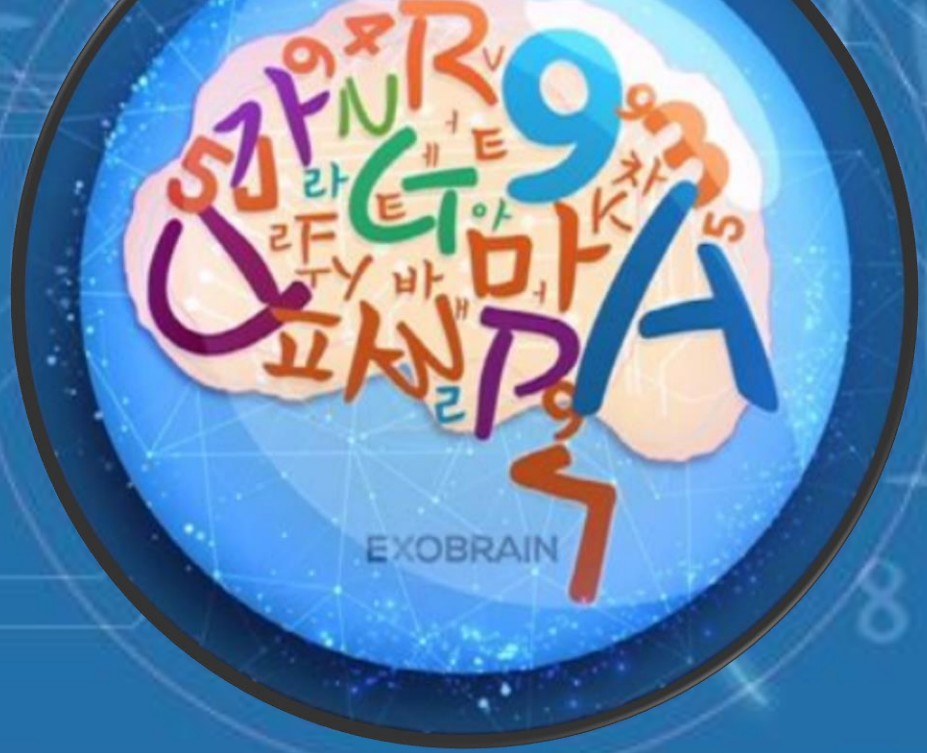


BERT 언어모델과 미세조정(Finetuning)

언어지능연구실 / 류지희 / 2022. 09. 19.

https://github.com/chrisjihee/DeepKorean.git

guest@129.254.164.137:git/transformers-4.12.5

BERT 언어모델 등장 과정

주요 참고문헌

- [Vaswani2017] A. Vaswani, et al., “Attention is all you need,” NIPS.
- [Han2022] K. Han, et al., “A survey on vision transformer,” IEEE Tr. PAMI.
- [Khan2022] S. Khan, et al., “Transformers in vision: a survey,” ACM Computing Surveys.
- [오일석2021] 오일석, 이진선, 파이썬으로 만드는 인공지능, 한빛아카데미.
- [오일석2022] 오일석, 컴퓨터비전과 딥러닝, 11장(비전 트랜스포머) [출간 예정]

(오일석 교수님 강의 자료를 교수님 동의 하에 발췌하였음을 밝힙니다)

인간의 주목

- 인간은 인식할 때 의도$_{intention}$에 따라 특정한 곳에 주목$_{attention}$
  - 질문에 따라 의도가 형성되고 의도에 따라 주목할 곳을 정함
  - 인간의 주목 맵 측정 [Das2016], 뇌가 어떻게 주목을 달성하는지 밝힘 [Corbetta2002]

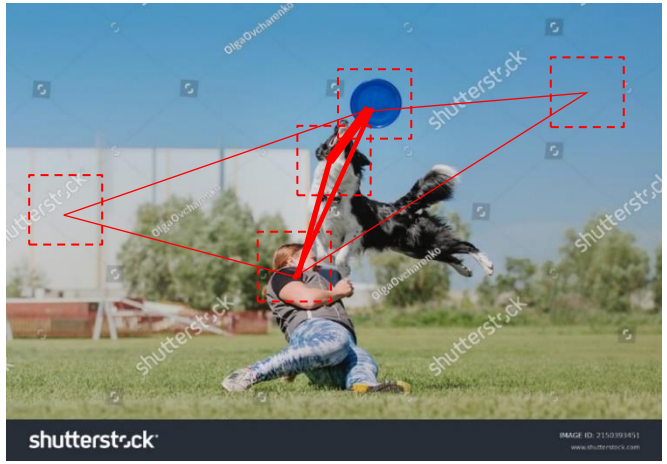
- 기계의 주목
  - 예) VQA(visual question answering) 시스템: 주목을 잘 처리해야 높은 성능 달성
  - 컴퓨터비전은 분류, 검출, 분할, 추적 등의 문제를 푸는데 주목 적용

딥러닝의 주목

- 주목
  - 영상의 중요한 부분에 더 큰 가중치를 주어 성능 개선
  - 그림에서 색의 농도로 주목하는 정도를 표시

주목

자기 주목



- 자기 주목self attention
  - 영상을 구성하는 요소 상호간에 주목을 결정
  - 그림에서는 선의 굵기로 주목하는 정도를 표시

자기 주목

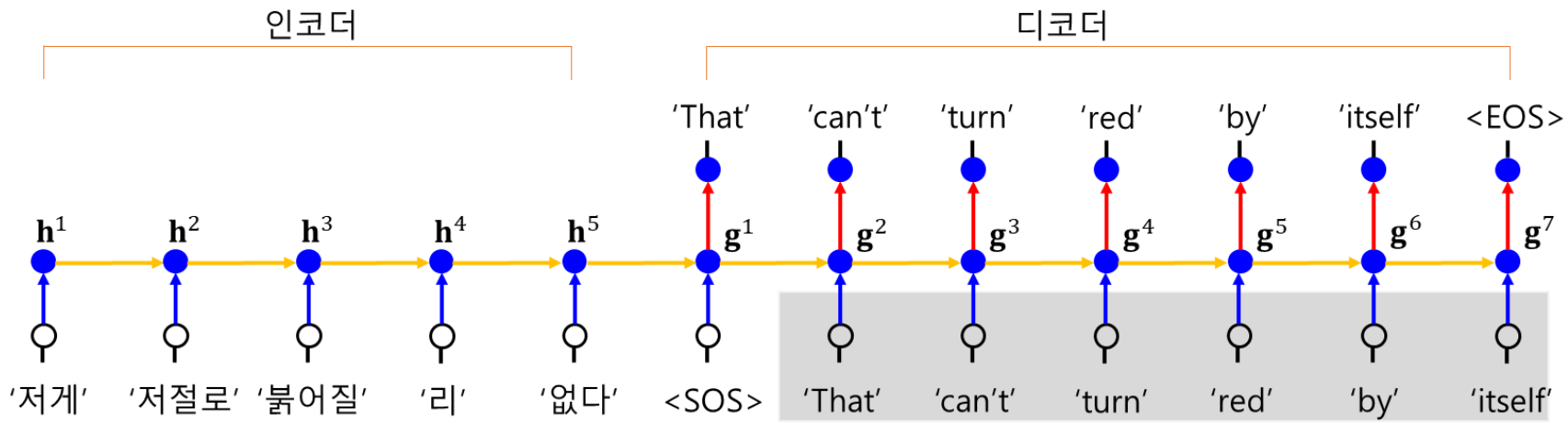
- 초기 자기 주목
  - Non-local 신경망은 자기 주목을 컴퓨터비전에 최초 도입한 논문 [Wang2018]
    - 아래 식으로 자기 주목 계산: 출력 특징 맵 $\mathbf{y}$는 입력 특징 맵 $\mathbf{x}$와 같은 크기
    - $\mathbf{y}$의 i 위치의 값 $\mathbf{y}_i$는 $\mathbf{x}$의 모든 위치의 값인 $g(\mathbf{x}_j)$의 가중치 합
    - 가중치 $f(\mathbf{x}_i, \mathbf{x}_j)$는 $\mathbf{x}_i$가 $\mathbf{x}_j$에 얼마나 주목해야하는지 나타내는 값
    - $f(\mathbf{x}_i, \mathbf{x}_j)$는 주로 내적과 같이 $\mathbf{x}_i$와 $\mathbf{x}_j$의 유사도를 측정해주는 함수 사용
    - 보통 컨볼루션층을 여러 번 거쳐 7*7처럼 작아진 특징 맵에 적용

$$\mathbf{y}_i = \frac{1}{C(\mathbf{x})} \sum_{\text{모든 } j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j)$$

- 2017년 트랜스포머 등장
  - 자연어처리 분야의 혁신
  - 2020~2022년에는 컴퓨터비전에 적용한 성공 사례 봇물

Seq2seq 모델의 등장

- 서츠케버의 seq2seq 모델
  - 2014년에 발표된 자연어처리 분야의 획기적 아이디어
  - 가변 길이의 문장을 가변 길이의 문장으로 변환, 예) 언어 번역
  - 자기 회귀 방식으로 동작
  - 큰 공헌 했지만 한계를 보임: 인코더의 마지막 은닉 상태만 디코더로 전달함

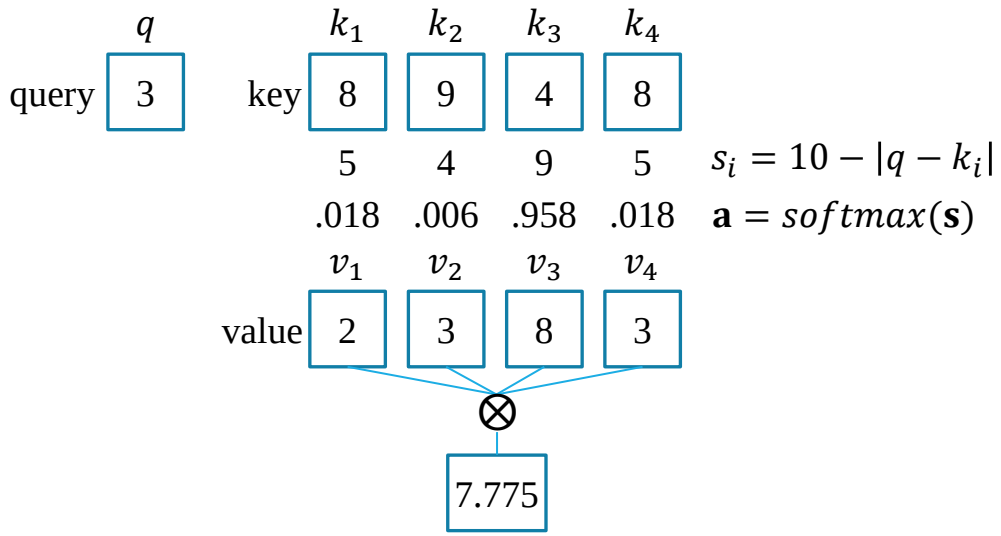


Query-Key-Value로 계산하는 주목

- 주목 계산 방법이 여럿인데, 최근에는 query-key-value 방식을 주로 사용
  - Query가 key와 유사한 정도를 측정하고 value를 가중 합함
  - 트랜스포머가 이 방법을 사용하므로 확실히 이해해야 함

