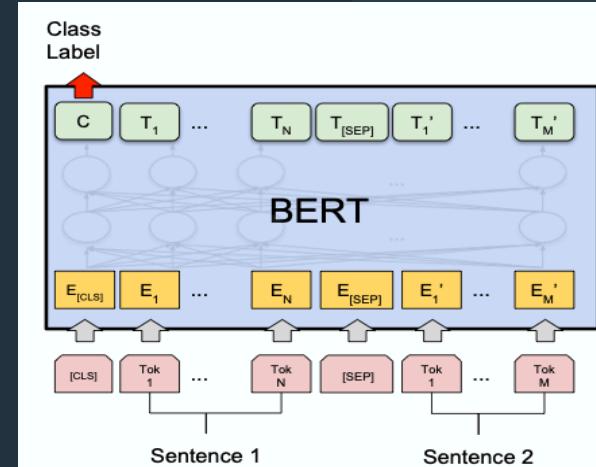
- input: Single sentence
- output: [CLS] token 사용해서 예측

Question Answering

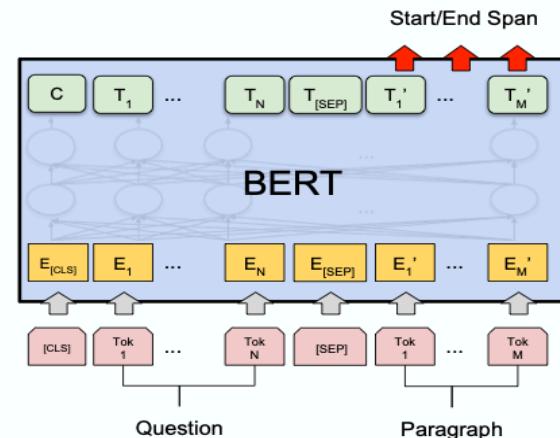
- input: question, paragraph
- output: paragraph내에서 Start/End Span 예측

Single Sentence Tagging Task

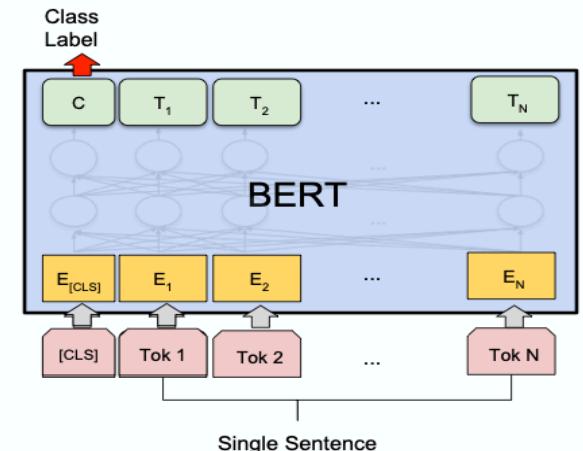
- input: Single sentence
- output: Token 별로 Tag 예측



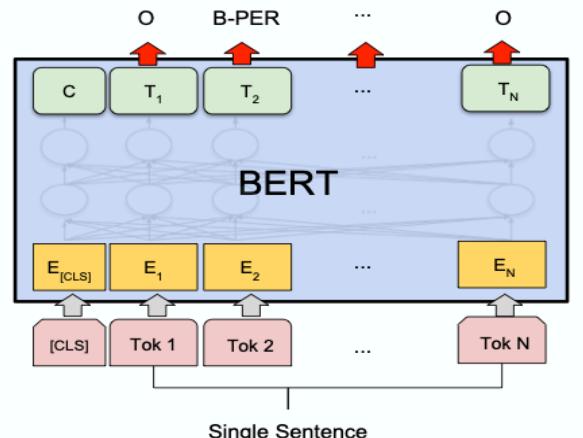
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(c) Question Answering Tasks:
SQuAD v1.1



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

GLUE Benchmark

Rank	Name	Model	URL	Score
1	T5 Team - Google	T5		89.7
2	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4
3	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.0
4	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8
5	Facebook AI	RoBERTa		88.5
6	XLNet Team	XLNet-Large (ensemble)		88.4
7	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6
8	GLUE Human Baselines	GLUE Human Baselines		87.1
12	Danqi Chen	SpanBERT (single-task training)		82.8

SpanBERT

12 Danqi Chen

SpanBERT (single-task training)



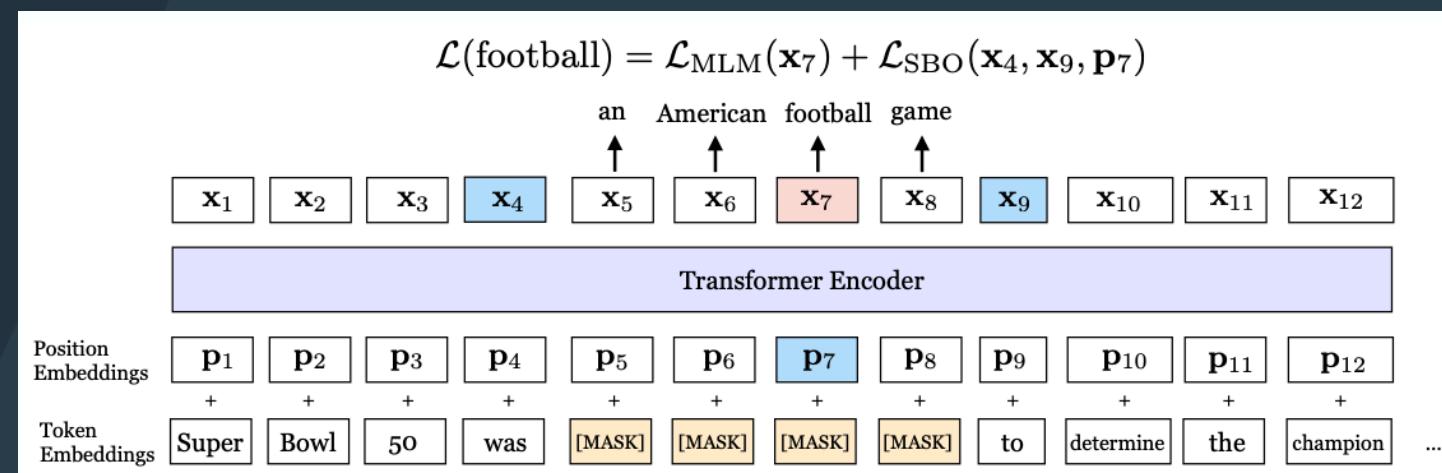
82.8

SpanBERT

2019년 발표

- Pretrain시 단일 토큰 마스크가 아닌 일정 영역(Span)을 Mask한 후 예측하도록 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

