



NLP Advanced

20기 정규세션

TOBIG'S 19기 임승섭

Contents



20기 정규세션

TOBIG'S 19기 임승섭

Unit 01 | Seq2Seq

Unit 02 | Attention mechanism

Unit 03 | Transformer

Unit 04 | 과제



20기 정규세션

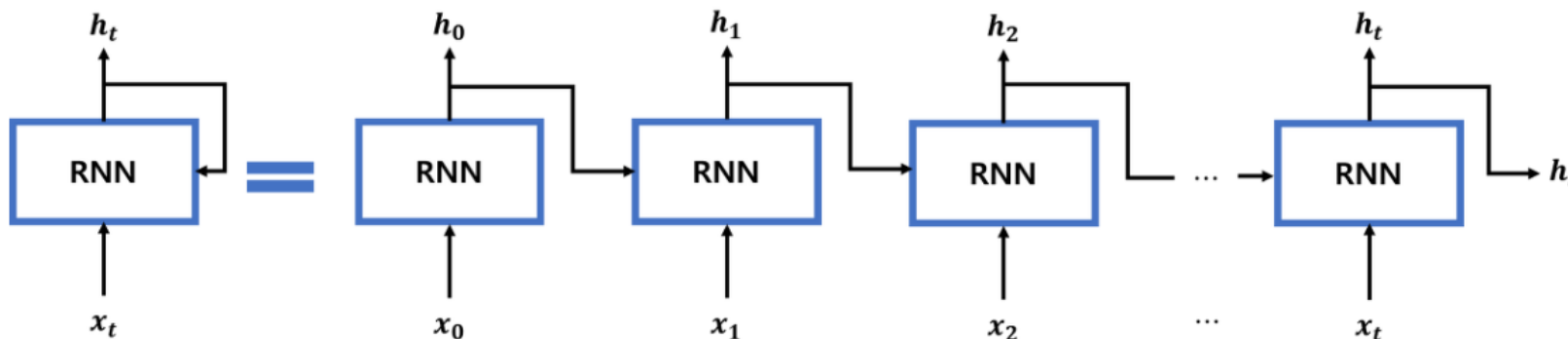
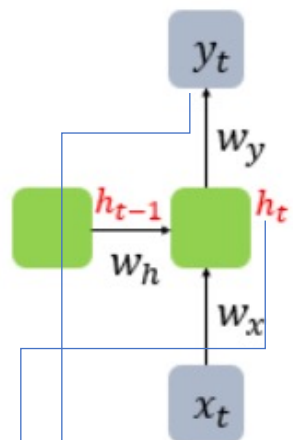
TOBIG'S 19기 임승섭

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 받음



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



Sequence-to-Sequence

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

NeurIPS 2014 (citation 23344회)



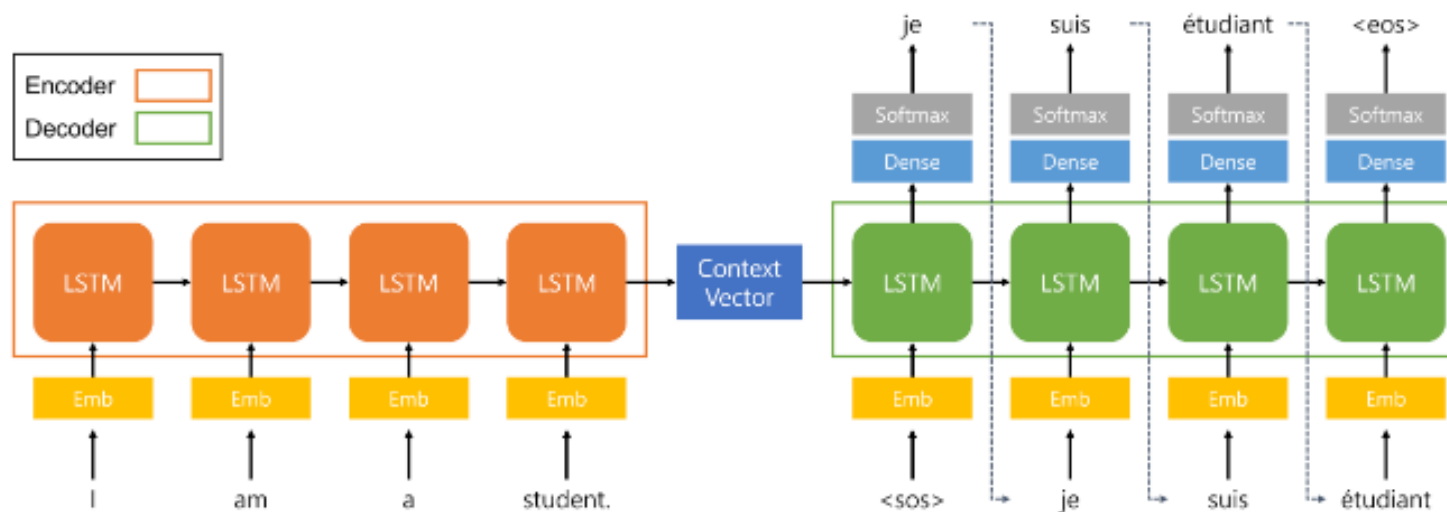
Ilya Sutskever
Co-Founder and Chief Scientist of OpenAI
Verified email at openai.com - [Homepage](#)
[Machine Learning](#) [Neural Networks](#) [Artificial Intelligence](#) [Deep Learning](#)

[FOLLOW](#)

TITLE	CITED BY	YEAR
Imagenet classification with deep convolutional neural networks A Krizhevsky, I Sutskever, GE Hinton Advances in neural information processing systems 25	140847 [*]	2012
Tensorflow: Large-scale machine learning on heterogeneous distributed systems M Abadi, A Agarwal, P Barham, E Brevdo, Z Chen, C Citro, GS Corrado, ... arXiv preprint arXiv:1603.04467	48049 [*]	2016
Dropout: a simple way to prevent neural networks from overfitting N Srivastava, G Hinton, A Krizhevsky, I Sutskever, R Salakhutdinov The journal of machine learning research 15 (1), 1929-1958	45538	2014
Distributed representations of words and phrases and their compositionality T Mikolov, I Sutskever, K Chen, GS Corrado, J Dean Advances in neural information processing systems 26	40342	2013
Sequence to sequence learning with neural networks I Sutskever, O Vinyals, QV Le Advances in neural information processing systems 27	23344	2014

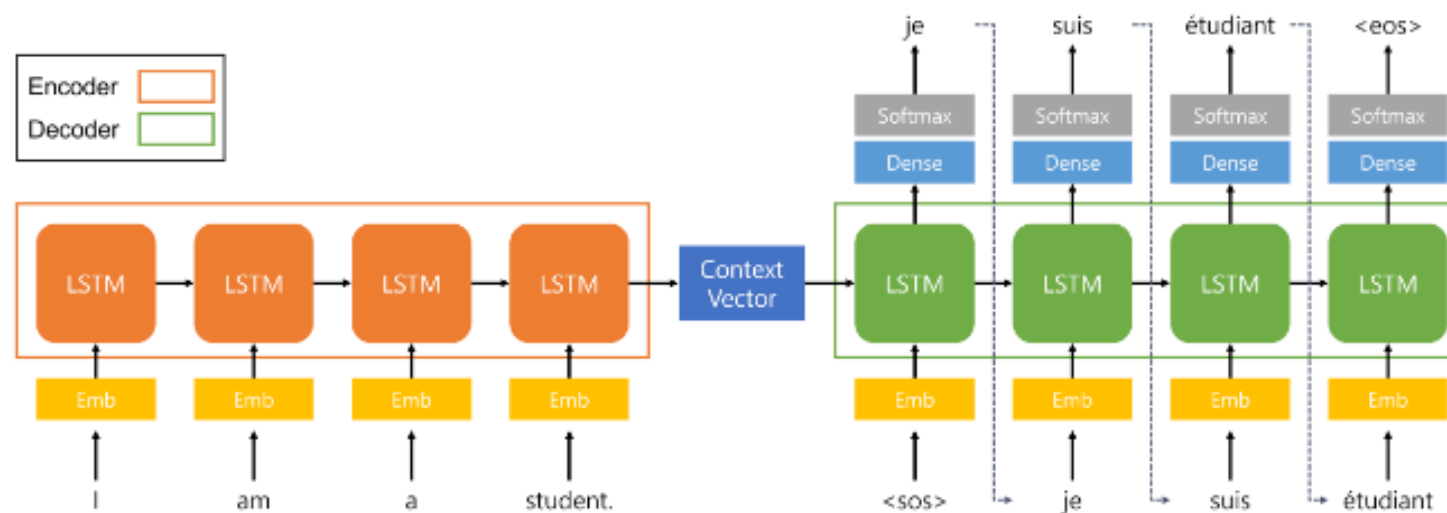
Seq2seq 모델의 동작 방법

- Seq2seq 모델은 번역, 요약과 같이 시퀀스(sequence)를 입력받아 시퀀스를 출력하는 task를 위해 고안된 RNN 기반 모델
- Seq2seq 모델은 '시퀀스를 받아들이는 부분'과 '시퀀스를 출력하는 부분'을 분리한 것이 특징
- 시퀀스를 받아들이는 부분을 인코더(Encoder), 시퀀스를 출력하는 부분을 디코더(decoder)라고 함
- 인코더는 입력 시퀀스를 받아들여 컨텍스트 벡터(context vector)로 불리는, 고정된 크기의 벡터로 변환
- 디코더는 인코더가 생성한 컨텍스트 벡터를 받아 출력 시퀀스를 출력



Seq2seq 모델이 동작(inference)하는 순서

1. 인코더의 은닉 상태를 적절한 값으로 초기화
2. 매 시점(time step)마다 단어의 임베딩이 입력되면 인코더는 이를 이용해 은닉 상태를 업데이트
3. 입력 시퀀스의 끝까지 이 과정을 반복하면 인코더의 최종 은닉 상태는 입력 시퀀스의 정보를 압축 요약한 정보를 담고 있게 됨
4. 디코더는 전달받은 컨텍스트 벡터로 자신의 은닉 상태를 초기화
5. 매 시점 자신(Decoder)이 바로 직전 시점에 출력했던 단어를 입력으로 받아, 은닉 상태를 업데이트하고, 이를 이용해 다음 단어를 예측



Seq2seq 모델의 학습 방법 - 교사 강요(teacher forcing)

- Seq2seq 모델을 학습(training)시킬 때, 이전 시점의 디코더 출력 단어를 다시 디코더의 입력값으로 사용하는 방식으로는 학습이 잘 되지 않음
- 모델 학습 시에는 디코더의 입력값으로 이전 시점의 디코더 출력 단어가 아닌 실제 정답 단어를 입력해줘야 함
- 이러한 방식을 teacher forcing이라고 함