Query-Key-Value로 계산하는 주목

- [예제 1] 1~10 자연수



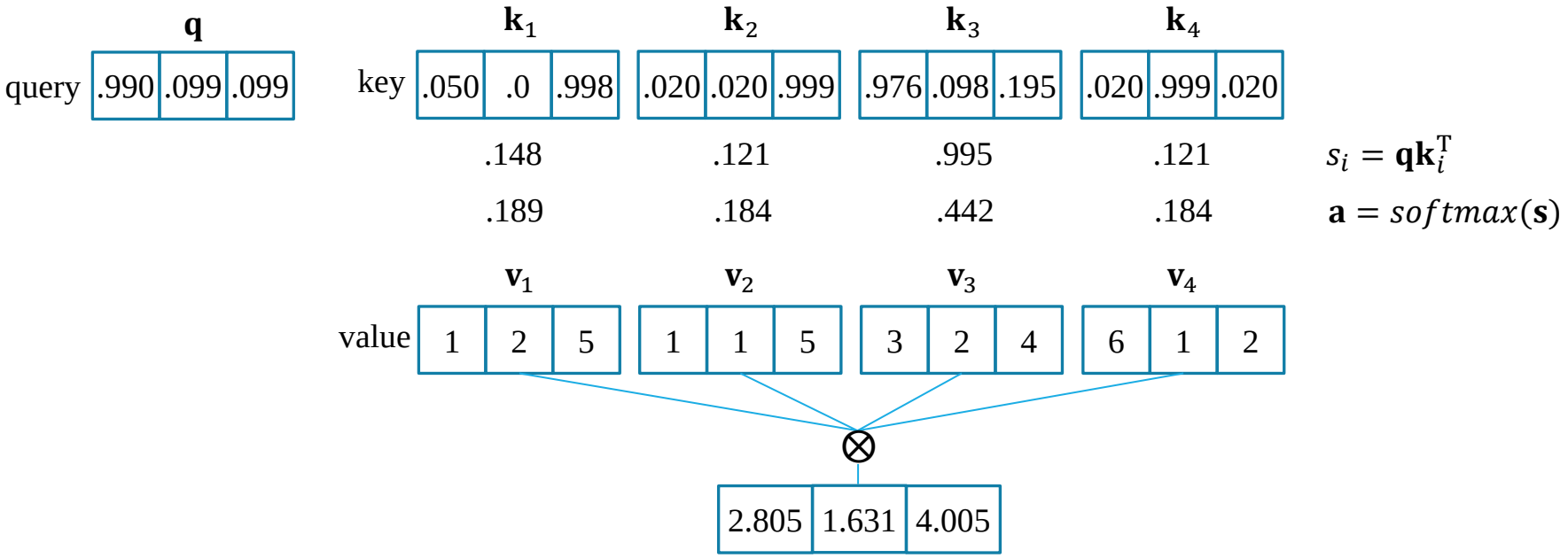
$$c = 0.018 * 2 + 0.006 * 3 + 0.958 * 8 + 0.018 * 3 = 7.775$$

- Query q는 세번째 key k3과 가장 유사하여 세번째 value k3에 가장 많이 주목함

$$c = softmax\big(similarity(q, k_1), similarity(q, k_2), \cdots, similarity(q, k_T)\big)\begin{pmatrix} v_1 \\ \vdots \\ v_T \end{pmatrix}$$

Query-Key-Value로 계산하는 주목

- [예제 2] 벡터로 확장



$$\mathbf{c} = 0.189 * (1 \;\; 2 \;\; 5) + 0.184 * (1 \;\; 1 \;\; 5) + 0.442 * (3 \;\; 2 \;\; 4) + 0.184 * (6 \;\; 1 \;\; 2) = (2.805 \;\; 1.631 \;\; 4.005)$$

Query-Key-Value로 계산하는 주목

- [예제 3] 행렬로 확장

$$\mathbf{q} = (0.990 \quad 0.099 \quad 0.099), \mathbf{K} = \begin{pmatrix} 0.050 & 0.000 & 0.998 \\ 0.020 & 0.020 & 0.999 \\ 0.976 & 0.098 & 0.195 \\ 0.020 & 0.999 & 0.020 \end{pmatrix}, \mathbf{V} = \begin{pmatrix} 1 & 2 & 5 \\ 1 & 1 & 5 \\ 3 & 2 & 4 \\ 6 & 1 & 2 \end{pmatrix}$$

주목 벡터

$$\mathbf{a} = softmax(\mathbf{qK}^{\mathrm{T}}) = softmax\left((0.990 \quad 0.099 \quad 0.099)\begin{pmatrix} 0.050 & 0.020 & 0.976 & 0.020 \\ 0.000 & 0.020 & 0.098 & 0.999 \\ 0.998 & 0.999 & 0.195 & 0.020 \end{pmatrix}\right)$$

$$= (0.189 \quad 0.184 \quad 0.442 \quad 0.184)$$

문맥 벡터

$$\mathbf{c} = softmax(\mathbf{qK}^{\mathrm{T}})\mathbf{V} = \mathbf{aV}$$

$$= (0.189 \quad 0.184 \quad 0.442 \quad 0.184)\begin{pmatrix} 1 & 2 & 5 \\ 1 & 1 & 5 \\ 3 & 2 & 4 \\ 6 & 1 & 2 \end{pmatrix}$$

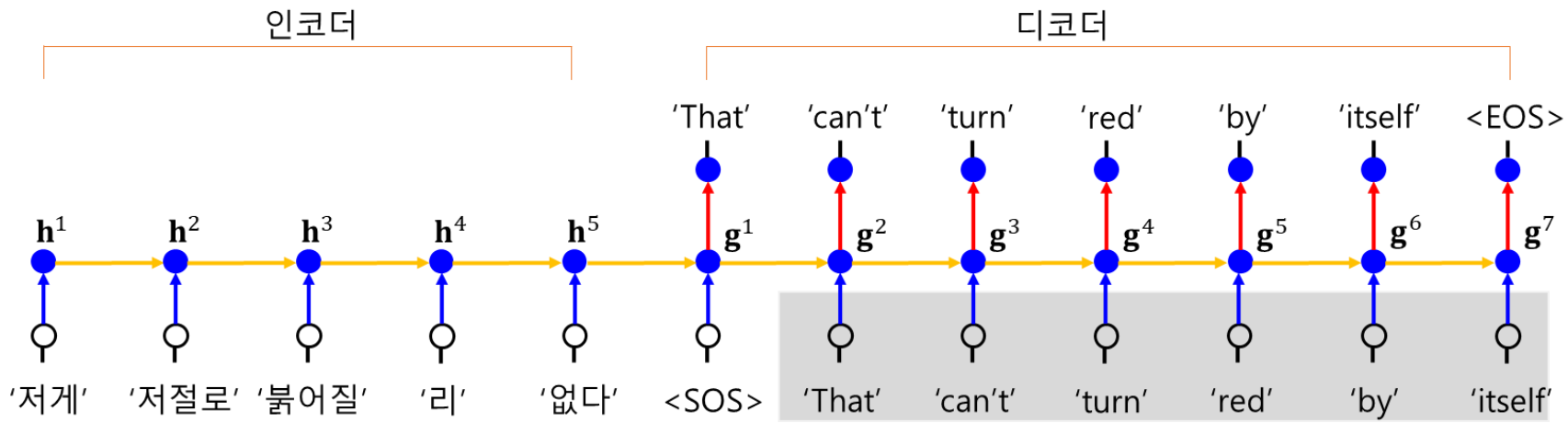
$$= (2.805 \quad 1.631 \quad 4.005)$$

$$\mathbf{c} = softmax(\mathbf{qK}^{\mathrm{T}})\mathbf{V}$$

남은 일은 무엇을 query로 하고, 무엇을 key로 하고, 무엇을 value로 할 지 결정하는 것뿐

주목을 반영한 seq2seq 모델

- 바다나우의 아이디어
  - 디코더가 인코더의 모든 상태 $\mathbf{h}^1$, $\mathbf{h}^2$, ...에 접근 허용하고 주목을 적용함
  - 예) 순간 6에서 인코더의 순간 2의 '저절로 ' 에 주목해야 'itself'를 제대로 생성할 수 있을것임. 디코더의 순간 6에서 주목 벡터 $\mathbf{a}=(0.01, 0.9, 0.02, 0.03, 0.04)$이면 바람직
  - 무엇을 query, key, value로 해야 이런 의도가 반영될까?



