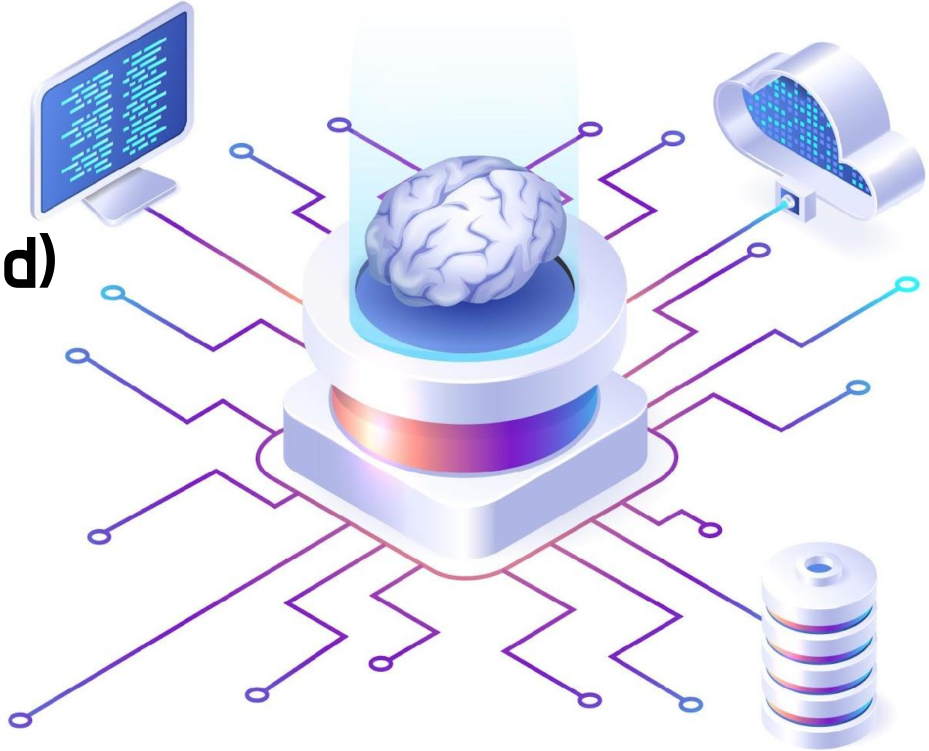
Transformer
(Attention is All You Need)

실무형 인공지능 자연어처리



Insight campus SeSAC

BERT 까지

단어의 표현

문맥의 표현

RNN

- 순환 구조로 시계열에 대한 고려 가능
- 장기문맥 손실 (Long dependency)

LSTM / GRU

- grad vanishing/exploding 문제 해결
- 장기문맥에 대한 고려

ElMo

- biLM (양방향 언어모델)
- Bidirectional, Stacking

CNN

- 이미지 분류에 많이 사용
- 시계열에 대한 고려 X

Seq2Seq

- Encoder - Decoder 구조
- Many to many
- 고정 문맥 벡터로 표현하여 정보 손실 발생
- Teacher forcing

Seq2Seq with Attention

- 집중해야할 정보를 고려 = attention
- LSTM 기반 attention
- Attention mechanism

Transformer
(Attention is all you need)

- LSTM을 제거. 속도향상
- Self-attention
- Multihead attention
- Positional encoding

BERT

- Wordpiece tokenizer
- Masked Language Model(MLM)
- Next Sentence Prediction (NSP)
- Position embedding
- Pre-trained model
- Fine tuning

1

Transformer

Attention is All You Need

Transformer 개요 (1)

- Transformer의 가장 큰 특징은 Convolution도, Recurrence도 사용하지 않음
- Since our model contains **no recurrence and no convolution**, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. (Vaswani et al., Attention Is All You Need, 2017)

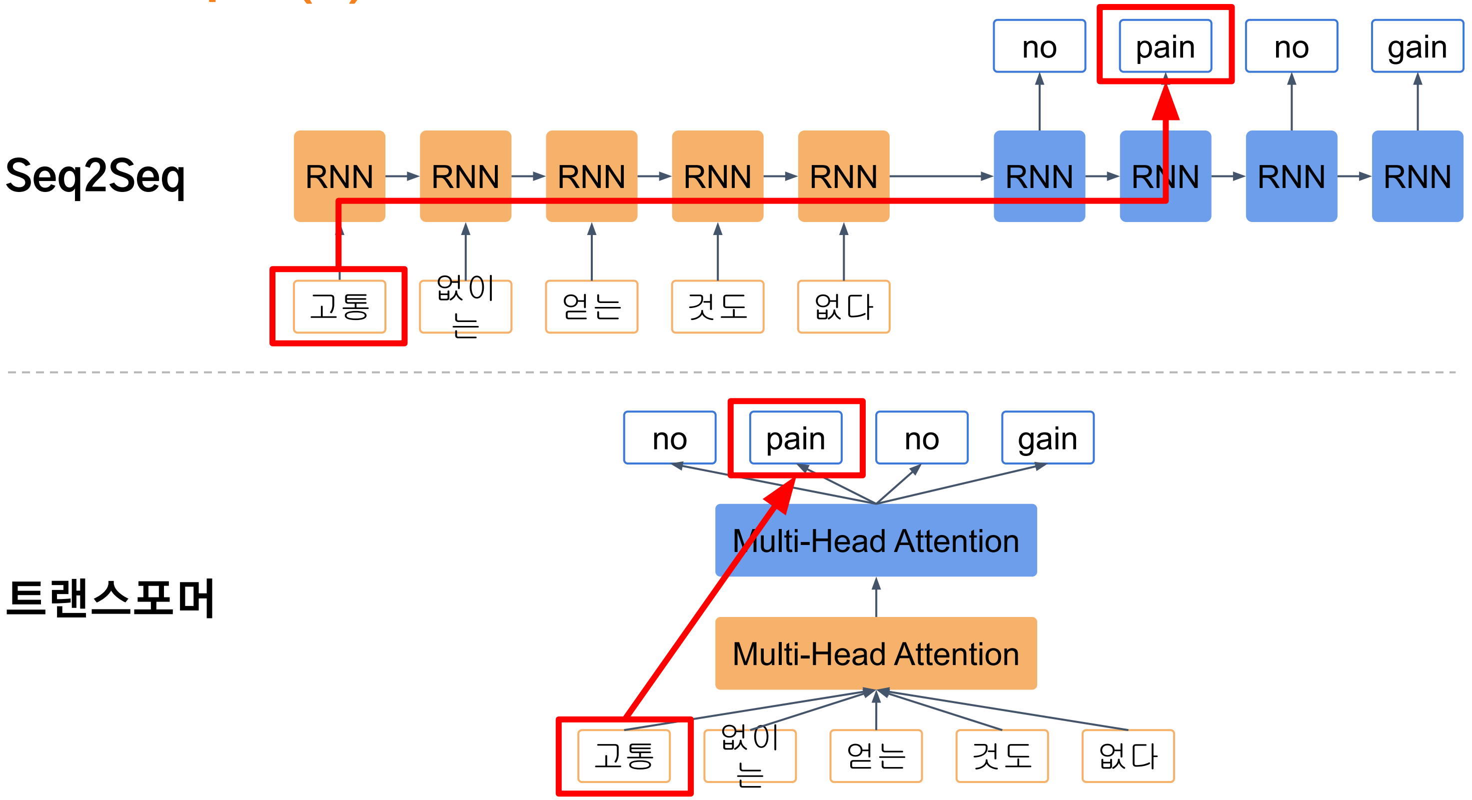
Insight campus SeSAC

Transformer 개요 (2)

- **Long-term dependency problem**
  어떤 정보와 다른 정보 사이의 거리가 멀 때 해당 정보를 이용하지 못하는 것 (RNN의 문제점)
  => Attention mechanism 으로 해결

- **Parallelization**
  RNN은 이전 hidden state를 사용함으로써 순차적으로 계산이 되어야함. (병렬화 불가능)
  => 행렬 연산 병렬화

Transformer 개요 (3)



RNN을 사용하지 않고 행렬 병렬연산으로 빠르게 학습이 가능

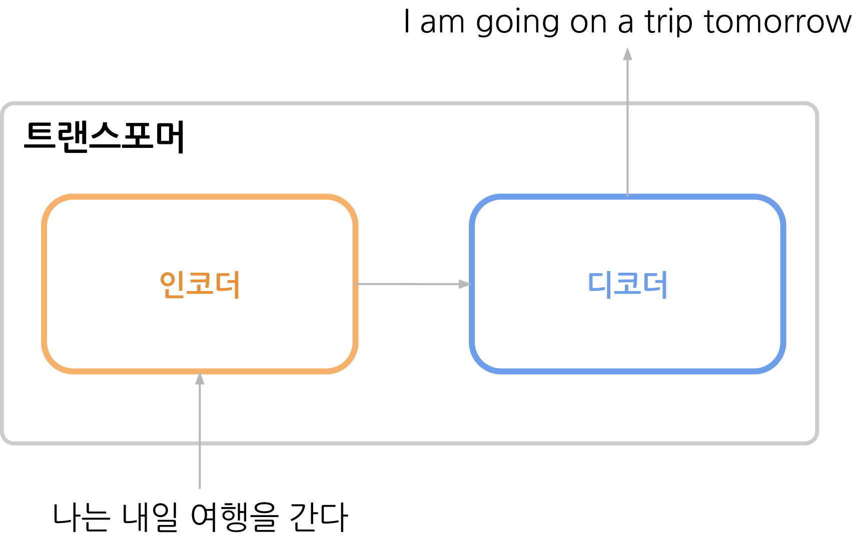
Transformer 개요 (4)

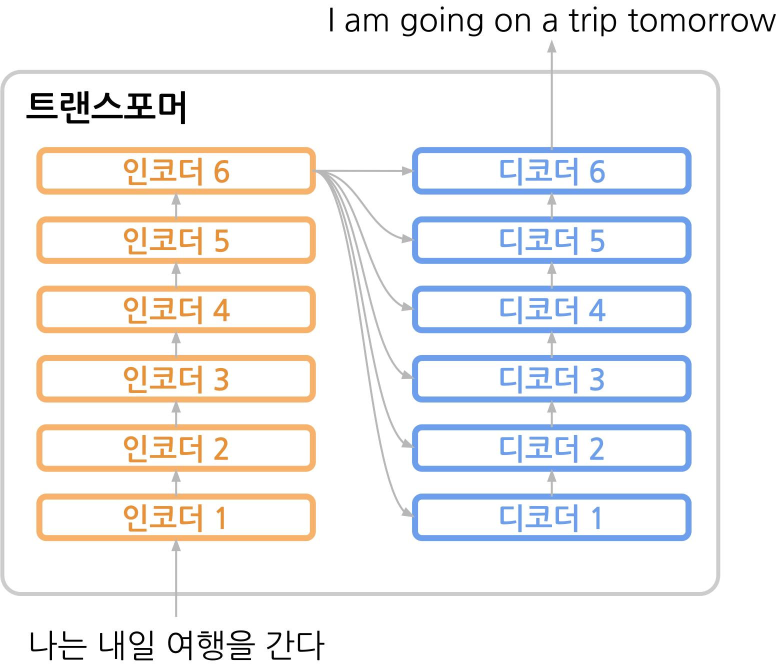
나는
내일
여행을
간다

→ **트랜스포머** →

I
am
going
on
a
trip
tomorrow

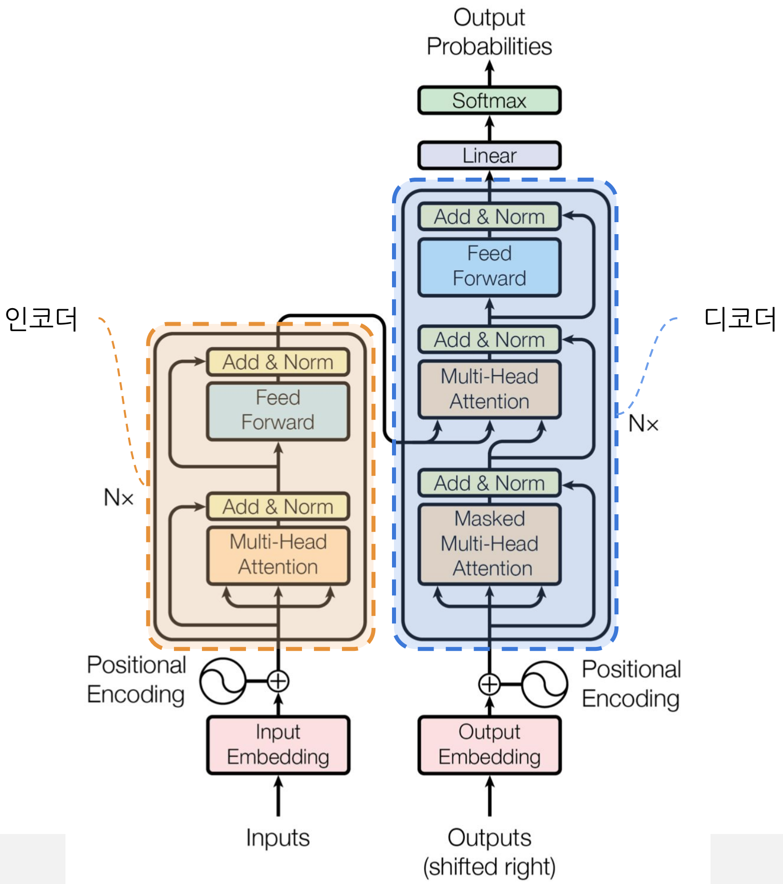
Transformer 개요 (4)





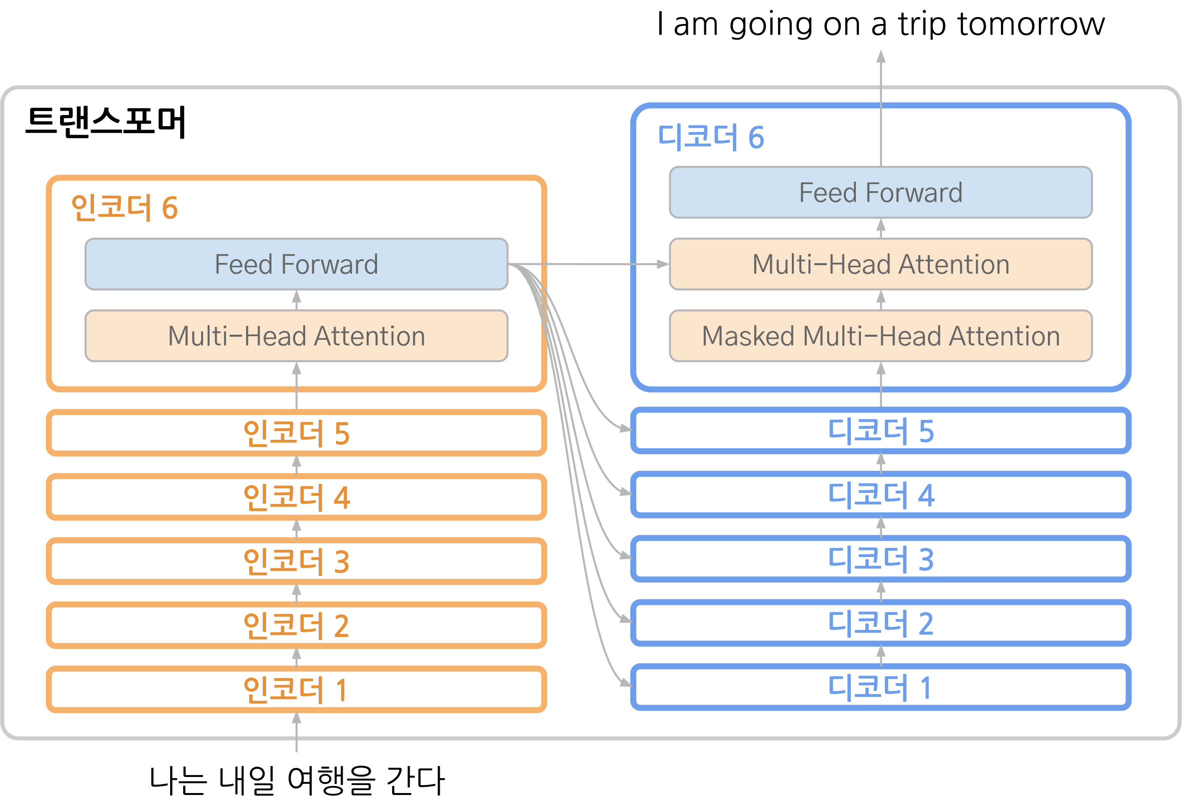
6개 layer의 encoder, 6개 layer의 decoder로 구성