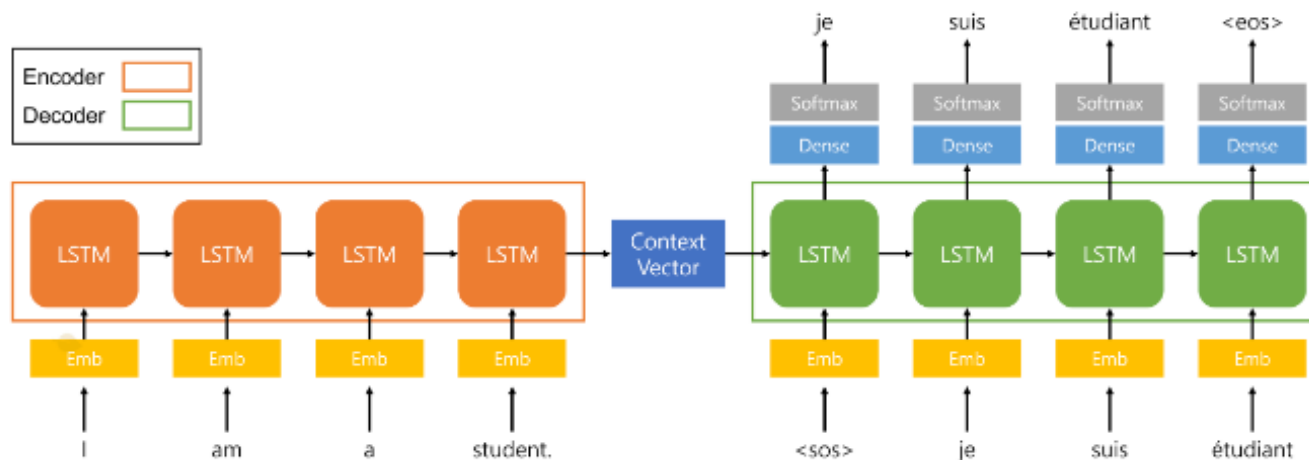


Fig.02 Seq2Seq - Teacher Forcing

Seq2Seq 모델의 학습은 교사 강요(teacher forcing) 방식으로 진행해야 한다. 즉 "<sos>je suis étudiant"가 입력되었을 때 "je suis étudiant<eos>"가 출력되어야 한다는 것을 디코더에게 직접 알려줘야 한다.

Seq2seq 모델의 한계

- 입력 시퀀스의 모든 정보를 하나의 고정된 크기의 벡터(컨텍스트 벡터)에 다 압축 요약하려 하다 보니 정보의 손실이 생길 수밖에 없음. 특히 시퀀스의 길이가 길다면 정보의 손실이 더 커짐
- RNN 구조로 만들어진 모델이다 보니, 필연적으로 gradient vanishing/exploding 현상이 발생



20기 정규세션

TOBIG'S 19기 임승섭

Unit 02

Attention mechanism

Unit 02 | Attention mechanism



20기 정규세션

TOBIG'S 19기 임승섭

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho Yoshua Bengio*
Université de Montréal

Unit 02 | Attention mechanism





20기 정규세션

TOBIG'S 19기 임승섭

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

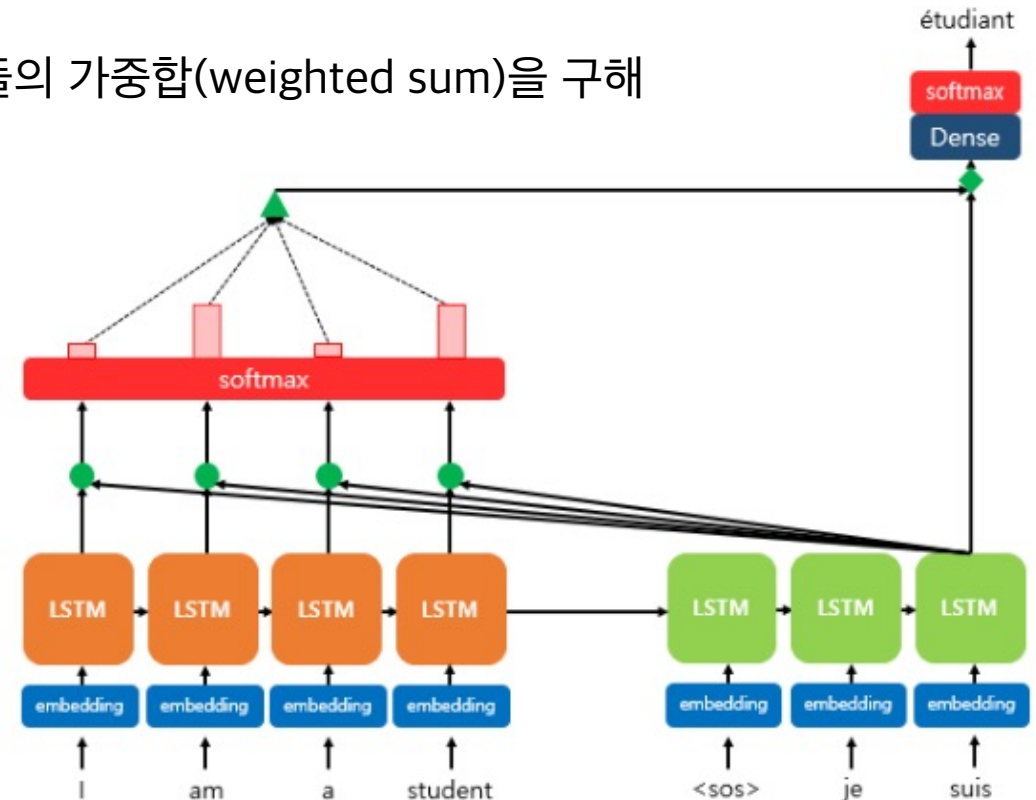
Attention: 풀고자 하는 Task에 핵심이 되는 정보를 찾아서 집중!

어텐션 메커니즘 (Attention Mechanism)

- 어텐션 메커니즘은 Seq2seq 모델의 문제점을 개선하기 위해 제안됨
- 어텐션 메커니즘의 아이디어는 고정된 크기의 벡터(컨텍스트 벡터) 하나에 입력 시퀀스의 모든 정보를 다 담아야 한다는 인코더의 부담을 덜어주기 위함
- 디코더에서 다음 단어 예측을 위해 **인코더의 매 시점 은닉 상태**들을 모두 사용하자 !
- 구체적으로, 어텐션 메커니즘은 다음을 가정
 - “단어 x 를 출력하기 직전의 디코더 은닉 상태는, 인코더가 입력 시퀀스에서 x 와 연관이 깊은 단어를 읽은 직후의 인코더 은닉 상태와 유사할 것이다.”

어텐션 메커니즘 (Attention Mechanism)

- Seq2seq + Attention 모델에서의 디코더는 다음과 같은 순서로 다음 단어를 예측
 1. 어느 시점의 인코더 은닉 상태에 조금 더 '집중'해야 하는지 찾기 위해, 현재 디코더의 은닉 상태와 매 시점 인코더의 은닉 상태들 간 '유사도'를 계산
 2. 이 유사도를 확률의 형태로 바꾸고, 그 값에 따라 인코더 은닉 상태들의 가중합(weighted sum)을 구해 '보정된 컨텍스트 벡터'를 구함
 3. '보정된 컨텍스트 벡터'를 이용해 다음 단어를 예측
- gradient vanishing/exploding 현상을 줄일 수 있음



어텐션 함수(Attention Function)

- 어텐션 함수는 주어진 쿼리(Query)에 대해서 모든 키(Key)와의 유사도를 각각 구함
- 구해낸 이 유사도를 키와 매핑되어 있는 각각의 값(Value)에 반영
- 유사도가 반영된 값(Value)을 모두 더해서 리턴 (Attention Value)

