



NLP Advanced

19기 정규세션

TOBIG'S 18기 국주현

Contents



19기 정규세션

TOBIG'S 18기 국주현

Unit 01 | Seq2Seq

Unit 02 | Attention mechanism

Unit 03 | Transformer

Unit 04 | 과제



19기 정규세션

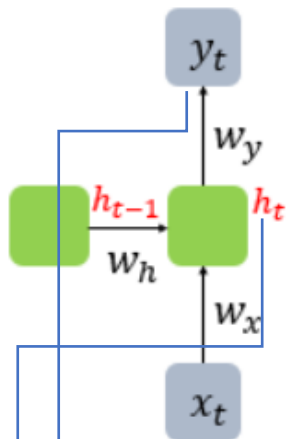
TOBIG'S 18기 국주현

Unit 01

Seq2Seq

순환 신경망(Recurrent Neural Network, RNN)

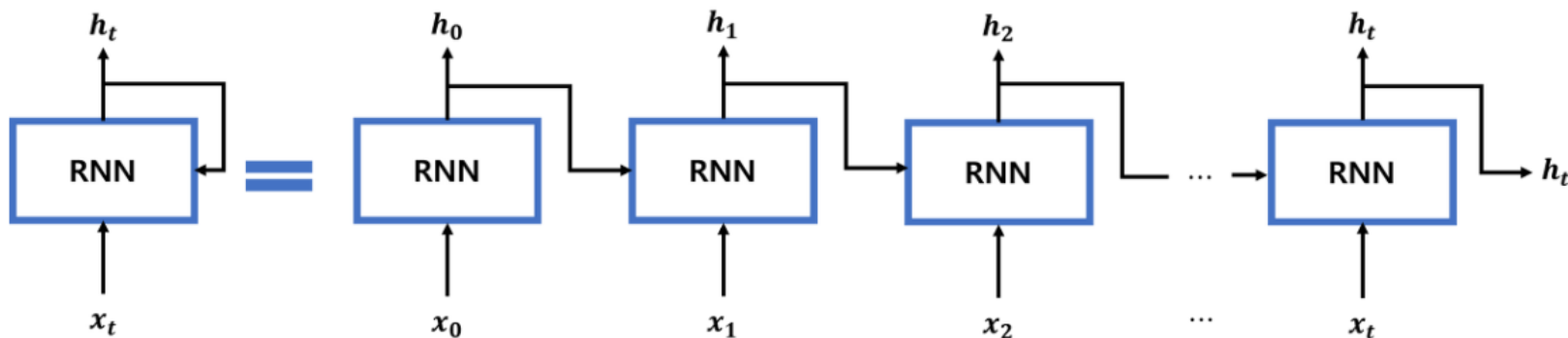
- 은닉층에서 나온 결과값이 다시 은닉층으로 돌아가 새로운 입력값과 연산을 수행하는 순환구조의 신경망
- 연속적인 시퀀스를 처리하기 위해 설계됨
- RNN은 시점에 따라서 입력을 받고, 현재 시점의 hidden state 인 h_t 를 계산하기 위해 직전 시점의 hidden state 인 h_{t-1} 을 입력 받는다.



출력층 : $y_t = f(W_y h_t + b)$

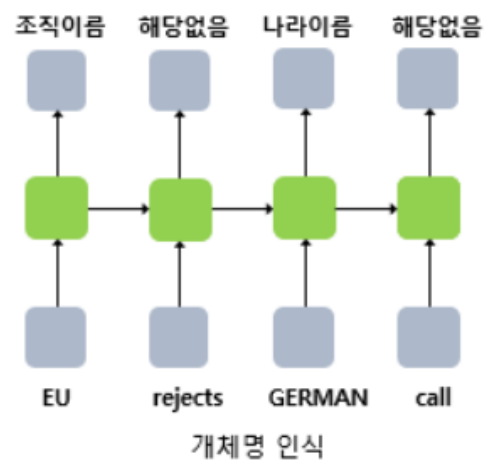
은닉층 : $h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$

이전 은닉층 해당 시점에서의 input



순환 신경망(Recurrent Neural Network, RNN)

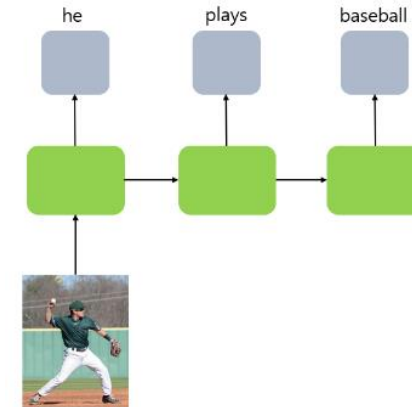
- Rnn은 설계하기 나름이지만 대표적으로 아래와 같은 유형이 있다.



다 대 다 (many to many)



다 대 일 (many to one)



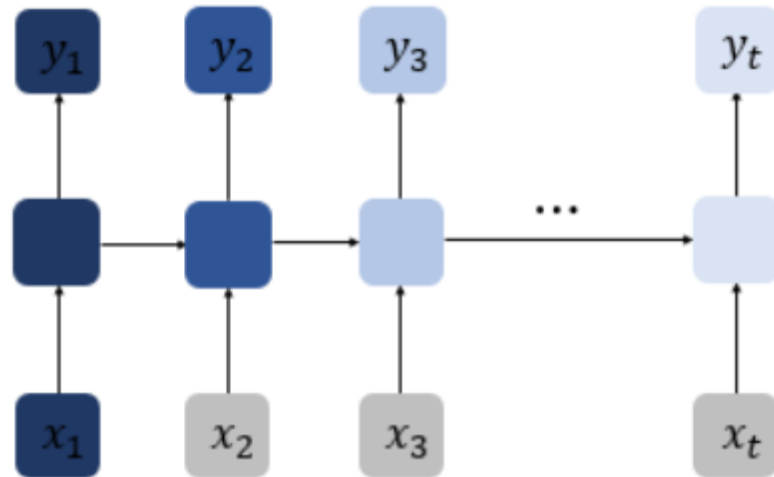
이미지 캡셔닝

일 대 다 (one to many)

- Nlp에서 각 시점(time step)의 입력은 주로 단어 벡터 또는 형태소 벡터가 된다.

Long – Term Dependency Problem

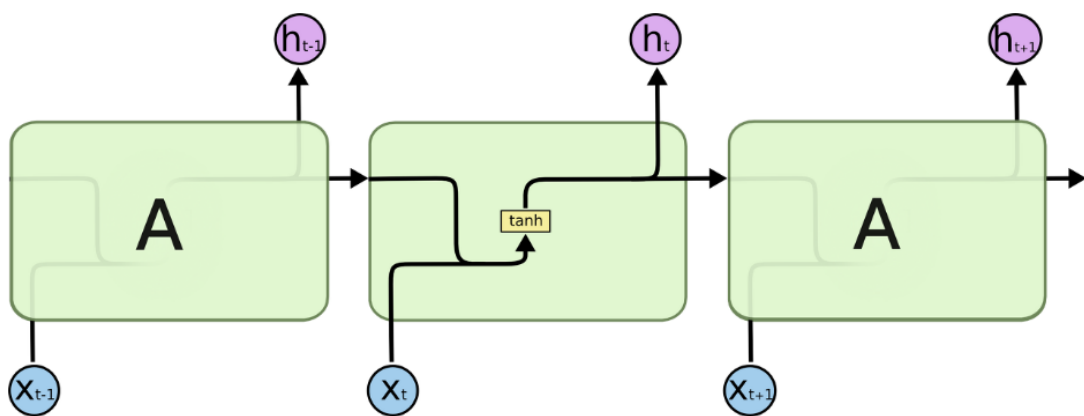
- 기존의 RNN은 반영해야하는 시점이 길어지면서 앞에 있던 정보가 소실되는 '장기 의존성 문제' 를 가지고 있다.
- 뒤로 갈수록 x_1 의 영향력이 점점 줄어드는 것을 확인 가능하다.



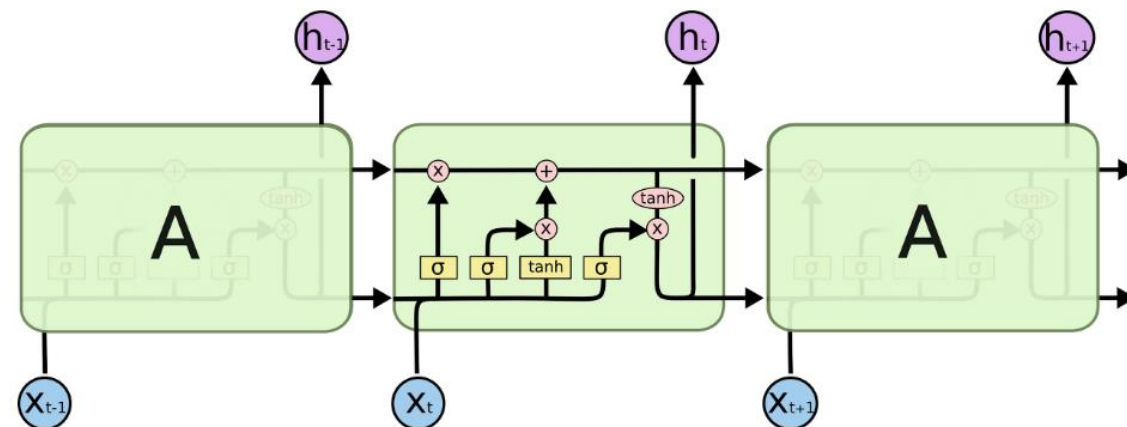
이를 개선하기 위한 방안이 LSTM!

LSTM

- RNN에 비해 긴 의존 기간을 필요로 하는 학습을 수행할 능력을 갖고 있음
- LSTM도 RNN과 같이 체인과 같은 구조를 가지고 있음
- LSTM은 단순한 neural network layer 한 층 대신에, 4개의 layer가 특별한 방식으로 서로 정보를 주고 받도록 되어 있음



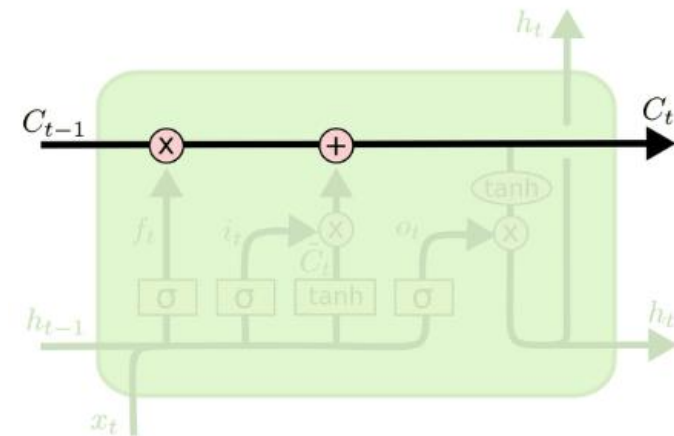
Vanila RNN



LSTM

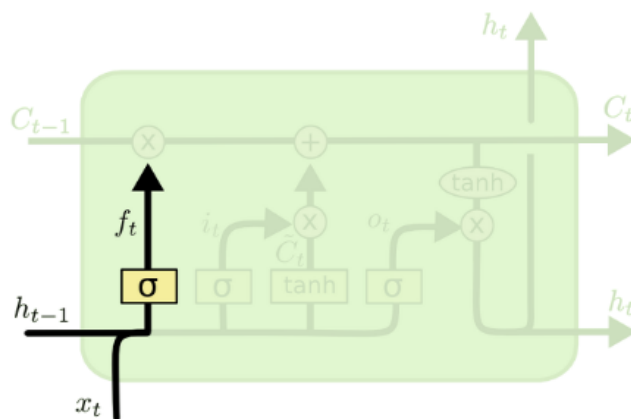
LSTM – cell state

- 이전 시점의 셀 상태가 다음 시점의 셀 상태를 구하기 위한 입력으로서 사용됨



LSTM – 삭제 게이트

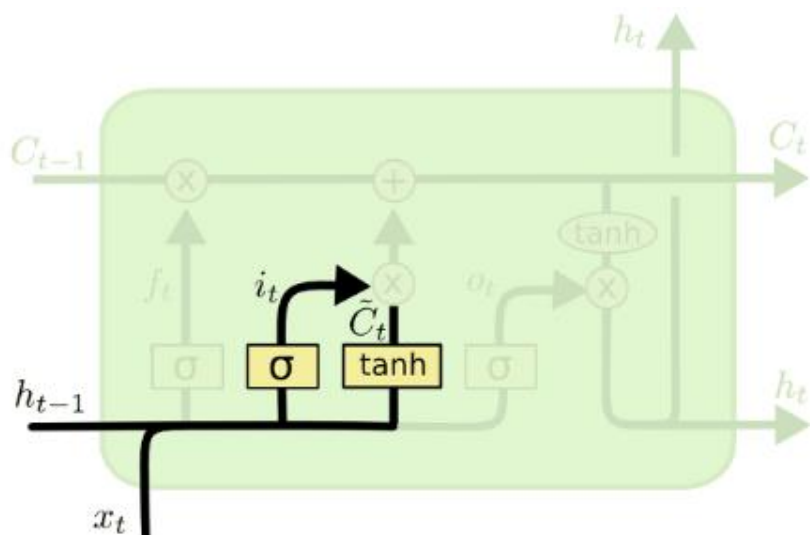
- LSTM의 첫 단계로, cell state로부터 어떤 정보를 버릴 것인지를 정하는 게이트
- sigmoid layer에 의해 결정
- h_{t-1} 과 x_t 를 받아서 0과 1 사이의 값을 C_{t-1} 에 보내준다