주목을 반영한 seq2seq 모델

- [예제 3] 가정된 상황의 행렬 연산

$$\mathbf{g}^5 = (0.2,0.9,0.0)$$
$$\mathbf{h}^1 = (0.1,0.0,0.8)\ \mathbf{h}^2 = (0.1,0.9,0.0)\ \mathbf{h}^3 = (0.0,0.1,0.8)\ \mathbf{h}^4 = (0.2,0.1,0.6)\ \mathbf{h}^5 = (0.9,0.0,0.1)$$

  - 이때 $\mathbf{g}^5$가 query이고, $\mathbf{h}^1, \mathbf{h}^2, \cdots, \mathbf{h}^5$가 key와 value
  - 이것을 행렬로 표현하면

$$\mathbf{q} = (0.2\ \ 0.9\ \ 0.0), \mathbf{K} = \begin{pmatrix} 0.1 & 0.0 & 0.8 \\ 0.1 & 0.9 & 0.0 \\ 0.0 & 0.1 & 0.8 \\ 0.2 & 0.1 & 0.6 \\ 0.9 & 0.0 & 0.1 \end{pmatrix}, \mathbf{V} = \begin{pmatrix} 0.1 & 0.0 & 0.8 \\ 0.1 & 0.9 & 0.0 \\ 0.0 & 0.1 & 0.8 \\ 0.2 & 0.1 & 0.6 \\ 0.9 & 0.0 & 0.1 \end{pmatrix}$$

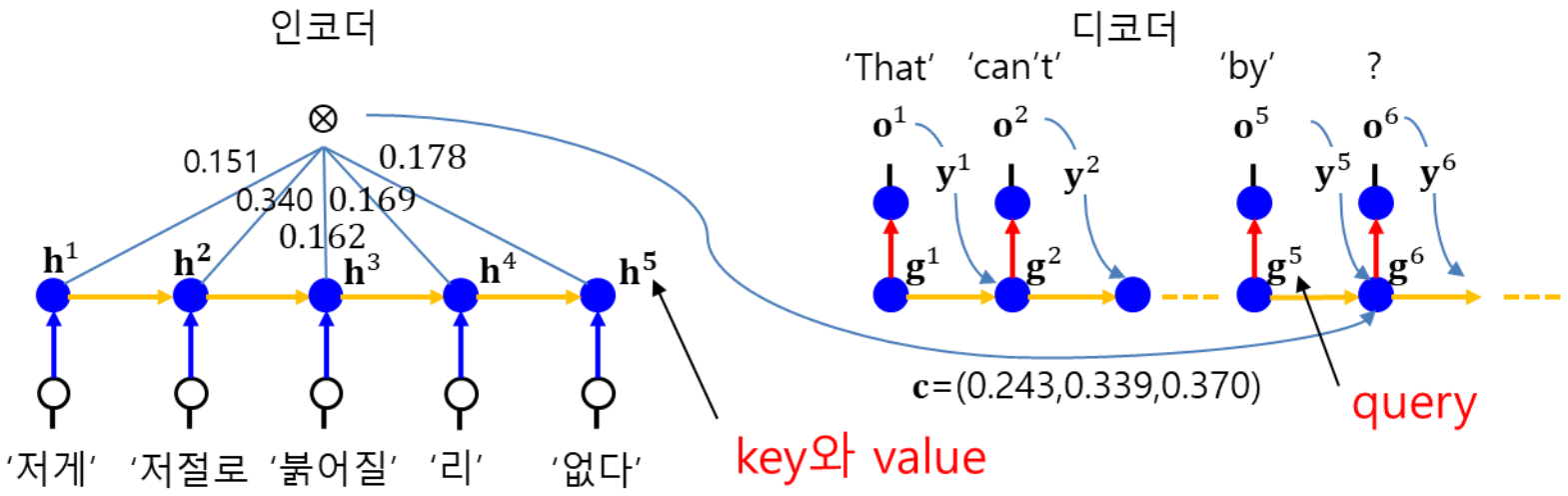
  - 식을 적용하면

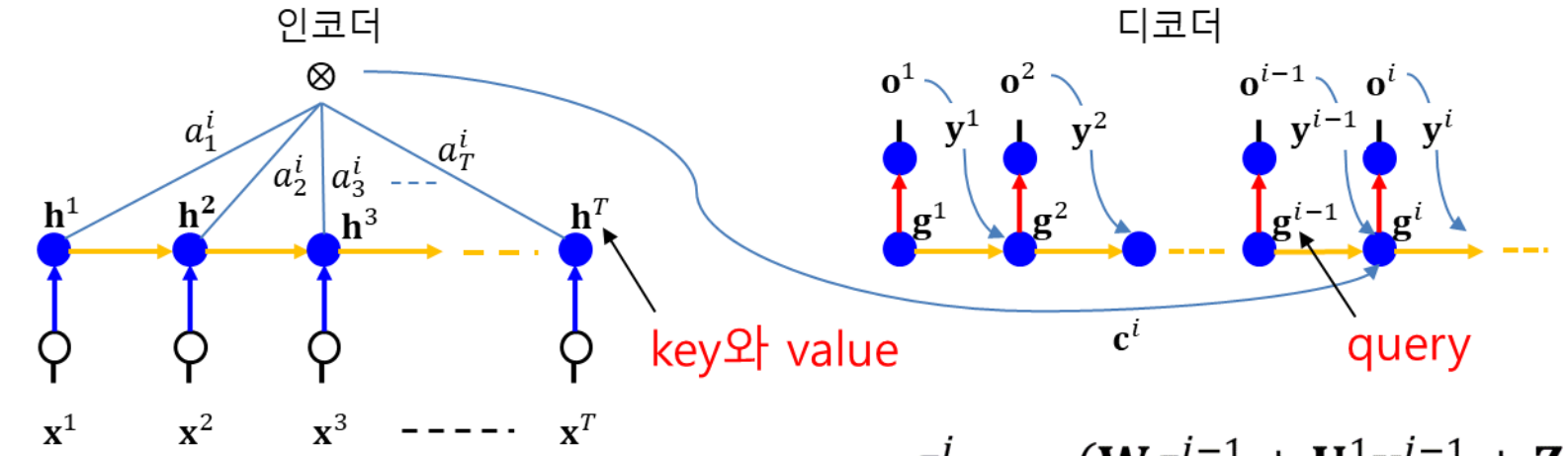
$$\mathbf{c} = softmax(\mathbf{q}\mathbf{K}^{\mathrm{T}})\mathbf{V} = softmax\left((0.2\ \ 0.9\ \ 0.0)\begin{pmatrix} 0.1 & 0.1 & 0.0 & 0.2 & 0.9 \\ 0.0 & 0.9 & 0.1 & 0.1 & 0.0 \\ 0.8 & 0.0 & 0.8 & 0.6 & 0.1 \end{pmatrix}\right)\begin{pmatrix} 0.1 & 0.0 & 0.8 \\ 0.1 & 0.9 & 0.0 \\ 0.0 & 0.1 & 0.8 \\ 0.2 & 0.1 & 0.6 \\ 0.9 & 0.0 & 0.1 \end{pmatrix}$$

$$= (0.151\ \ 0.340\ \ 0.162\ \ 0.169\ \ 0.178)\begin{pmatrix} 0.1 & 0.0 & 0.8 \\ 0.1 & 0.9 & 0.0 \\ 0.0 & 0.1 & 0.8 \\ 0.2 & 0.1 & 0.6 \\ 0.9 & 0.0 & 0.1 \end{pmatrix}$$

$$= (0.243\ \ 0.339\ \ 0.370)$$

주목을 반영한 seq2seq 모델



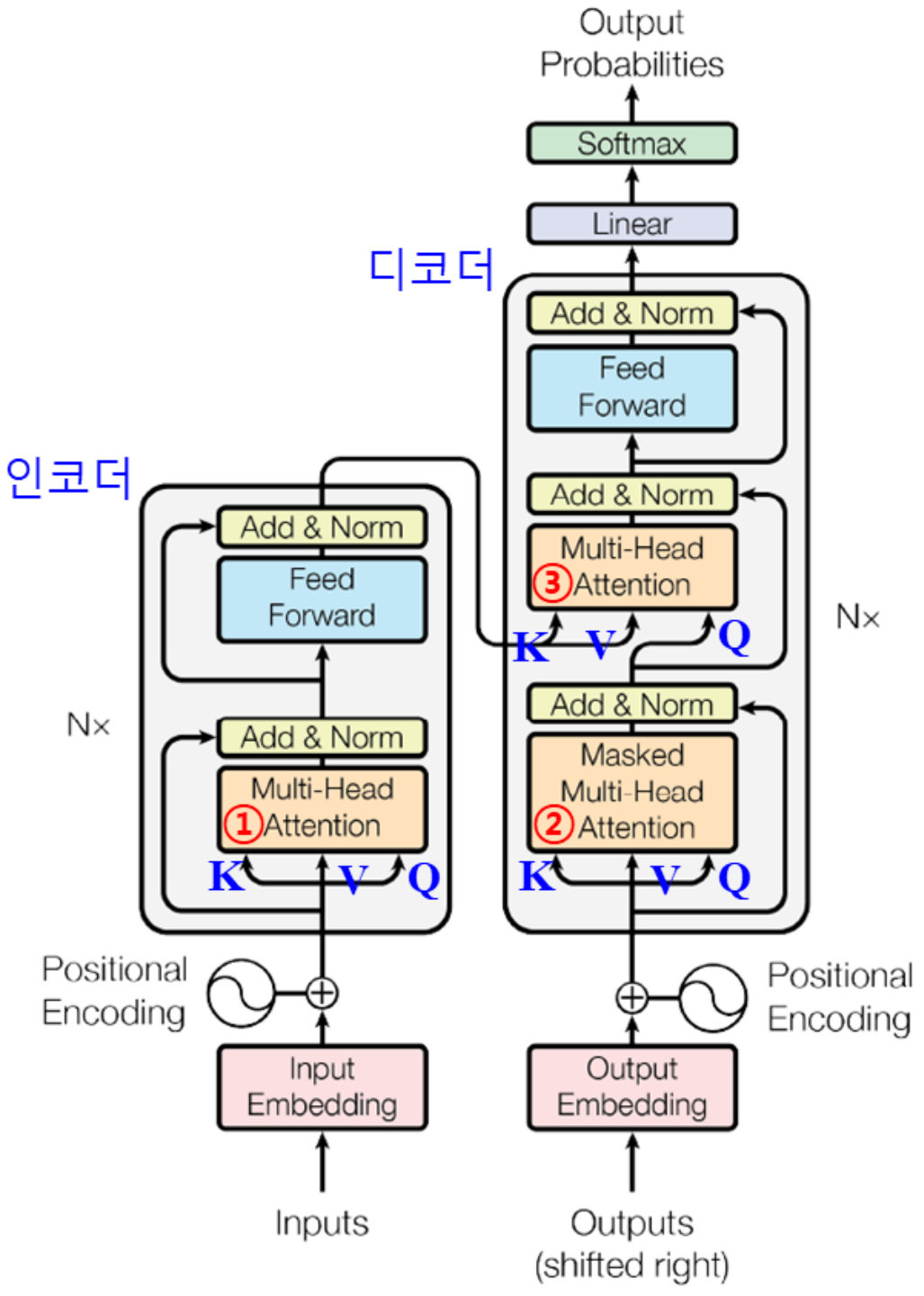


$$\left.\begin{aligned}\mathbf{g}^i &= \tau_1(\mathbf{W}\mathbf{g}^{i-1} + \mathbf{U}^1\mathbf{y}^{i-1} + \mathbf{Z}\mathbf{c}^i)\\ \mathbf{o}^i &= \tau_2(\mathbf{U}^2\mathbf{g}^i)\end{aligned}\right\}$$