Q = Query : t 시점의 디코더 셀에서의 은닉 상태

K = Keys : 모든 시점의 인코더 셀의 은닉 상태들

V = Values : 모든 시점의 인코더 셀의 은닉 상태들

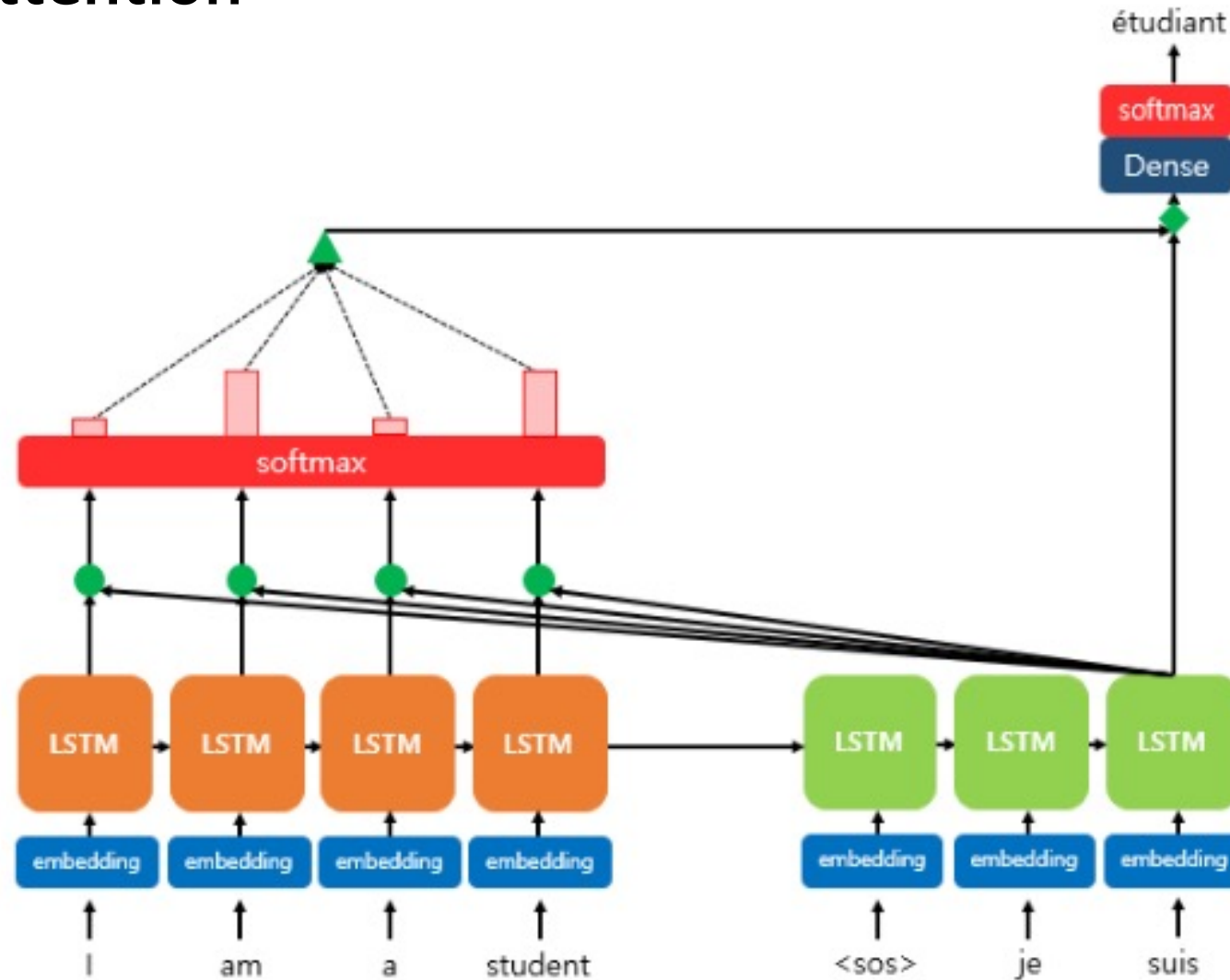
Unit 02 | Attention mechanism



20기 정규세션

TOBIG'S 19기 임승섭

Dot-Product Attention



Unit 02 | Attention mechanism

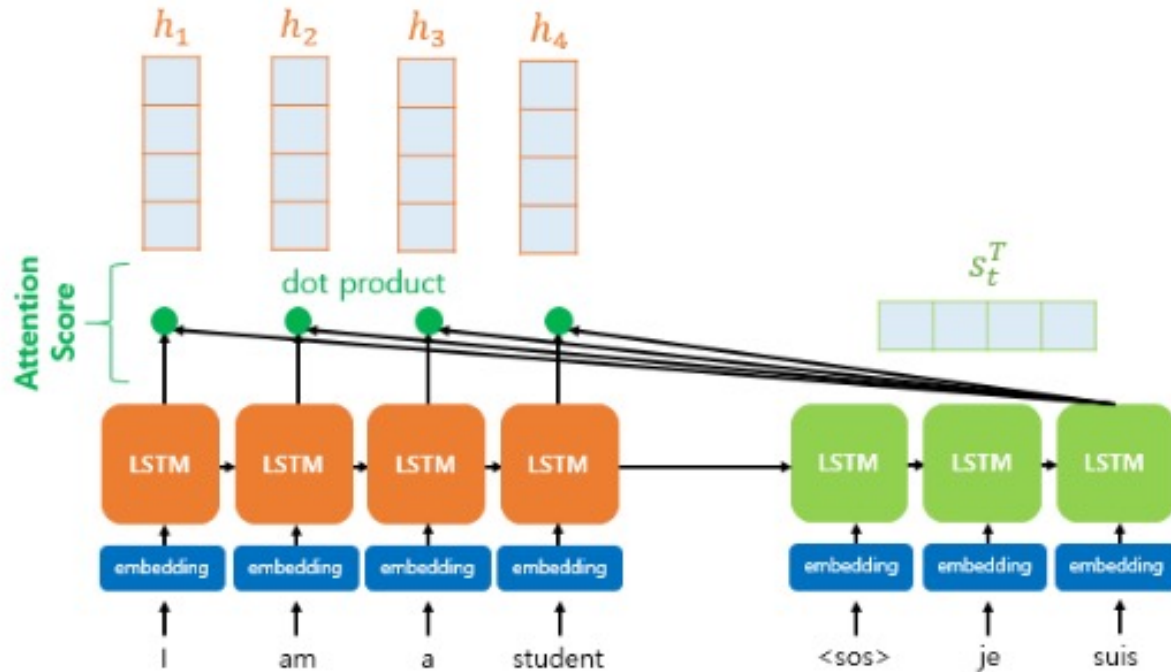


20기 정규세션

TOBIG'S 19기 임승섭

Dot-Product Attention

1) 어텐션 스코어(Attention Score)를 구한다.



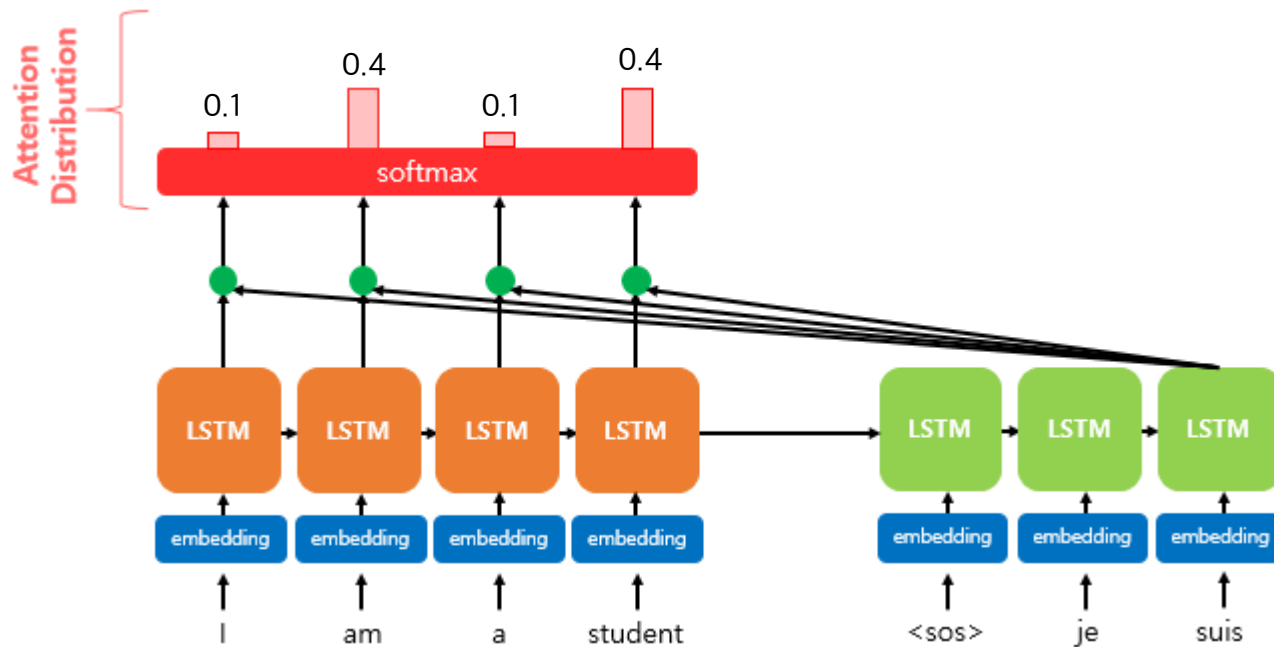
$$s_t^T \times h_i$$

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

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

Dot-Product Attention

2) 소프트맥스(softmax) 함수를 통해 어텐션 분포(Attention Distribution)를 구한다.



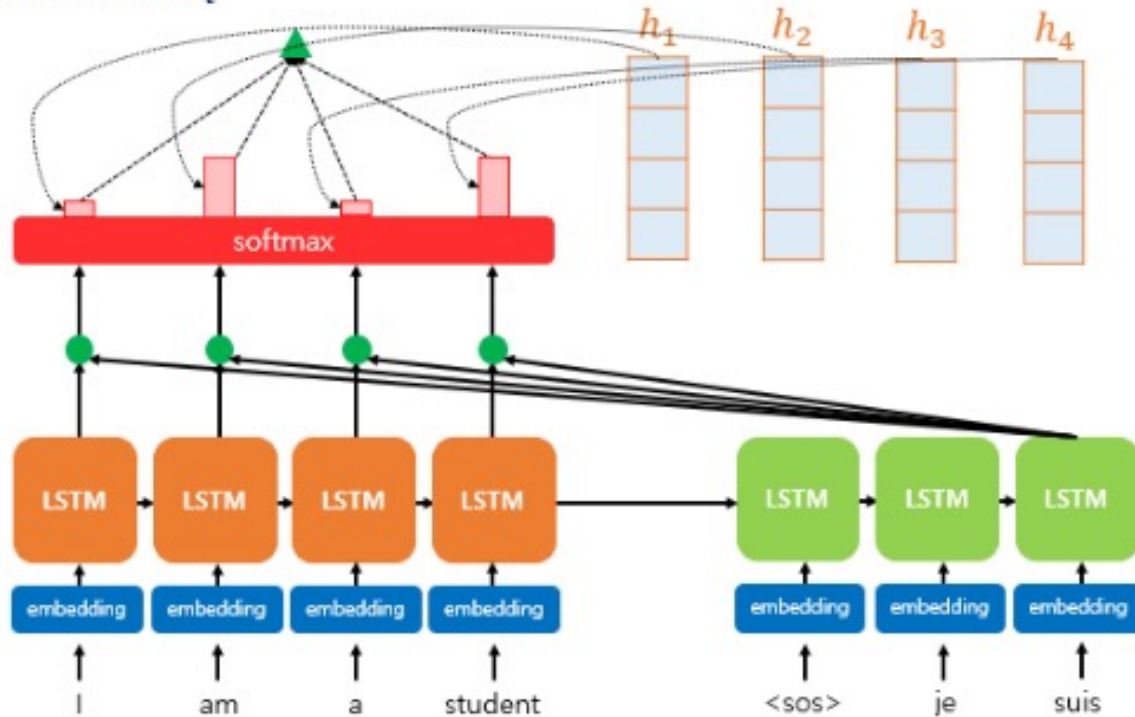
디코더 시점 t 에서의 어텐션 가중치의 모음값인 어텐션 분포를 α^t 이라고 할 때, α^t 을 식으로 정의하면 다음과 같다.

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

Dot-Product Attention

3) 각 인코더의 어텐션 가중치와 은닉 상태를 가중합하여 어텐션 값(Attention Value)을 구한다.