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

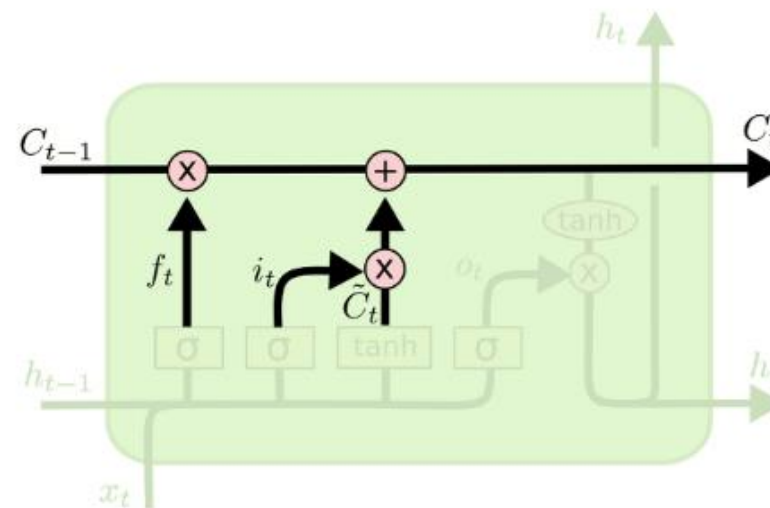
LSTM – 입력 게이트

- 새로운 정보 중 어떤 것을 cell state에 저장할 것인지를 정하는 게이트
- "input gate layer"라고 불리는 sigmoid layer가 어떤 값을 업데이트할 지정 : i_t
- tanh layer가 새로운 후보 값들인 \tilde{C}_t 라는 벡터를 만들고 cell state에 더할 준비를 하게 됨 : \tilde{C}_t



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

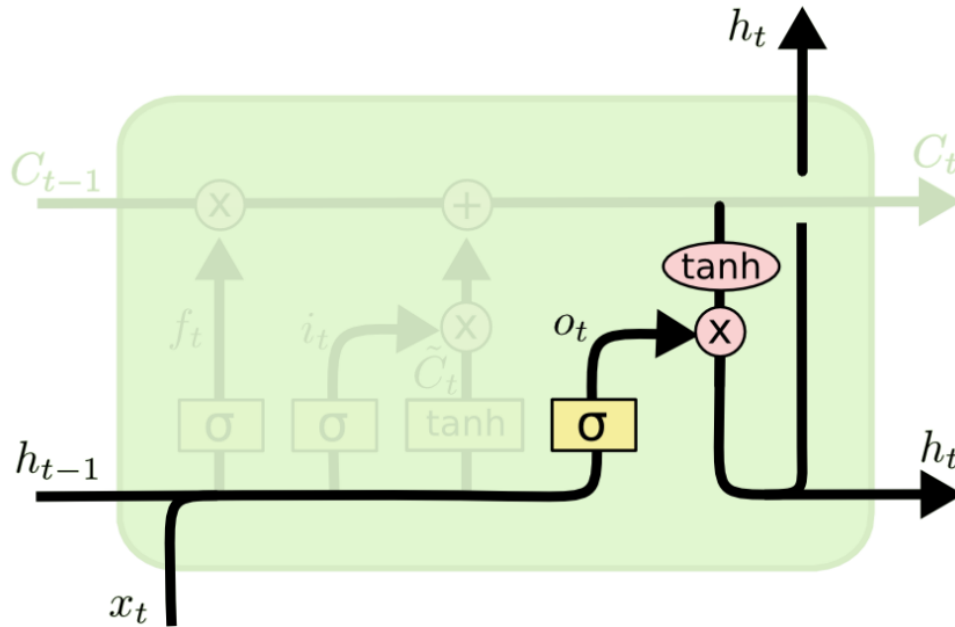


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Cell state 업데이트!

LSTM – 출력 게이트

- 현재 hidden state를 만드는 게이트
- 새로 만든 cell state에 tanh 를 적용하고, 이전 hidden state 와 현재 input 값을 가중치와 곱해준다.
- sigmoid gate의 output과 곱해주면 현재 hidden state 완성



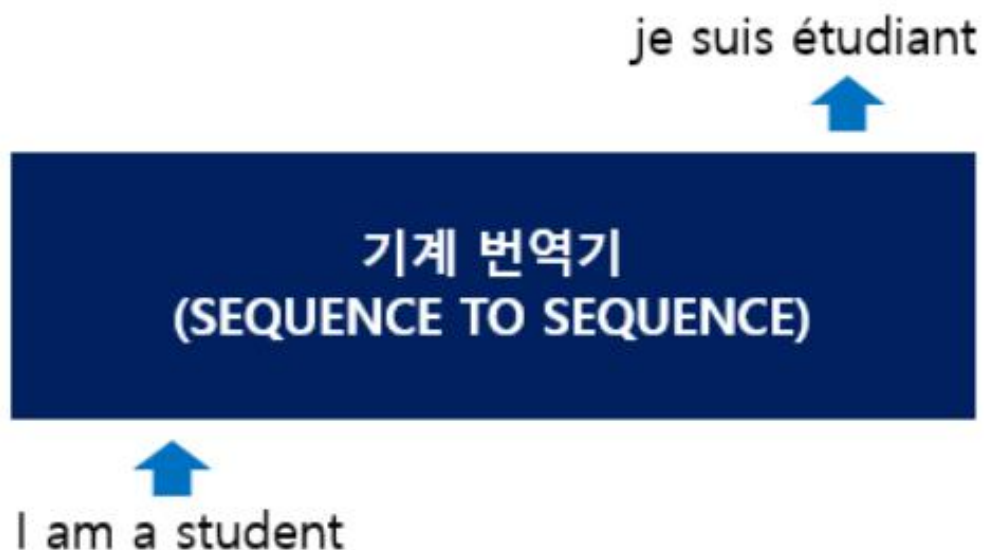
$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

Seq2Seq(Sequence-to-Sequence)

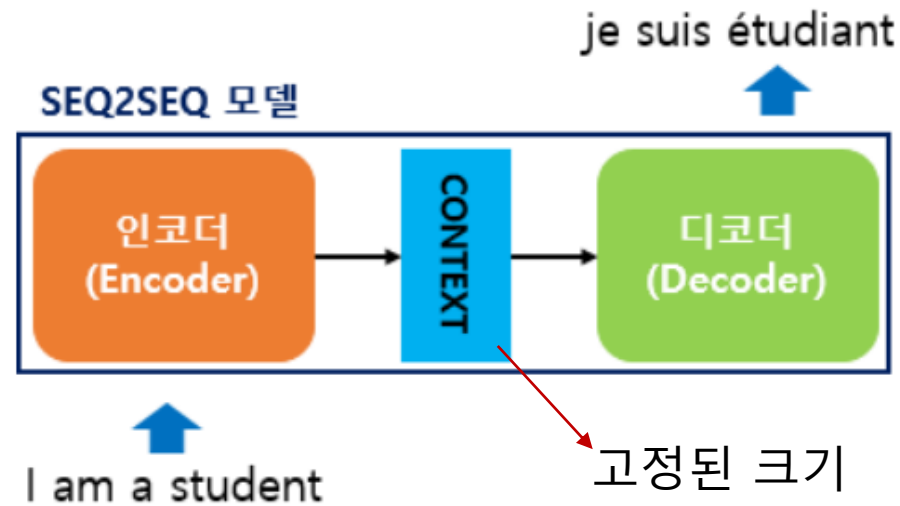
- 입력된 시퀀스로부터 다른 도메인의 시퀀스를 출력하는 다양한 분야에서 사용되는 모델

Ex) 챗봇(Chatbot)과 기계 번역(Machine Translation)



Seq2Seq(Sequence-to-Sequence)

- seq2seq는 크게 인코더와 디코더라는 두 개의 모듈로 구성됨
- 인코더는 입력 문장의 모든 단어들을 순차적으로 입력 받은 뒤에 마지막에 이 모든 단어 정보들을 압축해서 하나의 벡터(context vector)를 만들게 됨
- 디코더는 컨텍스트 벡터를 받아서 번역된 단어를 한 개씩 순차적으로 출력



- 하지만, seq2seq는 번역 scale이 커졌을 때 고정된 크기의 벡터로 인한 병목현상을 발생시킬 수 있음
- 그래서 등장한 기법이 어텐션을 활용한 기계번역!



19기 정규세션

TOBIG'S 18기 국주현

Unit 02

Attention machanism

Unit 02 | Attention mechanism





19기 정규세션

TOBIG'S 18기 국주현

어학사전

영어사전

attention 미국·영국 [ə'tenʃn]  영국식  ★★ [다른 뜻\(4건\)](#) | [예문보기](#)

1. 주의 (집중), 주목 2. 관심, 흥미 3. (관심을 끌기 위한) 행동

프랑스어사전

attention [atãsjõ]  ★★ [다른 뜻\(2건\)](#) | [예문보기](#)

[여성명사] 1. 주의(력),조심,긴장,관심 2. 친절,정중,배려 = amabilité,empressement,prévenance

Attention: 풀고자 하는 Task에 핵심이 되는 정보를 찾아서 집중!

Unit 02 | Attention mechanism



19기 정규세션

TOBIG'S 18기 국주현

긍정과 부정을 맞추는 영화 리뷰 분류 Task

훈련 데이터

리뷰1 : 와 이 영화 진심 존잼!!!

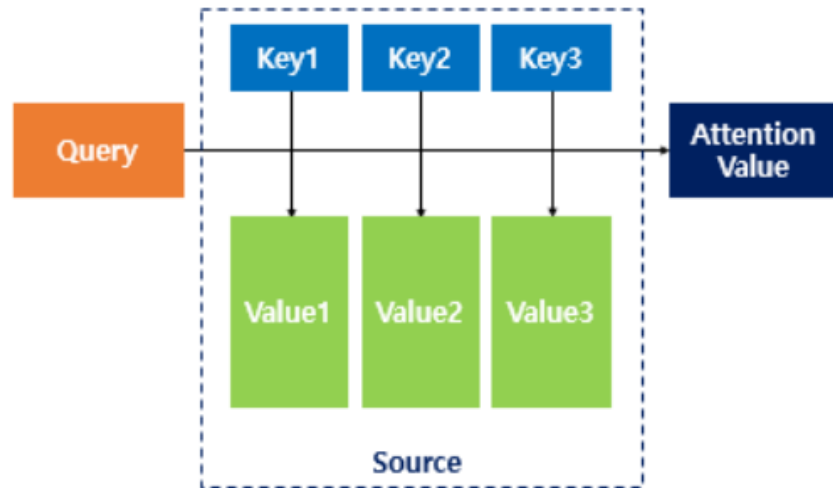
리뷰2 : 실망대망. 교과서적 연출로도 평타 이상은 가능
할 좀비소재를 이렇게까지...