트랜스포머의 등장

- “Attention is all you need” 논문
  - 구글 연구진이 2017년 발표
  - 자연어 처리를 위한 혁신적인 모델 “트랜스포머” 제안
  - 컨볼루션 신경망과 순환 신경망의 구성요소를 모두 없애고 주목만으로 신경망을 구현

트랜스포머의 기본 구조


Output Probabilities
Softmax
Linear
디코더
Add & Norm
Feed Forward
Add & Norm
Multi-Head ③ Attention
K V Q
Add & Norm
Masked Multi-Head ② Attention
K V Q
Nx
인코더
Add & Norm
Feed Forward
Add & Norm
Multi-Head ① Attention
K V Q
Nx
Positional Encoding
Input Embedding
Inputs
Positional Encoding
Output Embedding
Outputs (shifted right)

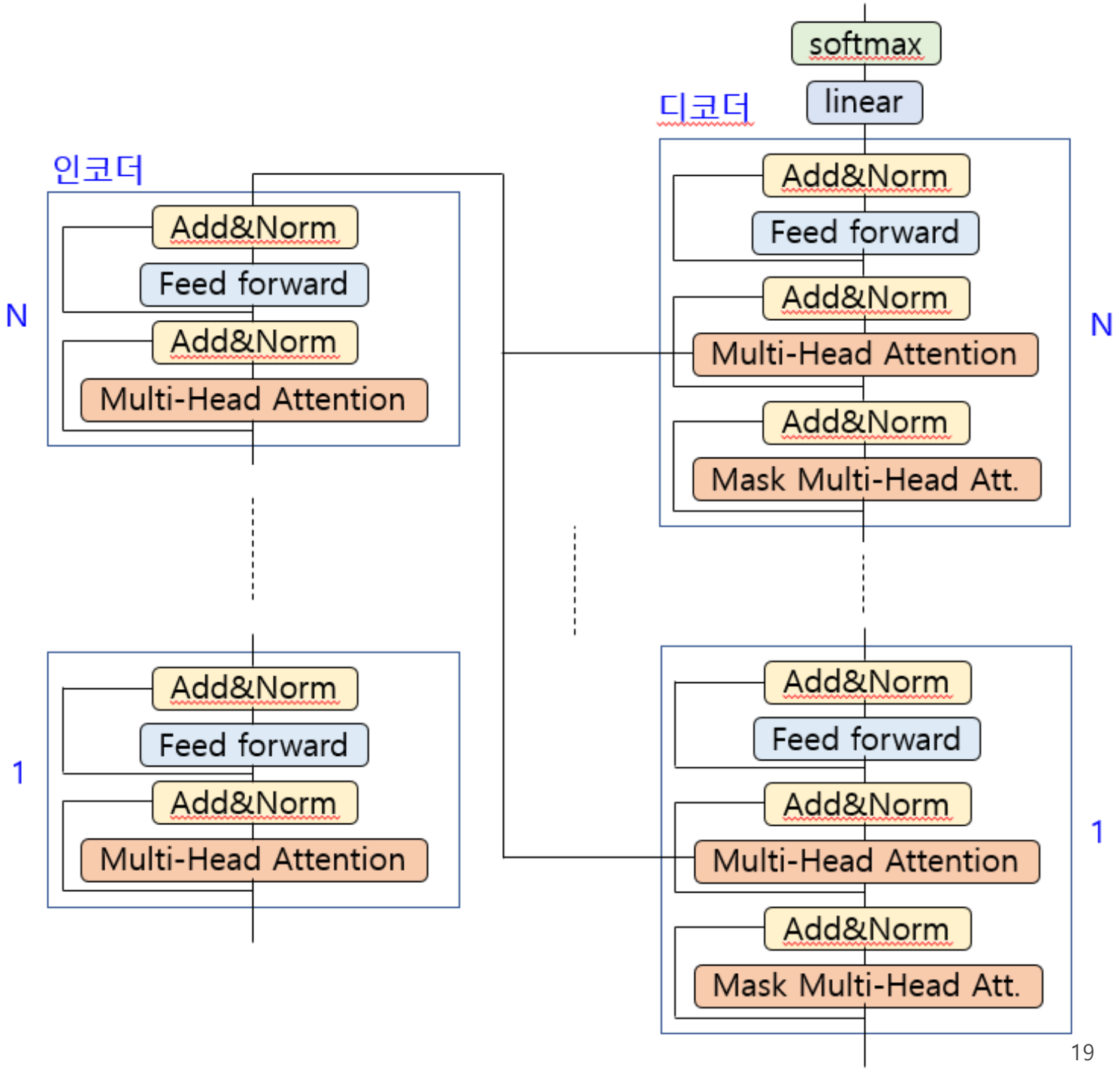
트랜스포머의 기본 구조

- 인코더와 디코더로 구성
  - 한영번역 트랜스포머라면, 인코더에 한국어 입력, 디코더에서 영어 출력
  - 문장의 모든 단어를 한꺼번에 입력 → 자기 주목이 가능해짐. 병렬 처리 유리
    - 단어는 임베딩을 통해 d_{model} 차원의 임베딩 벡터로 표현. 논문은 d_{model}=512
    - 위치 인코딩으로 위치 정보 추가
  - MHA층이 자기 주목을 담당. 헤드를 여러 개 사용하여 성능 향상
  - FF층
  - Add&Norm 적용
  - N개의 인코더 및 디코더층을 반복해서 쌓음 (N=6)

트랜스포머의 기본 구조

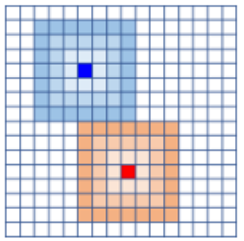
- 세 개의 MHA층이 주목을 담당
  - 인코더의 ①과 디코더의 ② 표시한 MHA층은 자기 주목을 담당
    - 문장을 구성하는 단어끼리 주목을 처리
  - 디코더의 ③ 표시한 MHA는 자기 주목이 아님
    - 인코더와 디코더 사이의 주목을 담당
    - 인코더의 상태 벡터가 key와 value, 디코더의 상태 벡터는 query로 작용함
- MHA로 처리하는 자기 주목은 트랜스포머의 핵심 아이디어!

트랜스포머의 전체 동작

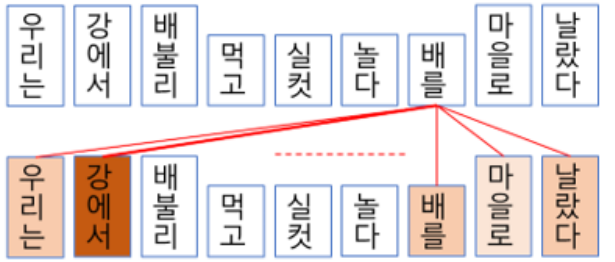

softmax
linear
디코더
인코더
Add&Norm
Feed forward
Add&Norm
Multi-Head Attention
N
Add&Norm
Feed forward
Add&Norm
Multi-Head Attention
Add&Norm
Mask Multi-Head Att.
N
Add&Norm
Feed forward
Add&Norm
Multi-Head Attention
1
Add&Norm
Feed forward
Add&Norm
Multi-Head Attention
Add&Norm
Mask Multi-Head Att.
1