Attention Value a_t



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

이러한 어텐션 값 a_t 는 인코더의 문맥을 포함하고 있다고 하여, 컨텍스트 벡터(context vector)라고도 불림

앞서 seq2seq에서는 인코더의 마지막 은닉 상태를 컨텍스트 벡터라고 부르는 것과 대조

Unit 02 | Attention mechanism

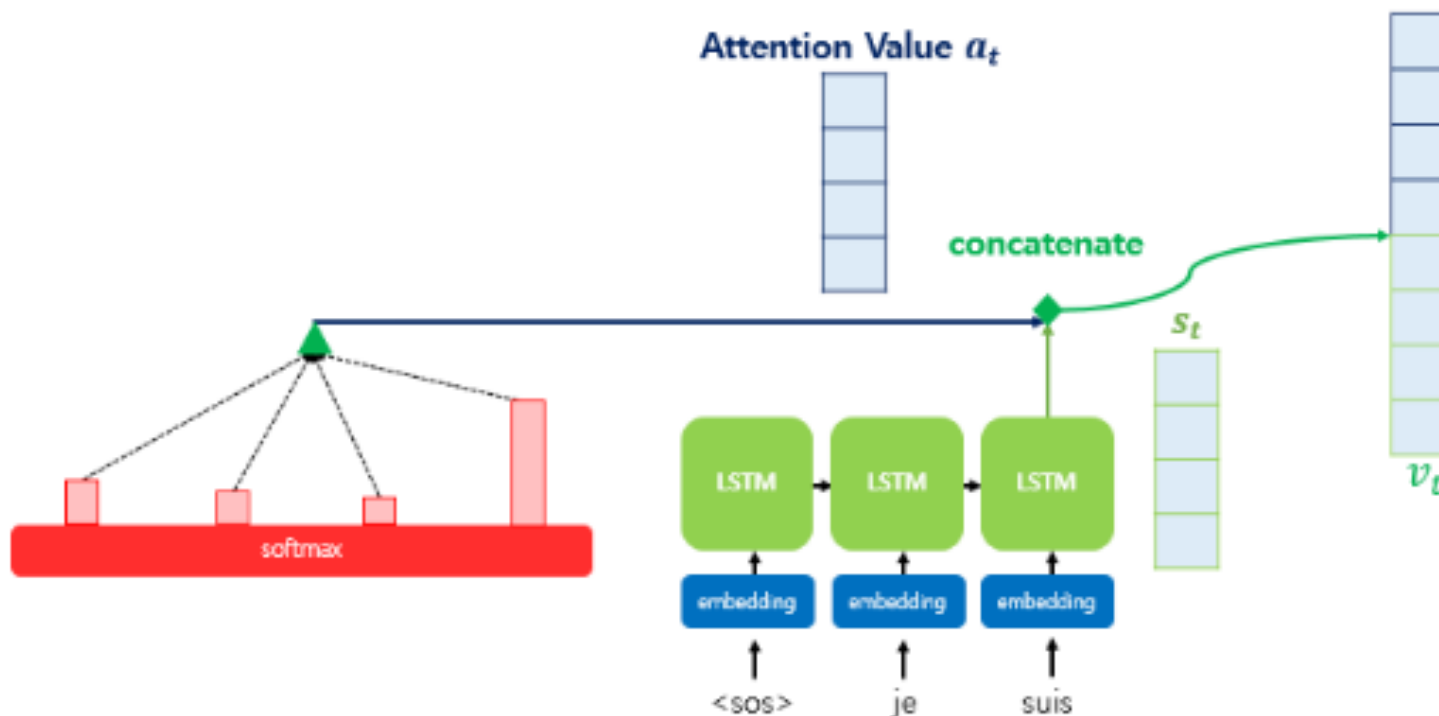


20기 정규세션

TOBIG'S 19기 임승섭

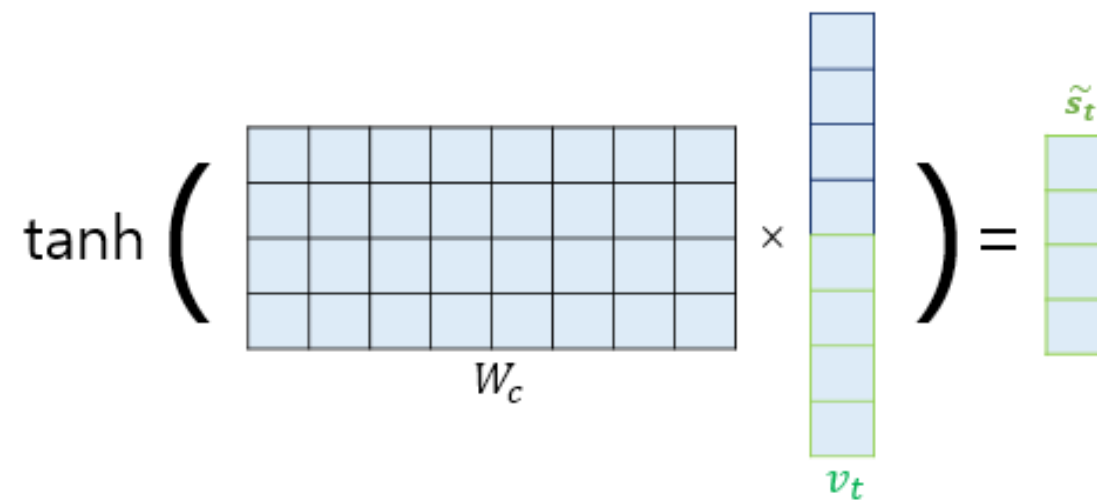
Dot-Product Attention

4) 어텐션 값과 디코더의 t 시점의 은닉 상태를 연결한다.(Concatenate)



Dot-Product Attention

5) 출력층 연산의 입력이 되는 \tilde{s}_t 를 계산합니다.


$$\tanh \left(\begin{matrix} \text{4x8 grid} \\ W_c \end{matrix} \times \begin{matrix} \text{10x1 column} \\ v_t \end{matrix} \right) = \begin{matrix} \text{4x1 column} \\ \tilde{s}_t \end{matrix}$$

- 논문에서는 v_t 를 바로 출력층으로 보내기 전에 신경망 연산을 한 번 더 추가
- 가중치 행렬과 곱한 후에 \tanh 함수를 지나도록 하여 출력층 연산을 위한 새로운 벡터인 \tilde{s}_t 를 얻음

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

Dot-Product Attention

6) \tilde{s}_t 를 출력층의 입력으로 사용합니다.

- \tilde{s}_t 를 출력층의 입력으로 사용하여 예측 벡터를 얻음

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



20기 정규세션

TOBIG'S 19기 임승섭

Unit 03

Transformer

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

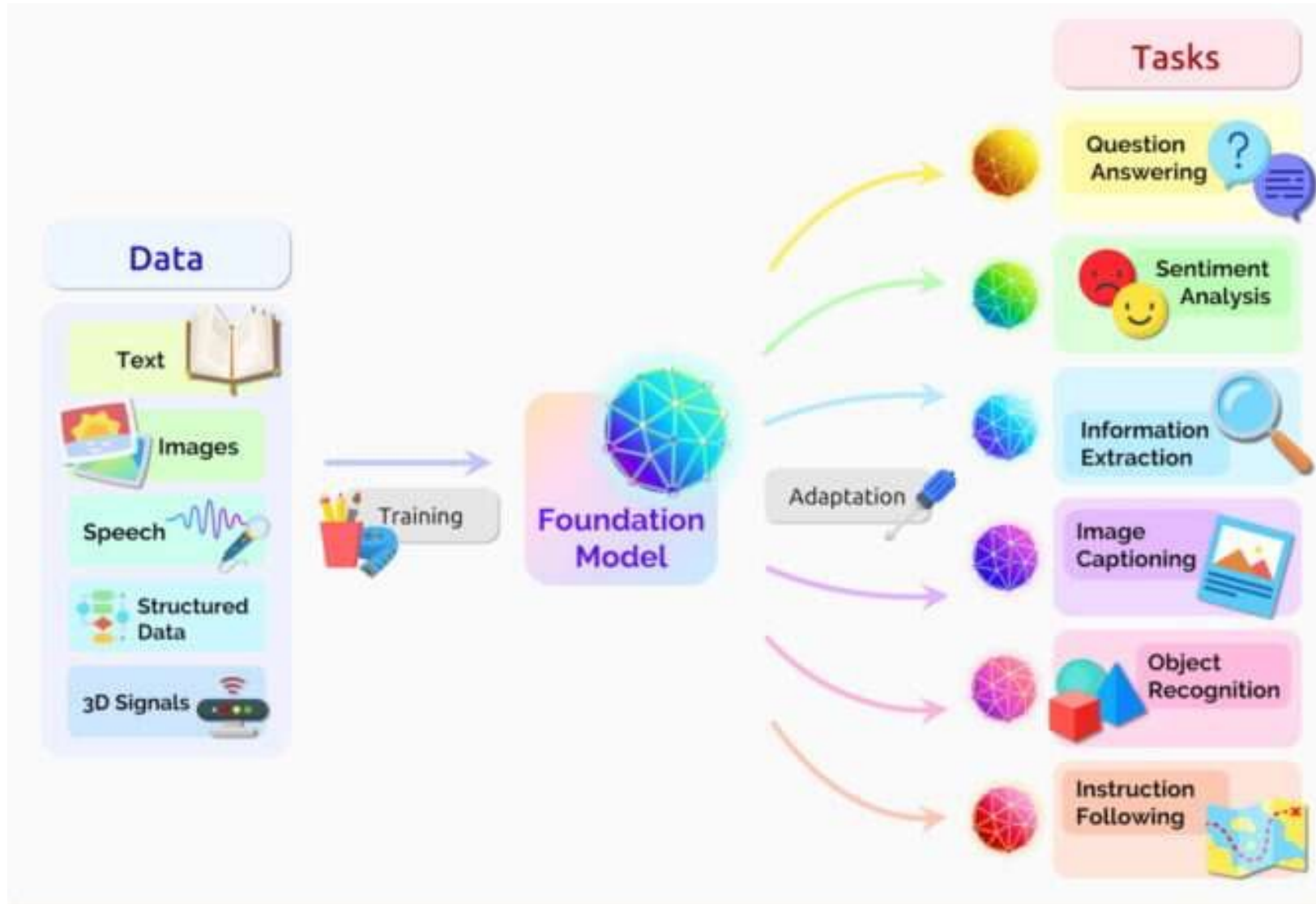
Unit 03 | Transformer



20기 정규세션

TOBIG'S 19기 임승섭

파운데이션 모델이라고도 불리는 트랜스포머는 여러 데이터 소스와 함께 다양한 영역에서 활용되고 있음



Unit 03 | Transformer



20기 정규세션

TOBIG'S 19기 임승섭

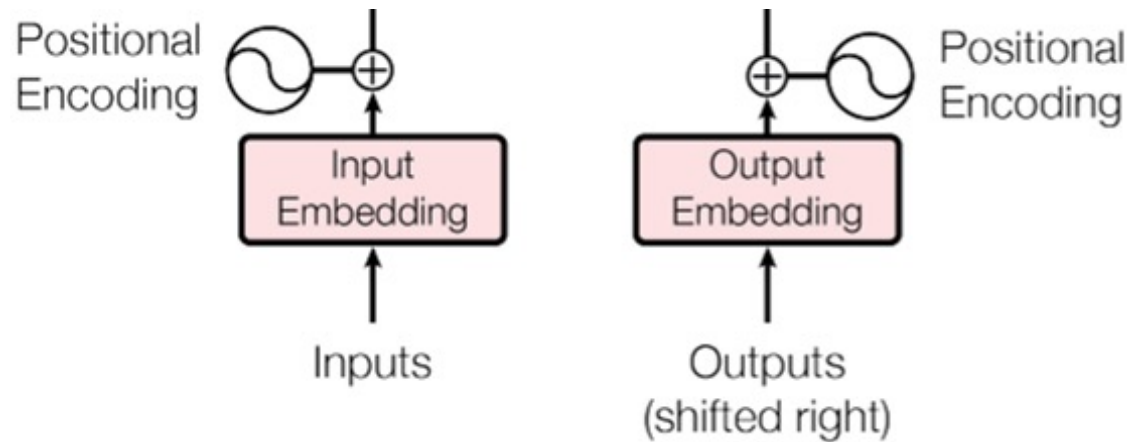
- 트랜스포머 모델의 아이디어는 Seq2seq + Attention 모델에서 RNN 구조를 제거하는 것
- 즉, RNN을 사용하지 않고 오직 어텐션 연산만 사용한 모델이 트랜스포머 모델

왜 RNN을 제거했을까

- 시퀀스 형태의 데이터는 각 항목의 값 뿐만 아니라 그 순서도 중요
- "John loves Sarah"와 "Sarah loves John"은 단어는 모두 같지만 순서가 달라 의미가 다름
- 즉, 시퀀스 형태의 데이터를 다루기 위해서는 순서 정보를 처리할 수 있는 모델을 사용해야 함
- 기존에는 시퀀스 형태의 데이터를 처리하기에 적합한 RNN을 사용하였으나, 단점이 많음
 - **병렬화 문제** : RNN은 그 구조상 순차적으로 입력을 처리해야 하기에 병렬화가 불가능. 이 때문에 대규모의 데이터셋을 이용한 학습이 불가능(학습 시간이 너무 길어짐)
 - **long distance dependency 문제** : 시퀀스에서 멀리 떨어진 항목들 간의 관계성은 gradient vanishing/exploding 문제로 학습이 잘 되지 않음