- $L(an) = L_{MLM}(x_5) + L_{SBO}(x_4, x_9, p_5)$
- $L(American) = L_{MLM}(x_6) + L_{SBO}(x_4, x_9, p_6)$
- $L(football) = L_{MLM}(x_7) + L_{SBO}(x_4, x_9, p_7)$
- $L(game) = L_{MLM}(x_8) + L_{SBO}(x_4, x_9, p_8)$



MT-DNN (Multi-Task Deep Neural Networks)



7

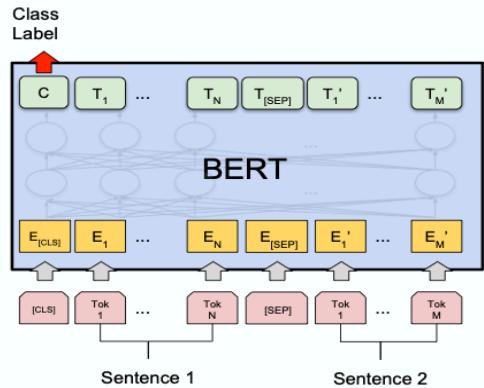
Microsoft D365 AI & MSR AI

MT-DNN-ensemble

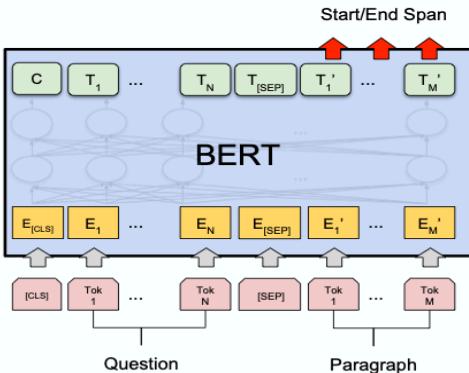


87.6

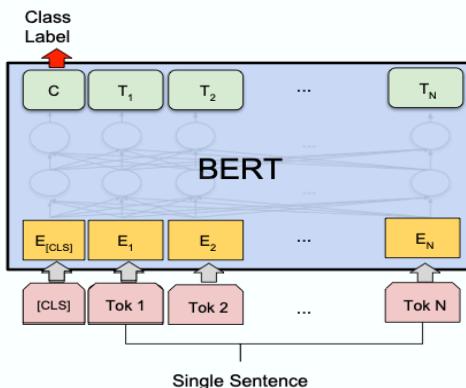
BERT vs. MT-DNN



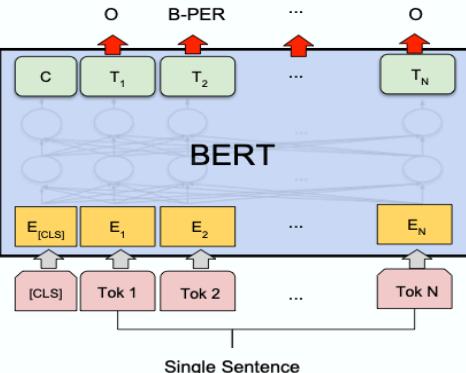
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(c) Question Answering Tasks:
SQuAD v1.1



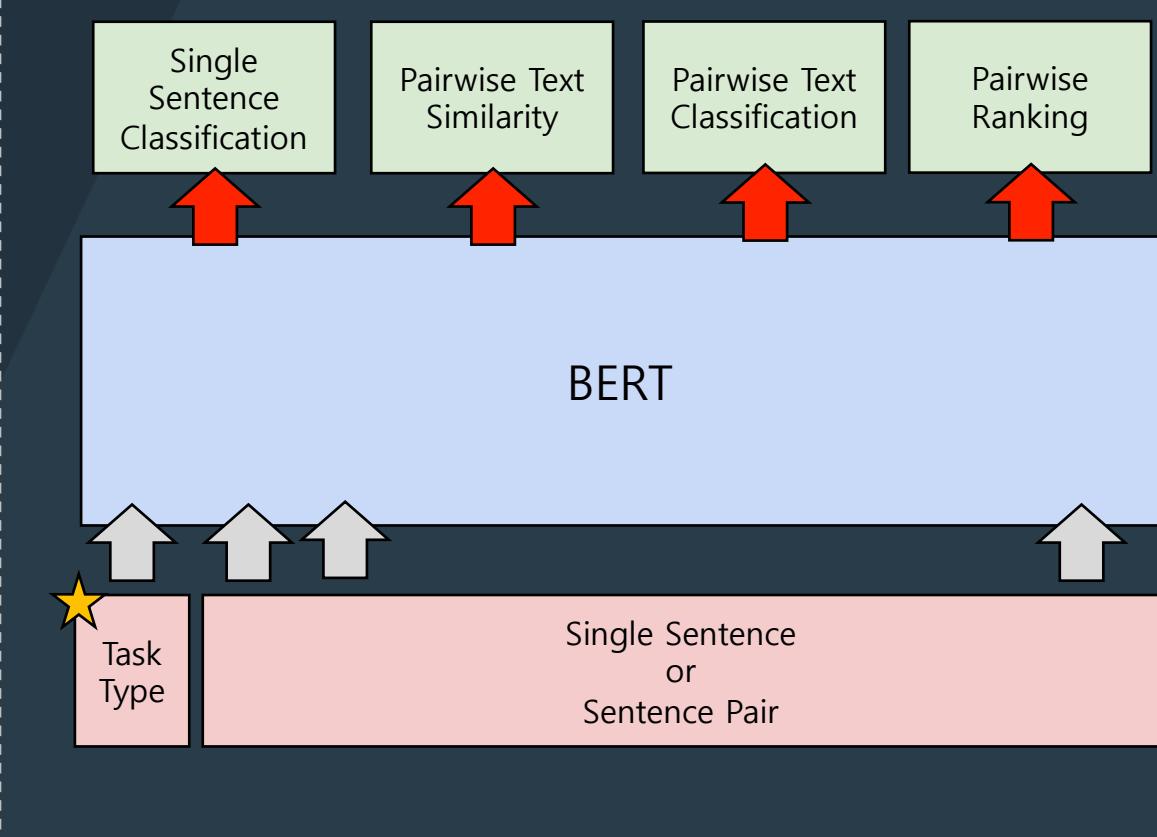
(b) Single Sentence Classification Tasks:
SST-2, CoLA