트랜스포머의 특성

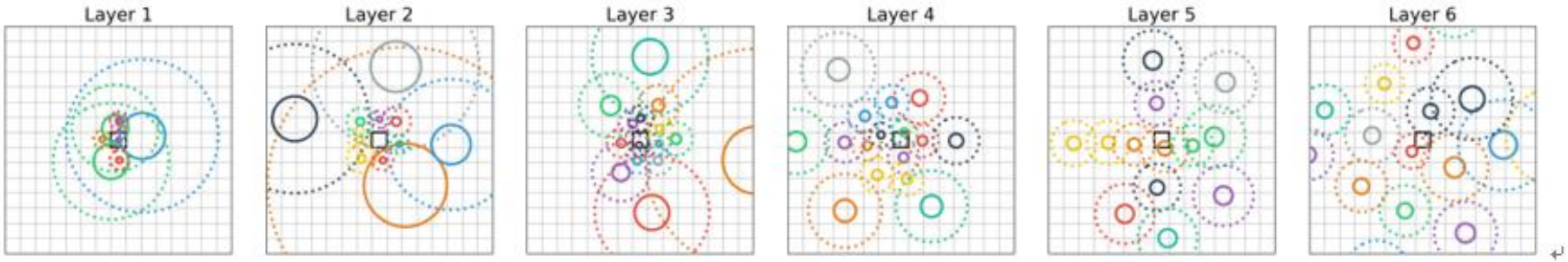
- 장거리 의존long-range dependency



[그림 11-28] 컨볼루션 연산을 통한 장거리 의존



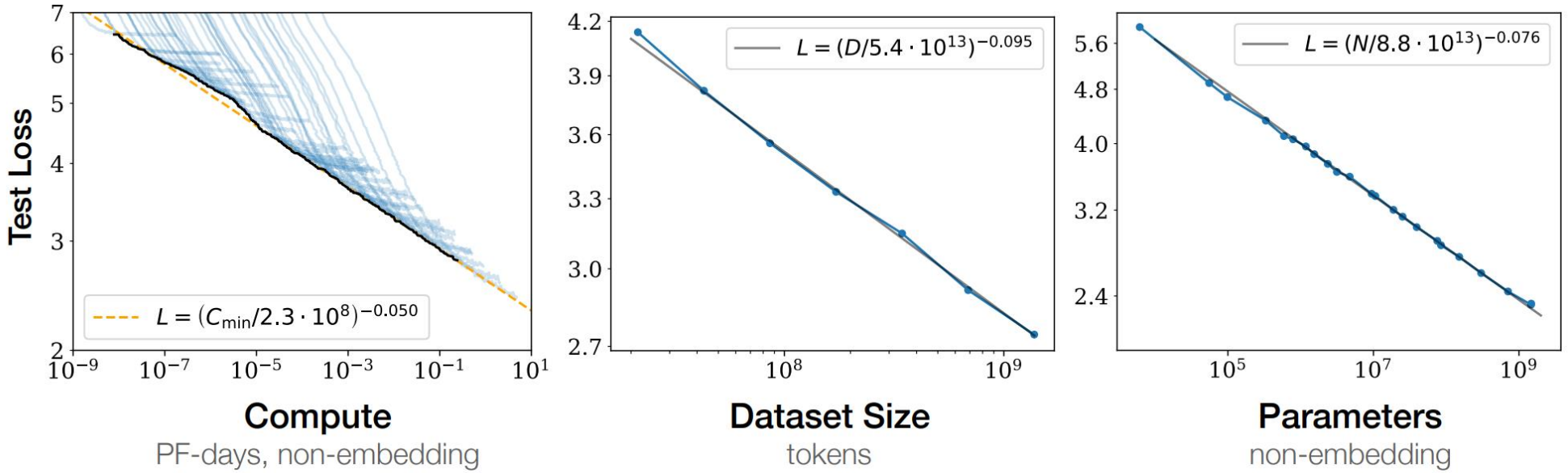
[그림 11-29] 트랜스포머은 자기 주목을 통해 장거리 의존을 표현



[그림 11-30] 지역 정보와 전역 정보를 모두 능숙하게 처리하는 트랜스포머 [Cordonnier2020]

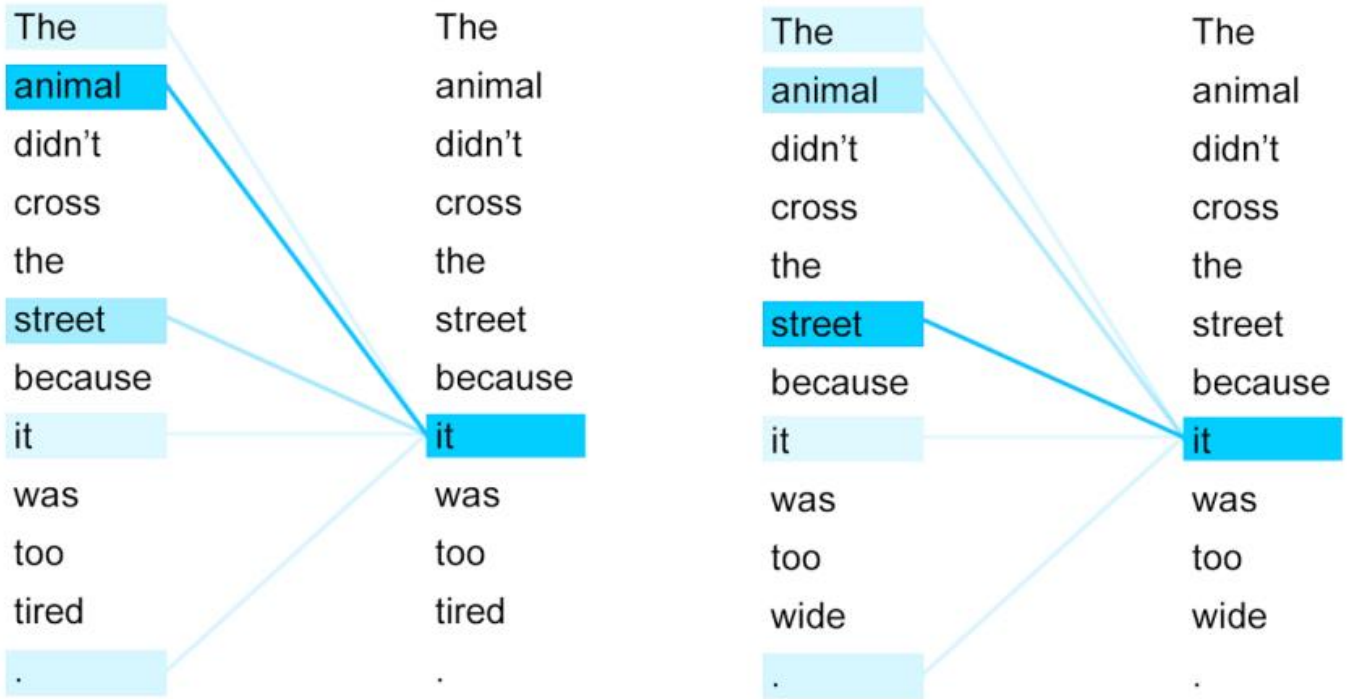
트랜스포머의 특성

- 확장성scalability
  - 초기 트랜스포머는 40M개를 조금 넘는 수준, 이후 초거대 모델 등장
    - GPT-3: 175B개
    - Switch 트랜스포머: 1.6T개
  - 모델 크기, 데이터셋 크기, 학습 계산 자원 사이의 관계 분석 연구 발표
    - 큰 모델을 사용하는 것이 유리하다는 근거 제시