어떻게 RNN을 제거했나 : Positional Encoding

- RNN 구조에서는 순서 정보가 자연스럽게 모델에 입력됐지만, 어텐션 연산에서는 순서 정보가 고려되지 않음
- 그래서 트랜스포머 모델에서는 input embedding에 positional encoding이라 불리는, 입력 임베딩과 같은 차원의 위치 정보를 담고 있는 벡터를 더해줌



$$f(p) = \begin{cases} \sin\left(\frac{p}{10000^{i/d}}\right) & (i = 2k) \\ \cos\left(\frac{p}{10000^{(i-1)/d}}\right) & (i = 2k + 1) \end{cases}$$

Unit 03 | Transformer



20기 정규세션

TOBIG'S 19기 임승섭

트랜스포머 모델의 구조

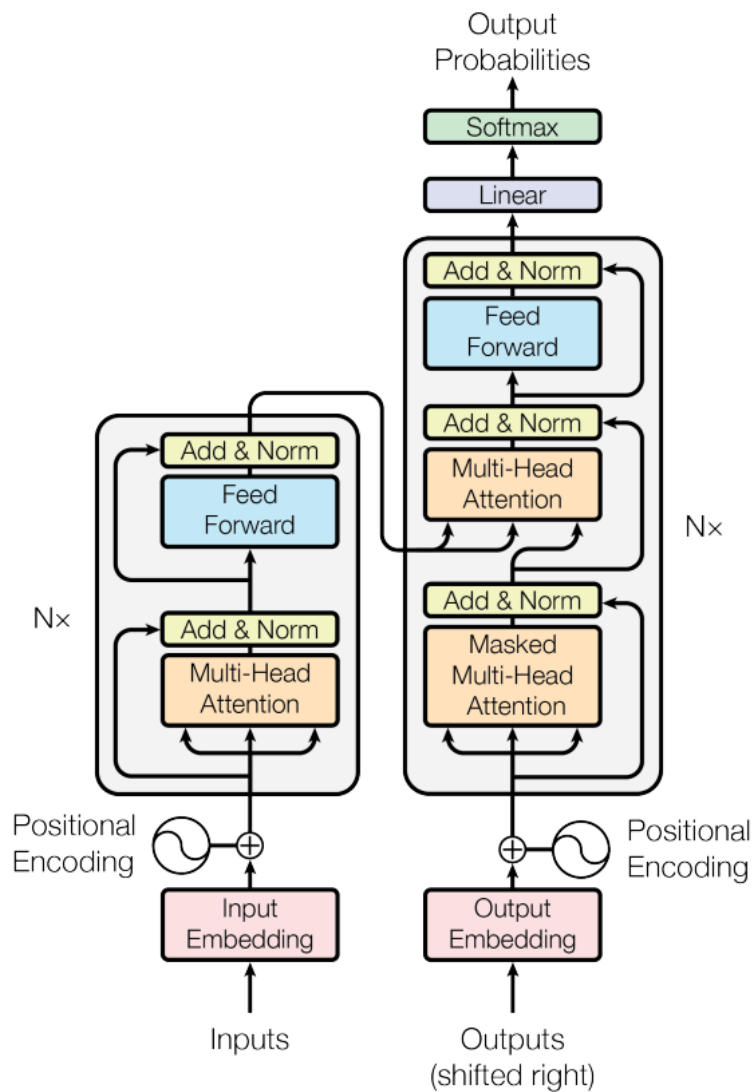
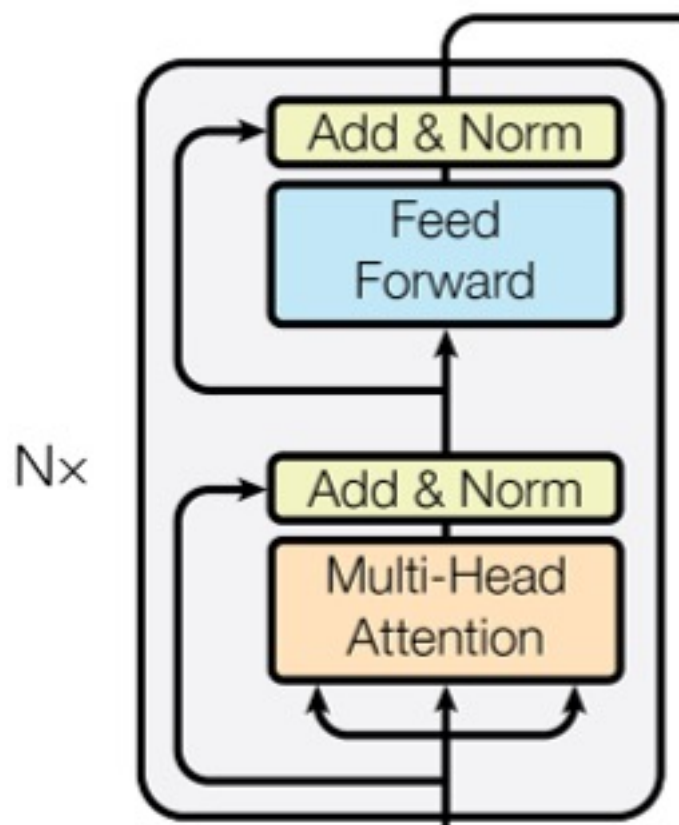


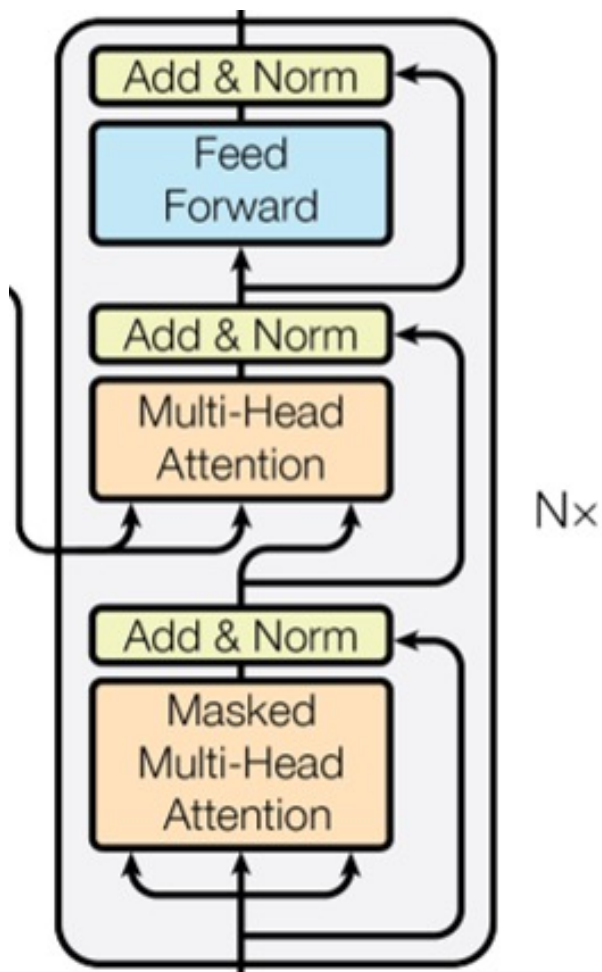
Figure 1: The Transformer - model architecture.

트랜스포머 모델의 구조

- 트랜스포머 인코더는 인코더 레이어를 N개 쌓은 구조
- 이전 인코더 레이어의 출력은 다음 인코더 레이어의 입력으로 사용



트랜스포머 모델의 구조



- 트랜스포머 디코더는 디코더 레이어를 N 개 쌓은 구조
- 이전 디코더 레이어의 출력은 다음 디코더 레이어의 입력으로 사용

Unit 03 | Transformer



20기 정규세션

TOBIG'S 19기 임승섭

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”

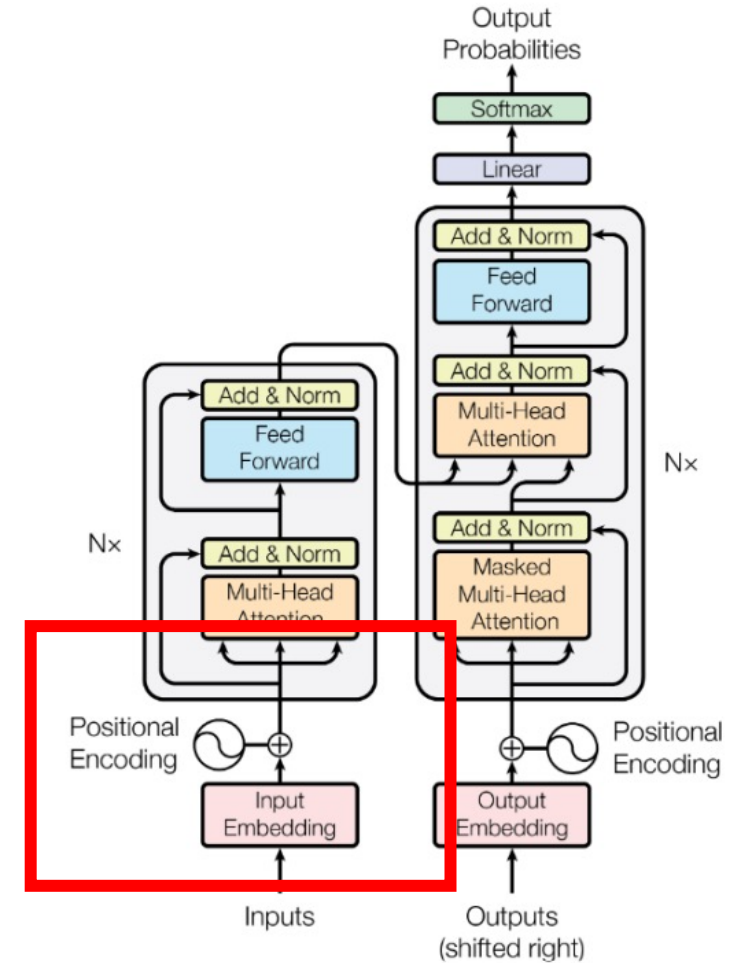
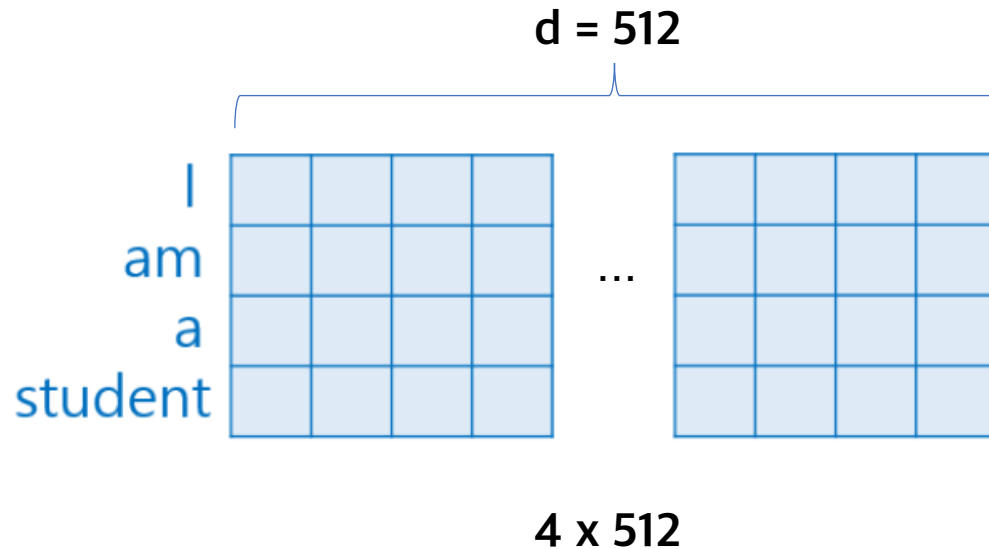


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



20기 정규세션

TOBIG'S 19기 임승섭

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

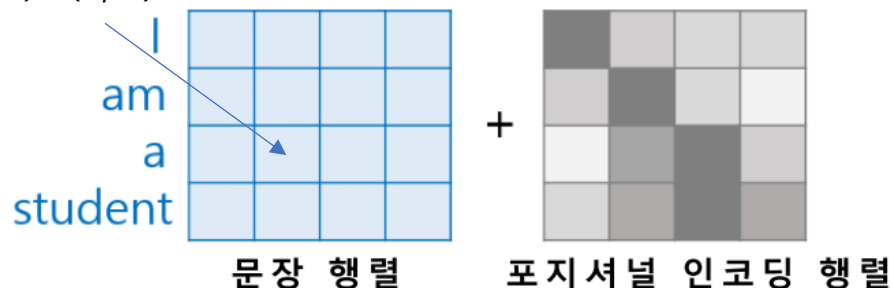
- Why?)