Transformer 개요 (5)



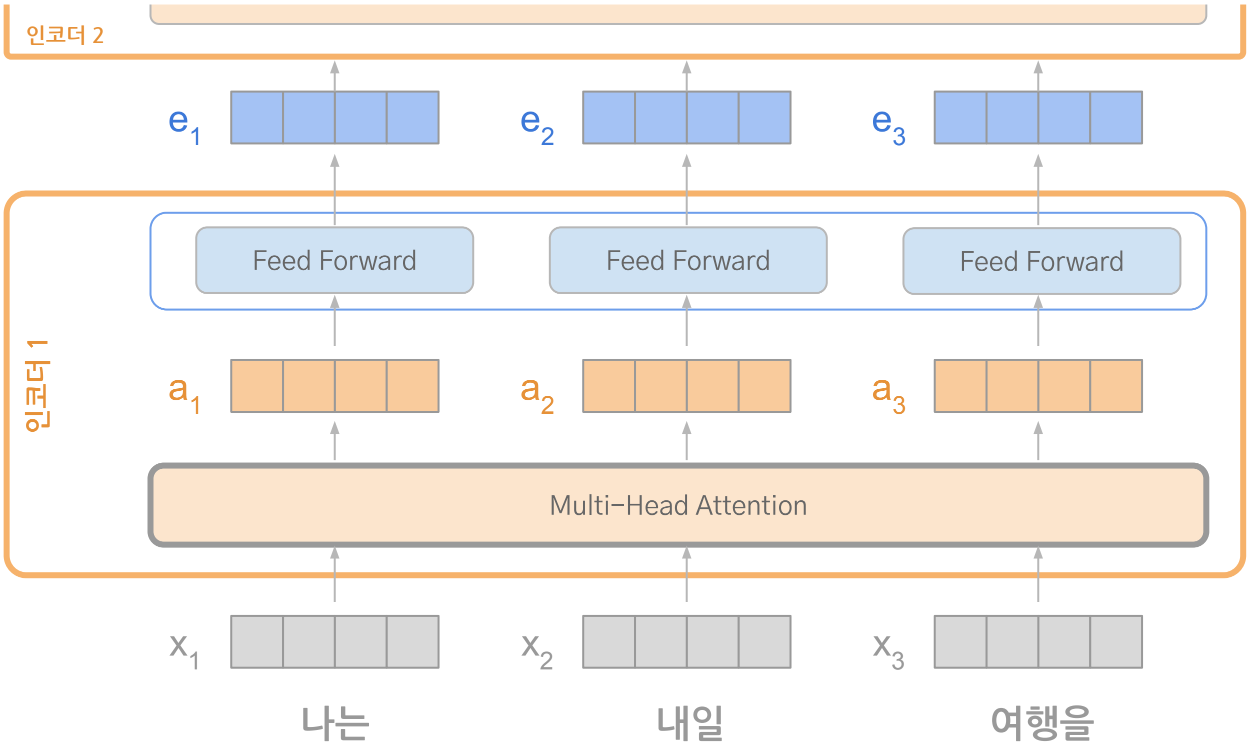
2 Encoder

Transformer 구조

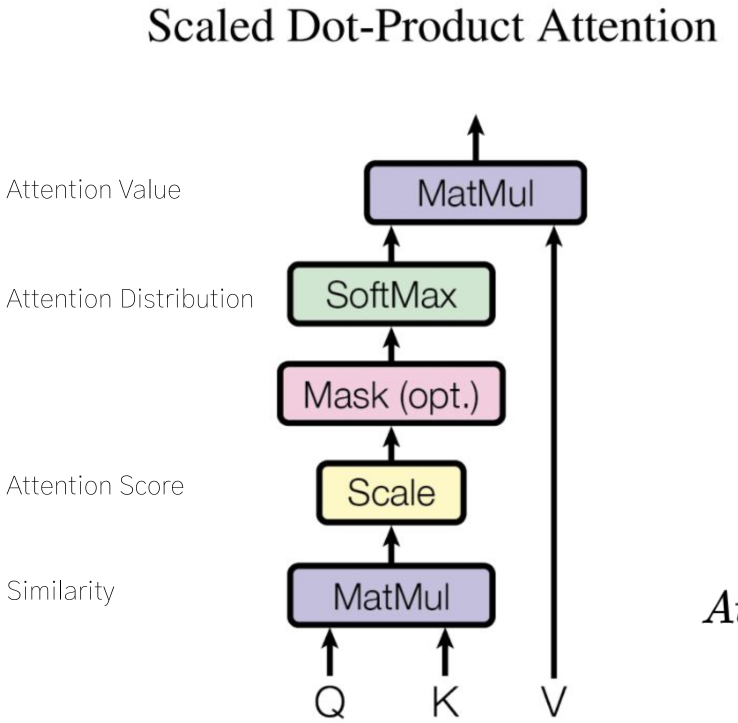


Insight campus SeSAC

Transformer 인코더 - Multi-Head Attention



Self Attention (1) - Scaled Dot-Product Attention



- 연산 Dependency 가 줄어 빠른 연산 가능
- 병렬화 가능 연산 증가
- long-range의 term들의 dependency도 학습가능
- 스케일을 조정해 주는 이유는 내적 행렬의 분산을 줄여 개별 소프트맥스 값이 지나치게 작아지는 문제를 방지

- QK^T : Q(query)와 K(key)의 유사도를 의미
- sqrt(d_k) : K(key)의 차원수로 나누어 scaling

단어간 유사도

$$Attention(Q, K, V) = softmax_k\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

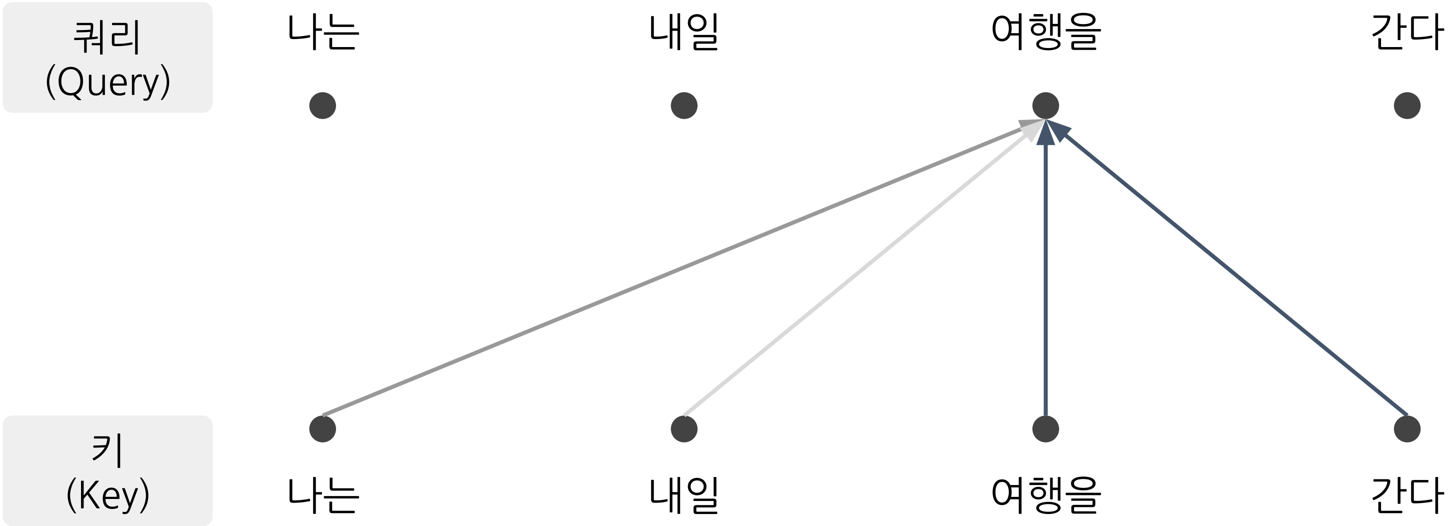
어텐션 분포 어텐션 스코어

벡터의 내적과 코사인 유사도

$$A \cdot B = \|A\| \|B\| \cos\theta$$

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Self Attention (2) - 예제



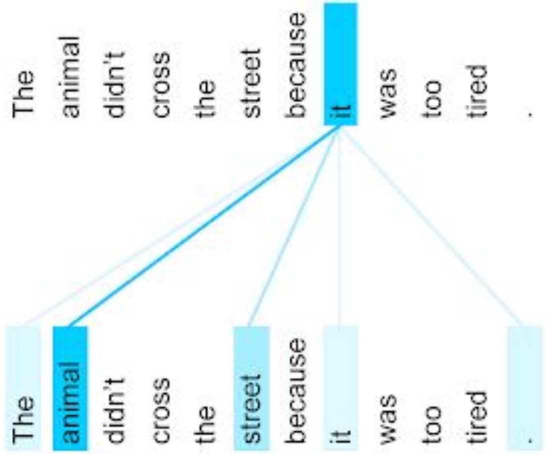
Attention 학습을 통해 결과에 대한 해석과 설명이 가능

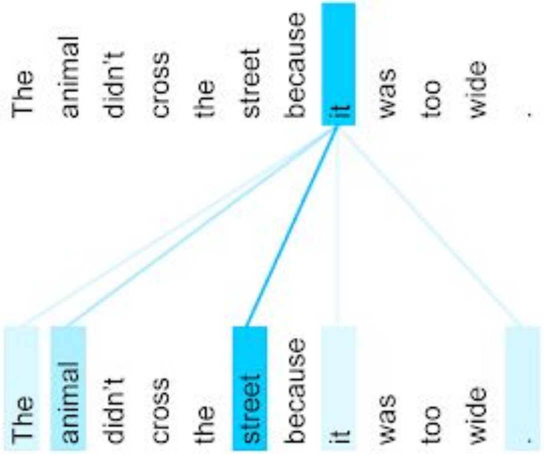
Insight campus SeSAC

Self Attention (3)

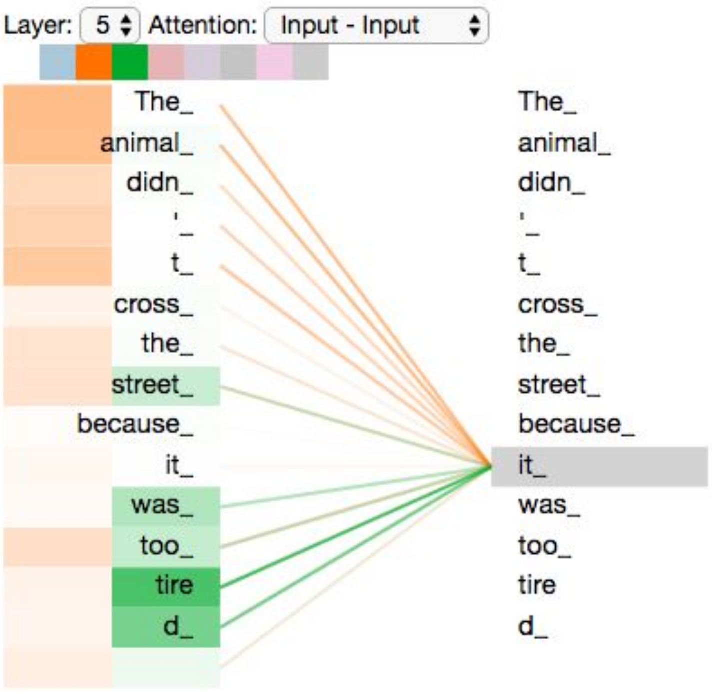
The animal didn't cross the street because it was too tired.
⇒ 동물은 길을 건너지 않았다. 왜냐하면 그것(it)은 너무 피곤하기 때문이다.

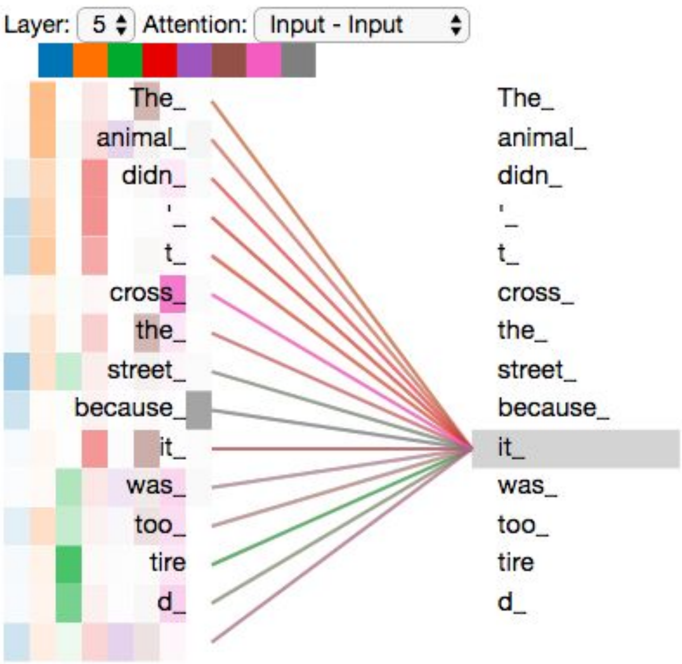
The animal didn't cross the street because it was too wide.
⇒ 동물은 길을 건너지 않았다. 왜냐하면 그것(it)이 너무 넓기 때문이다.