예측 결과

긍정

부정

Attention mechanism



- 1) 주어진 특정 시점 t 의 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 유사도를 각각 구함!
- 2) 구해낸 이 유사도를 키와 맵핑 되어있는 각각의 '값(Value)'에 반영
- 3) 그리고 유사도가 반영된 '값(Value)'을 모두 더해서 리턴!
이를 어텐션 값(Attention Value)

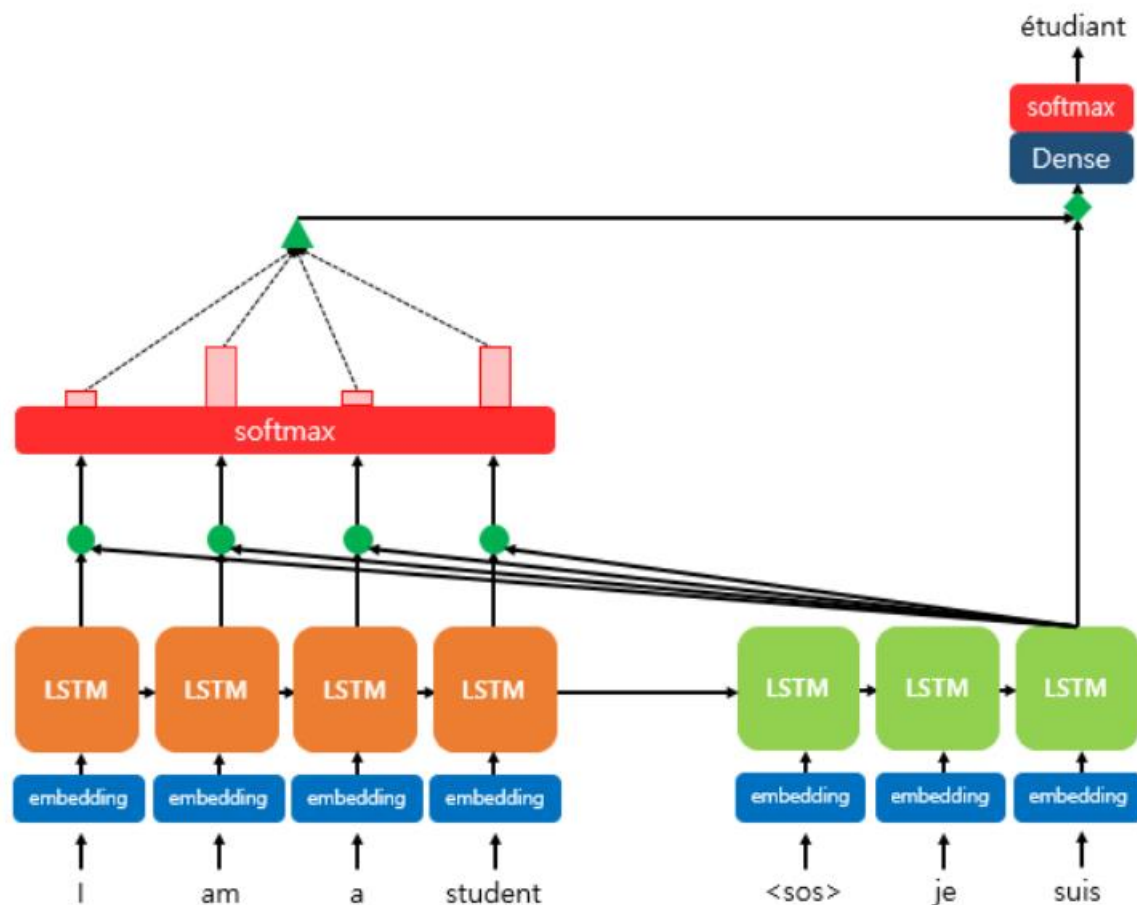
seq2seq 어텐션 알고리즘이 쓰인 모델에서

Q = Query : t 시점의 디코더 셀에서의 은닉 상태

K = Keys : 모든 시점의 인코더 셀의 은닉 상태들

V = Values : 모든 시점의 인코더 셀의 은닉 상태들

닷-프로덕트 어텐션(Dot-Product Attention)



- 가장 기본적인 어텐션 연산
- seq2seq에서 사용되는 닷-프로덕트 어텐션과 다른 어텐션의 차이는 주로 중간 수식이 조금 다른데, 메커니즘 자체는 거의 유사

닷-프로덕트 어텐션(Dot-Product Attention)

1) 어텐션 스코어(Attention Score) 구하기

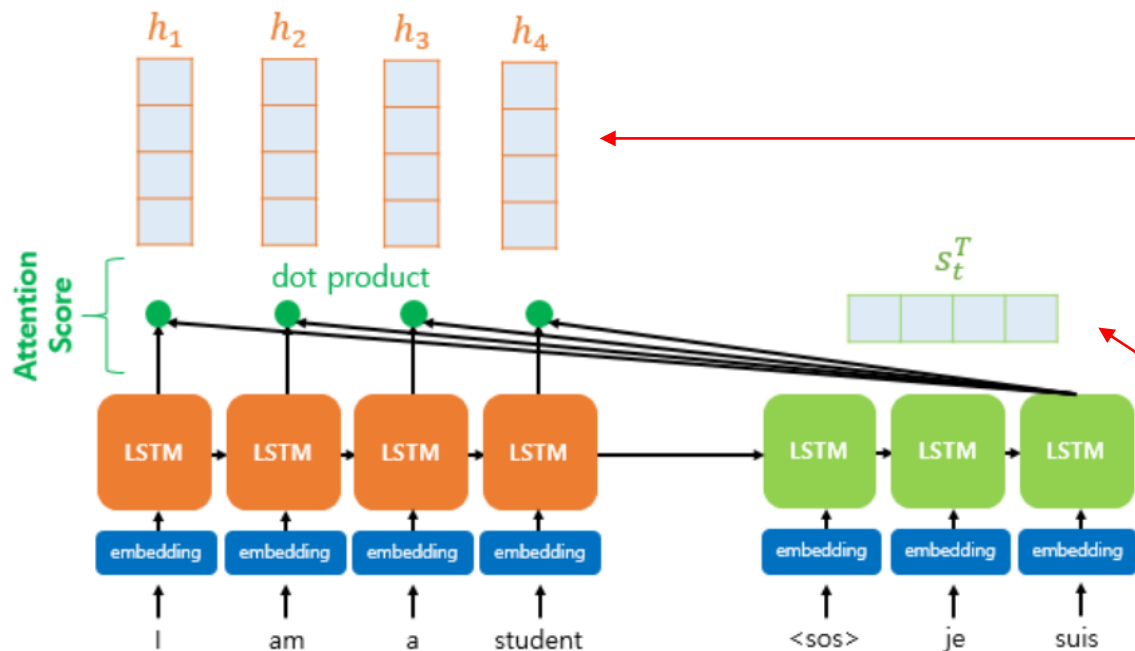
- 의미: 현재 디코더의 시점 t 에서 단어를 예측하기 위해, 인코더의 모든 은닉 상태 각각이 디코더의 현재 시점의 은닉상태 s_t 와 얼마나 유사한지를 판단하는 스코어 값

- h_1, h_2, \dots, h_N : 인코더의 은닉 상태(hidden state)

- s_t (Q: 쿼리) : 디코더의 은닉 상태

- **Attention score**

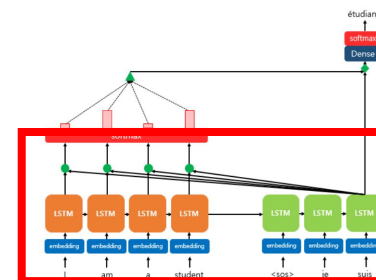
s_t 를 전치(transpose)하고 각 은닉 상태와(h) 내적(dot product)을 수행



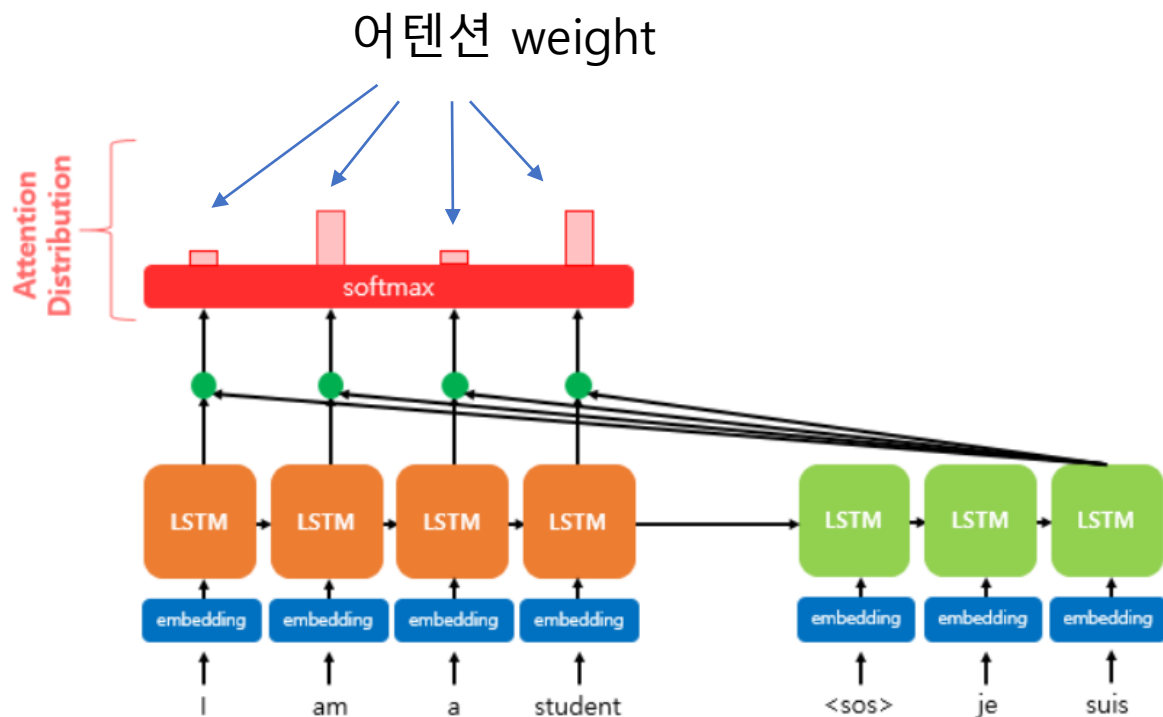
$$s_t^T \times h_i$$

$$\text{score}(s_t, h_i) = s_t^T h_i$$

$$e^t = [s_t^T h_1, \dots, s_t^T h_N]$$



닷-프로덕트 어텐션(Dot-Product Attention)

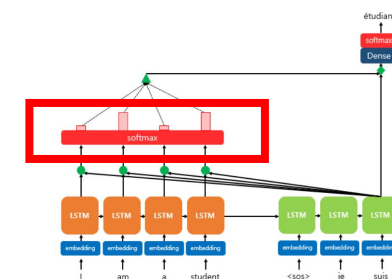


2) 어텐션 분포(Attention Distribution) 구하기

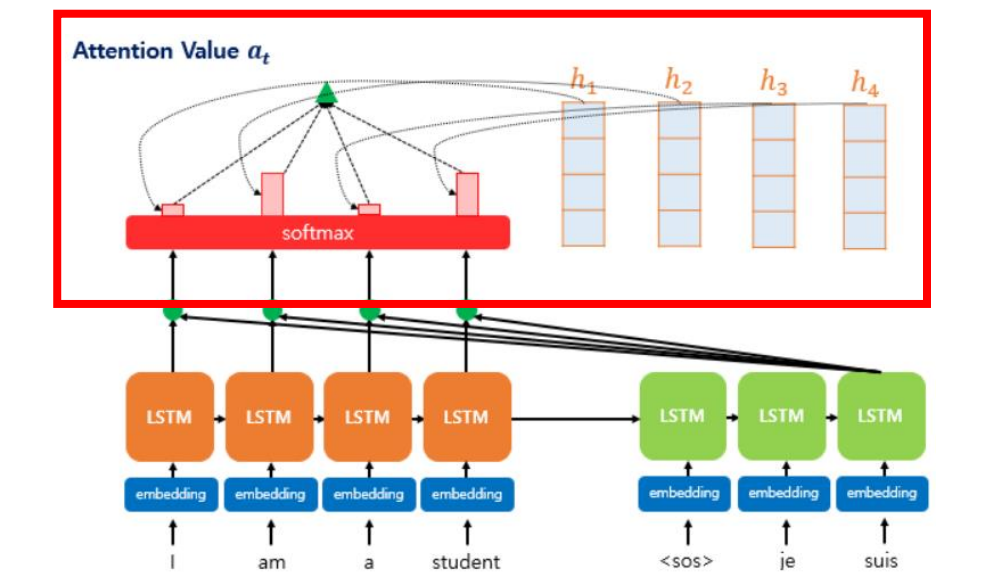
- 의미: Attention score(e^t)에 소프트맥스 함수를 적용하여, 모든 값을 합하면 1이 되는 분포

$$\alpha^t = \text{softmax}(e^t)$$

- 여기서 각각의 값은 attention weight 라고 불림



닷-프로덕트 어텐션(Dot-Product Attention)