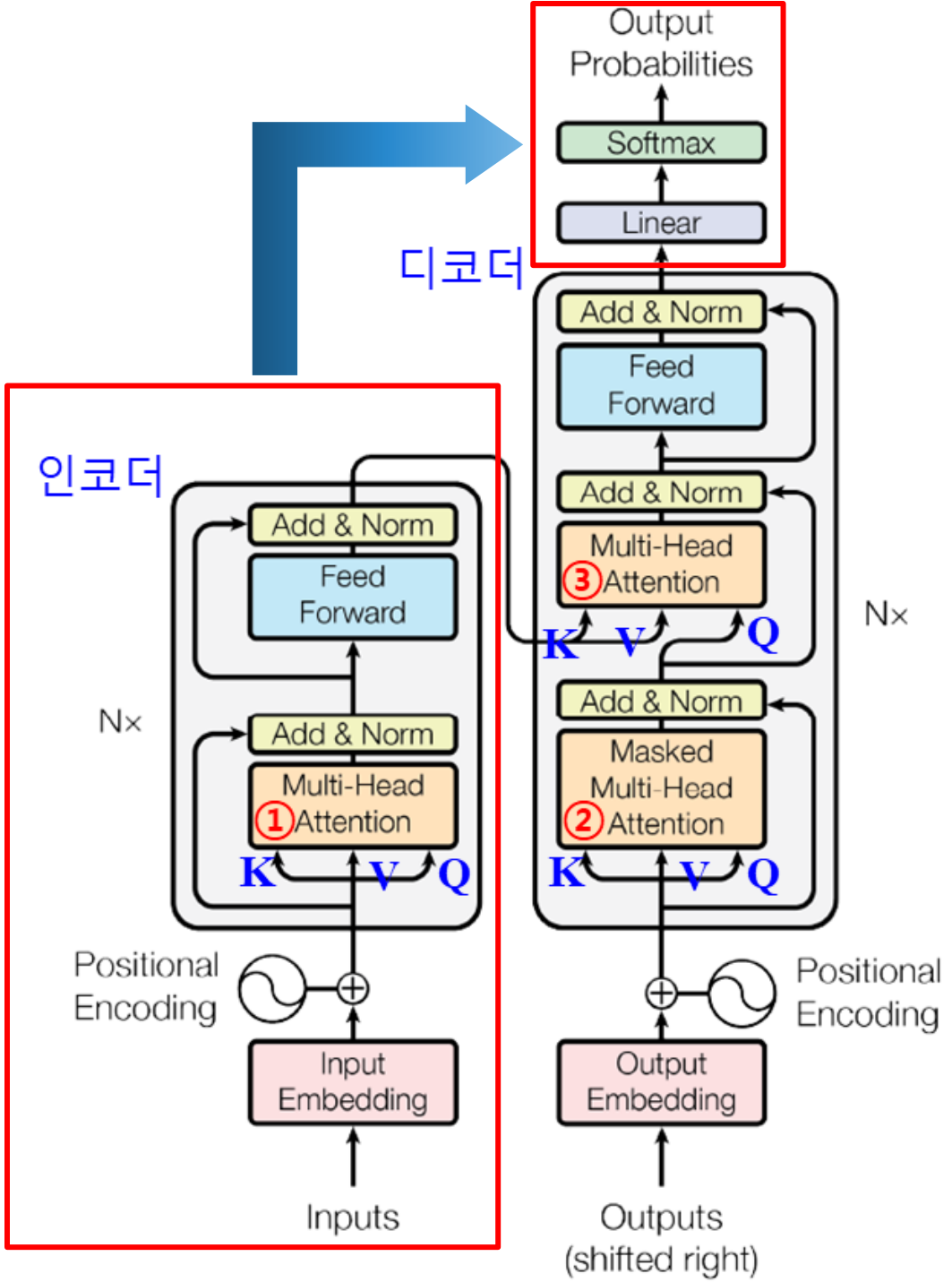


자기 주목 결과 시각화 예시

Encoder Self-Attention

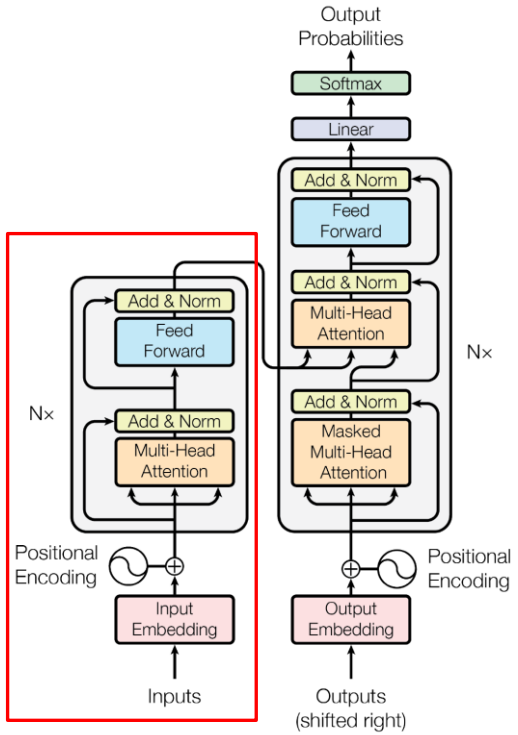


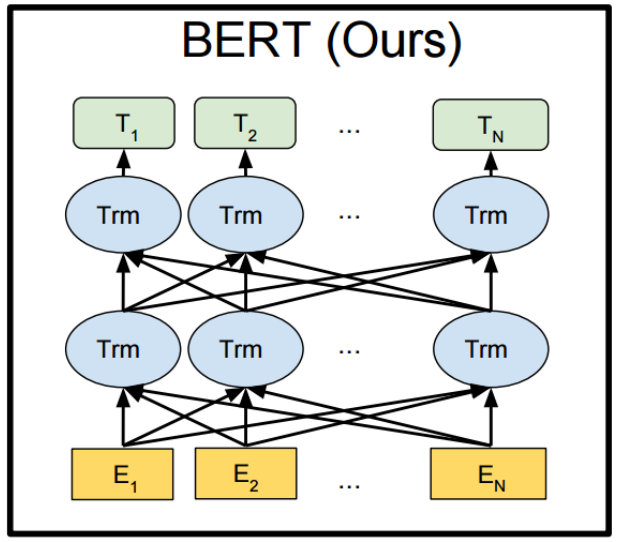
BERT의 등장

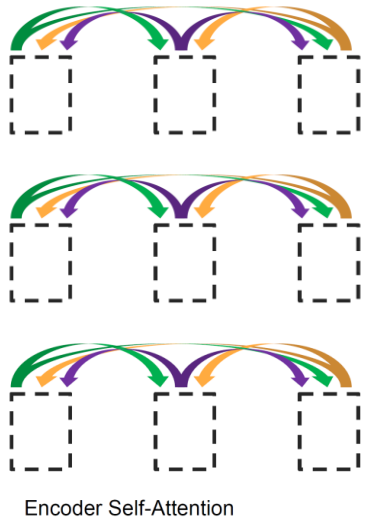

Output Probabilities
Softmax
Linear
디코더
Add & Norm
Feed Forward
Add & Norm
Multi-Head ③ Attention
K V Q
Add & Norm
Masked Multi-Head ② Attention
K V Q
Nx
인코더
Add & Norm
Feed Forward
Add & Norm
Multi-Head ① Attention
K V Q
Nx
Positional Encoding
Input Embedding
Inputs
Positional Encoding
Output Embedding
Outputs (shifted right)

BERT 모델

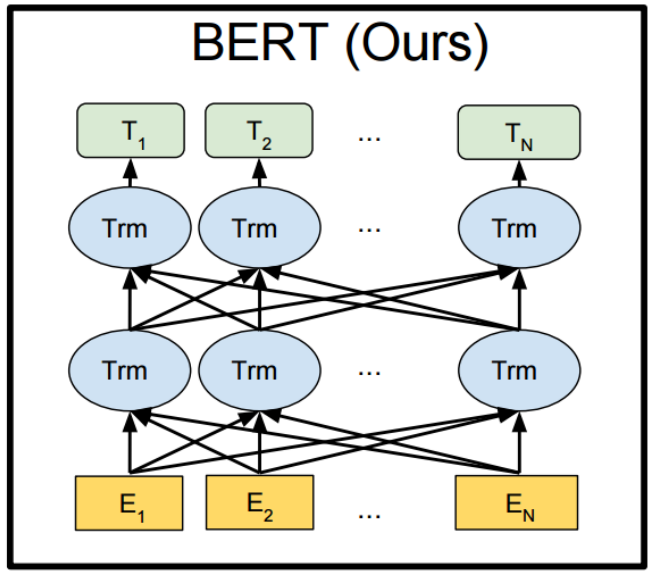
- BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
  - Pre-training of Deep Bidirectional Transformers for Language Understanding (Google AI Language, 2018)

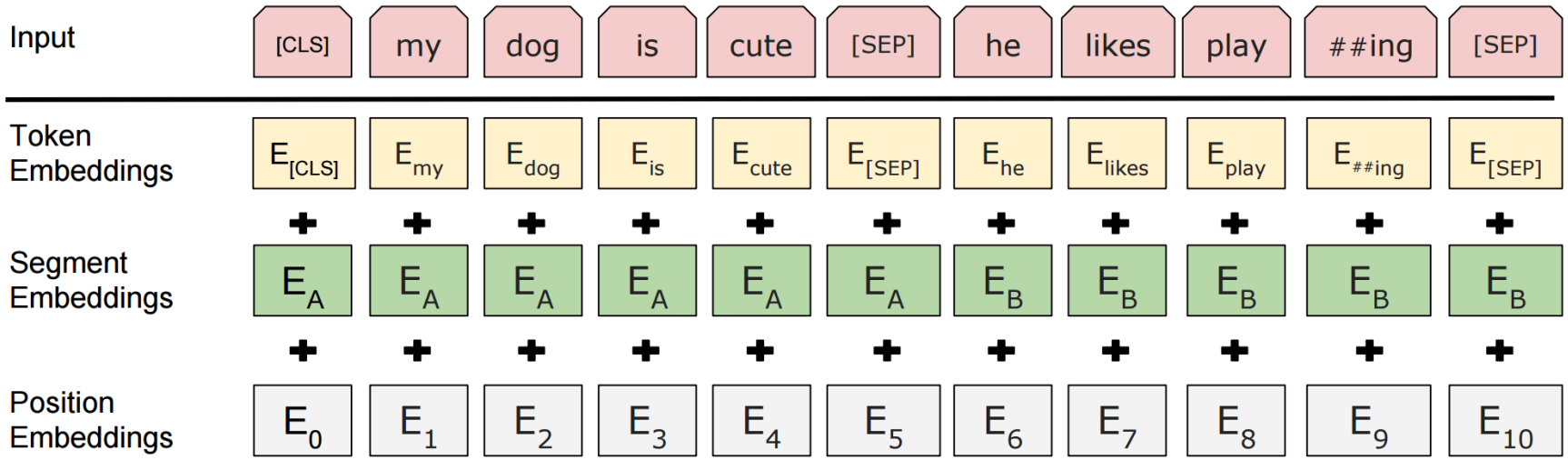






BERT 입력


BERT (Ours)
T1
T2
...
TN
Trm
Trm
...
Trm
Trm
Trm
...
Trm
E1
E2
...
EN


Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
E[CLS] Emy Edog Eis Ecute E[SEP] Ehe Elikes Eplay E##ing E[SEP]
Segment Embeddings
EA EA EA EA EA EA EB EB EB EB EB
Position Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

(학습) 1단계 : Pre-training

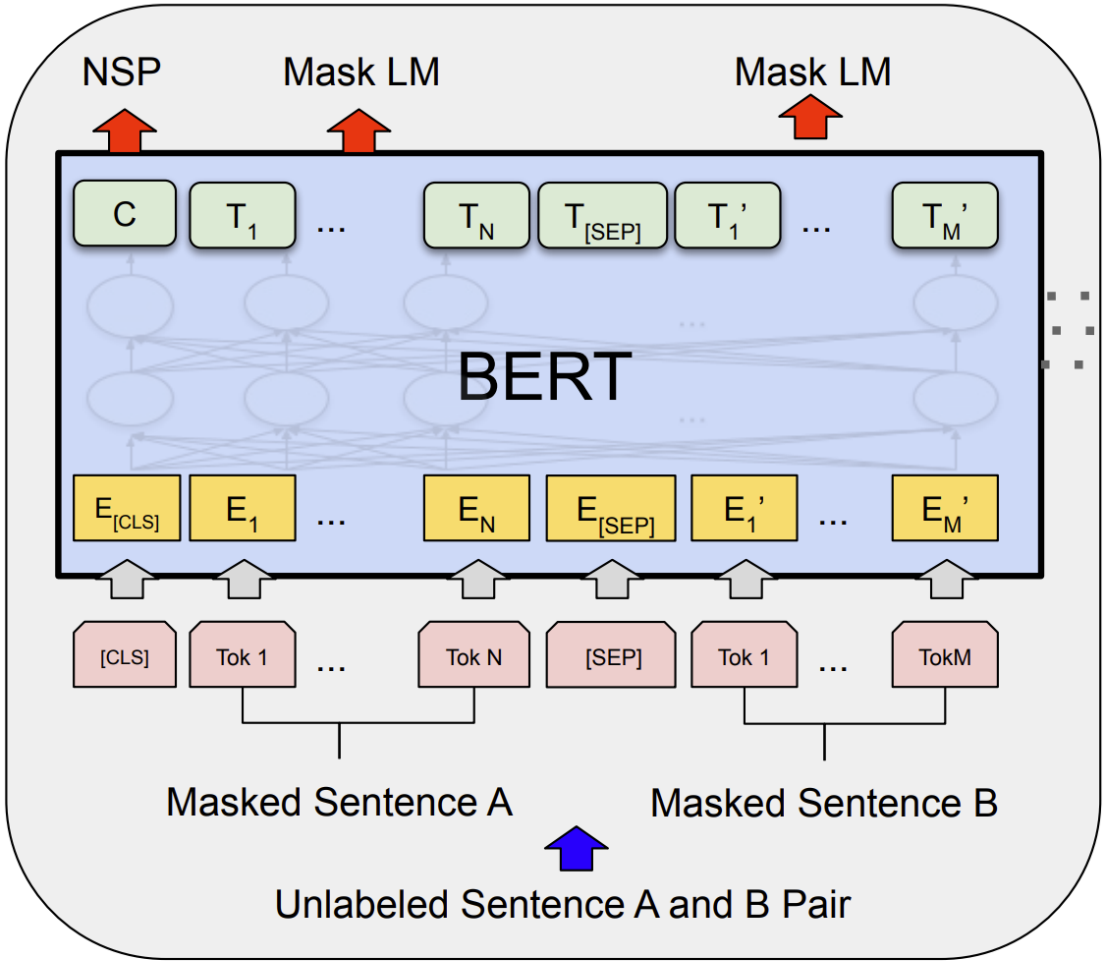
- Task#1: Masked LM

Input: the man went to the [MASK1] . he bought a [MASK2] of milk.
Labels: [MASK1] = store; [MASK2] = gallon

- Task#2: Next Sentence Prediction

Sentence A: the man went to the store .
Sentence B: he bought a gallon of milk .
Label: IsNextSentence