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

BERT

MTDNN



MT-DNN

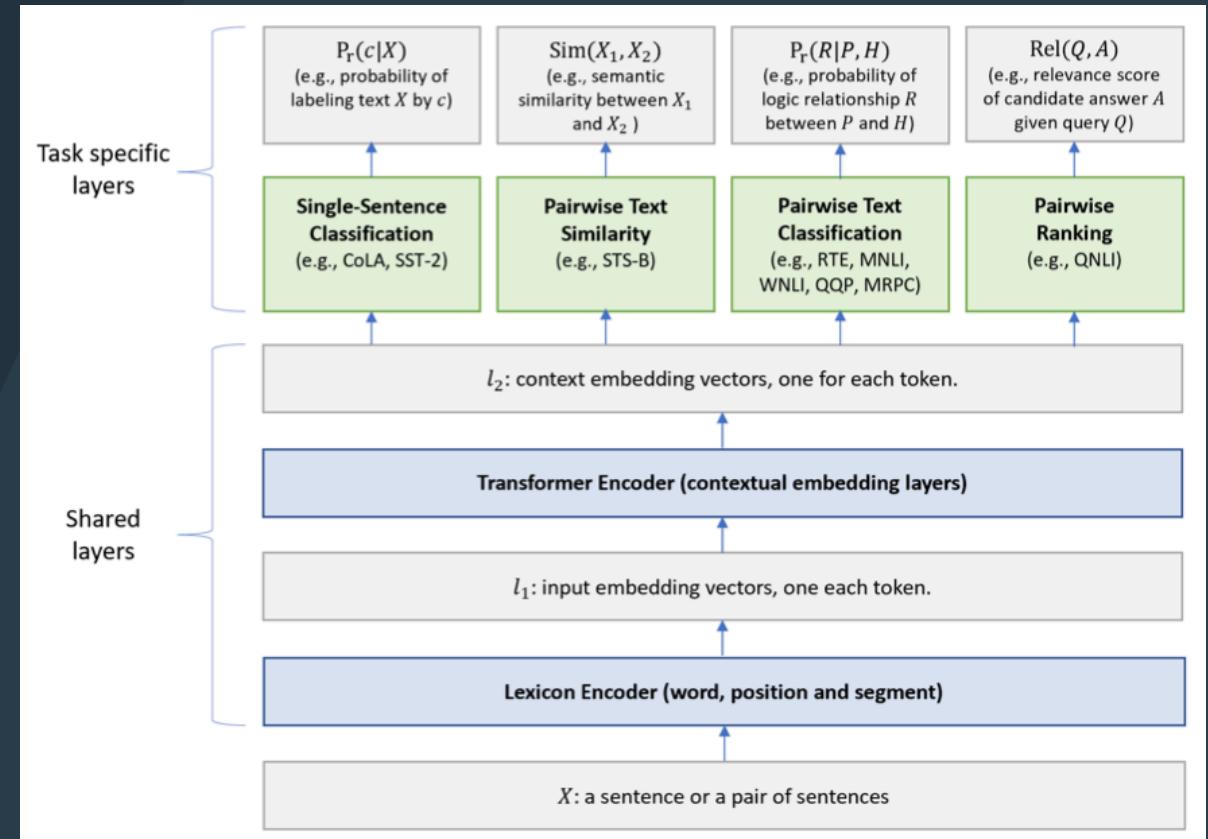
2019년 Microsoft에서 발표
하나의 BERT 모델이 다양한 Downstream Task를
지원하도록 하여 동시에 학습 함

pretraining stage

- Bert 훈련 절차와 동일함
- 기존 BERT 모델을 사용하여도 됨

multi-task learning stage

- GLUE, SNLI, SciTail 등의 데이터를 이용해 추가로 학습 함
- Downstream task마다 따로 모델을 만들지 않고 여러 Task를 한 모델에서 동시에 지원하도록 함



XLNet

(Generalized Autoregressive Pretraining)

6 XLNet Team

XLNet-Large (ensemble)



88.4

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 예측
하도록 하는 학습 방법

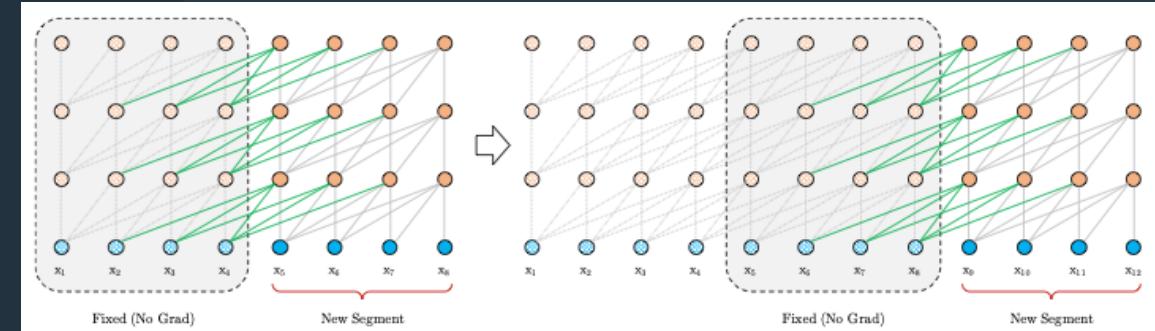
Transformer-XL

- Segment Recurrence
- Relative position embedding

XLNet-Transformer-XL

Segment Recurrence

- 이전 모델은 최대길이 초과하는 문장의 경우는 일부를 버려야 함
- XLNET은 긴 문장을 여러 Segment로 분리하고 이전 segment의 hidden state를 mem에 저장한 후 다음 segment에서 사용하는 방식으로 recurrence 하게 처리 함