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

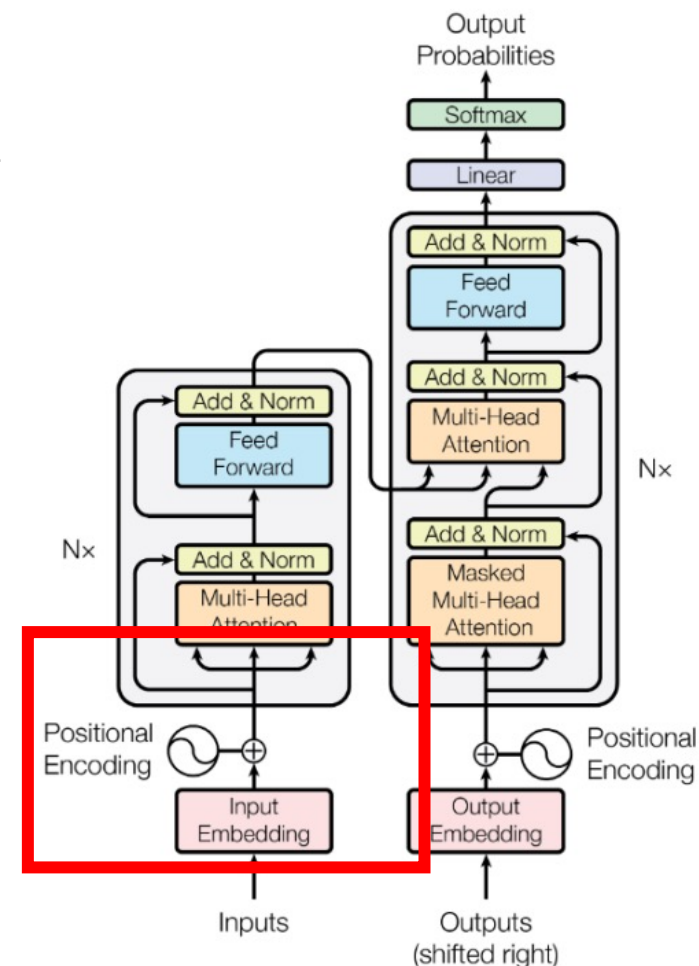


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



20기 정규세션

TOBIG'S 19기 임승섭

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 적용
- Residual connection은 어떠한 연산의 결과를 연산의 입력과 다시 더해주는 것을 의미. 그 이후 정규화를 진행
=> $LayerNorm(x + Sublayer(x))$

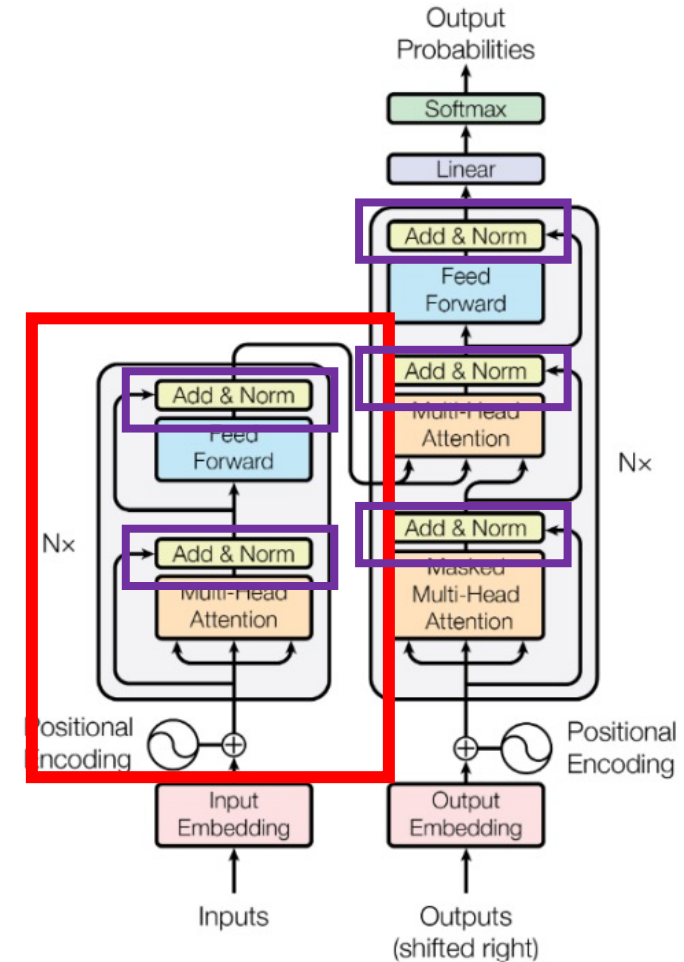


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer

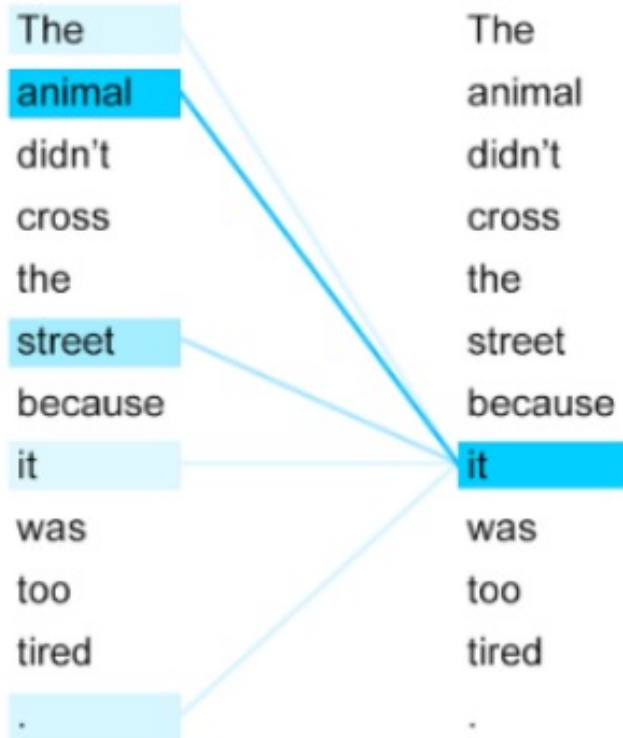


20기 정규세션

TOBIG'S 19기 임승섭

Self-attention

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



‘It’ 이 ‘animal’ 과 연관되었을 확률이 높다는 것을 찾아내기 위함!