3) 어텐션 값 (Attention Value) 구하기

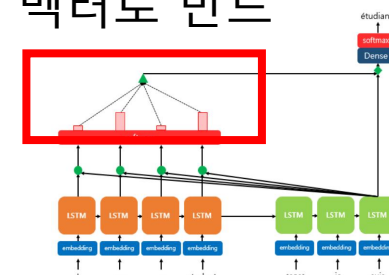
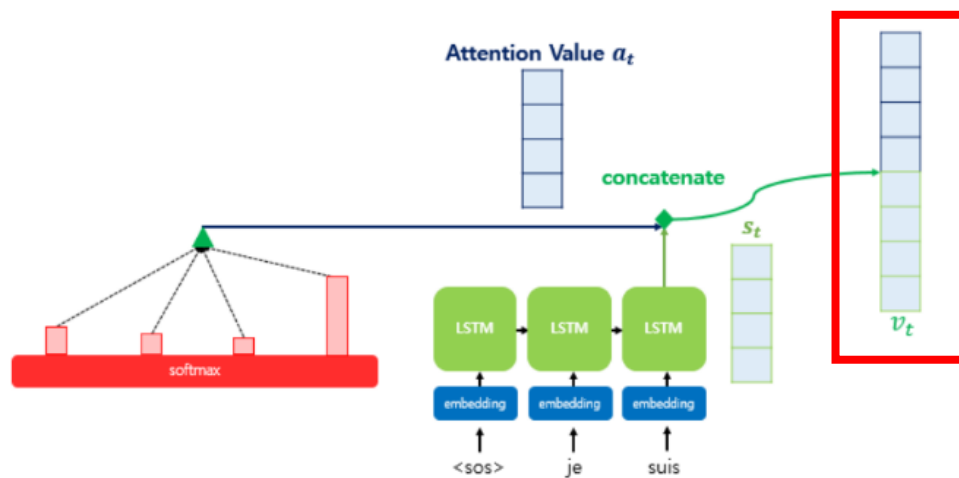
- 의미: 각 인코더의 은닉 상태와 어텐션 가중치 값들을 곱하고, 최종적으로 모두 더한 것

$$a_t = \sum_{i=1}^N \alpha_i^t h_i$$

- 어텐션 값 a_t 은 종종 인코더의 문맥을 포함하고 있다고하여, **컨텍스트 벡터(context vector)**라고도 불림

4) 어텐션 값과 디코더의 t 시점의 은닉 상태를 연결

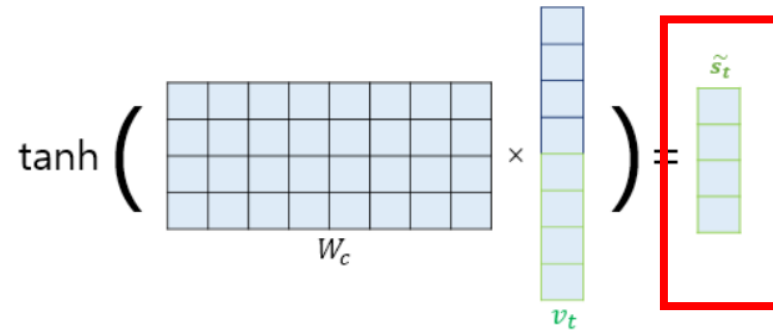
- a_t 를 s_t 와 결합 (concatenate)하여 하나의 벡터로 만드는 작업을 수행하는 것 -> 벡터 v_t 생성



닷-프로덕트 어텐션(Dot-Product Attention)

5) 출력층 연산의 입력 계산

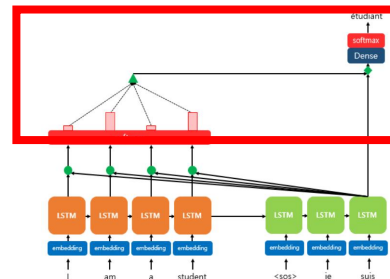
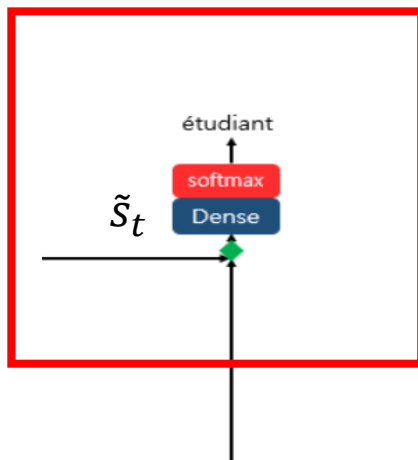
- 가중치 행렬과 곱한 후에 \tanh 함수를 지나도록 하여 출력층 연산을 위한 새로운 벡터인 \tilde{s}_t 계산



$$\tilde{s}_t = \tanh(\mathbf{W}_c[a_t; s_t] + b_c)$$

- \tilde{s}_t 를 출력층의 입력으로 사용하여 예측 벡터를 얻을 수 있음

$$\hat{y}_t = \text{Softmax}(W_y \tilde{s}_t + b_y)$$





19기 정규세션

TOBIG'S 18기 국주현

Unit 03

Transformer

Attention is all you need

2017, 66919회 인용

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion

Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

Unit 03 | Transformer



19기 정규세션
TOBIG'S 18기 국주현

Transformer

- 기본적으로 기계 번역을 위해 제안된 모델
- 인코더 - 디코더 구조

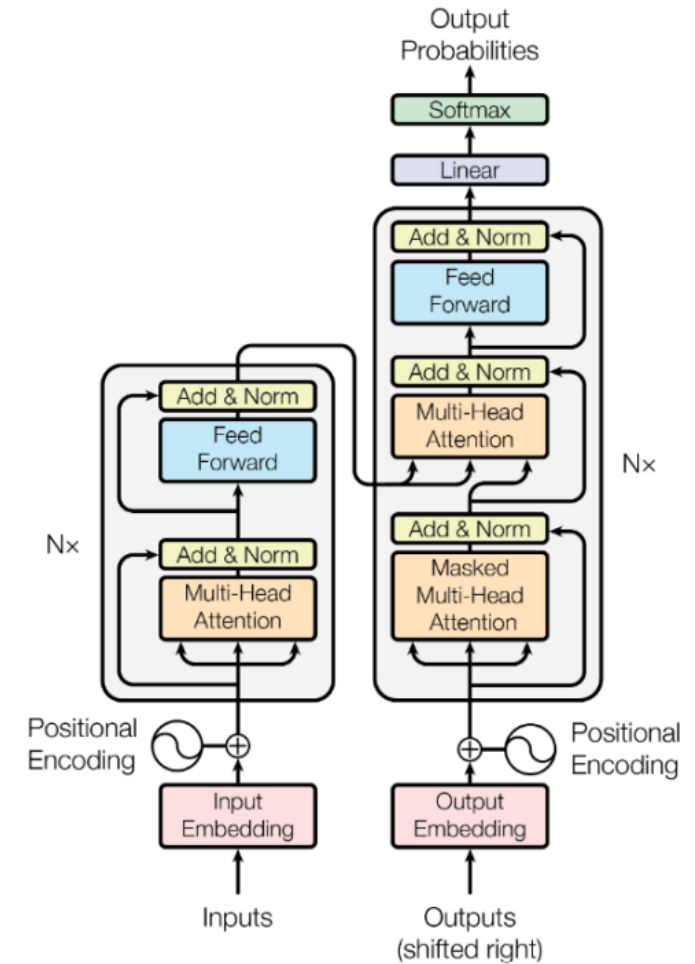
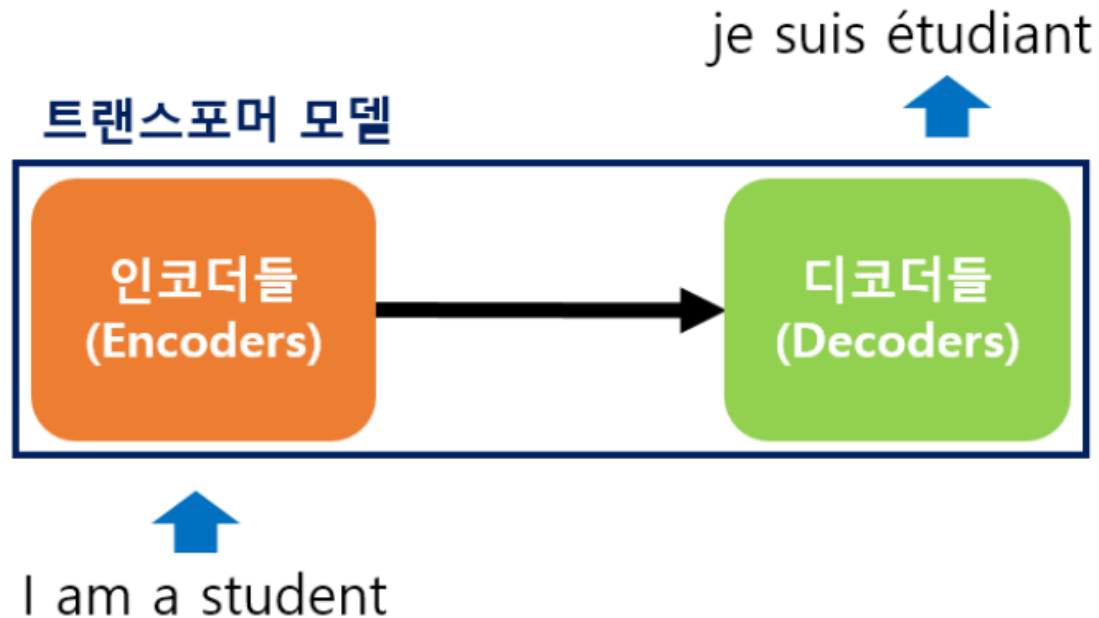


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



19기 정규세션

TOBIG'S 18기 국주현

Input Embedding

- Input에 입력된 데이터를 컴퓨터가 이해할 수 있도록 행렬 값으로 바꾸는 과정
- Word embedding , glove, fasttext 를 사용해서 단어 -> 벡터 변환
- Ex) "I am a student"

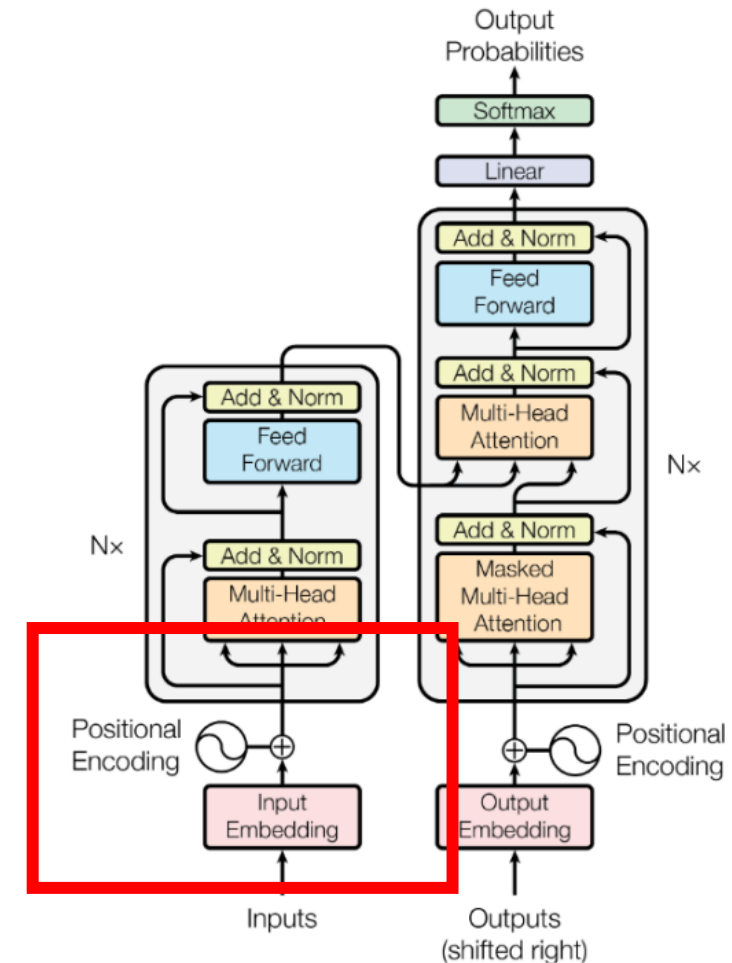
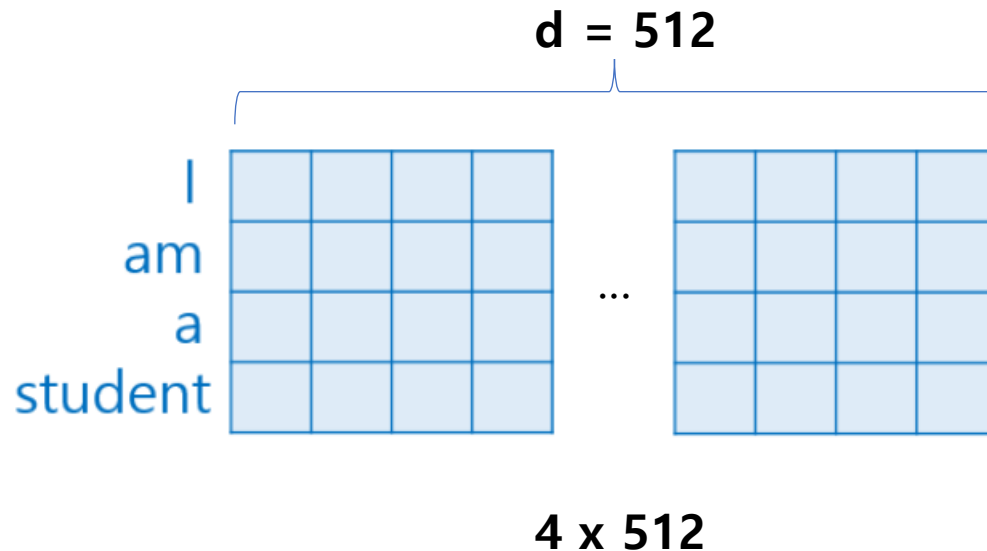