Self Attention (4)

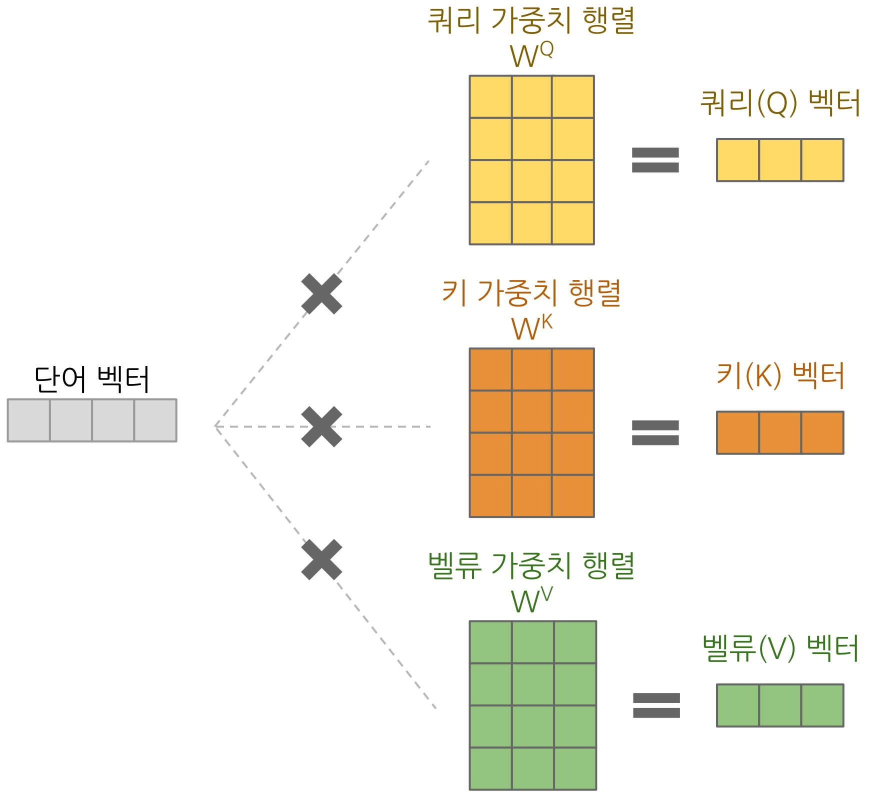




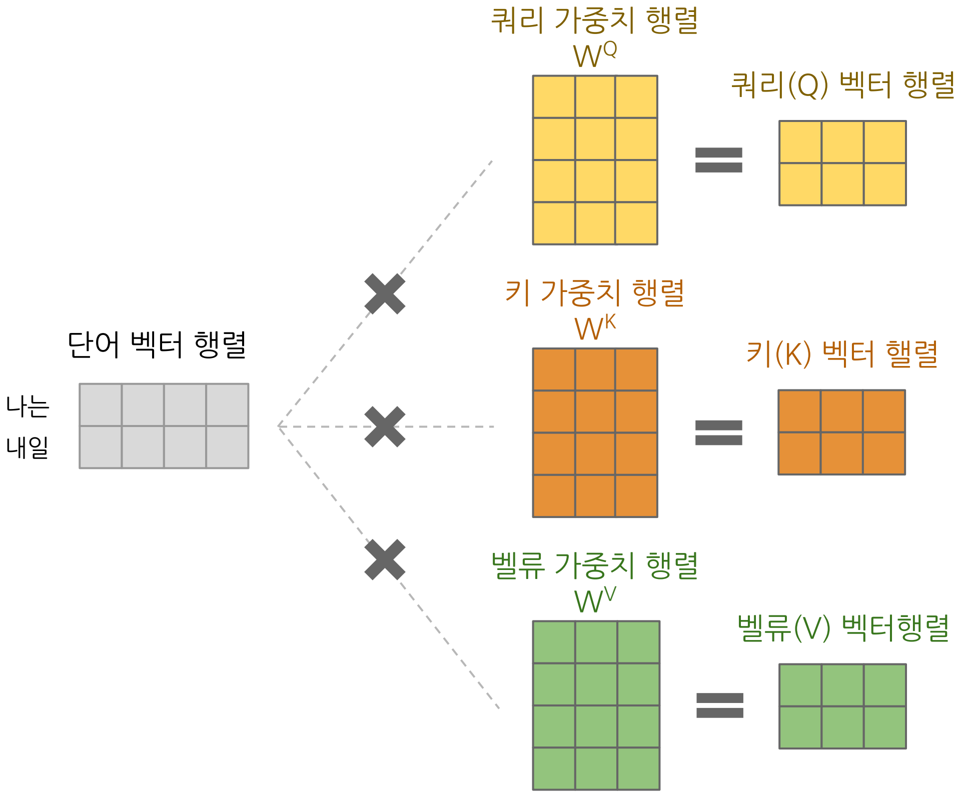
https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb#scrollTo=OJKU36QAfqOC

Self Attention이 고려된 Vector가 생성

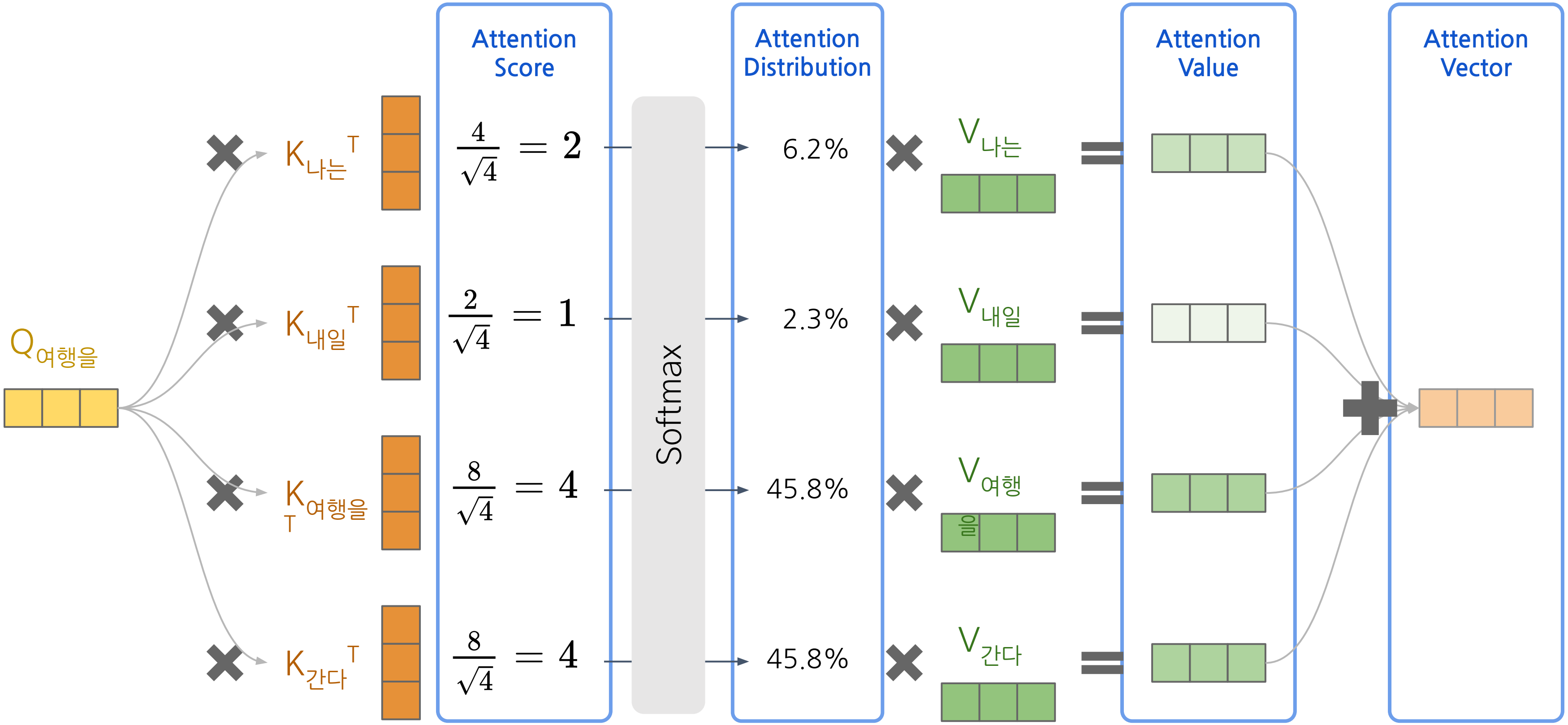
Scaled Dot-Product Attention (1) - Query, Key, Value



Scaled Dot-Product Attention (2) - Query, Key, Value

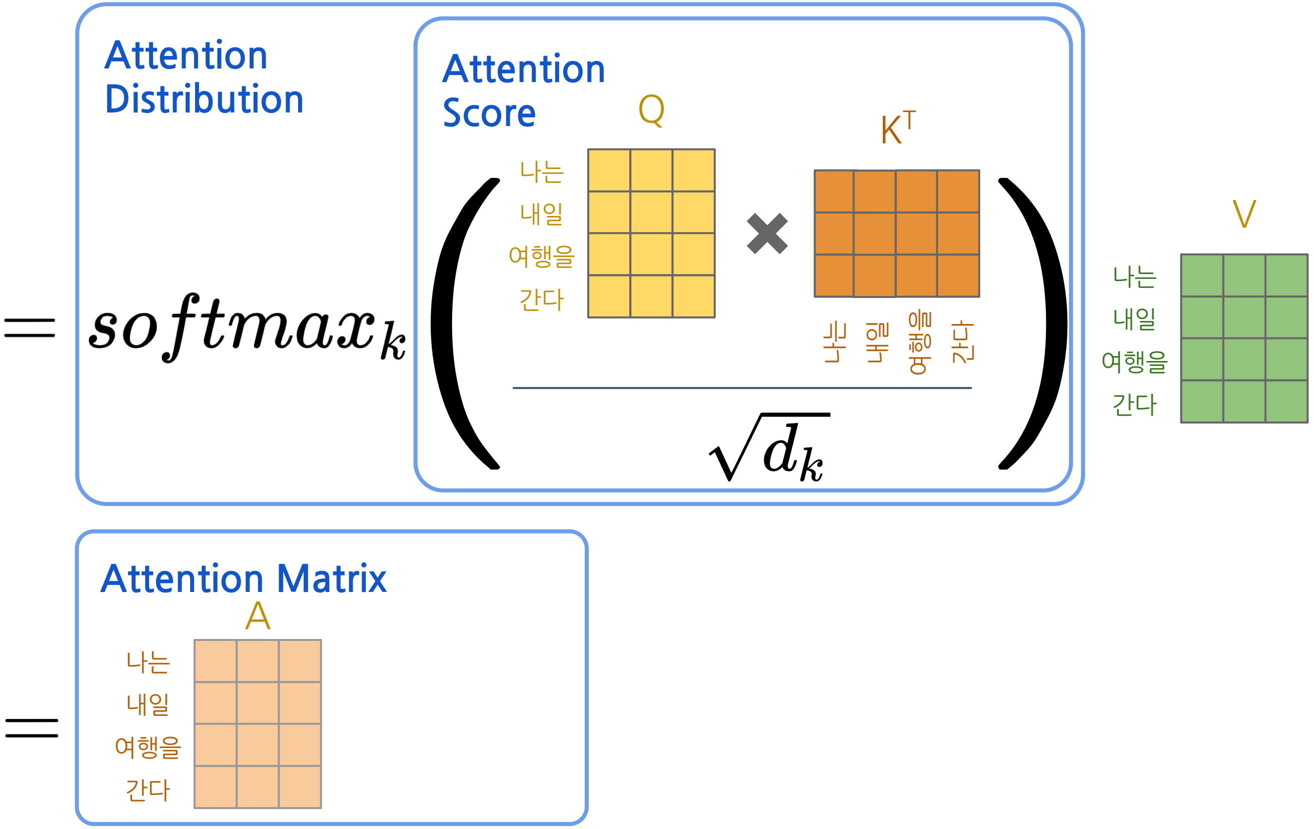


Scaled Dot-Product Attention (3) - Query, Key, Value



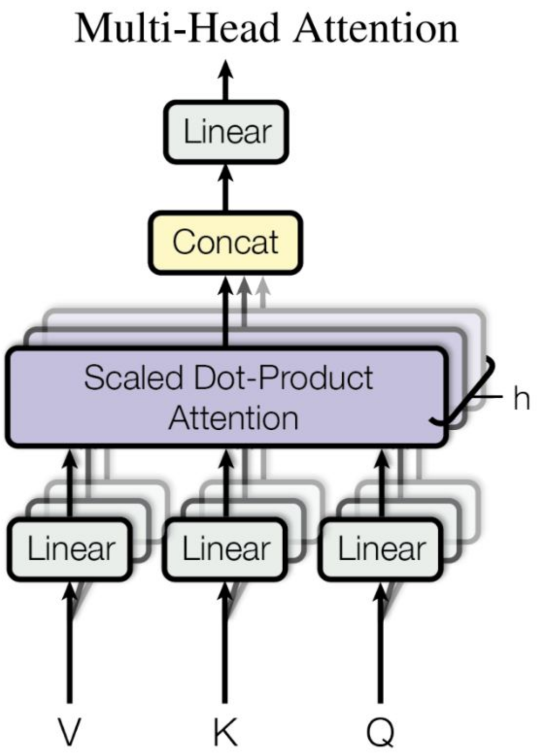
Scaled Dot-Product Attention (4) - Query, Key, Value

$$Attention(Q, K, V) = softmax_k\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Insight campus SeSAC