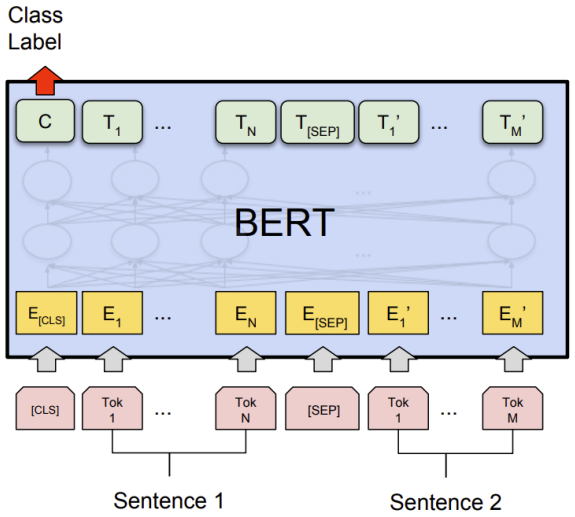
Sentence A: the man went to the store .
Sentence B: penguins are flightless .
Label: NotNextSentence



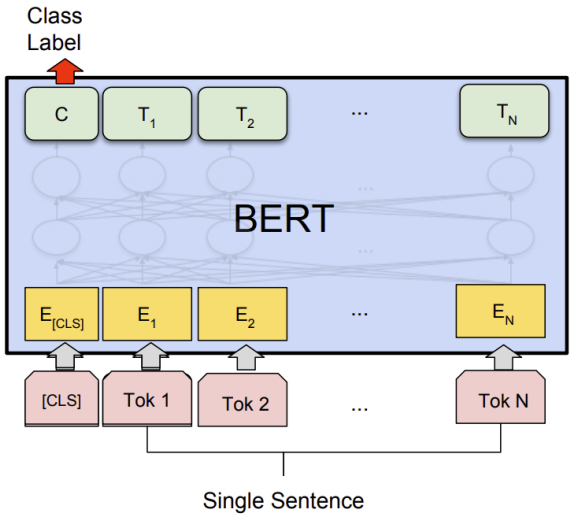
Pre-training

(학습) 1단계 : Pre-training

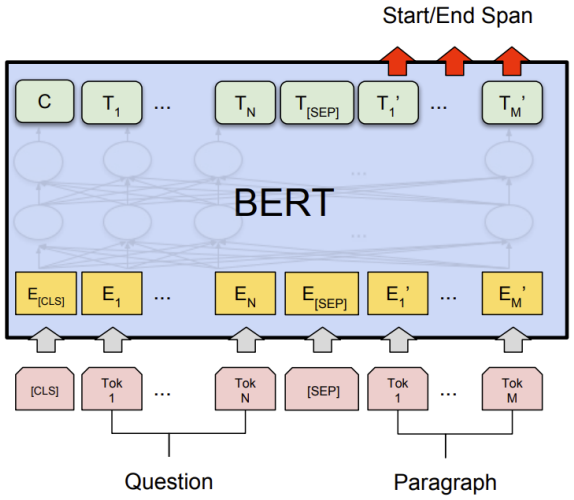
(학습) 2단계 : Fine-tuning



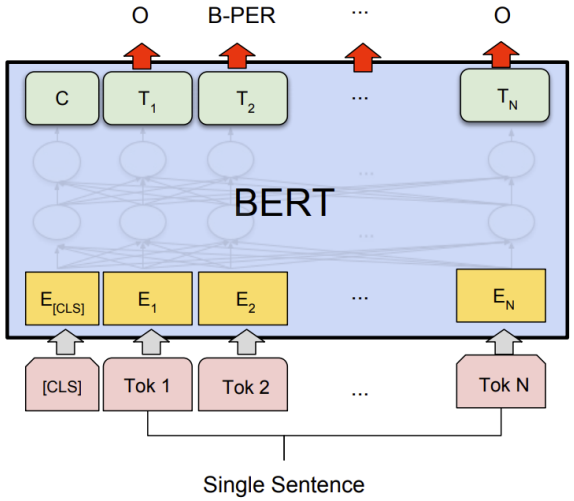
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



(c) Question Answering Tasks: SQuAD v1.1



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

BERT 언어모델 후속 연구

언어 처리 길이 확장 연구

- Long Range Arena 벤치마크 결과

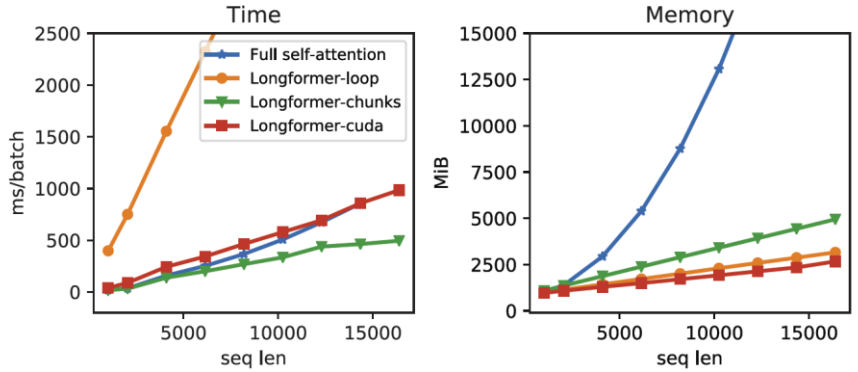
Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Transformer	36.37	64.27	57.46	42.44	71.40	FAIL	54.39
Local Attention	15.82	52.98	53.39	41.46	66.63	FAIL	46.06
Sparse Trans.	17.07	63.58	**59.59**	**44.24**	71.71	FAIL	51.24
Longformer	35.63	62.85	56.89	42.22	69.71	FAIL	53.46
Linformer	35.70	53.94	52.27	38.56	76.34	FAIL	51.36
Reformer	**37.27**	56.10	53.40	38.07	68.50	FAIL	50.67
Sinkhorn Trans.	33.67	61.20	53.83	41.23	67.45	FAIL	51.39
Synthesizer	36.99	61.68	54.67	41.61	69.45	FAIL	52.88
BigBird	36.05	64.02	59.29	40.83	74.87	FAIL	**55.01**
Linear Trans.	16.13	**65.90**	53.09	42.34	75.30	FAIL	50.55
Performer	18.01	65.40	53.82	42.77	**77.05**	FAIL	51.41
Task Avg (Std)	29 (9.7)	61 (4.6)	55 (2.6)	41 (1.8)	72 (3.7)	FAIL	52 (2.4)

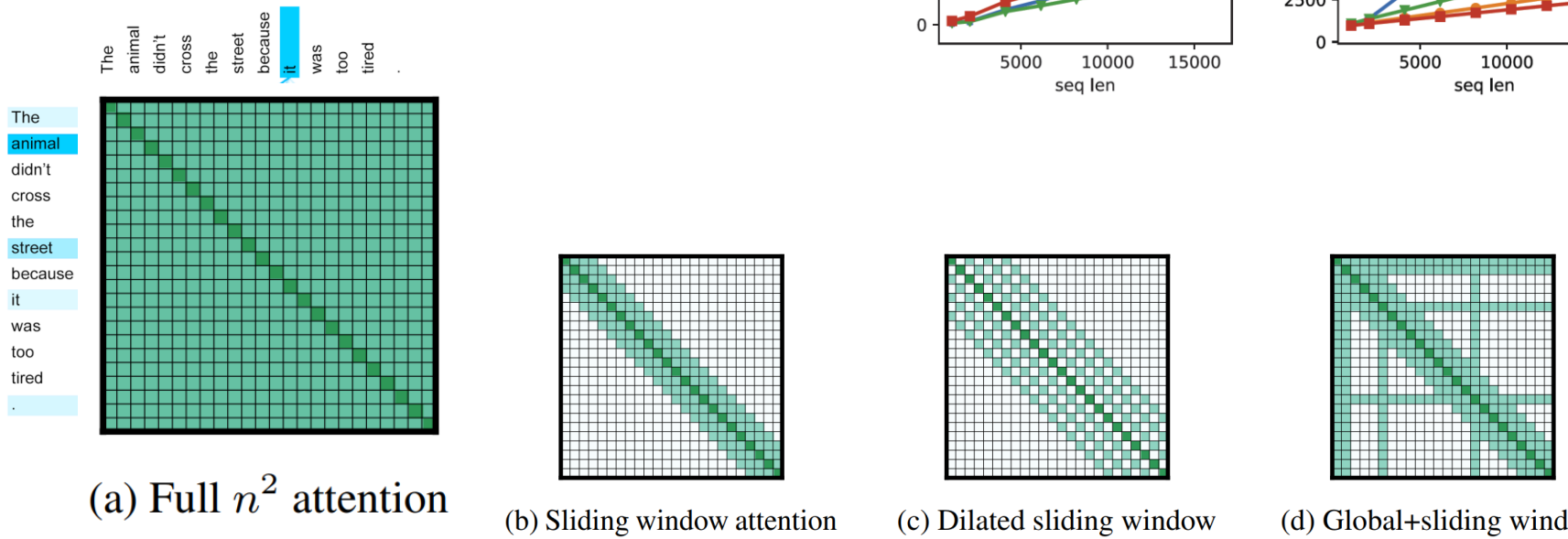
Table 1: Experimental results on Long-Range Arena benchmark. Best model is in boldface and second best is underlined. All models do not learn anything on Path-X task, contrary to the Pathfinder task and this is denoted by FAIL. This shows that increasing the sequence length can cause seriously difficulties for model training. We leave Path-X on this benchmark for future challengers but do not include it on the Average score as it has no impact on relative performance.