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



19기 정규세션

TOBIG'S 18기 국주현

Input Embedding

- **Positional encoding** 더해주기

Why?) 기존 rnn, lstm 과 다르게 문장을 병렬처리 -> sequential 데이터를 다루기 위해서는 위치 정보 값을 반영 해줘야 함

- **Positional encoding** 은 sine & cosine 함수 사용

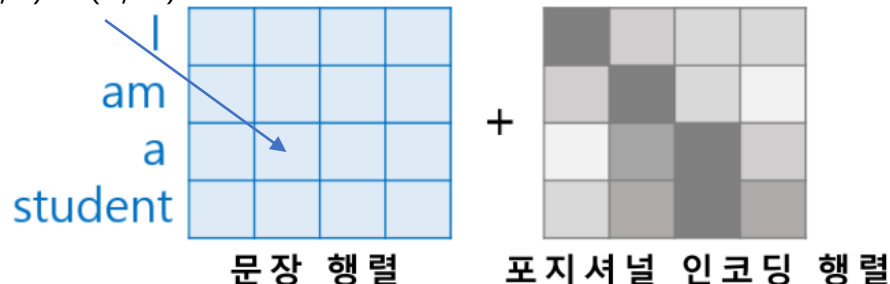
- Why?)

의미정보가 변질되지 않기 위해서는 Positional encoding 값이 너무 크면 안됨. s&c 함수는 -1~1 사이를 반복하는 주기함수

- But, 주기함수들이 같아질 수 있음

=> 방지하기 위해 다양한 주기의 s&c 함수 사용 (512차원이므로 512개 사용)

(pos, i) = (3, 2)



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

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

- Pos는 입력 문장에서의 임베딩 벡터 위치
- i는 임베딩 벡터 내의 차원의 인덱스
- dmodel은 전체 임베딩 벡터 차원(512)
- i 인덱스가 짝수인 경우에는 sine, 홀수인 경우에는 cosine

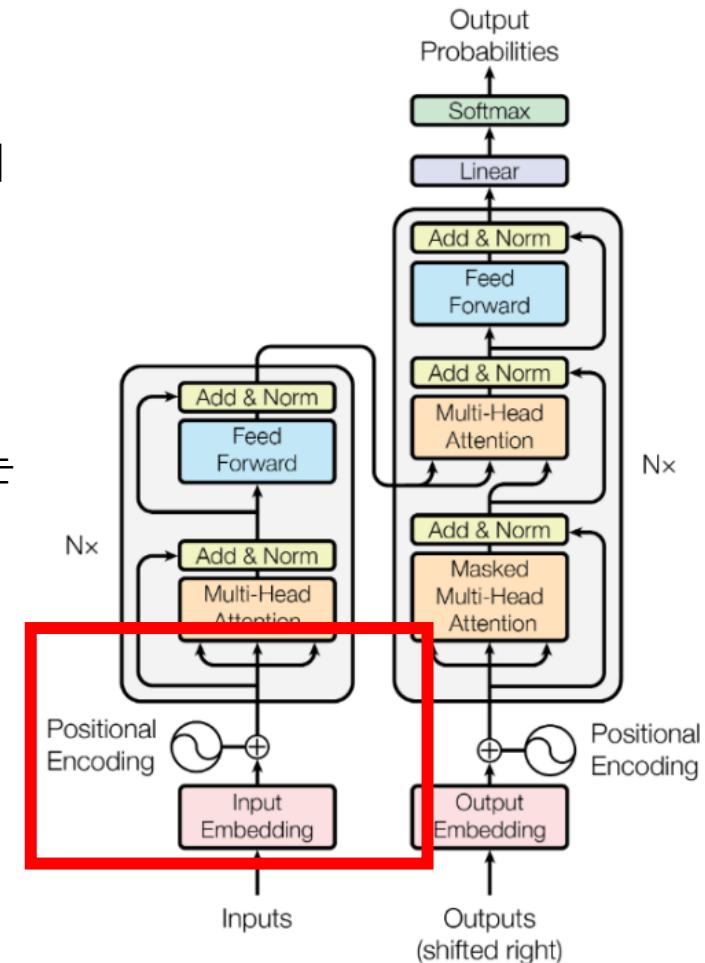


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



19기 정규세션

TOBIG'S 18기 국주현

Encoder

- 여러 개의 인코더 레이어가 중첩되어 사용 (본 논문 : N=6)
- 각 레이어는 2개의 서브 레이어로 구성
=> **멀티 헤드 어텐션, 피드 포워드 네트워크**
- 인코더 디코더의 서브 레이어가 끝날 때마다 Residual connection 과 Norm 두가지 연산을 적용해줌
- Residual connection은 어떠한 연산의 결과를 연산의 입력과 다시 더해주는 것을 의미함. 그 이후 정규화를 진행함
=> $LayerNorm(x + Sublayer(x))$

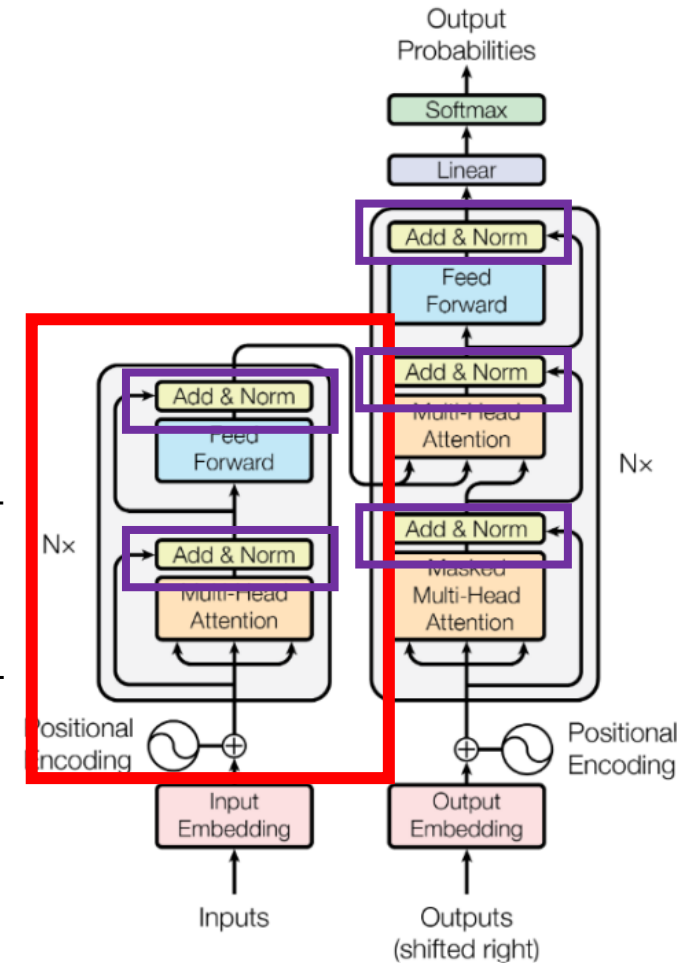


Figure 1: The Transformer - model architecture.

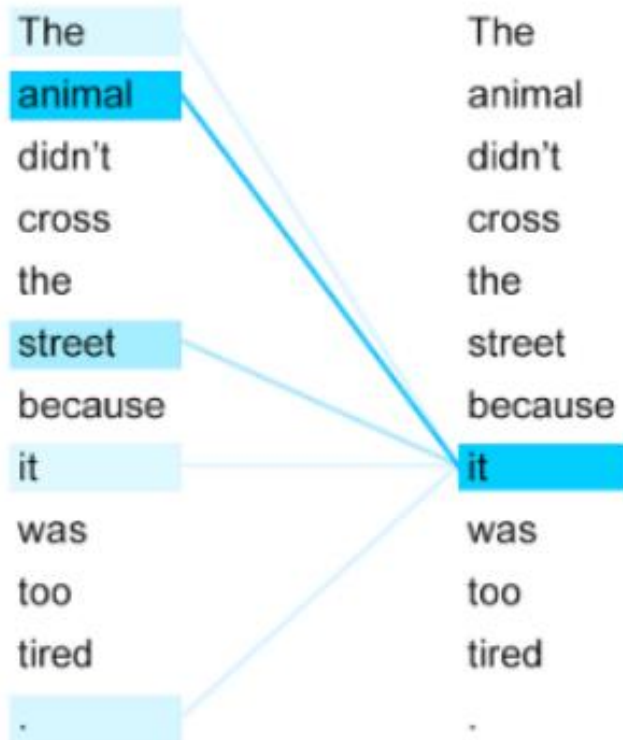
Unit 03 | Transformer



19기 정규세션
TOBIG'S 18기 국주현

Self-attention

- 입력 시퀀스의 특정 단어를 처리할 때, 다른 단어들이 각각 얼마나 영향을 주는 지 계산하는 과정



'It' 이 'animal' 과 연관되었을 확률이 높다는 것을 찾아내기 위함!