Multi-Head Attention (1)



$$MultiHead(Q,K,V) = Concat(head_1,\dots,head_h)W^O$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

$$QW_i^Q = [d_Q x d_{model}] x [d_{model} x d_k] = [d_Q x d_k]$$

$$KW_i^K = [d_K x d_{model}] x [d_{model} x d_k] = [d_K x d_k]$$

$$VW_i^V = [d_V x d_{model}] x [d_{model} x d_v] = [d_V x d_v]$$

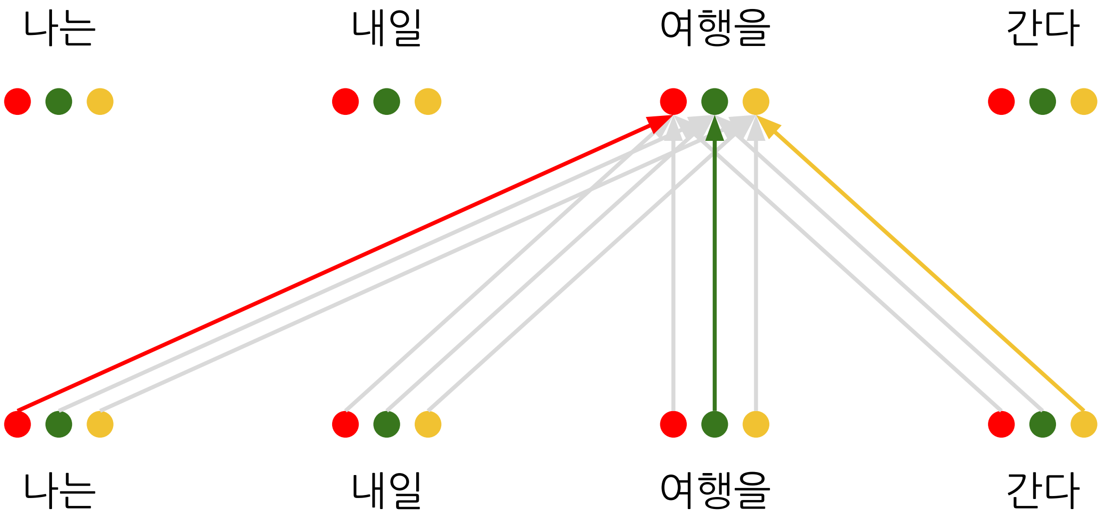
$$\downarrow$$

$$Attention(QW_i^Q, KW_i^K, VW_i^V) = [d_V x d_v]$$

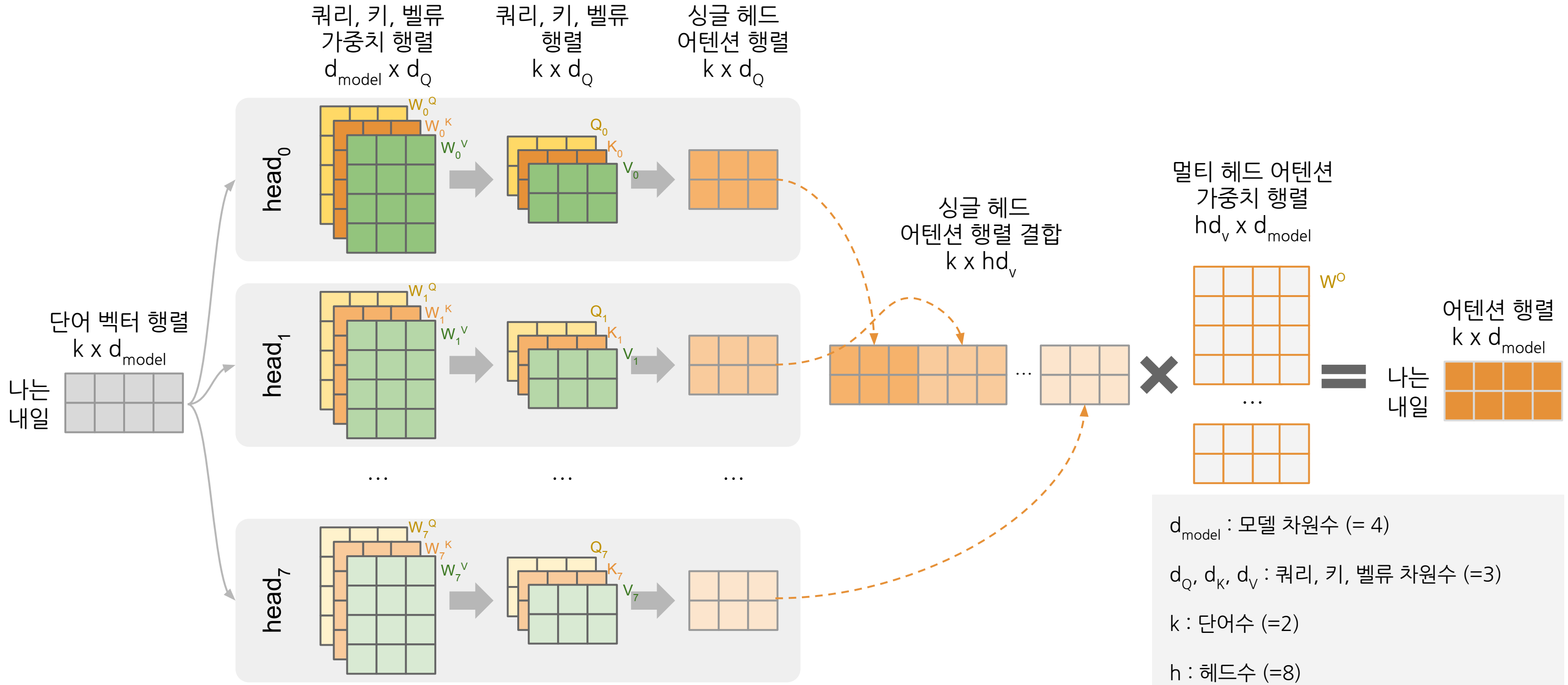
$$\downarrow$$

$$Concat(QW_i^Q, KW_i^K, VW_i^V)W^O = [d_V x hd_v] x [hd_v x d_{model}] = [d_V x d_{model}]$$

Multi-Head Attention (2) - 예제



Multi-Head Attention (3) - 과정



d_{model} : 모델 차원수 (= 4)

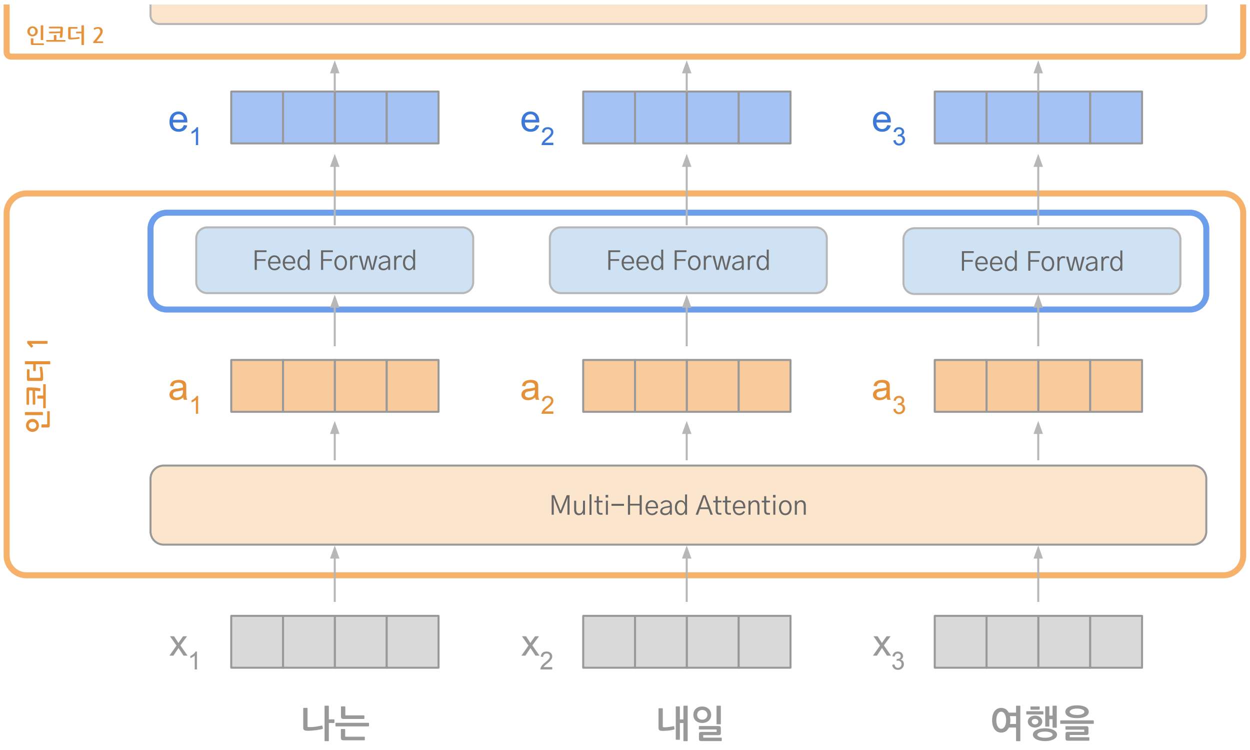
d_Q, d_K, d_V : 쿼리, 키, 벨류 차원수 (=3)

k : 단어수 (=2)

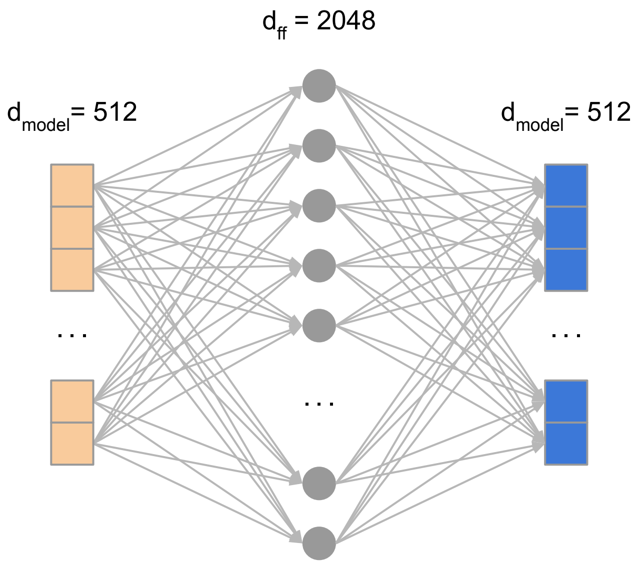
h : 헤드수 (=8)