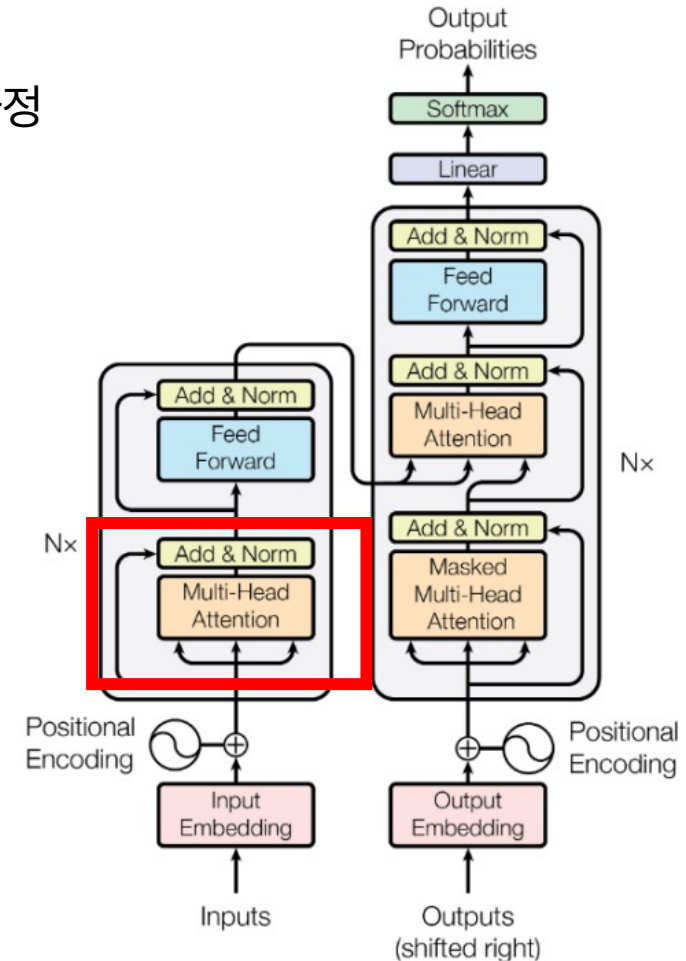


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



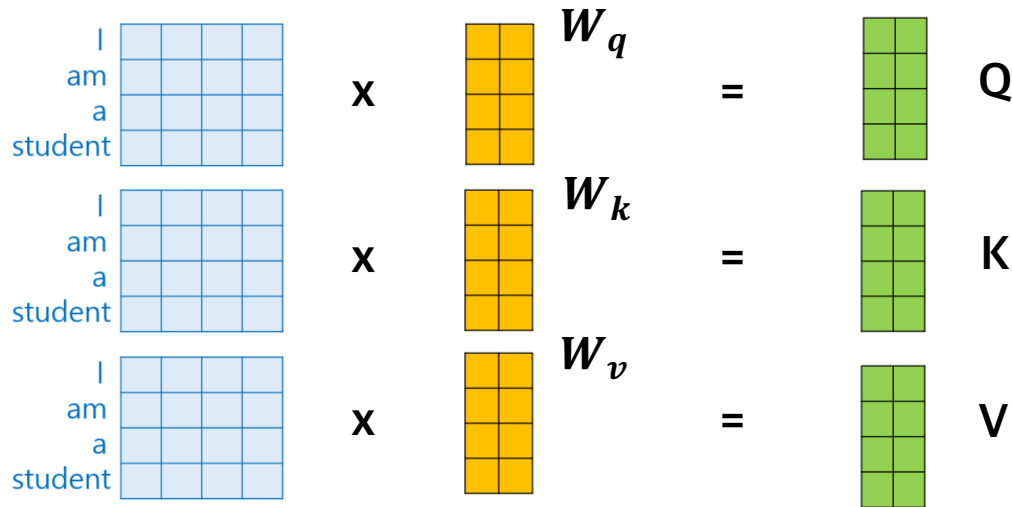
20기 정규세션

TOBIG'S 19기 임승섭

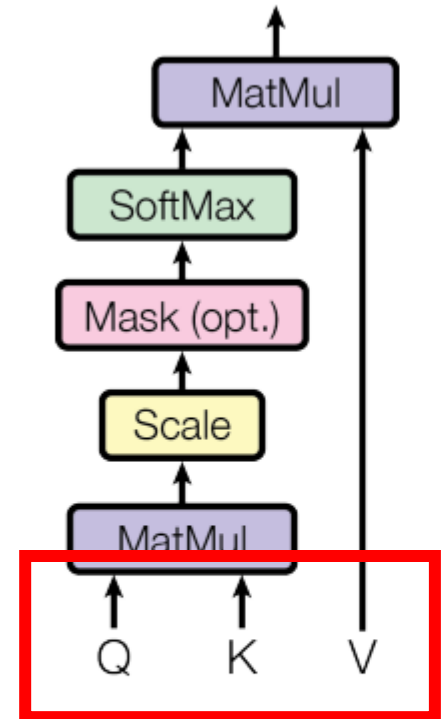
Self-attention

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



20기 정규세션

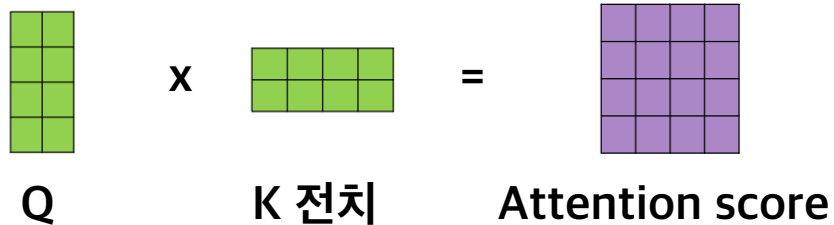
TOBIG'S 19기 임승섭

Self-attention

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

2) Attention score 계산

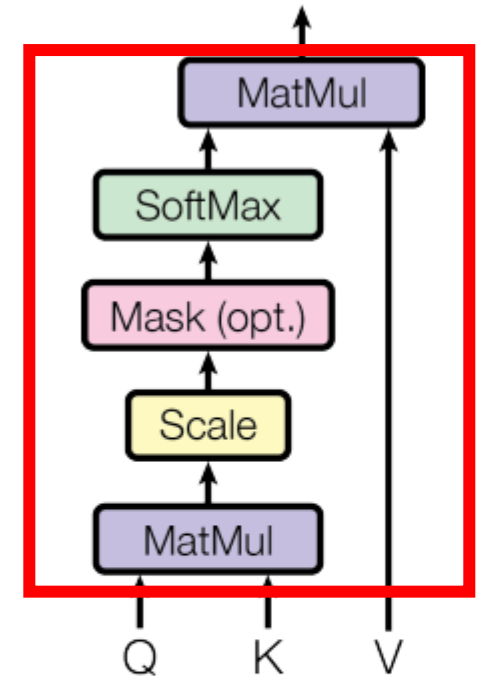
- 생성된 Q 와 K 를 곱해줌



3) Scaling 및 self-attention 계산

- Scaling 진행 (d_k 는 차원을 의미, 그냥 self-attention일 땐 512, multi-head attention ... 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.

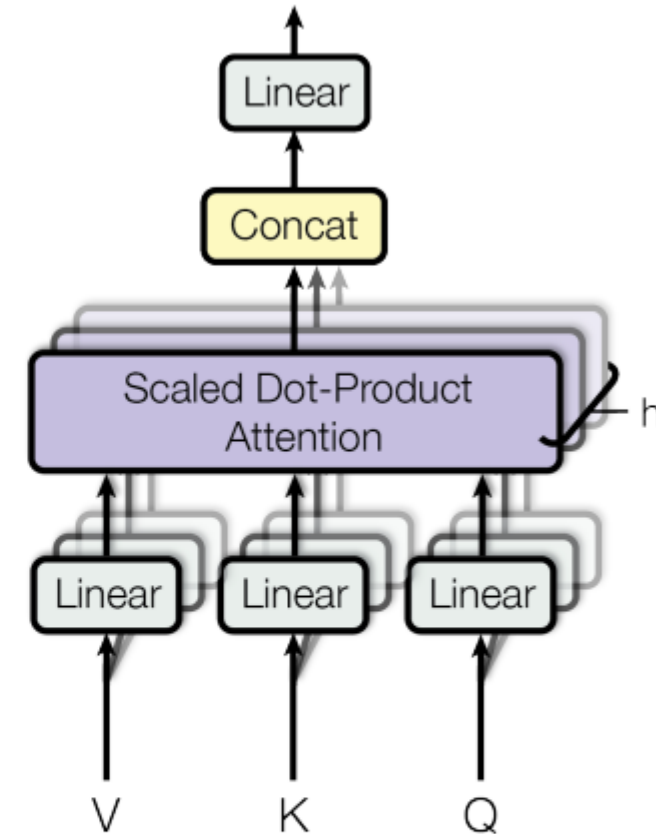
어떤 헤드에서는 it과 animal 의 연관성을 높게 볼 것이고,

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!

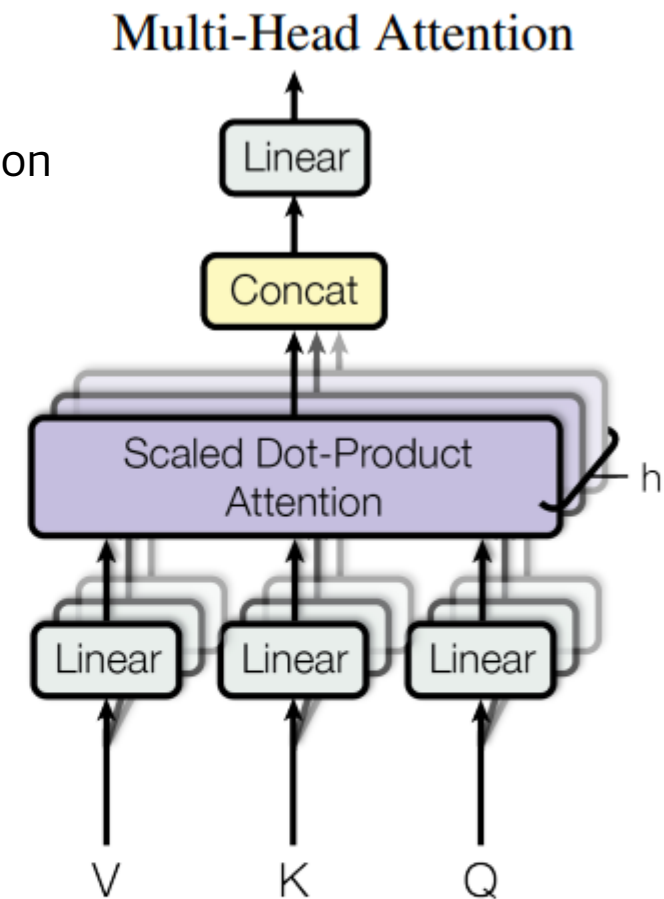
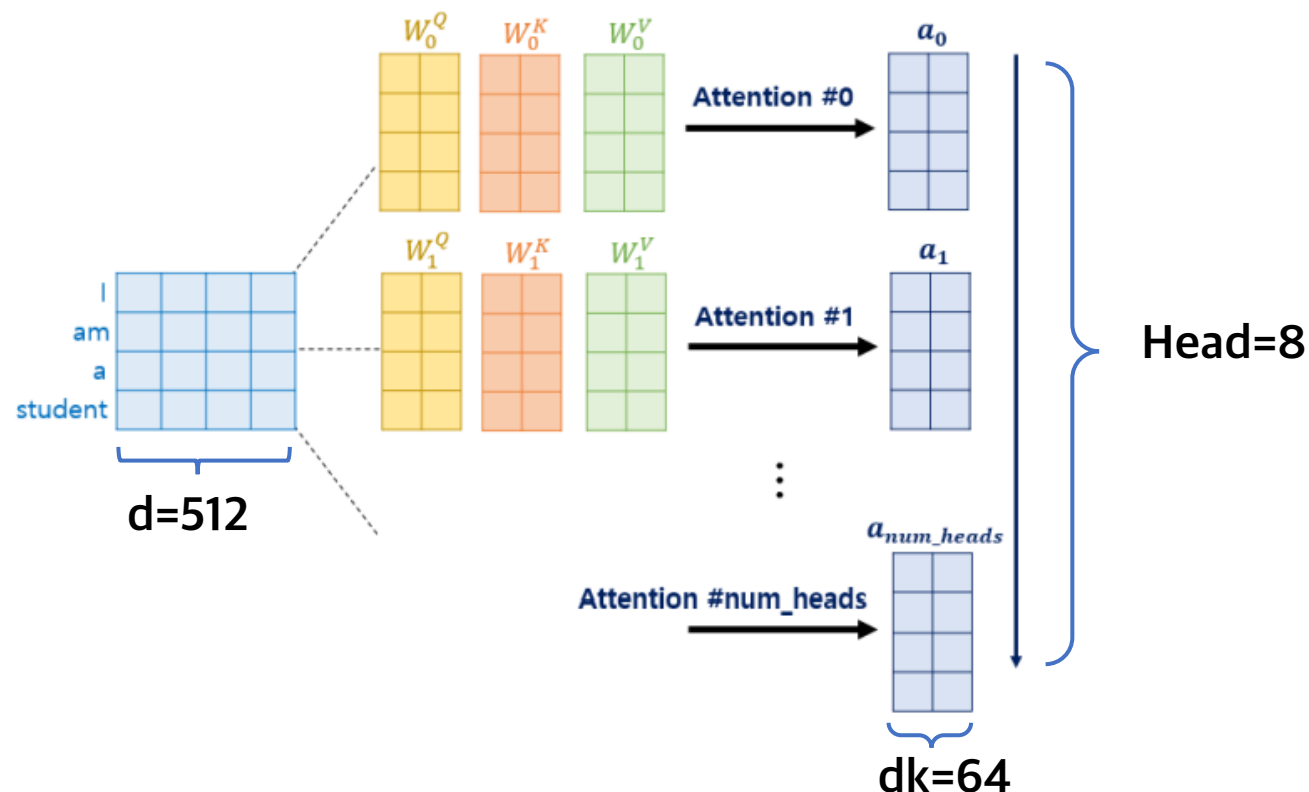
- 본 논문에서는 head 수를 8로 설정하고, Self-attention 을 병렬로 계산한다음 concat(연결) 진행!