Relative Position Embedding

- 이전 모델은 각 단어마다 절대 위치를 입력에 position embedding을 추가해서 결정 함
- XLNET은 입력에 주던 position embedding을 제거 하고, query와 key 사이의 상대적인 거리를 position embedding 으로 사용 함
- 일정 거리 이상은 최대 또는 최소 값으로 사용

-4
-5
-5
-5

-3
-4
-5
-5

-2
-3
-4
-5

-1
-2
-3
-4

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

XLNet-Two-Stream Self Attention

1 Content Stream

단어 자신의 정보를 포함

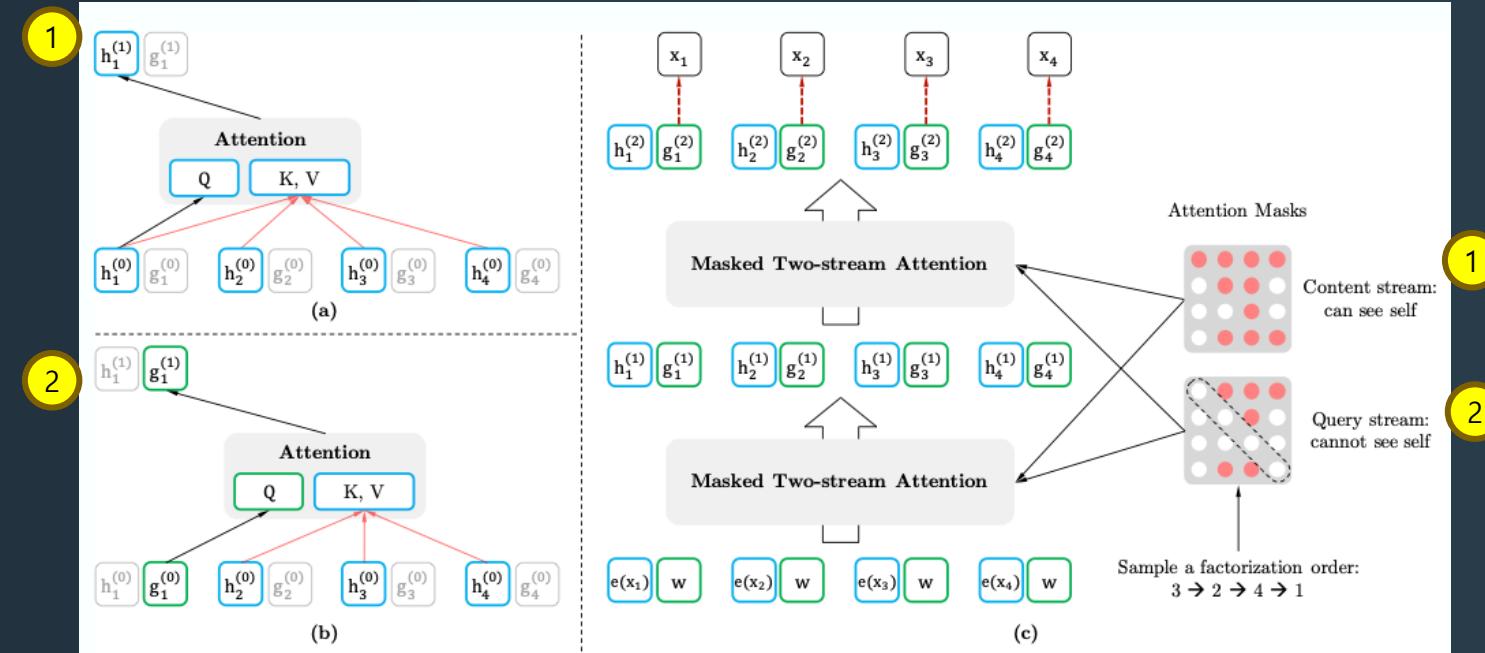
$$h_{z_t}^{(m)} \leftarrow \text{Attention}(Q = h_{z_t}^{(m-1)}, \text{KV} = h_{z < t}^{(m-1)}; \theta)$$

2 Query Stream

단어 자신의 정보를 제외 함

단어 예측을 위해 사용

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



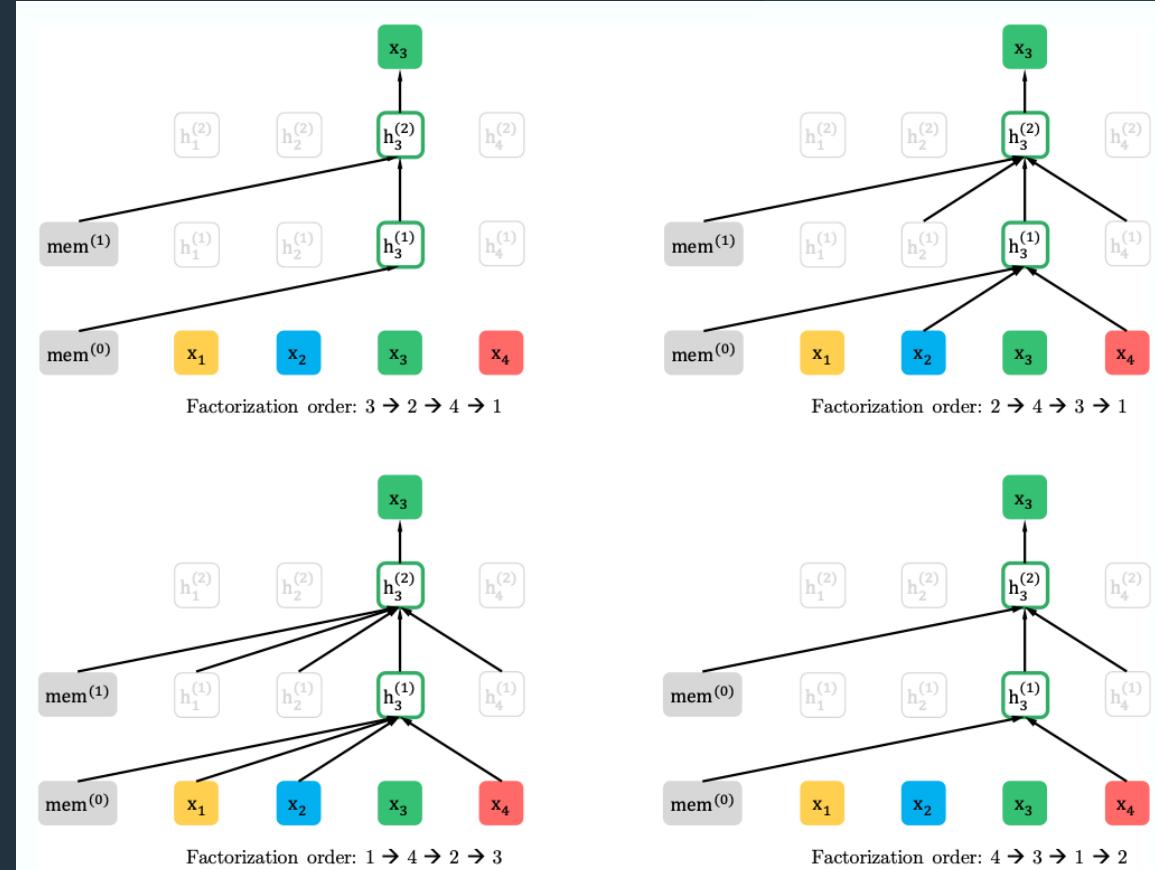
단어를 예측하기 위해서 예측할 단어의 정보가 없는 g 와
단어의 정보를 포함한 h 가
둘다 필요 함