Insight campus SeSAC

Transformer 인코더 - Feed Forward



Position-wise Feed-Forward Networks



$$FFX(x) = \max\,(0, xW_1 + b_1)W_2 + b_2$$

$$x$$

$$F_1 = xW_1 + b_1$$

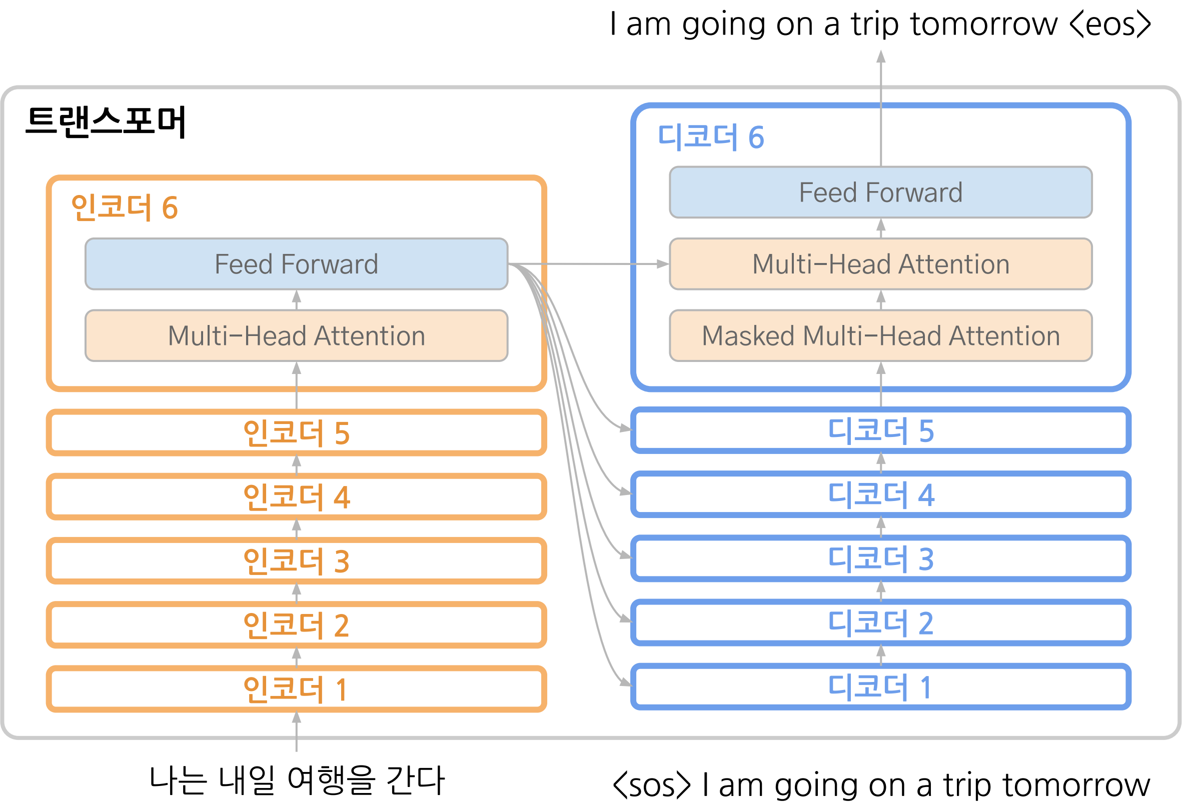
$$F_2 = max(0, F_1)$$

활성 함수 : ReLU

$$F_3 = F_2W_2 + b_2$$

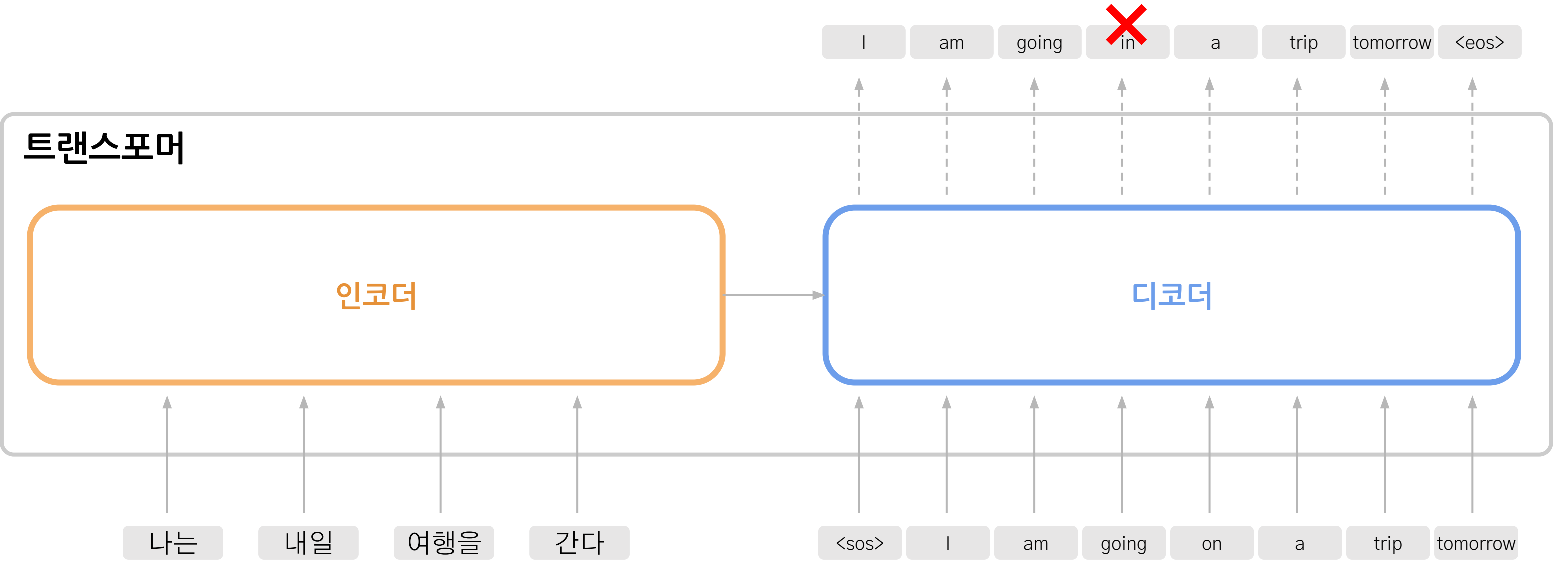
3 Decoder

Transformer 구조 - 학습



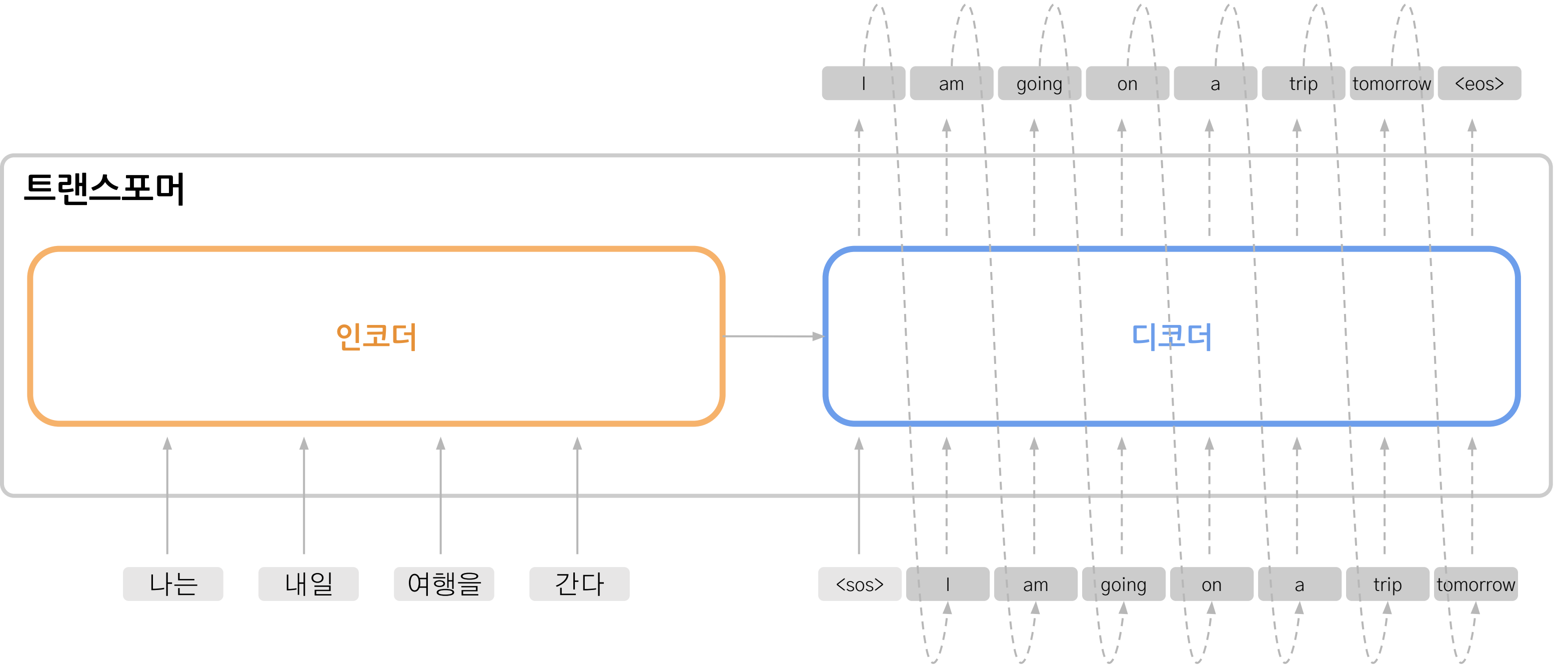
Insight campus SeSAC

Transformer - 학습 (Teacher forcing)



예측 학습을 위해 정답을 디코더에 입력으로 사용 (Teacher forcing)

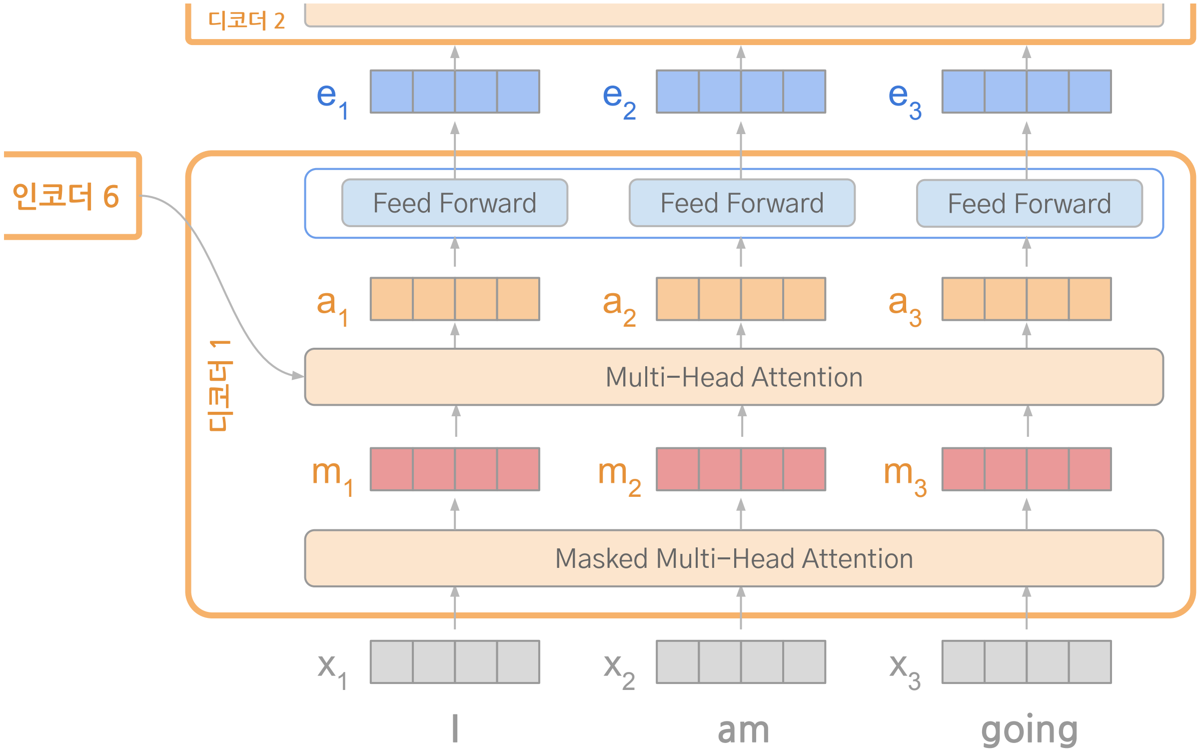
Transformer - 예측



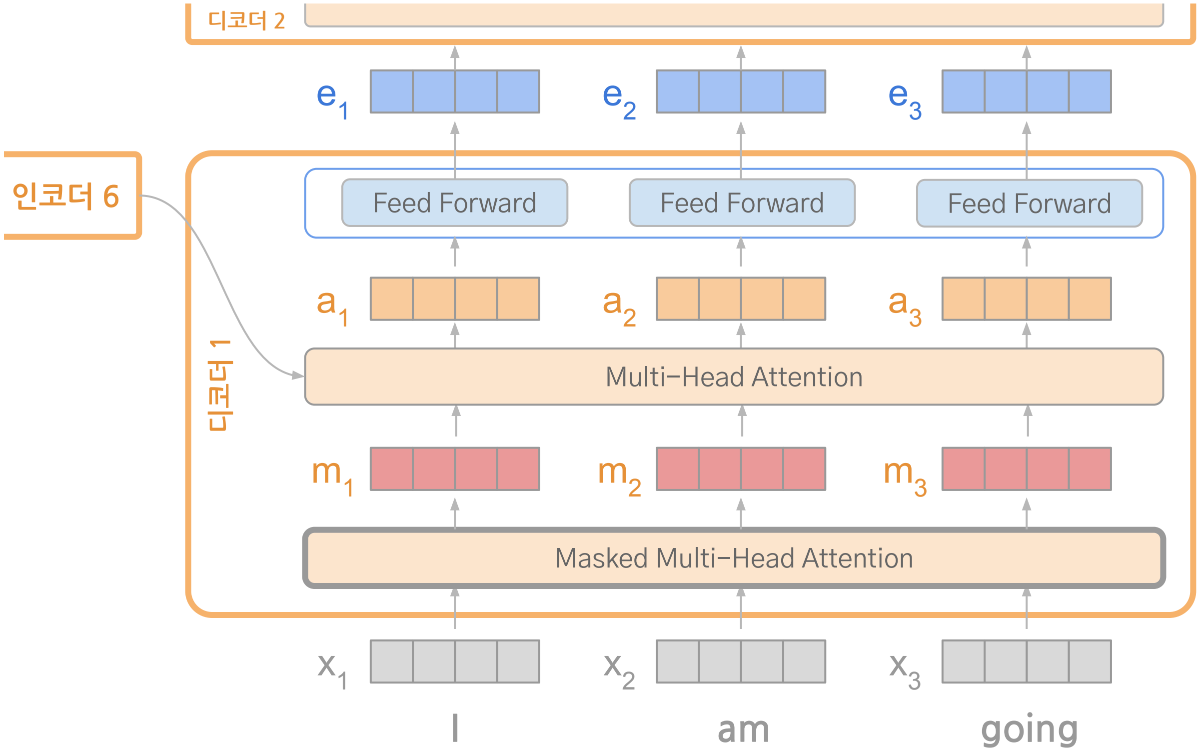
<sos>를 입력하면 첫번째 단어를 예측하고 예측한 단어로 다음 단어를 예측

Insight campus SeSAC

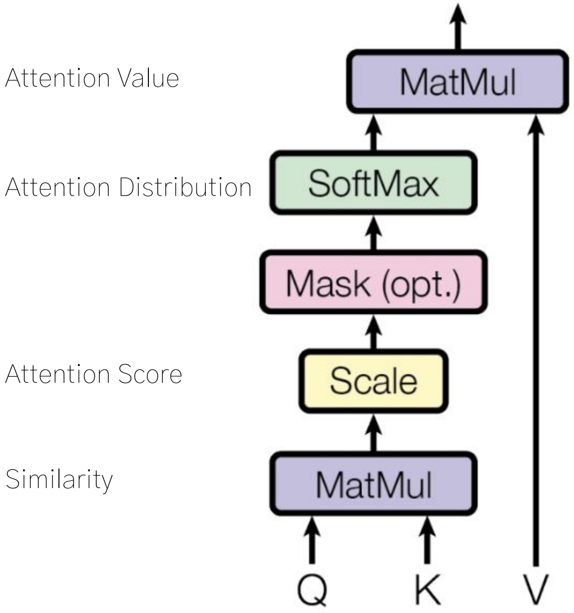
Transformer 디코더



Masked Multi-Head Attention



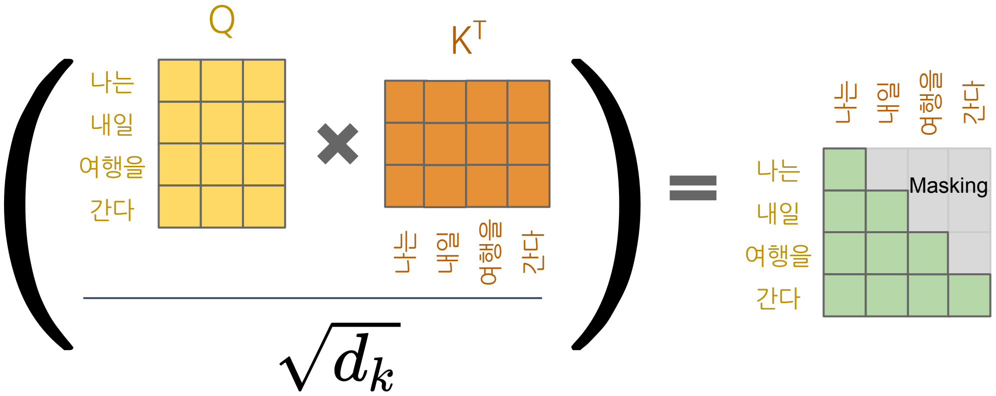
Scaled Dot Product Attention (Masking)



단어간 유사도

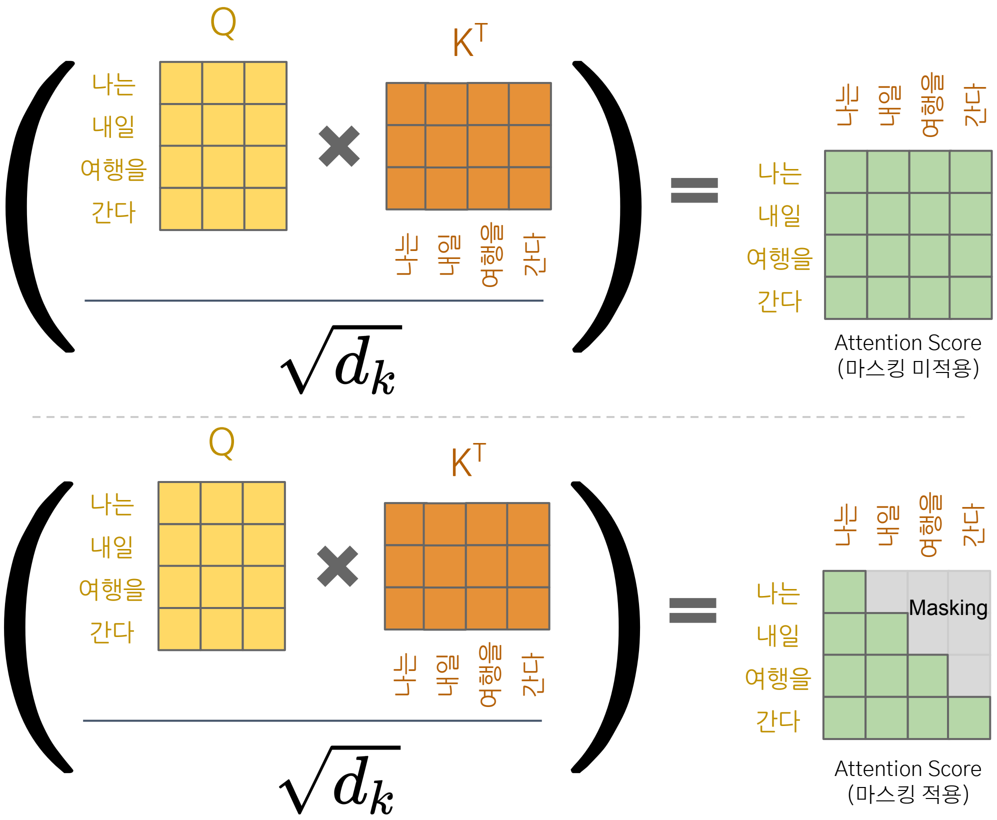
$$Attention(Q, K, V) = softmax_k\left(\frac{\boxed{QK^T}}{\sqrt{d_k}}\right)V$$