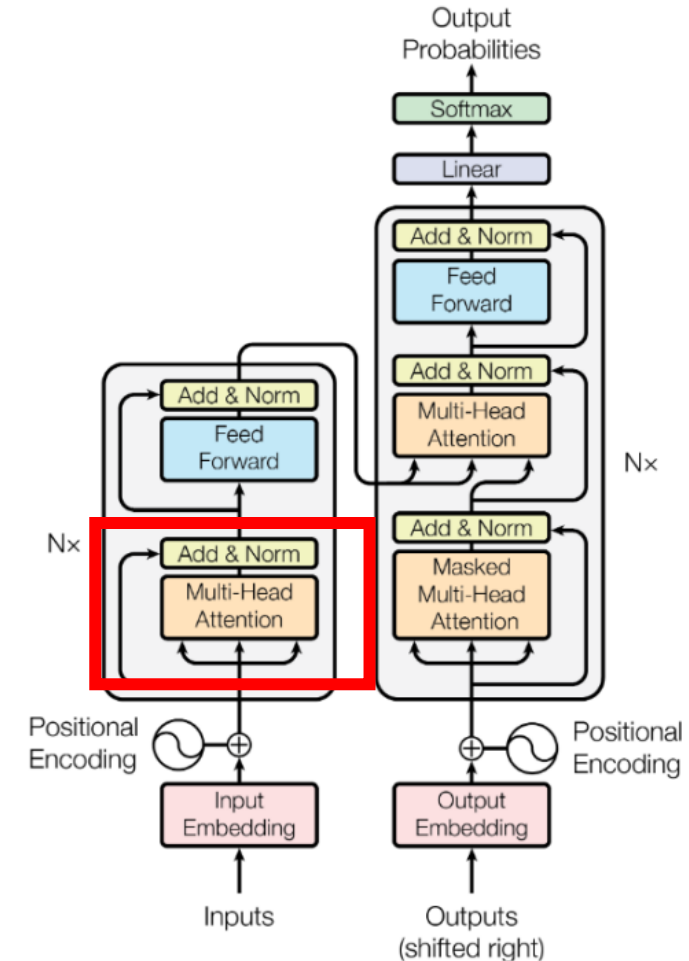


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



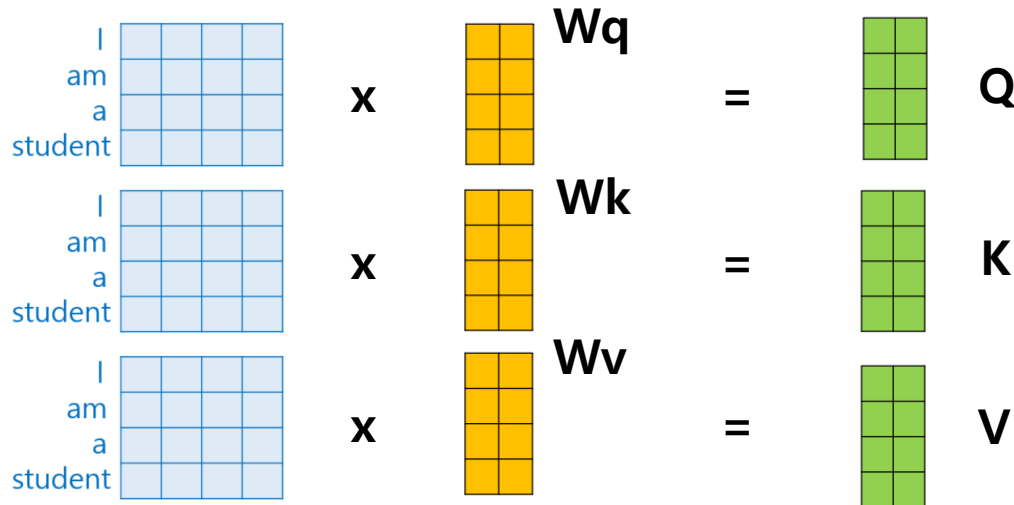
19기 정규세션

TOBIG'S 18기 국주현

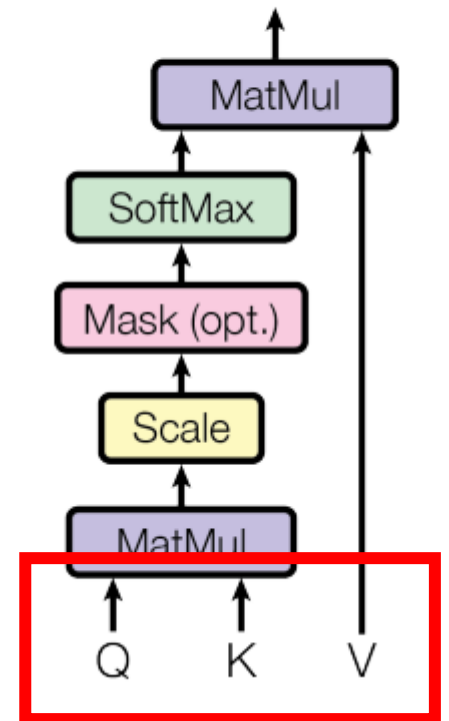
Self-attention

1) 각 인코더의 input vector로부터 3개의 벡터 생성

Input embedding(word embedding + pos) 에다 weight 곱해주기



Scaled Dot-Product Attention



- **Query:** 현재 처리중인 단어에 대한 벡터 (다른 단어와의 연관된 정도를 계산하기 위한 기준이 되는 값)
- **Key:** 단어와의 연관된 정도를 결정하기 위해 query와 비교하는데 사용되는 벡터
- **Value:** 특정 key에 해당하는 입력 시퀀스의 정보(가중치 벡터)

Unit 03 | Transformer



19기 정규세션

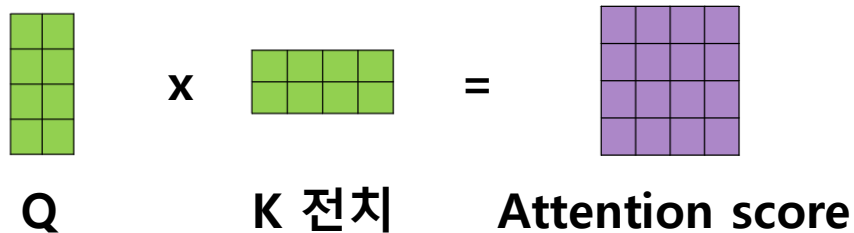
TOBIG'S 18기 국주현

Self-attention

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

2) Attention score 계산

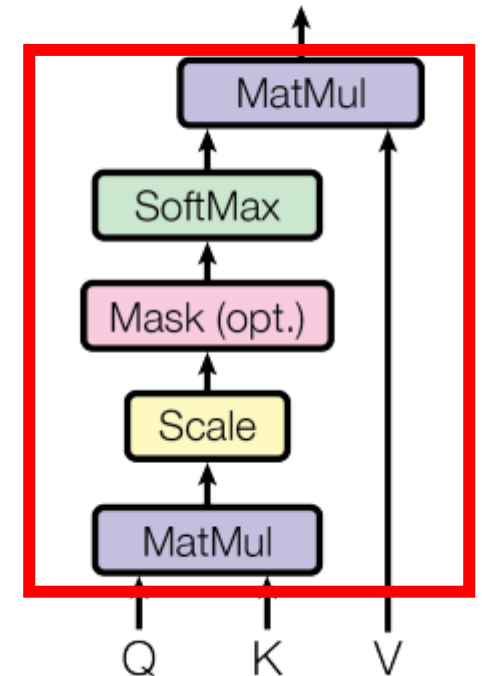
- 생성된 Q 와 K 를 곱해줌



3) Scaling 및 self-attention 계산

- Scaling 진행 (d_k 는 차원을 의미, 그냥 self-attention일 땐 512, multi-head attention에서 헤드 수가 8일 땐 64를 의미)
- 그 값을 softmax 후 value와 곱하면 self-attention 계산 끝

Scaled Dot-Product Attention



Multi-head Self-attention

- 트랜스포머는 한 번의 어텐션을 하는 것보다 어텐션을 병렬로 여러 번 사용하는 것이 더 효과적이기 때문에 여러 헤드로 나눠서 병렬로 계산

Why?) 병렬로 하면 다른 시각으로 정보들을 수집 가능

Ex)

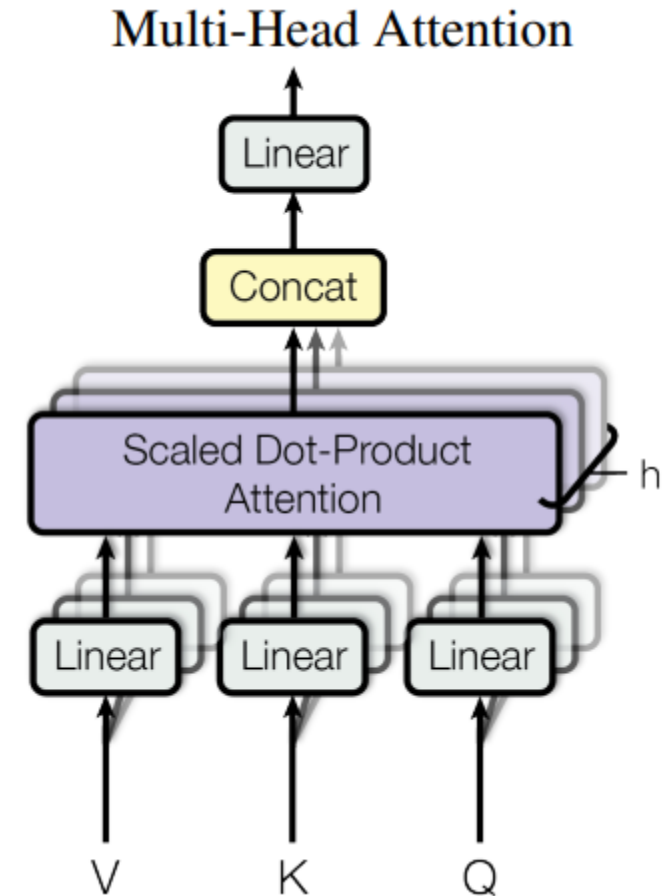
The **animal** didn't cross the street because **it** was too tired.

어떤 헤드에서는 it과 animal 의 연관성을 높게 볼 것이고,

The animal didn't cross the **street** because **it** was too tired.

또다른 헤드에서는 it과 street 의 연관성을 더 높게 볼 것이라 병렬로 어텐션을 수행하면 여러 시각에서 접근 가능함!

- 본 논문에서는 head 수를 8로 설정하고, Self-attention 을 병렬로 계산한다음 concat(연결) 진행!



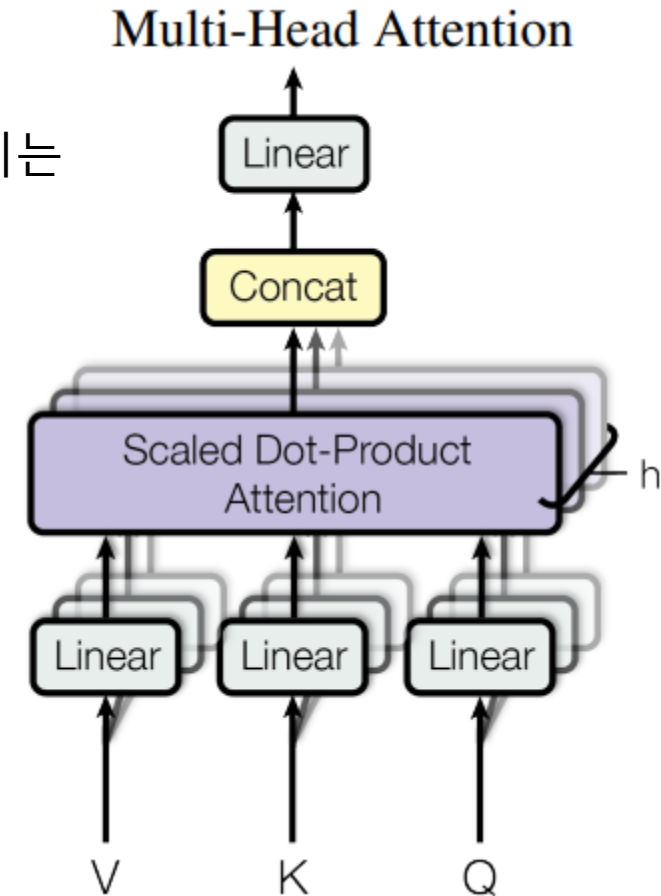
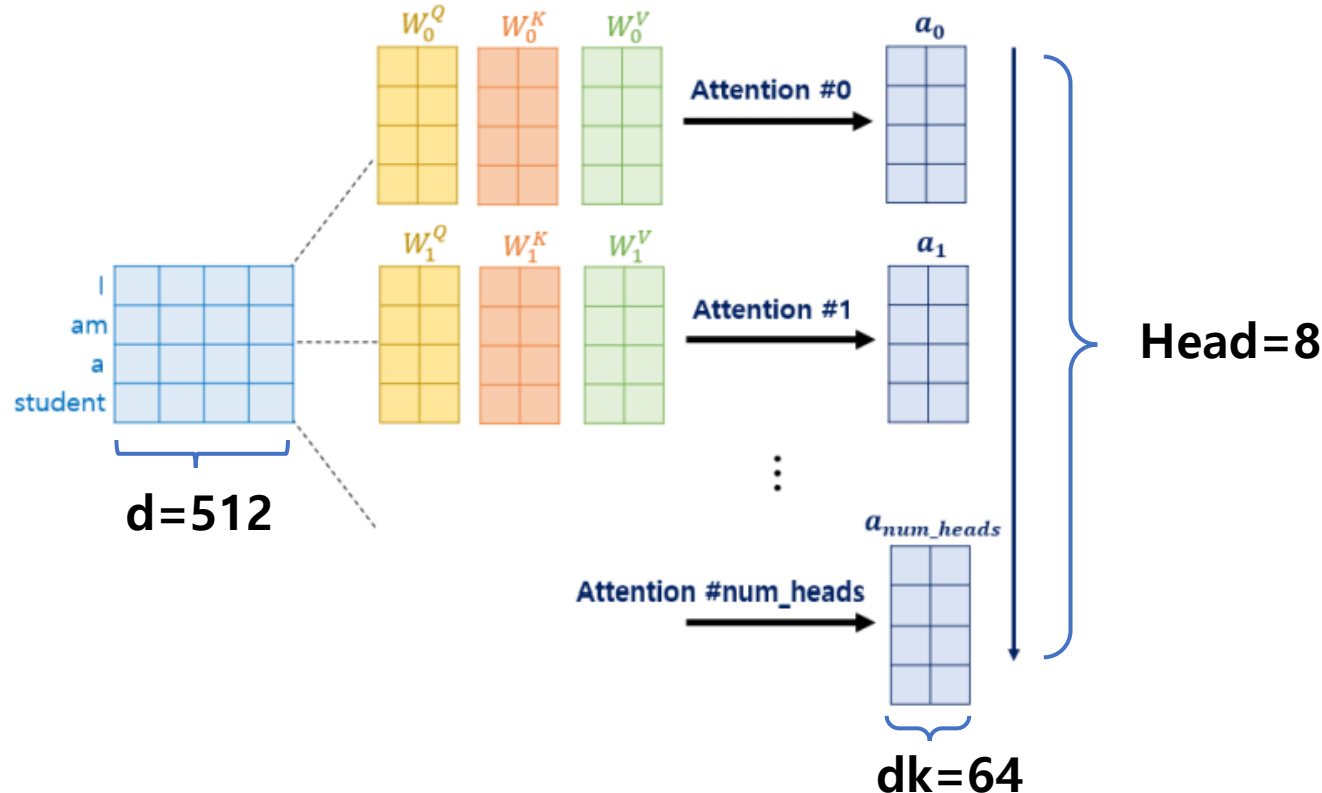
Unit 03 | Transformer