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)
-----	------	-------	------	--------



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)
-----	------	-------	------	--------

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

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

더 다양한 단어들의 관계를 볼 수 있음

XLNet-Permutation LM

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

$3 \rightarrow 2 \rightarrow 4 \rightarrow 1$
입력: mem
출력: a(3)을 예측

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

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

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

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

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

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

$4 \rightarrow 3 \rightarrow 1 \rightarrow 2$
입력: mem, boy(4)
출력: a(3)을 예측

더 다양한 단어들의 관계를 볼 수 있음

RoBERTa

(A Robustly Optimized BERT Pretraining Approach)

5 Facebook AI

RoBERTa



88.5

RoBERTa

2019년 Facebook에서 발표

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

BERT

- 16G 데이터 사용
- Book Corpus
- English Wikipedia

RoBERTa

- 160G 데이터 사용
- Book Corpus, English Wikipedia (16G)
- CC-News (76G)
- Open Web Text (38G)
- Stories (31G)

RoBERTa-Static vs. Dynamic Masking

Static Masking

- Masking된 학습 데이터를 저장한 후 이 데이터를 반복해서 학습하는 방법

Dynamic Masking

- Masking 하지 않은 학습 데이터를 저장한 후 이 데이터를 매 epoch마다 동적으로 Masking 해서 학습하는 방법
- Static Masking에 비해서 Dynamic Masking이 성능이 약간 좋음

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

RoBERTa-Training with large batches

256 Batch size

- Steps 1M
- 총 학습 256,000,000 ($256 * 1,000,000$)

2K Batch size

- Steps 125K
- 총 학습 256,000,000 ($2,048 * 125,000$)
- 가장 성능이 좋음

8K Batch size

- Steps 31K
- 총 학습 256,000,000 ($8,192 * 31,250$)

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

ALBERT (A LITE BERT)

2 ALBERT-Team Google Language ALBERT (Ensemble)



89.4

ALBERT

2019년 Google에서 발표

BERT와 비교해서 parameter 수를 엄청나게 줄이면서도 성능은 더 좋아짐
(BERT Large 기준 1/18)

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

Cross-layer parameter sharing
Layer간에 parameter를 공유하도록 해서
parameter 개수를 줄임

Factorized embedding parameterization

Embedding의 크기를 줄이고 중간에 linear-layer를
추가해서 parameter 개수를 줄임

Sentence order prediction

동일한 단락 내의 문장의 순서를 바뀐 여부를 예측
하는 학습방법을 사용 함

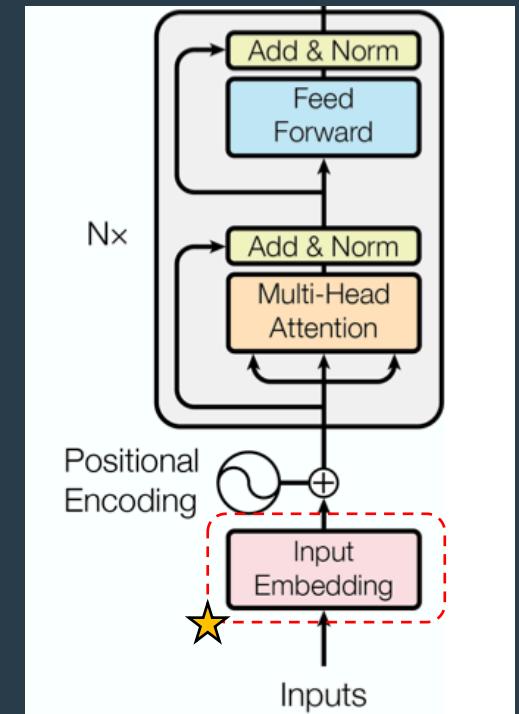
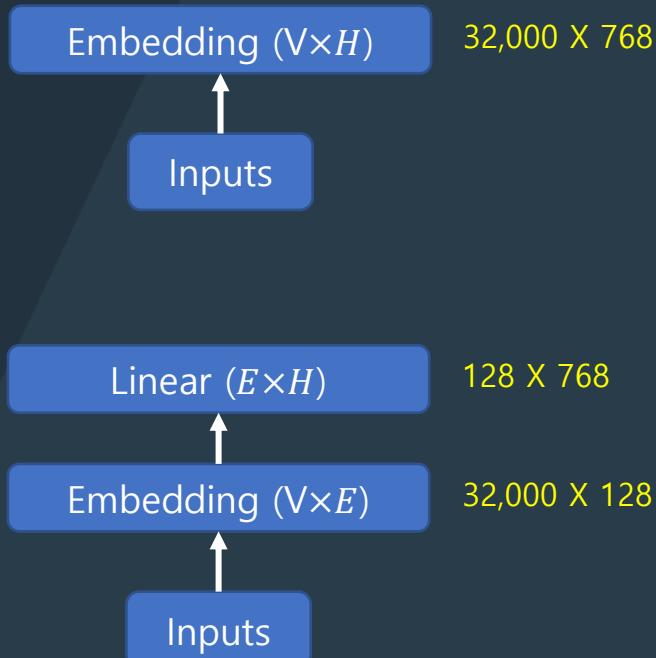
ALBERT-Factorized embedding parameterization