Longformer: The Long-Document Transformer

Longformer: The Long-Document Transformer

Iz Beltagy* **Matthew E. Peters*** **Arman Cohan***
Allen Institute for Artificial Intelligence, Seattle, WA, USA
{beltagy,matthewp,armanc}@allenai.org





(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

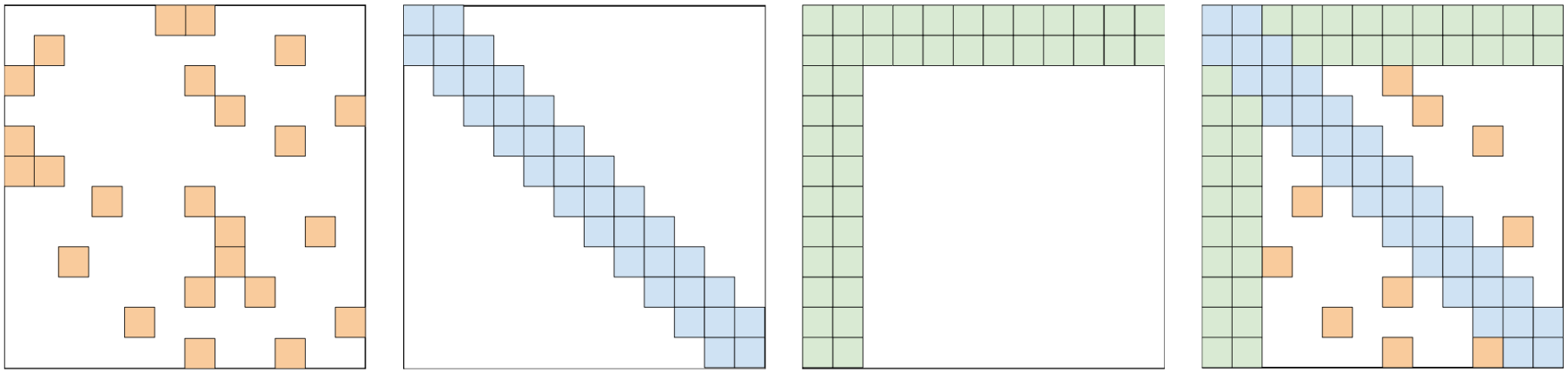
Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer.

Big Bird: Transformers for Longer Sequences

Big Bird: Transformers for Longer Sequences

Manzil Zaheer MANZILZ@GOOGLE.COM
Guru Guruganesh GURUG@GOOGLE.COM
Google Research,
Mountain View, CA, USA

Avinava Dubey AVINAVADUBEY@GOOGLE.COM
Joshua Ainslie, Chris Alberti, Santiago Ontanon,
Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang,
Amr Ahmed
Google Research, USA



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Big Bird: Transformers for Longer Sequences

Model	HotpotQA			NaturalQ		TriviaQA		WikiHop
	Ans	Sup	Joint	LA	SA	Full	Verified	MCQ
HGN [27]	**82.2**	88.5	**74.2**	-	-	-	-	-
GSAN	81.6	88.7	73.9	-	-	-	-	-
ReflectionNet [33]	-	-	-	77.1	**64.1**	-	-	-
RikiNet [62]	-	-	-	75.5	59.5	-	-	-
Fusion-in-Decoder [40]	-	-	-	-	-	**84.5**	90.3	-
SpanBERT [43]	-	-	-	-	-	79.1	86.6	-
MRC-GCN [88]	-	-	-	-	-	-	-	78.3
MultiHop [14]	-	-	-	-	-	-	-	76.5
Longformer [8]	81.2	85.8	73.2	-	-	77.3	85.3	81.9
BIGBIRD-ETC	81.2	**89.1**	73.6	**77.7**	57.8	80.9	**90.8**	**82.3**

Table 5: Fine-tuning results on **Test** set for QA tasks. The Test results (F1 for HotpotQA, Natural Questions, TriviaQA, and Accuracy for WikiHop) have been picked from their respective leaderboard. For each task the top-3 leaders were picked not including BIGBIRD-etc. **For Natural Questions Long Answer (LA), TriviaQA Verified, and WikiHop, BIGBIRD-ETC is the new state-of-the-art**. On HotpotQA we are third in the leaderboard by F1 and second by Exact Match (EM).

BERT 모델 경량화 연구

- DistilBert: **a distilled version of BERT** (Oct. 2019)
  - general purpose language representation model
    - BERT를 40% 정도 줄이고, 60%나 빠르게 연산, 97%의 성능 유지

- (참고) Knowledge Distillation [Bucila et al., 2006, Hinton et al., 2015]
  - larger model(teacher model)로부터 compact model(student model)을 만들어내는 방법
  - student를 학습하기 위해서 teacher의 output을 그대로 이용
  - 작은 모델을 바로 학습시키는 것보다 의미있는 이유는 near-zero인 확률들도 학습할 수 있기 때문

BERT 모델 경량화 연구

- DistilBert: **a distilled version of BERT** (Oct. 2019)
  - student layer의 구조: BERT와 비슷함
    - token type embedding, pooler layer 없음
    - transformer block 두배로 줄임
    - initialization은 teacher의 레이어 두개당 하나 이용

Table 1: **DistilBERT retains 97% of BERT performance.** Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors. BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	77.6	48.9	84.3	88.6	89.3	89.5	71.3	91.7	91.2	43.7
DistilBERT	76.8	49.1	81.8	90.2	90.2	89.2	62.9	92.7	90.7	44.4

Table 2: **DistilBERT yields to comparable performance on downstream tasks.** Comparison on downstream tasks: IMDb (test accuracy) and SQuAD 1.1 (EM/F1 on dev set). D: with a second step of distillation during fine-tuning.

Model	IMDb (acc.)	SQuAD (EM/F1)
BERT-base	93.46	81.2/88.5
DistilBERT	92.82	77.7/85.8
DistilBERT (D)	-	79.1/86.9

Table 3: **DistilBERT is significantly smaller while being constantly faster.** Inference time of a full pass of GLUE task STS-B (sentiment analysis) on CPU with a batch size of 1.

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

Encoder-Decoder Structure

- BERT
  - encoder structure

- BART, T5
  - encoder-decoder structure
  - achieved good results on both generative and classification tasks

BART

- BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension (Oct 2019, Facebook AI)
  - https://arxiv.org/abs/1910.13461
- 다양한 denoising 방법 실험
  - Original text : ABC. DE.
  - **Token Masking**: 임의의 token을 [MASK]로 교체함

    => [MASK] token이 무엇인지 예측해야 함
  - **Token Deletion**: 임의의 token을 삭제함

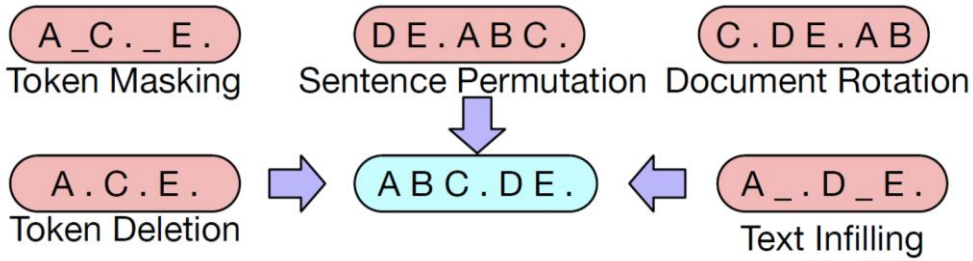
    => 삭제한 token의 위치를 찾아야 함
  - **Text Infilling**: span length를 뽑아 하나의

    [MASK] token으로 대체함

    => [MASK]로 대체된 token에

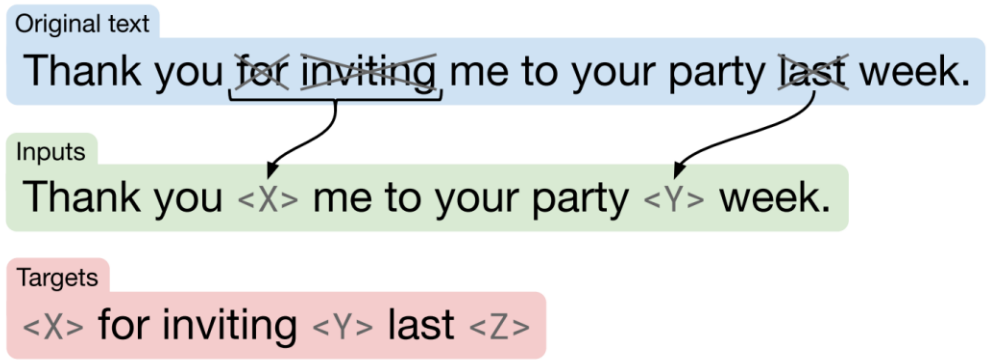
    몇개의 token이 존재할지 예측해야 함
  - **Sentence Permutation**: 문장의 순서를 랜덤으로 섞음
  - **Document Rotation**: 하나의 token을 뽑은 후, 그 token을 시작점으로 회전함

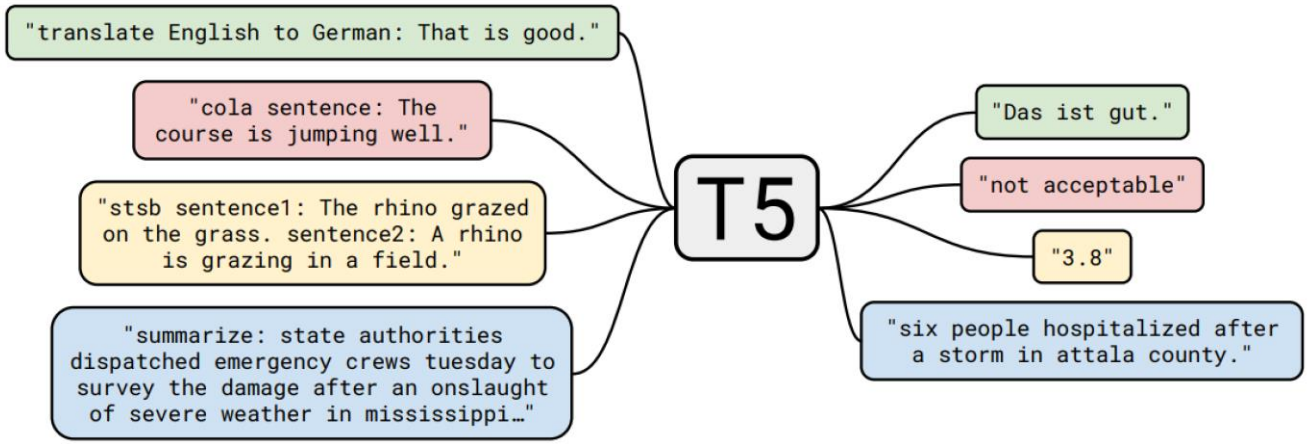
    => 문서의 start point를 찾도록 학습시킴



Text-To-Text-Transfer-transformer (T5)

- Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (Journal of Machine Learning Research 2020, Google)
- a unified framework that converts every text processing problem into "text-to-text" problem





감사합니다