어텐션 분포 어텐션 스코어

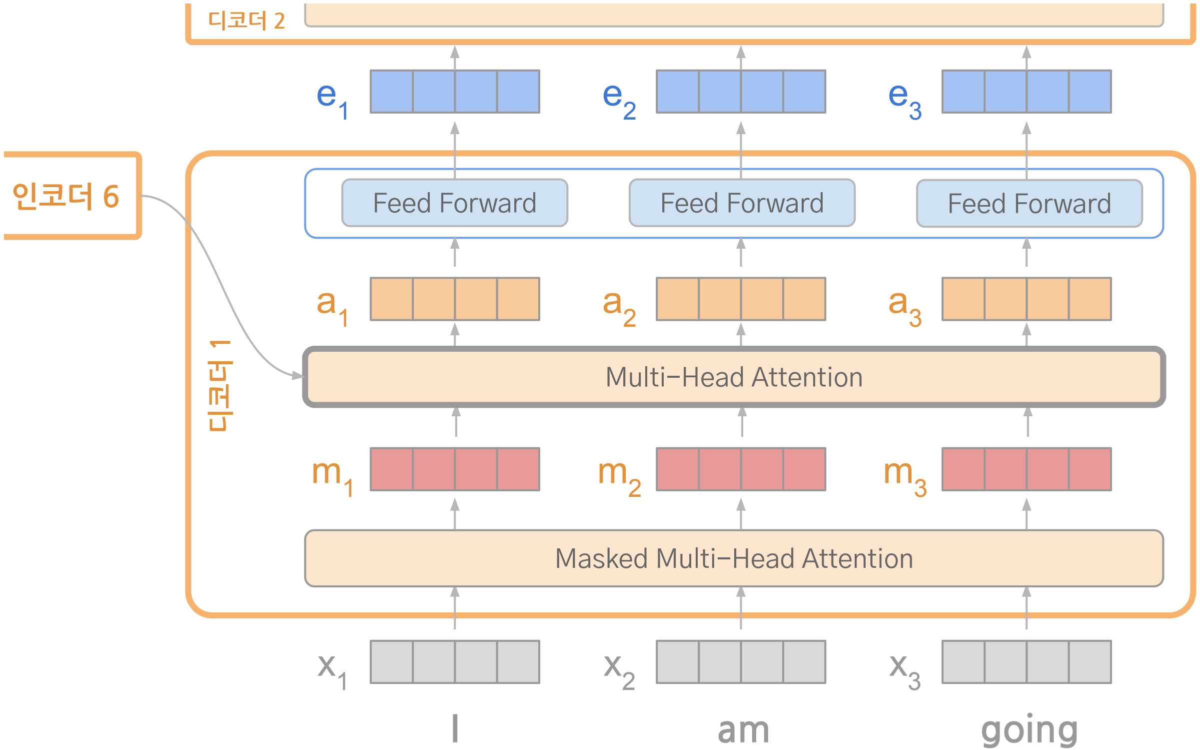


Decoding 시 다음 단어의 Attention을 고려하지 못하도록 Masking

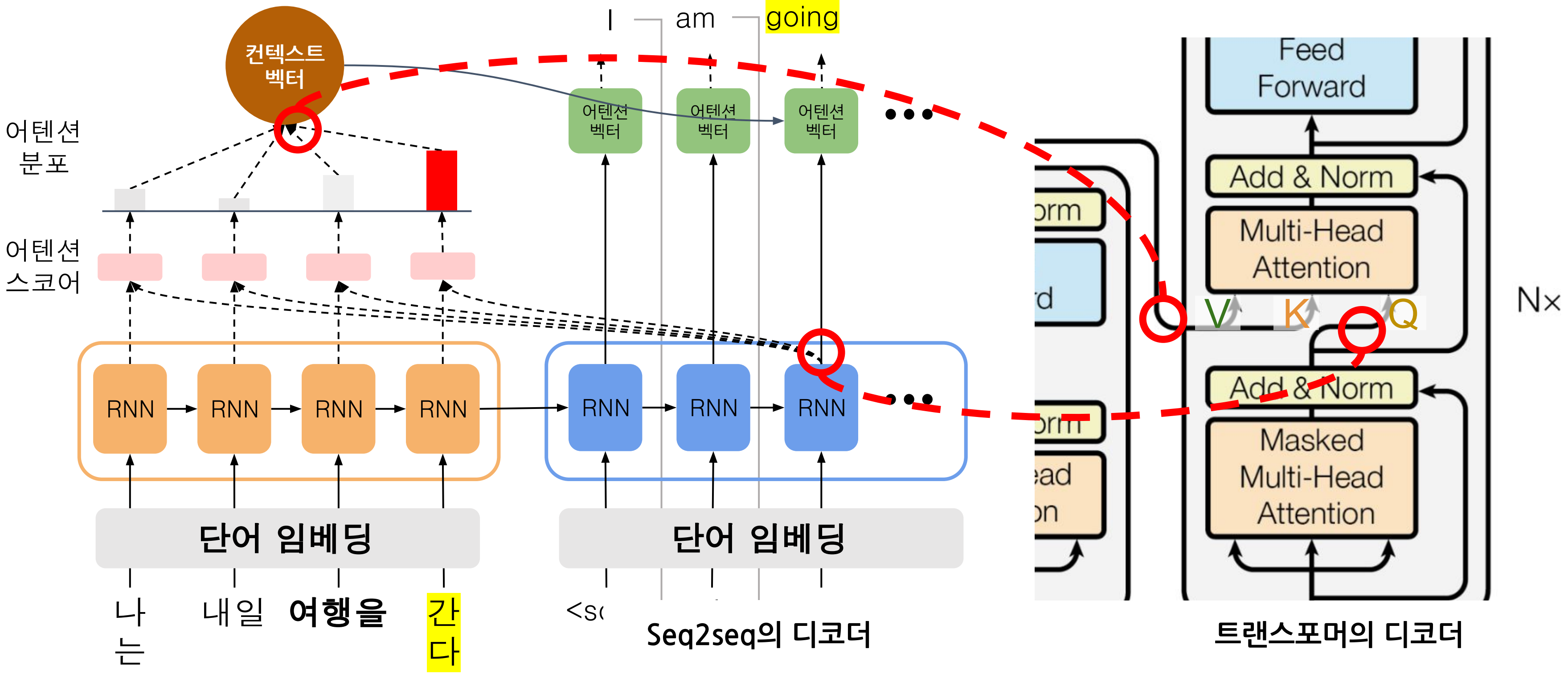
Scaled Dot Product Attention (Masking)



Encoder-Decoder Multi-Head Attention



Transformer 개요 (6)

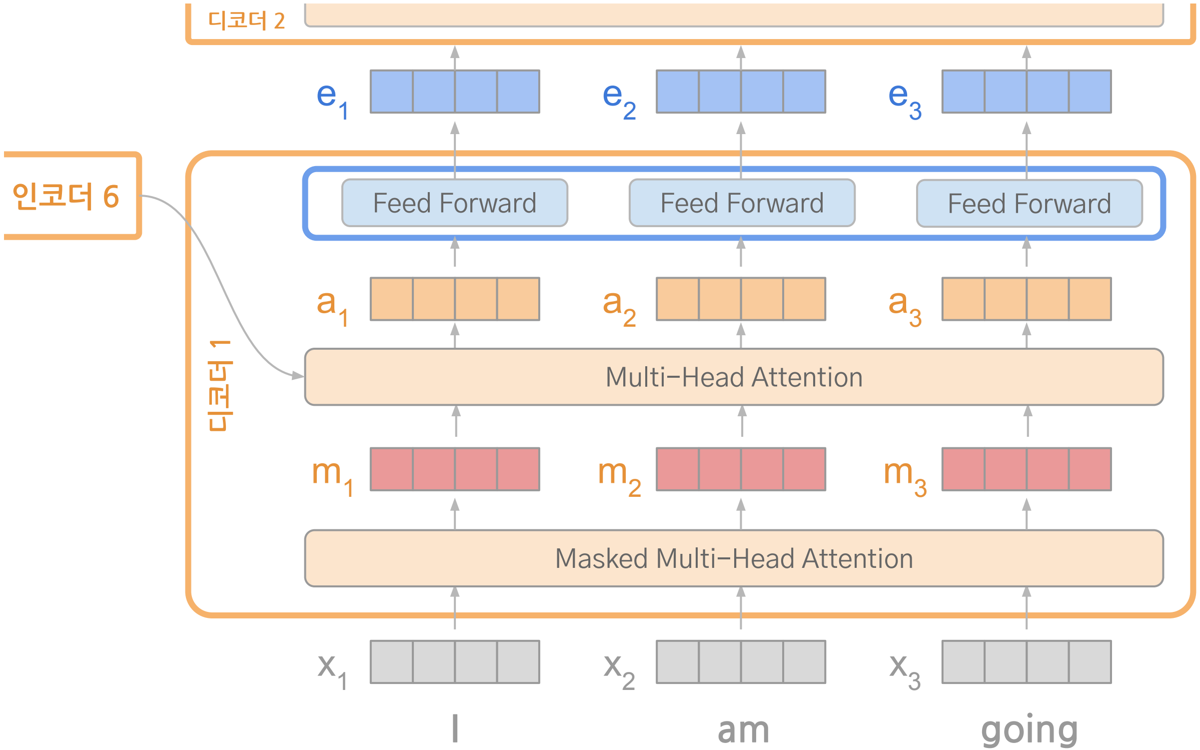


Seq2seq의 디코더

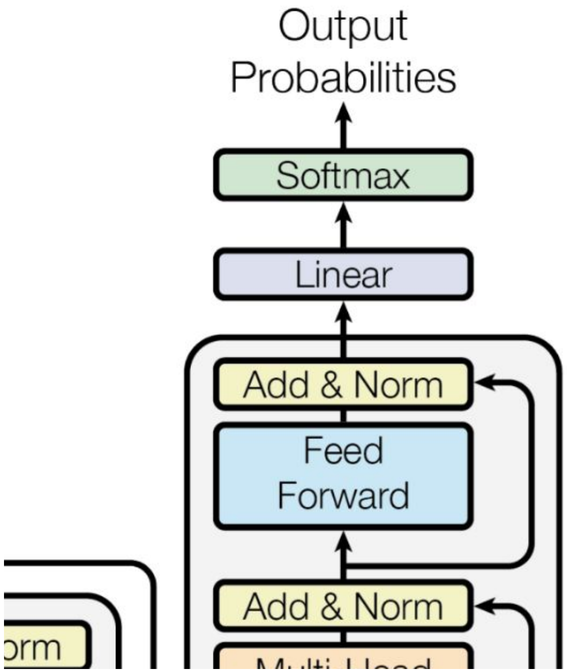
트랜스포머의 디코더

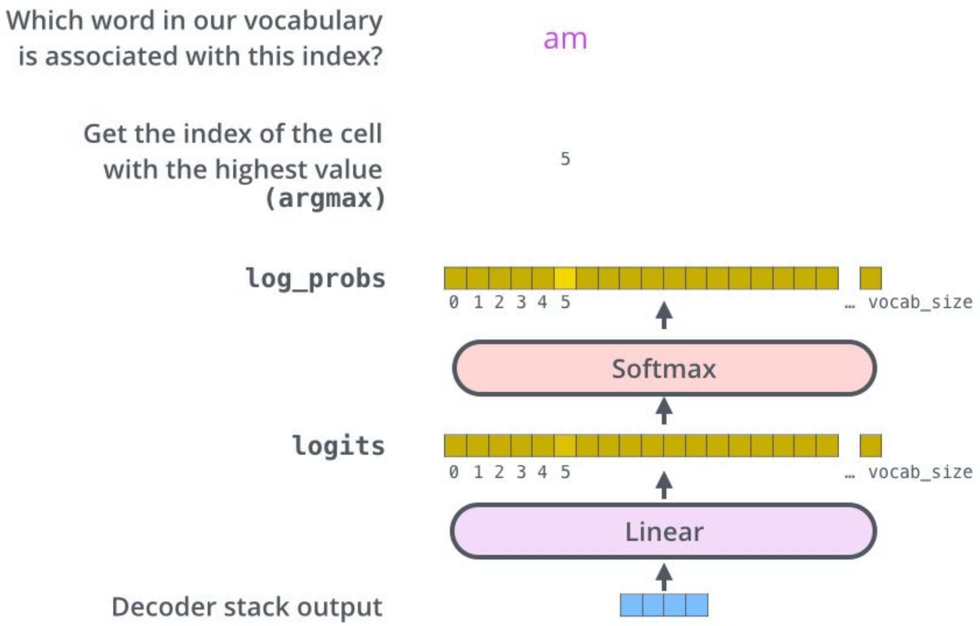
기존 Seq2Seq with Attention과 구조는 다르지만 유사한 적용