BERT-BASE

- vocab: 32,000
 - hidden: 768
 - parameter: 24,576,000 (32,000 * 768)

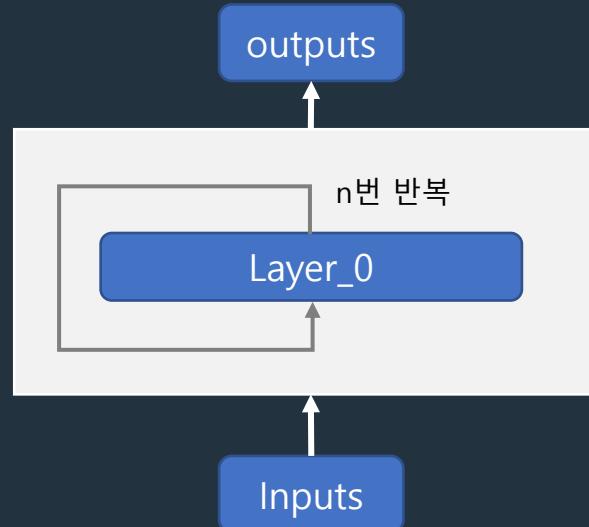
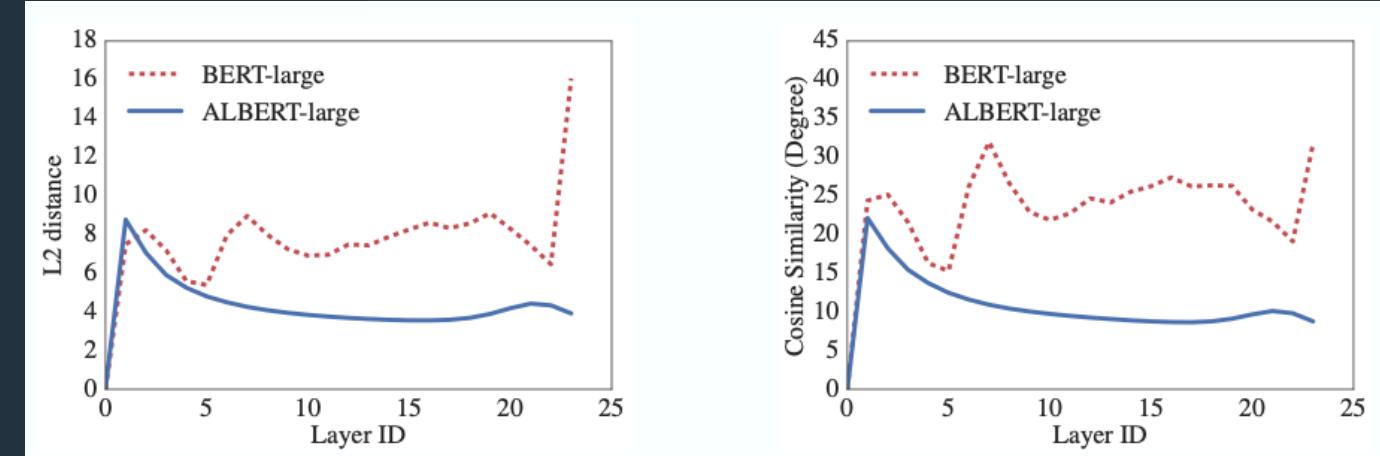
ALBERT-BASE

- vocab: 32,000
 - hidden: 128
 - linear: 128 x 768
 - parameter: 4,194,304 (32,000 * 128 + 128 * 768)

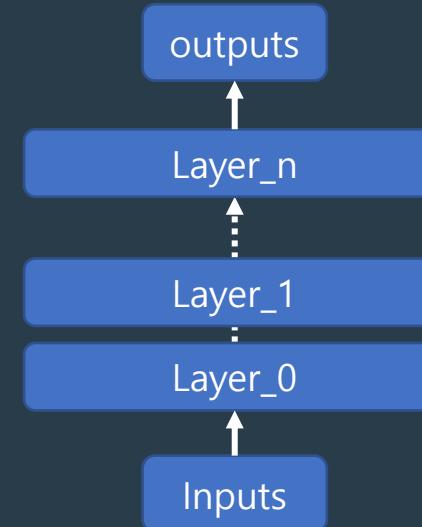


ALBERT-Cross-layer parameter sharing

- Layer간에 Parameter를 공유 함
- Layer 입/출력 간의 L2 distance와 cosine similarity를 비교해 보면 ALBERT가 BERT에 비교해서 상당히 안정적임



ALBERT



BERT

ALBERT-Sentence order prediction

BERT

- NSP는 False의 경우 다른 단락에서 문장을 가지고 옴
- 두 문장 간의 주제를 비교하는 문제로 MLM에 비해
서 문제가 너무 쉬워 학습이 잘 안됨

sentence_a = "신용 대출을 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

ALBERT

- SOP는 False일 경우 동일한 단락의 연속된 문장의
순서를 바꿈
- 두 문장 간의 의미를 비교하는 문제로 NSP에 비해서
어려운 문제 임

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

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

T5

(Text-to-Text Transfer Transformer)