19기 정규세션
TOBIG'S 18기 국주현

Multi-head Self-attention

- 본 논문에서는 임베딩 벡터 차원을 512로 두고 실험 진행
- 병렬로 처리해야하기에, multi-head self-attention 진행 시 하나의 head에서 이뤄지는 attention 연산의 차원은 512를 head 개수인 8 만큼 나눈 64 가 됨



Unit 03 | Transformer



19기 정규세션
TOBIG'S 18기 국주현

Position-wise Feed-Forward Networks

- 단순 피드포워드 신경망을 의미

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

- linear transformation 2개로 구성, Max() 부분은 **ReLU activation**을 의미
- inner-layer(은닉층)의 차원 $d_{ff}=2048$
- 입력 값과 출력 값은 512로 동일한 차원

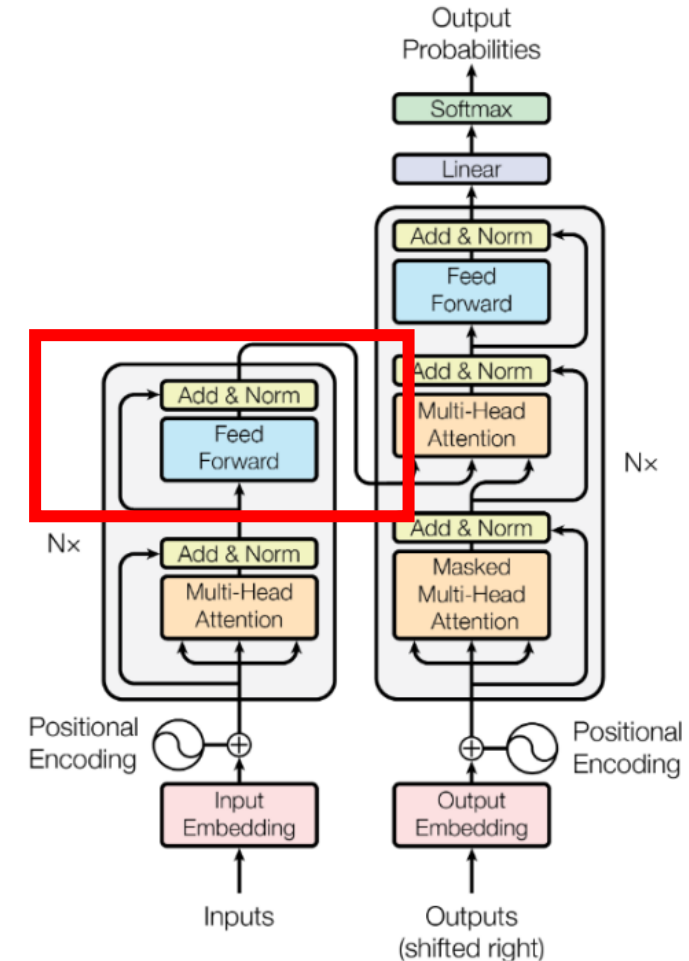
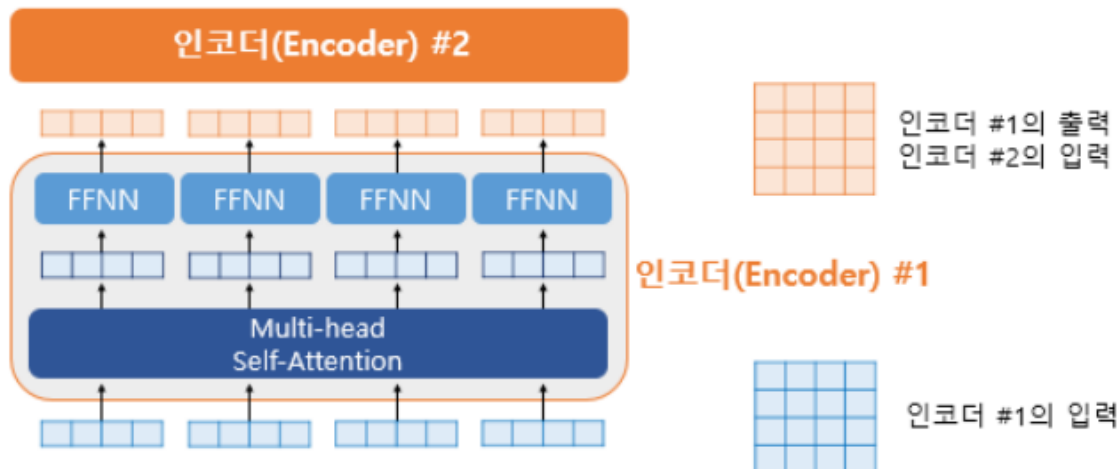


Figure 1: The Transformer - model architecture.

19기 정규세션
TOBIG'S 18기 국주현

- 여러 개의 디코더 레이어가 중첩되어 사용 (본 논문 : N=6)
- 인코더와 다르게, 두가지의 self-attention이 존재
- 인코더와 마찬가지로 positional encoding을 더해주고, 서브 레이어 이후 Residual connection과 layer normalization 과정 수행
- 디코더의 입력에는 시작 토큰 <sos>과 종료 토큰 <eos> 이 존재

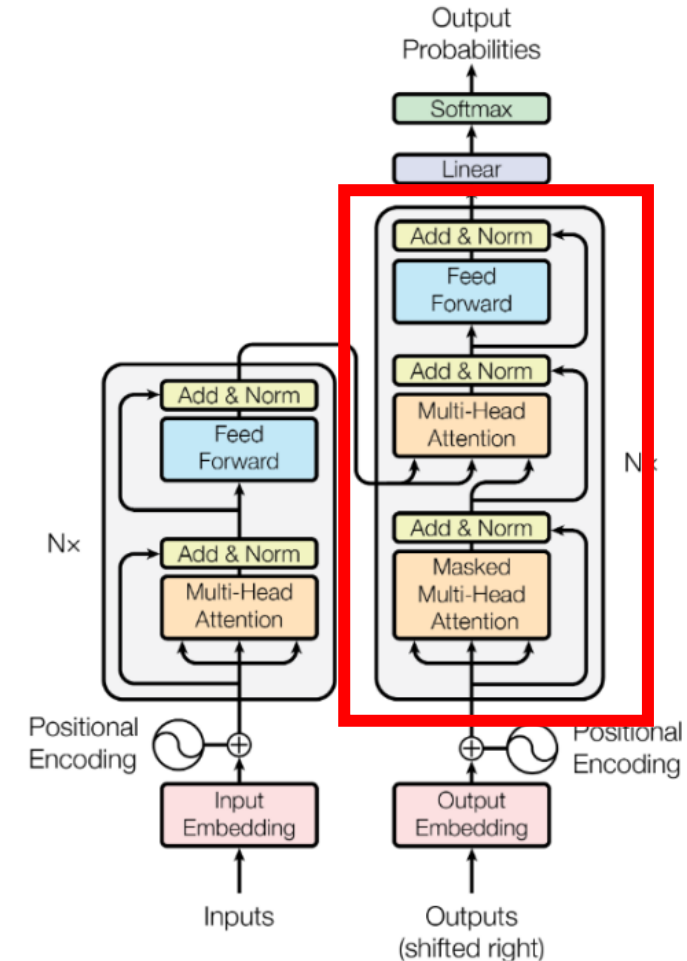
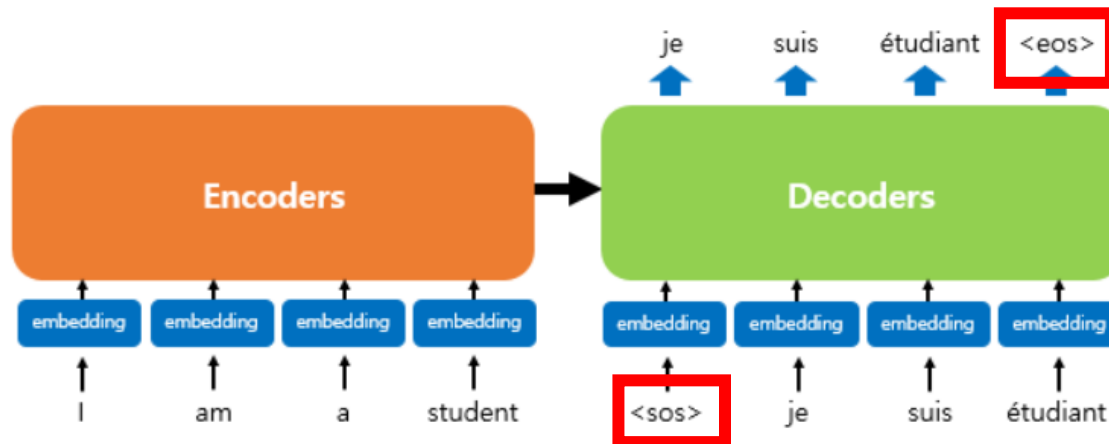


Figure 1: The Transformer - model architecture.

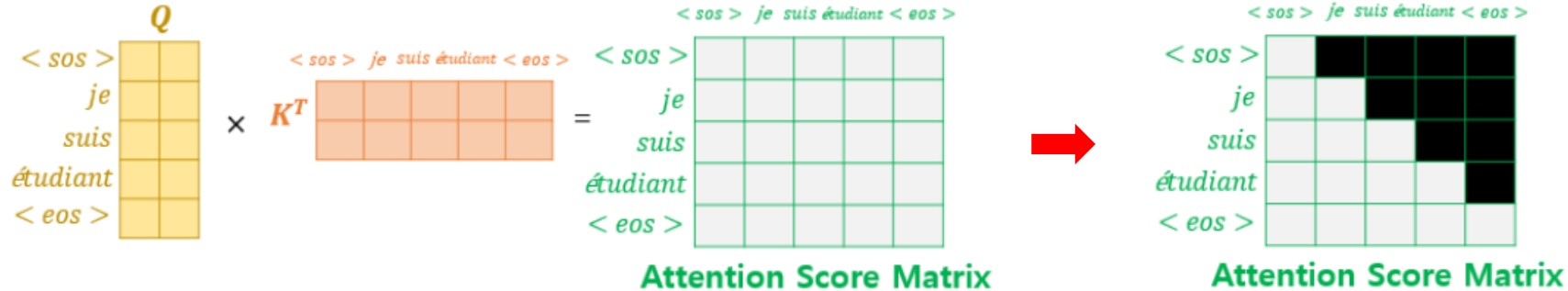
Unit 03 | Transformer



19기 정규세션
TOBIG'S 18기 국주현

Masked multi-head attention

- 디코더의 Masked multi-head attention 은 근본적으로 self-attention과 동일
- 차이점은 attention score matrix 에 직각 삼각형의 마스킹을 해줌!



마스킹을 하는 이유는?

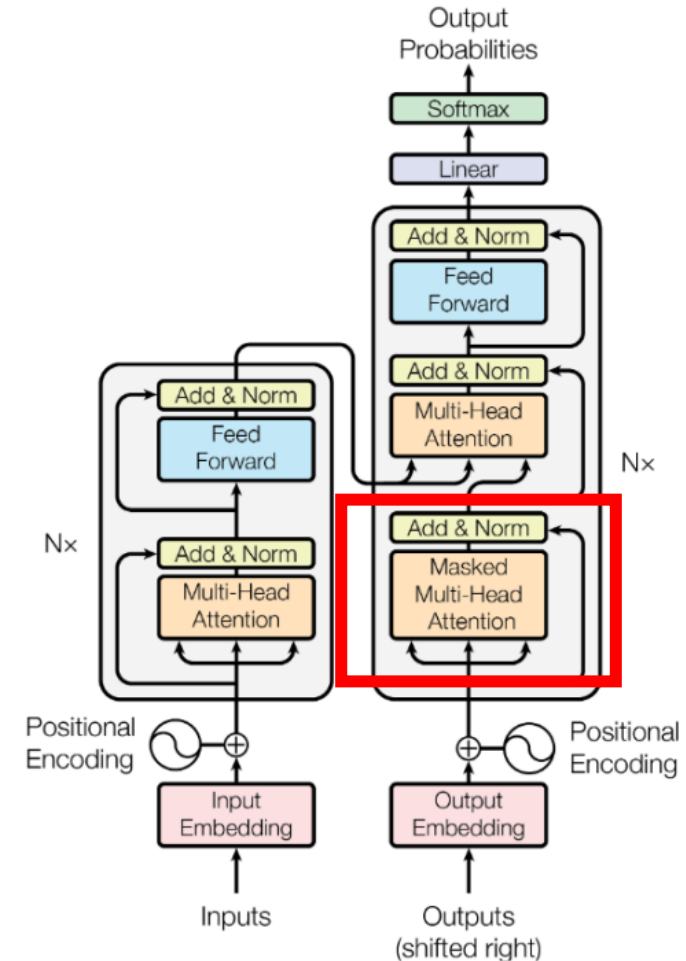


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



19기 정규세션
TOBIG'S 18기 국주현

Masked multi-head attention

- 디코더는 훈련 과정에서 실제 예측할 문장 행렬을 입력으로 넣어 줌.
- 하지만, self-attention을 할 때, 미래 시점의 단어들을 참고하면 안됨!