Encoder-Decoder Multi-Head Attention

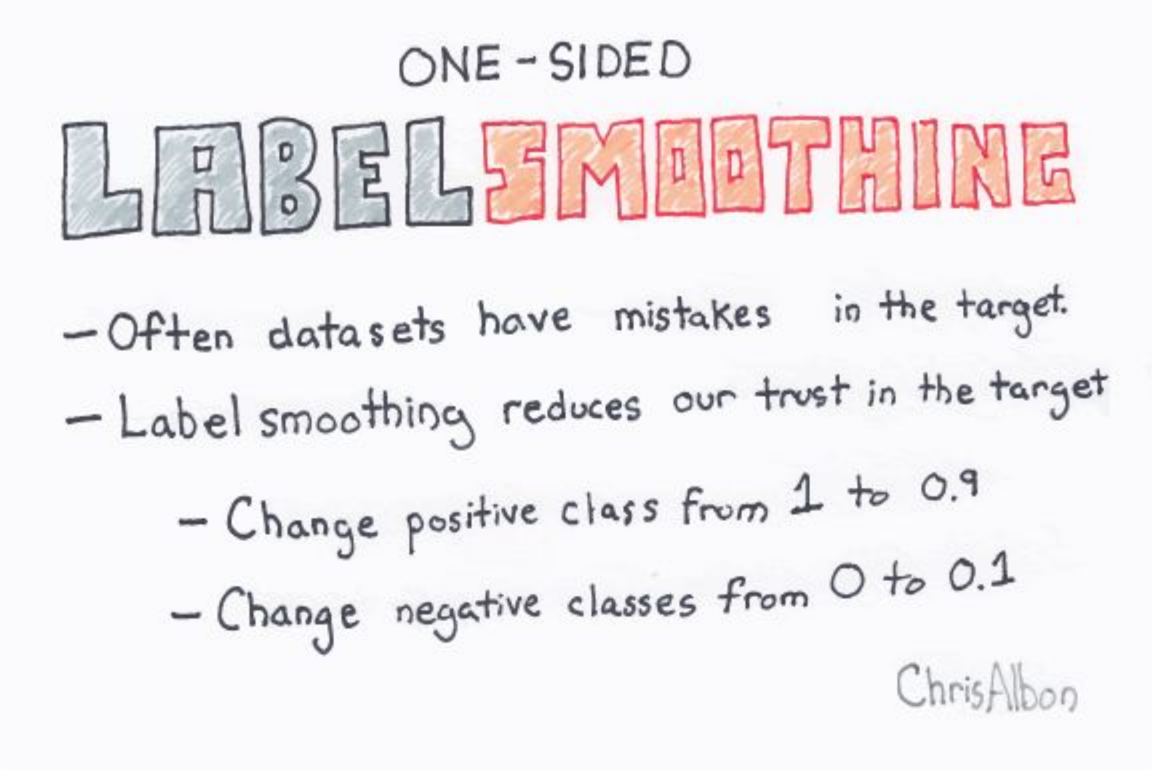


Label Smoothing

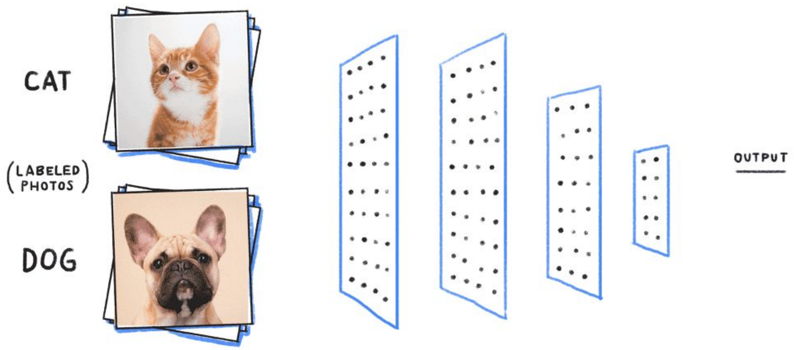


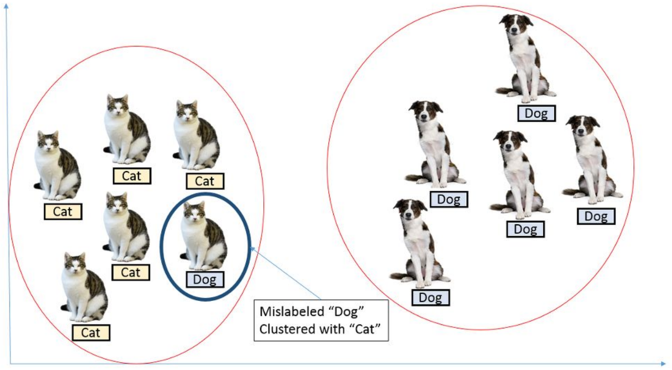


Label Smoothing



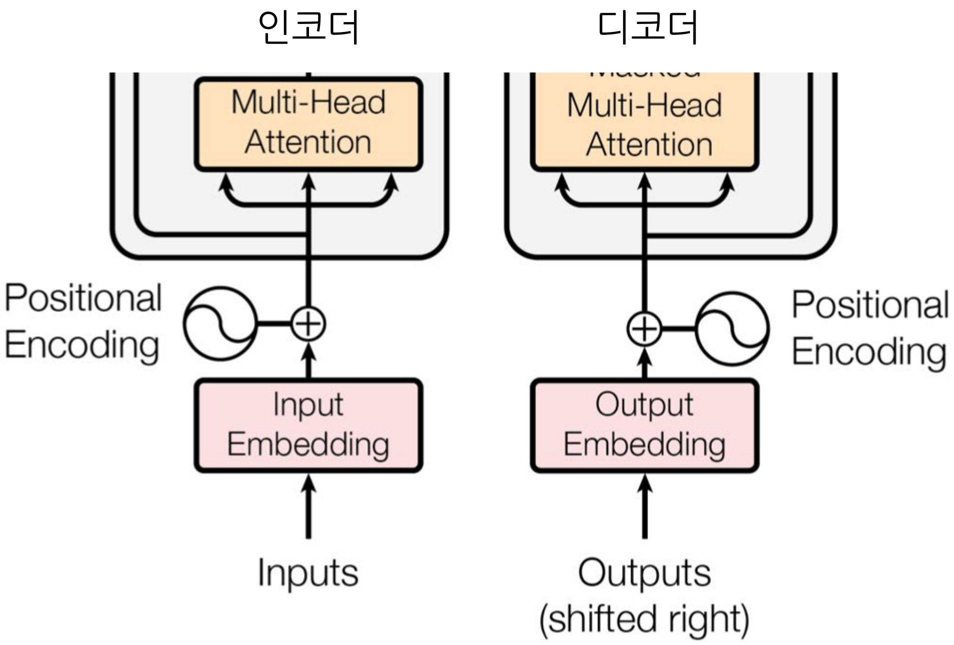
Label Smoothing



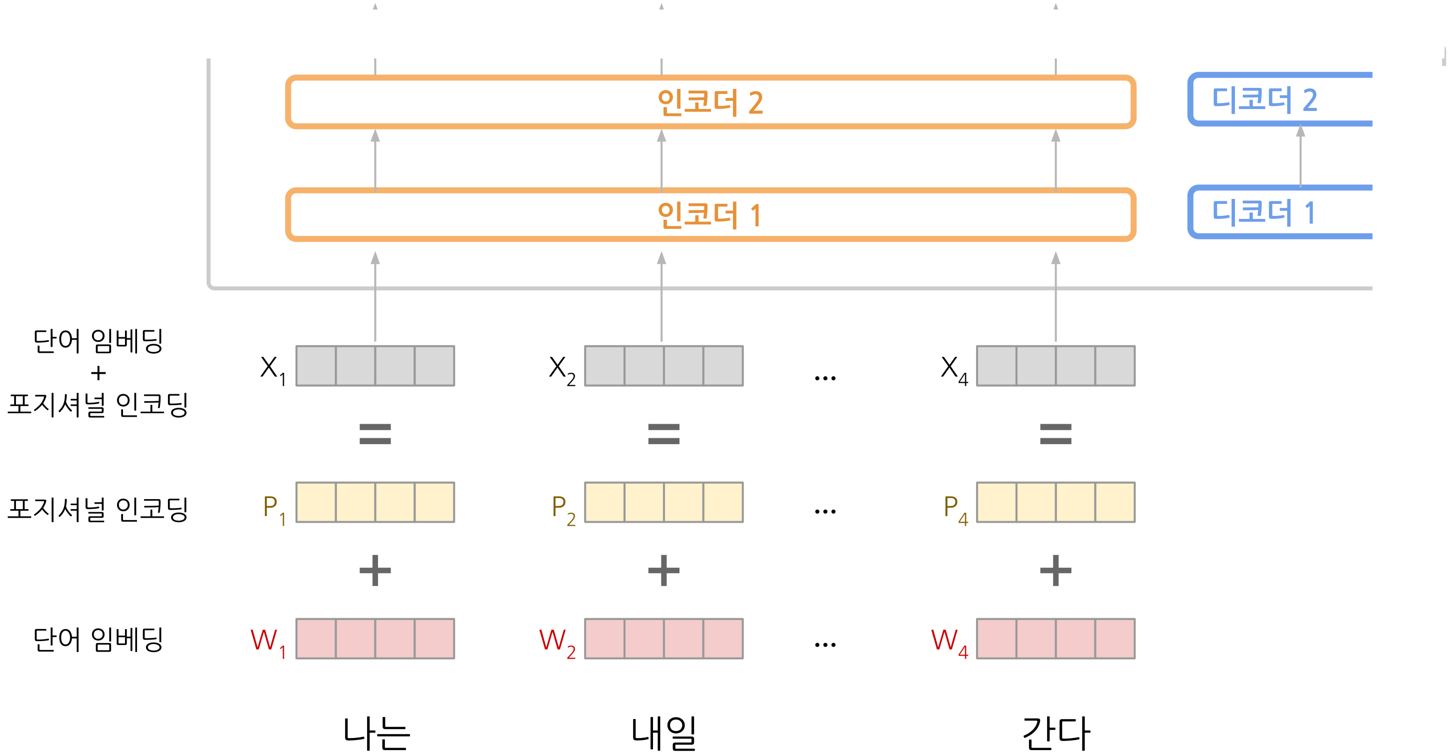


4 Input Embedding

Transformer 개요 (5)

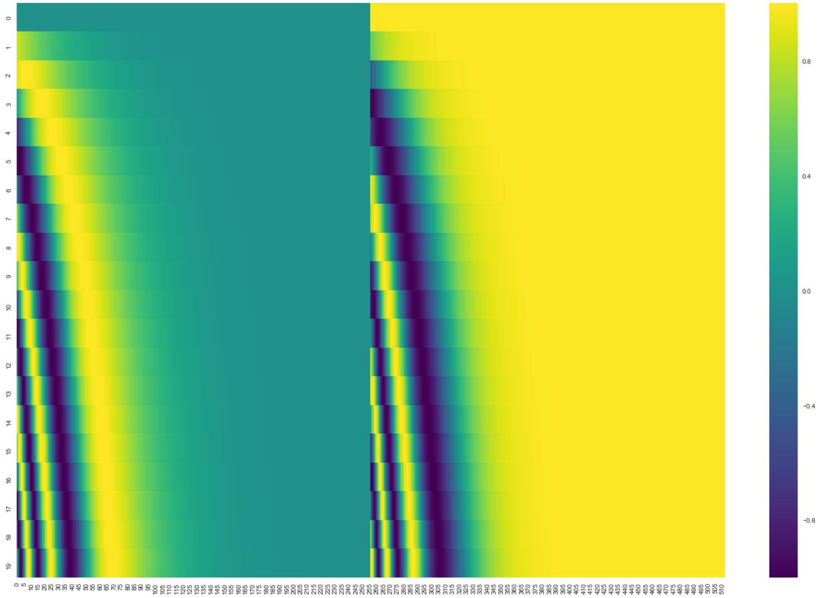


Transformer 구조 - 학습



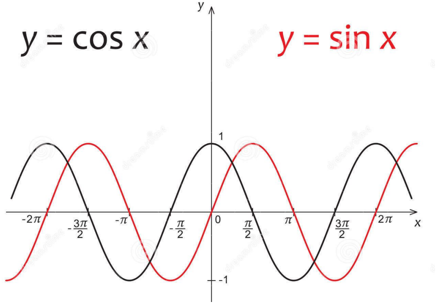
위치(Position)에 대한 절대적 위치를 표현하는 것이 아니라 대변할 수 있는 encoding

Positional Encoding (3)

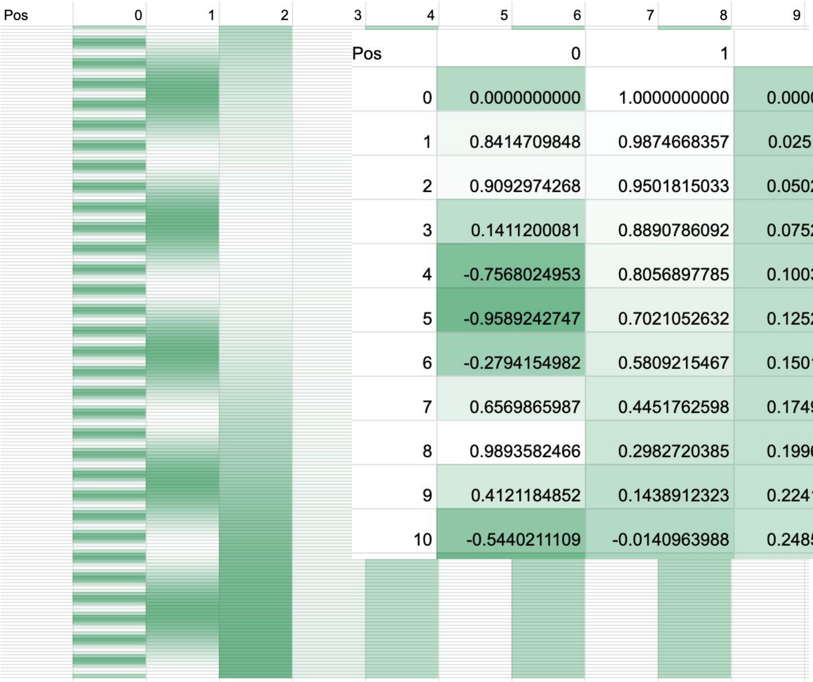


$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

$$\left[sin\left(\frac{pos}{10000^0}\right), cos\left(\frac{pos}{10000^0}\right), sin\left(\frac{pos}{10000^{2/4}}\right), cos\left(\frac{pos}{10000^{2/4}}\right)\right]$$
$$\left[sin\left(pos\right), cos\left(pos\right), sin\left(\frac{pos}{100}\right), cos\left(\frac{pos}{100}\right)\right]$$



Positional Encoding (4)



Pos	0	1	2	3	4	5	6	7	8	
0	0.0000000000	1.0000000000	0.0000000000	1.0000000000	0.0000000000	1.0000000000	0.0000000000	1.0000000000	0.0000000000	1.000000
1	0.8414709848	0.9874668357	0.0251162229	0.9999920755	0.0006309573	0.9999999950	0.0000158489	1.0000000000	0.0000003981	1.000000
2	0.9092974268	0.9501815033	0.0502165994	0.9999683023	0.0012619144	0.9999999800	0.0000316979	1.0000000000	0.0000007962	1.000000
3	0.1411200081	0.8890786092	0.0752852930	0.9999286807	0.0018928709	0.9999999550	0.0000475468	1.0000000000	0.0000011943	1.000000
4	-0.7568024953	0.8056897785	0.1003064873	0.9998732112	0.0025238267	0.9999999200	0.0000633957	0.9999999999	0.0000015924	1.000000
5	-0.9589242747	0.7021052632	0.1252643958	0.9998018949	0.0031547815	0.9999998750	0.0000792447	0.9999999999	0.0000019905	1.000000
6	-0.2794154982	0.5809215467	0.1501432720	0.9997147328	0.0037857350	0.9999998200	0.0000950936	0.9999999999	0.0000023886	1.000000
7	0.6569865987	0.4451762598	0.1749274192	0.9996117263	0.0044166871	0.9999997550	0.0001109425	0.9999999998	0.0000027868	1.000000
8	0.9893582466	0.2982720385	0.1996012004	0.9994928770	0.0050476373	0.9999996800	0.0001267915	0.9999999998	0.0000031849	1.000000
9	0.4121184852	0.1438912323	0.2241490484	0.9993581869	0.0056785856	0.9999995950	0.0001426404	0.9999999997	0.0000035830	1.000000
10	-0.5440211109	-0.0140963988	0.2485554753	0.9992076581	0.0063095316	0.9999995000	0.0001584893	0.9999999997	0.0000039811	1.000000

위치(Position)에 대한 Unique한 벡터가 생성됨. 동일 위치는 같은 벡터가 생성됨

Positional Encoding (5)

	0번째 인덱스	1번째 인덱스	2번째 인덱스	3번째 인덱스
포지셔널 인코딩	0.0 | 1.0 | 0.0 | 1.0	0.84 | 0.56 | 0.80 | 0.62	0.90 | -0.35 | 0.95 | -0.22	0.14 | -0.96 | 0.34 | -0.90
	+	+	+	+
단어 임베딩				
	나는	내일	여행을	간다