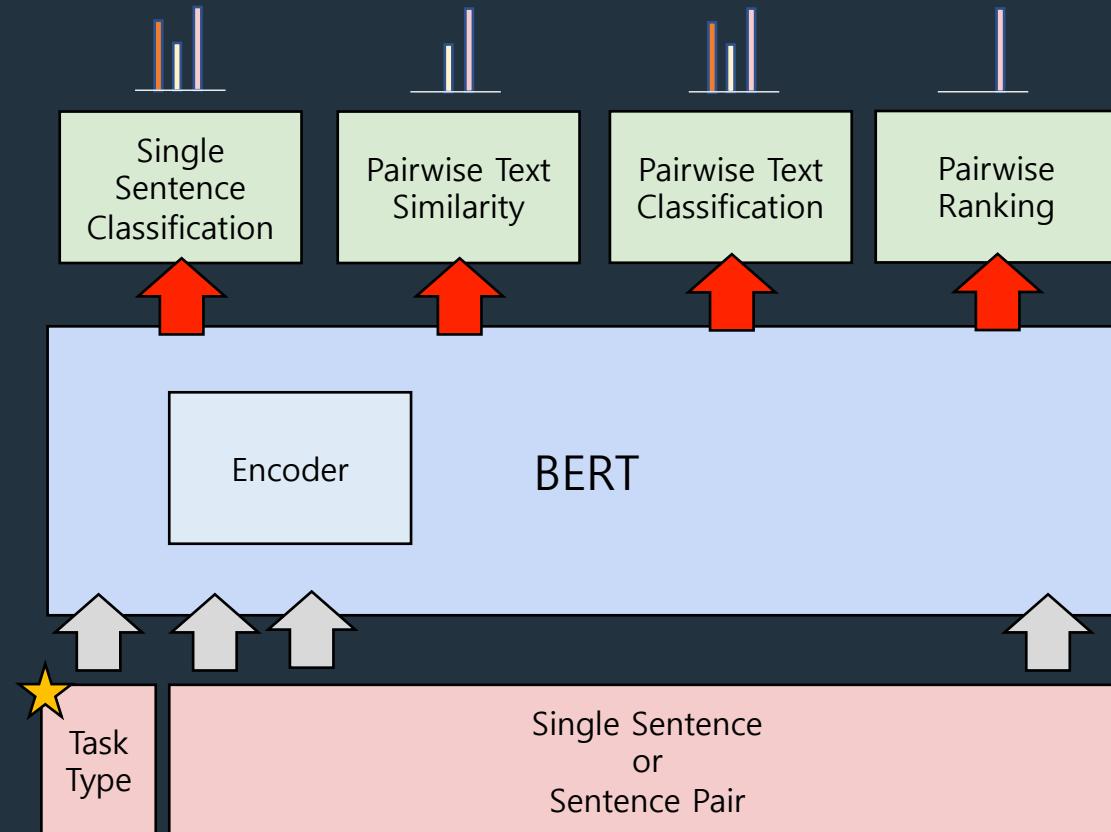
1 T5 Team - Google

T5

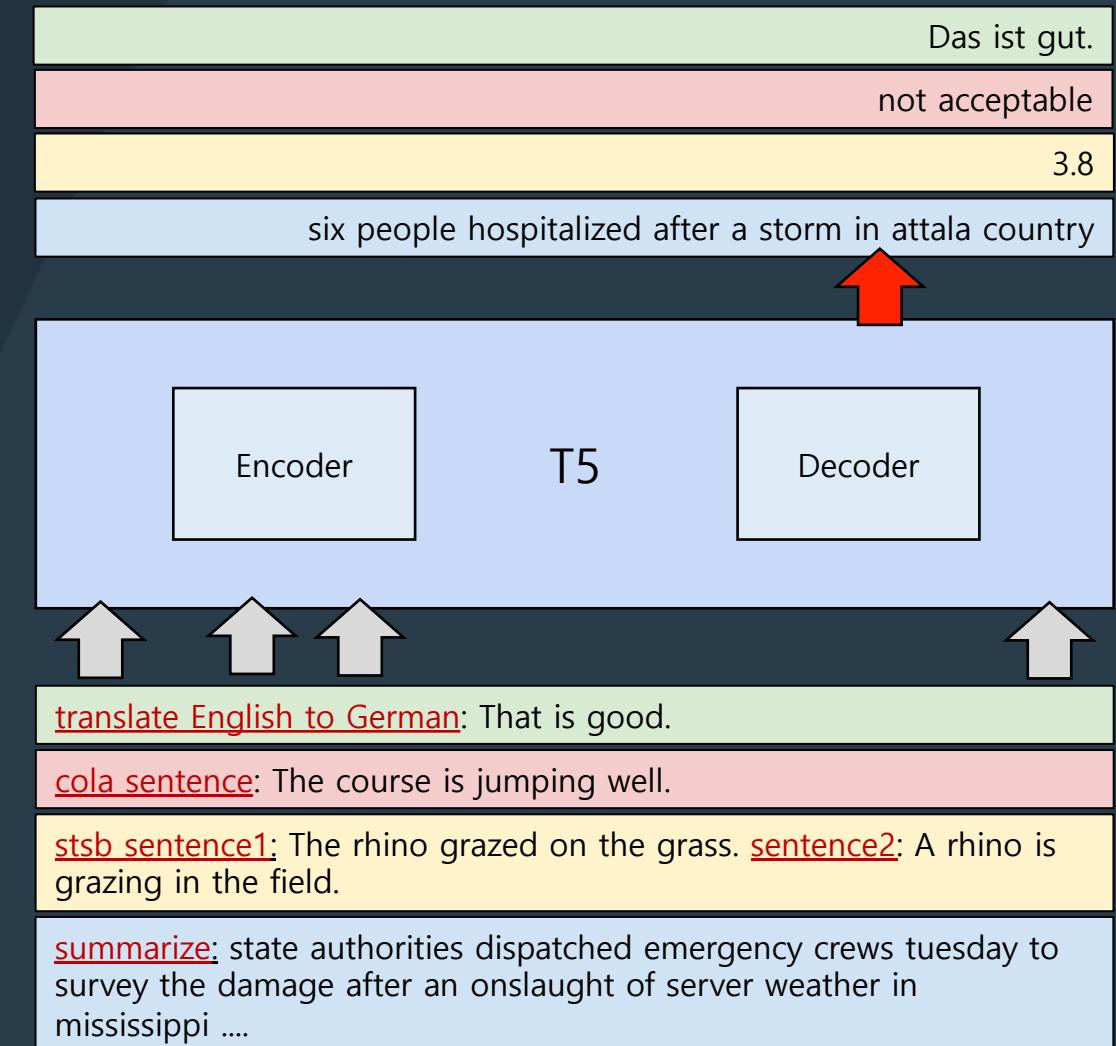


89.7

MT-DNN vs. T5



MTDNN



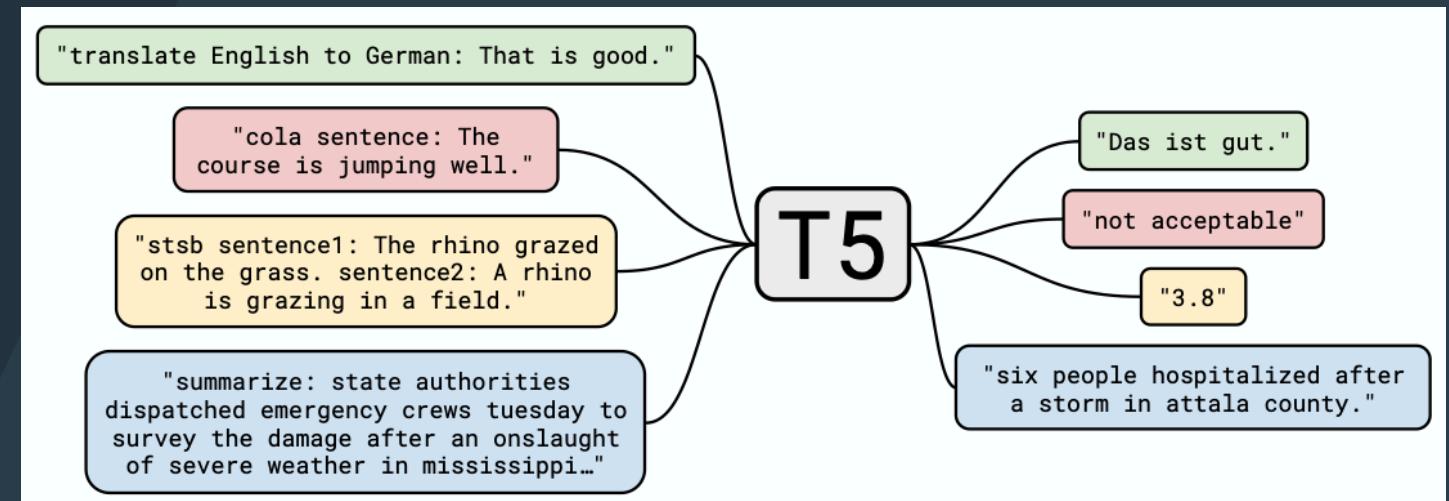
T5

T5

2019년 Google에서 발표
Standard Transformer (Encoder/Decoder) 사용함

Text to Text

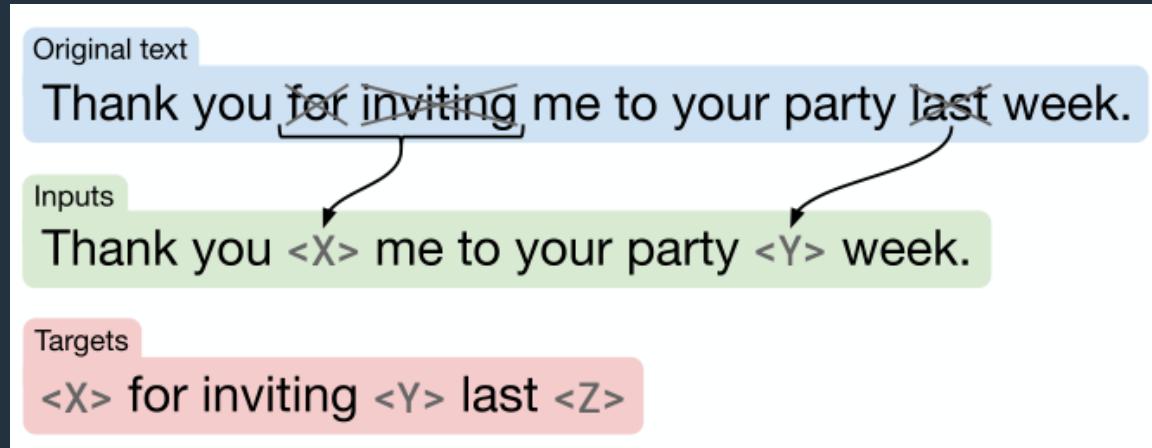
- 입력과 출력을 모두 Text로 처리하도록 하는 간단한 구조를 사용



T5

Denoising objectives

- 일정 영역을 MASK한 후 예측하도록 함



C4(Colossal Clean Crawled Corpus)

- 1개월간 Crawling한 데이터 20TB중 정해진 룰에 따라 데이터를 정제 함
- 750GB 학습 데이터 사용

Summary

GPT

- Transformer Decoder 사용
- 이전 단어를 보고 다음 단어를 예측하는 LM

BERT

- Transformer Encoder 사용
- 입력 단어에 일정 비율로 Mask를 한 후 단어를 예측하는 MLM
- 두 문장이 동일 단락에 속한 문장인지를 예측하는 NSP

SpanBERT

- BERT 사용
- 입력 단어의 일정 영역(Span)을 Mask한 후 예측 (MLM + SBO)
- NSP를 사용하지 않음

MT-DNN

- BERT 사용