=> 대처 방안으로 마스킹

How?) attention score에 softmax를 취할 때, 미래 시점 단어들은 $\text{softmax}(-\infty)$, 즉 0이 되게끔 적용함

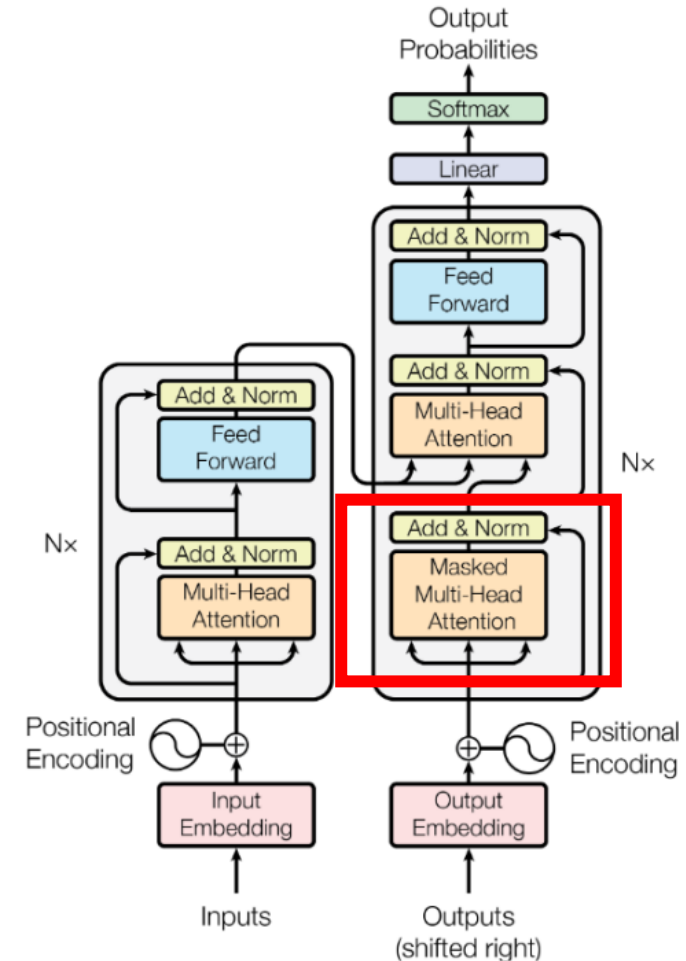
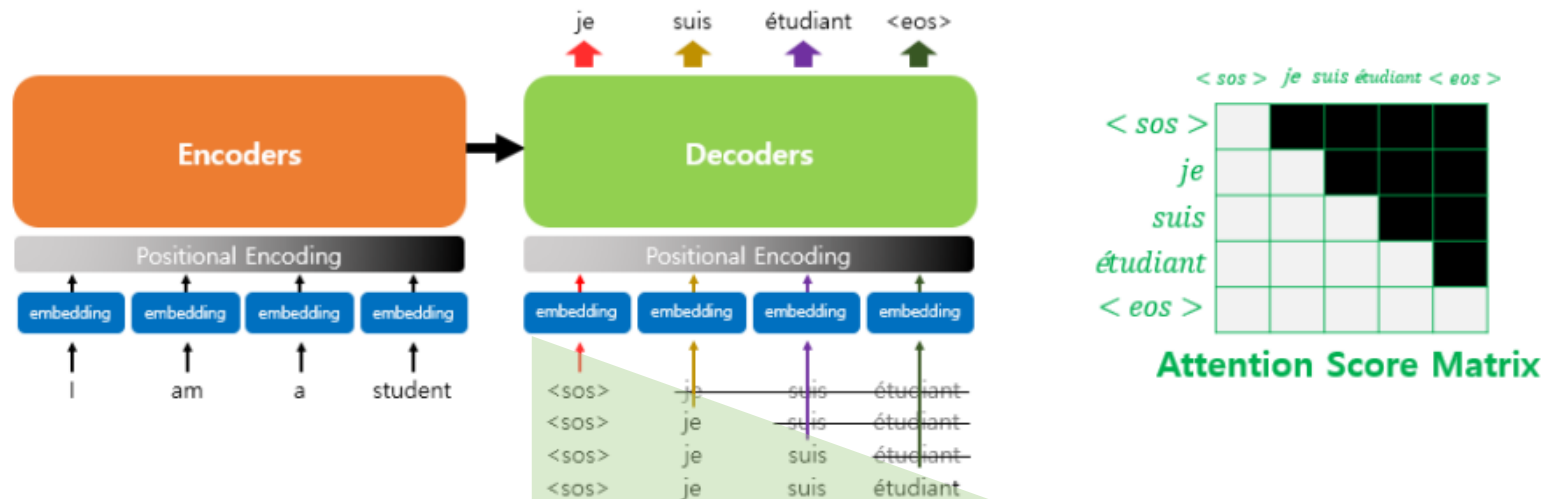


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



19기 정규세션
TOBIG'S 18기 국주현

Encoder-Decoder attention

- 인코더 디코더 어텐션은 Q와 K, V 가 다름
- Q는 디코더의 첫번째 서브 레이어 결과 행렬, K와 V 는 인코더의 아웃풋 행렬
- 인코더의 정보가 디코더로 넘어가는 과정
- 출력 단어(번역)를 만들기 위해 소스 문장(input data)에서 어떤 정보에 초점을 맞출지 학습하는 과정

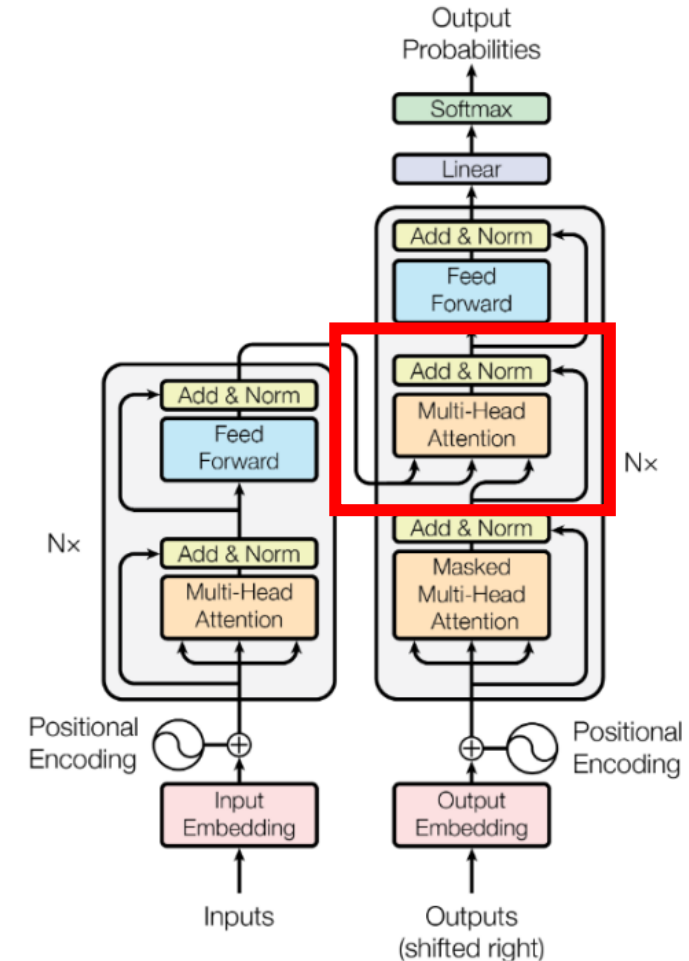
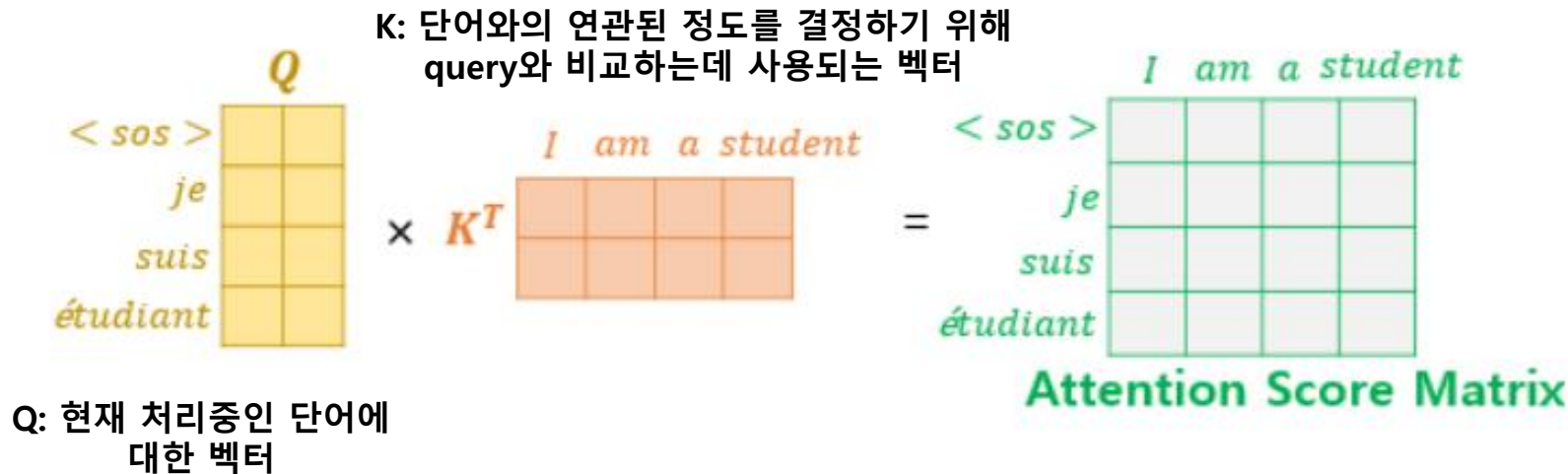


Figure 1: The Transformer - model architecture.



19기 정규세션

TOBIG'S 18기 국주현

Unit 04

과제

과제 1: 여러분의 연구가 궁금합니다!

- 컨퍼런스 프로젝트 시작이 얼마 남지 않았습니다.
- 본인의 연구 분야 또는 앞으로 해보고 싶은 연구 분야를 선정한다음
- 해당 분야에 대해 연구 계획서를 작성해주세요(워드 기준 11pt, 한페이지 이상)

과제 2: 트랜스포머 기반 후속 논문(ex: bert, vit) 1개 읽기 및 정리해주세요!

- 후속 연구 관련 논문 1개를 선정한 다음
- 정리해주세요(Pdf 5장 이상으로 정리하기, 노션 및 개인 블로그 링크를 올려도 됨)

과제 1 or 과제 2 둘 중 하나 선택해서 하기!

Reference



19기 정규세션
TOBIG'S 18기 국주현

[\[RNN\] RNN을 알아봅시다\[밑바닥부터 시작하는 딥러닝2 참고\]-I am yumida :: AIBLOG \(tistory.com\)](#)

[17기 nlp 정규세션 자료](#)

[딥 러닝의 가장 기본적인 시퀀스 모델 RNN \(Recurrent Neural Network\) \(tistory.com\)](#)

[08-01 순환 신경망\(Recurrent Neural Network, RNN\) - 딥 러닝을 이용한 자연어 처리 입문 \(wikidocs.net\)](#)

[Long Short-Term Memory \(LSTM\) 이해하기 :: 개발새발로그 \(tistory.com\)](#)

[15-01 어텐션 메커니즘 \(Attention Mechanism\) - 딥 러닝을 이용한 자연어 처리 입문 \(wikidocs.net\)](#)

[트랜스포머 transformer positional encoding \(blossominkyung.com\)](#)

[이영아의 트랜스포머](#)

[16-01 트랜스포머\(Transformer\) - 딥 러닝을 이용한 자연어 처리 입문 \(wikidocs.net\)](#)