5 잔차연결 & 정규화

Insight campus SeSAC

잔차연결 & 정규화

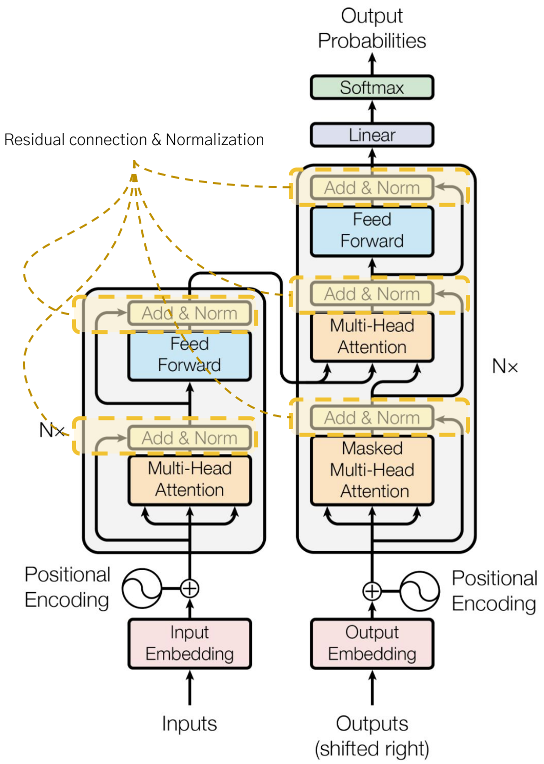
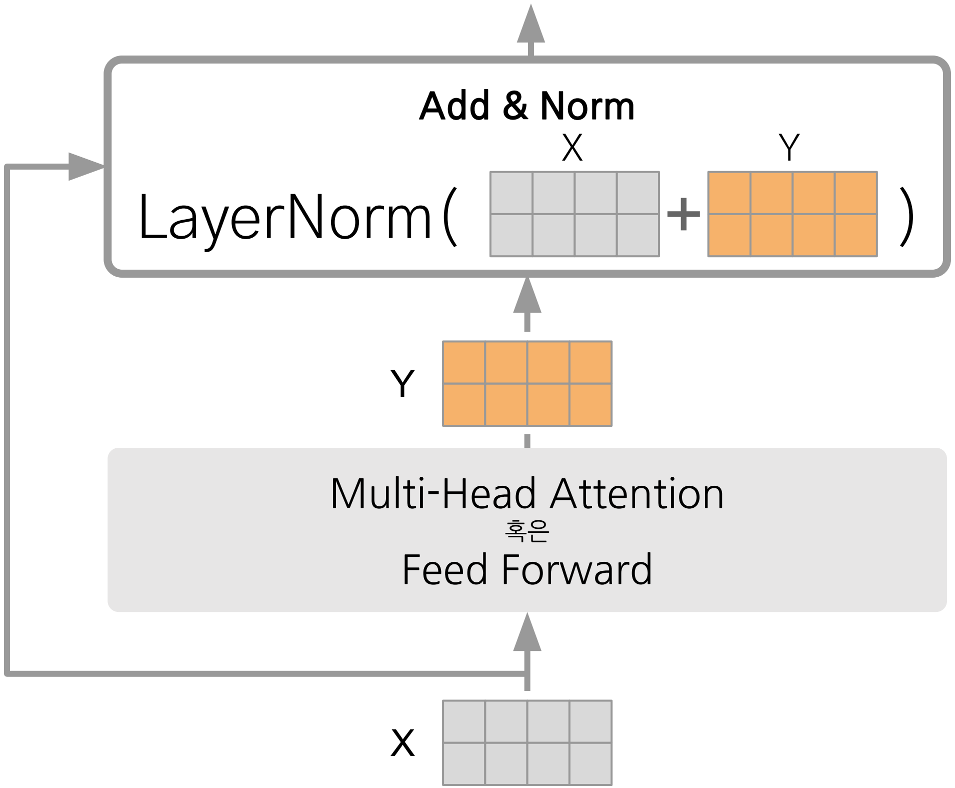


Figure 1: The Transformer - model architecture.

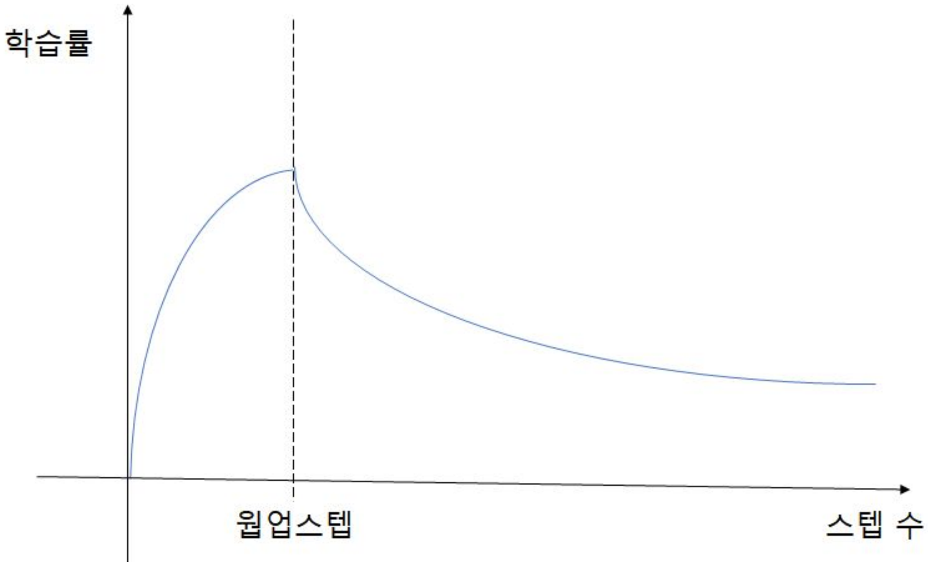
잔차연결 & 정규화



입력 행렬이 서브레이어를 통과했을때 정보가 유실되는 것을 막고자 잔차연결

6 학습

트랜스포머의 학습



$$lrate = d_{model}^{-0.5} \cdot min(stepnum^{-0.5}, stepnum \cdot warmupsteps^{-1.5})$$

웜업스텝 전까지 학습률을 올렸다가 웜업스텝에 도달하면 학습률을 감소며 학습

7 트랜스포머 리뷰

Transformer Review (1

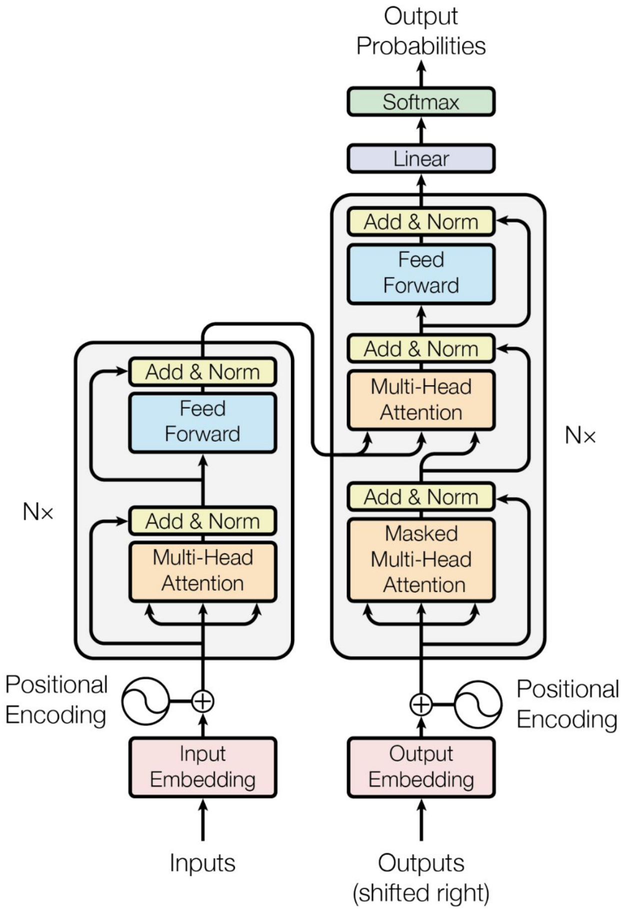
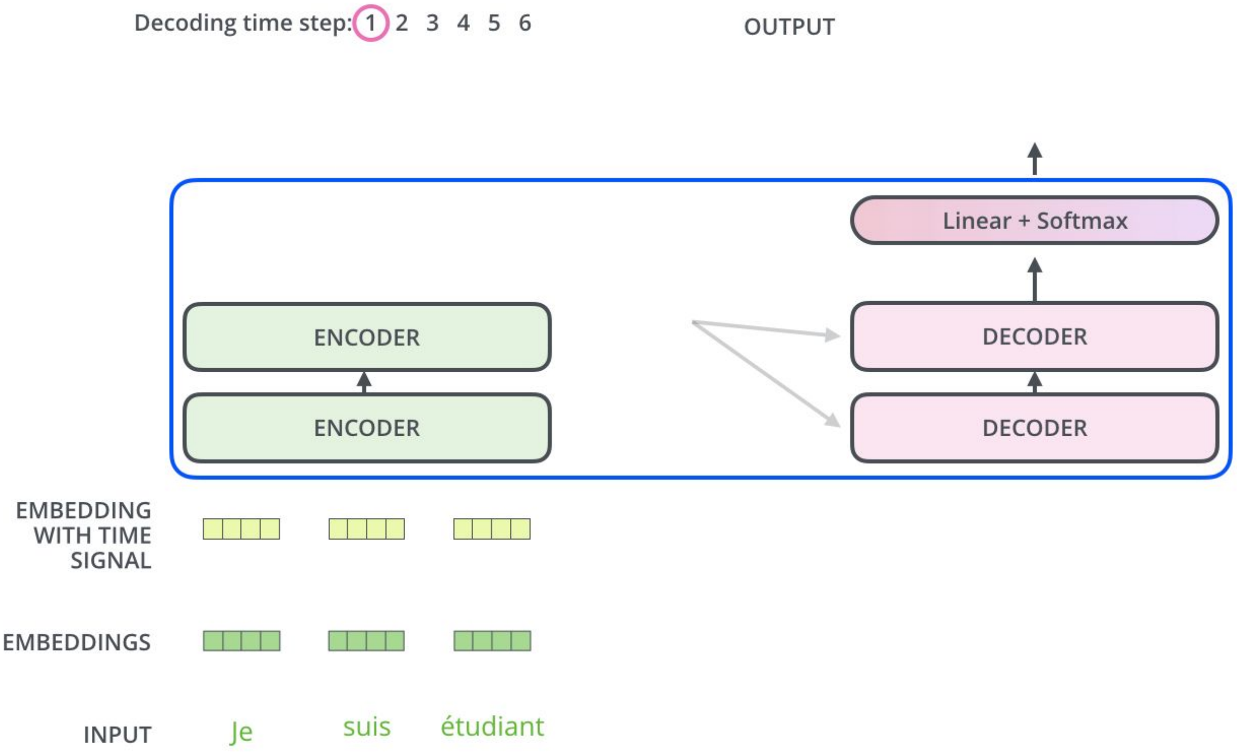
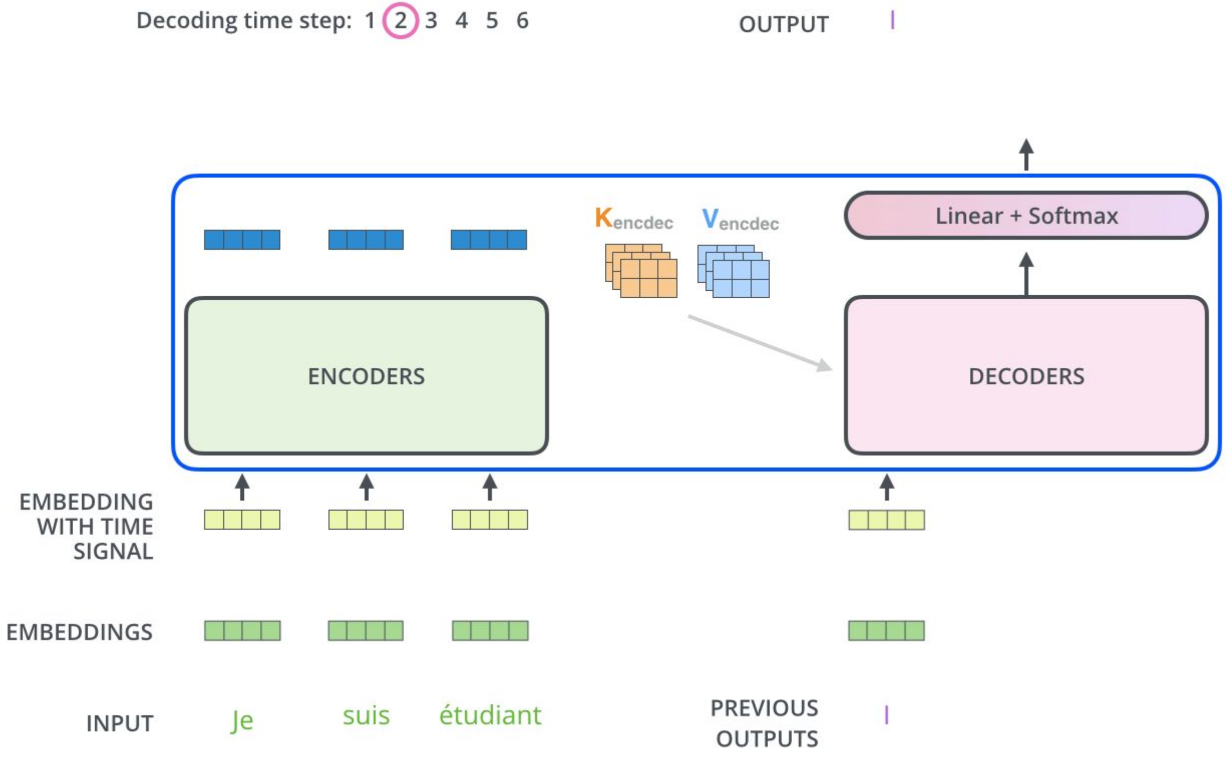


Figure 1: The Transformer - model architecture.

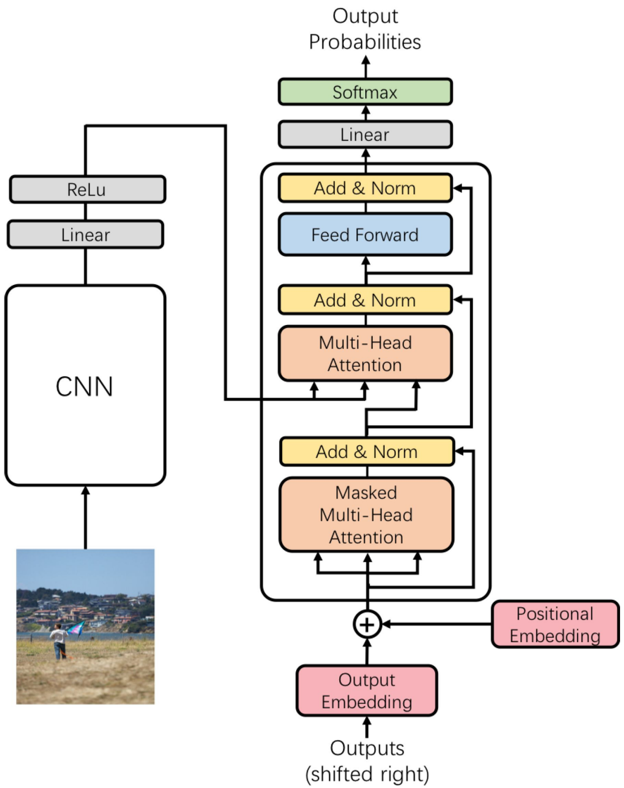
Transformer Review (2)



Transformer Review (3)



Transformer 활용



https://www.mdpi.com/2076-3417/8/5/739/html

감사합니다.

daniel@fins.ai