- 한 모델에서 여러 Downstream Task 지원하도록 구성한 후 동시에 학습

XLNet

- Transformer Decoder 사용
- 단어 순서를 섞은 후 예측하는 Permutation LM 사용
- 긴 문장을 처리하기 위해 memory 사용 (Segment Recurrence)
- Query, Key 간의 상대적 위치 사용 (Relative position embedding)

RoBERTa

- BERT 사용
- BERT 16G 보다 10배 많은 데이터 사용 (160G)
- BERT 256 보다 큰 배치 사용 (2K)
- Dynamic Mask 사용
- NSP를 사용하지 않음

ALBERT

- BERT 사용
- 작은 Embedding에 Linear를 추가 함
- Layer간에 Parameter를 공유 함
- NSP 대신 문장의 순서를 맞추는 더 어려운 SOP 사용

T5

- Standard Transformer 사용 (Encoder / Decoder)
- 입력과 출력을 모두 Text로 처리 함
- Mask된 단어를 맞추는 Denoising objective 사용
- 인터넷을 Crawling하여 많은 학습 데이터 사용 (750G)

사람은 어떻게 기억 하는가? →



감사합니다

cchyun@gmail.com

GitHub

<https://github.com/paul-hyun/transformer-evolution>

Visualization

<https://app.wandb.ai/cchyun/transformer-evolution>