Multi-Head Attention



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



20기 정규세션
TOBIG'S 19기 임승섭

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

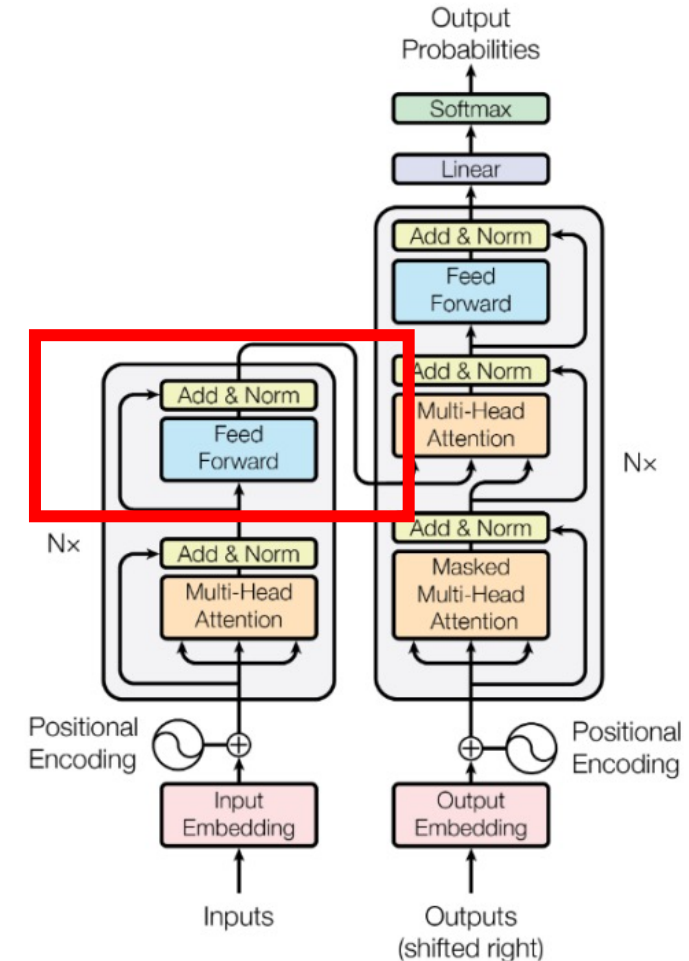
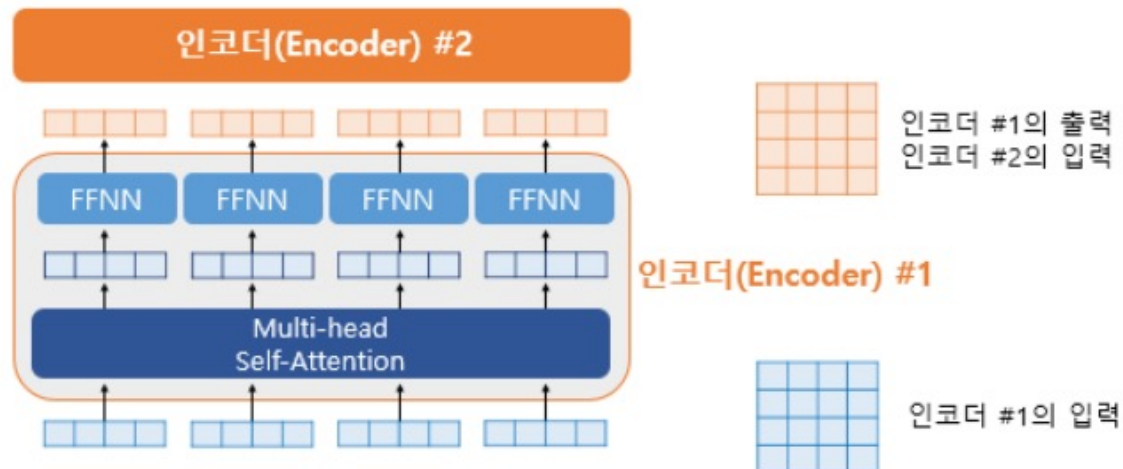


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



20기 정규세션

TOBIG'S 19기 임승섭

Decoder

- 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

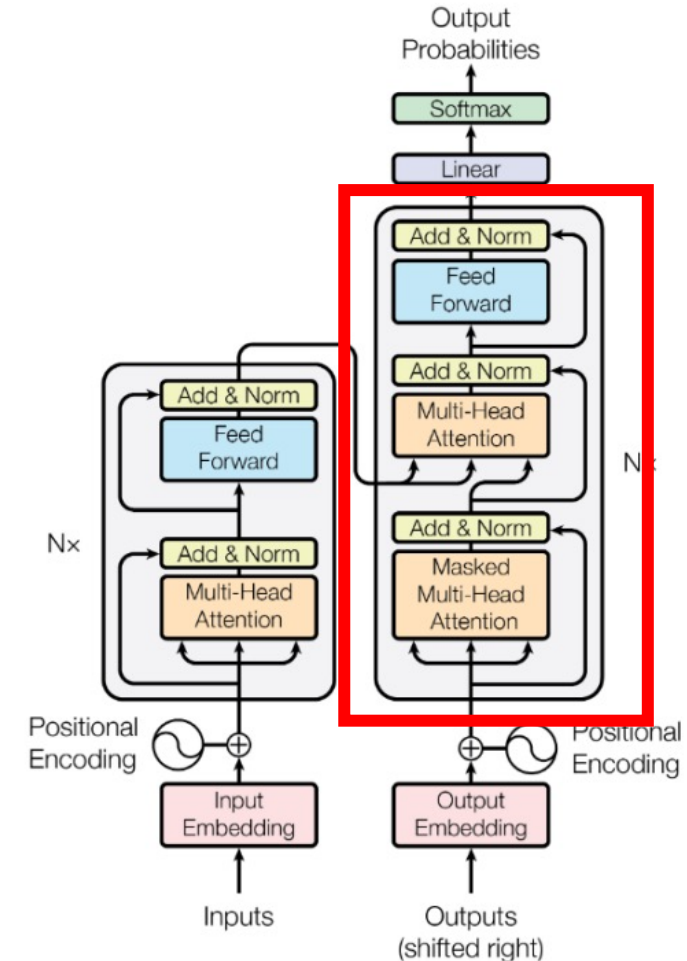
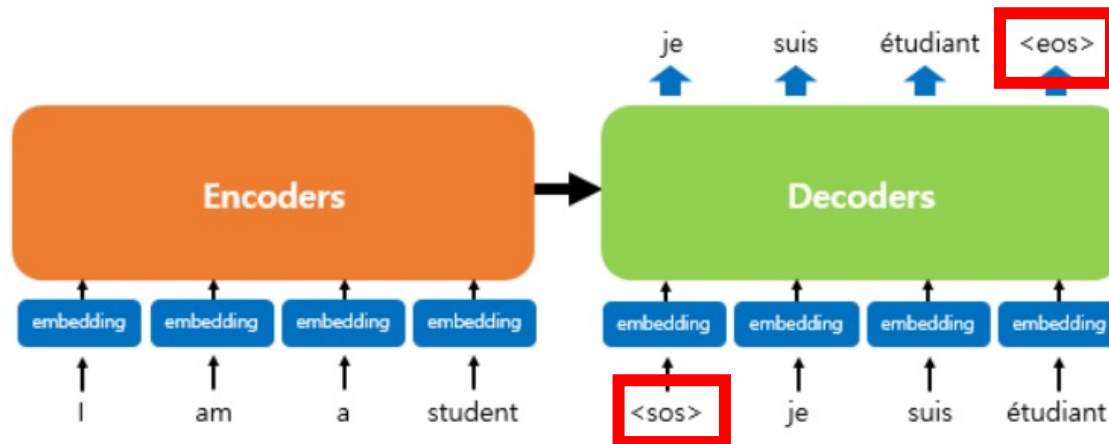


Figure 1: The Transformer - model architecture.

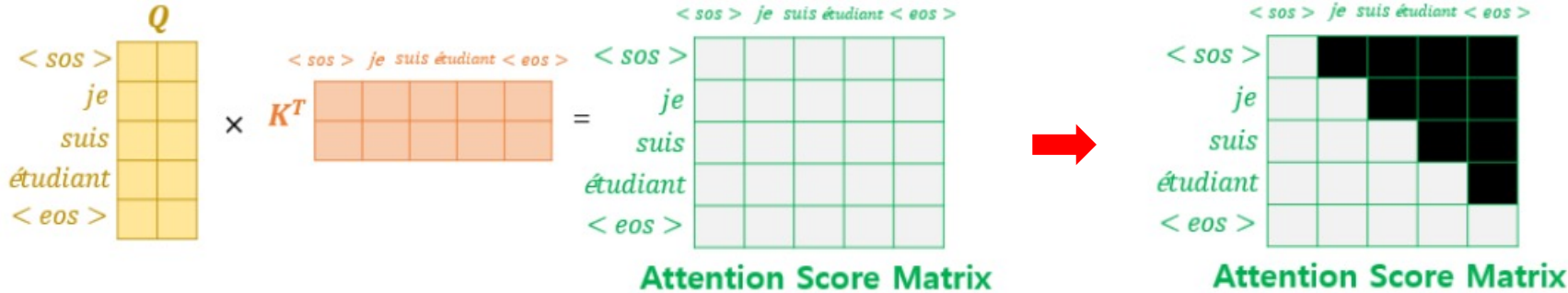
Unit 03 | Transformer



20기 정규세션
TOBIG'S 19기 임승섭

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?

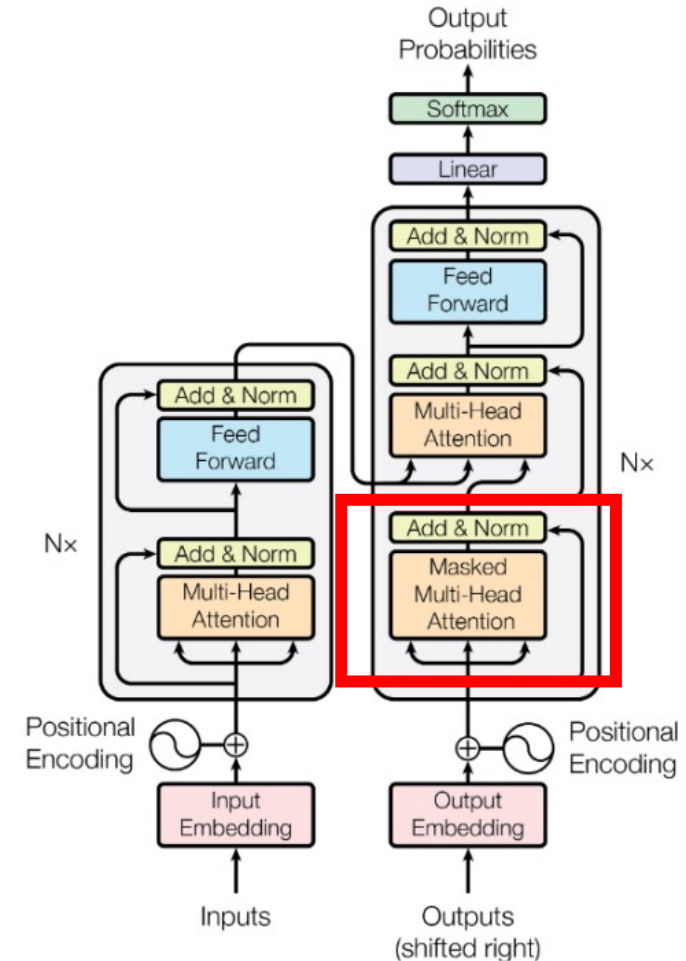


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



20기 정규세션

TOBIG'S 19기 임승섭

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

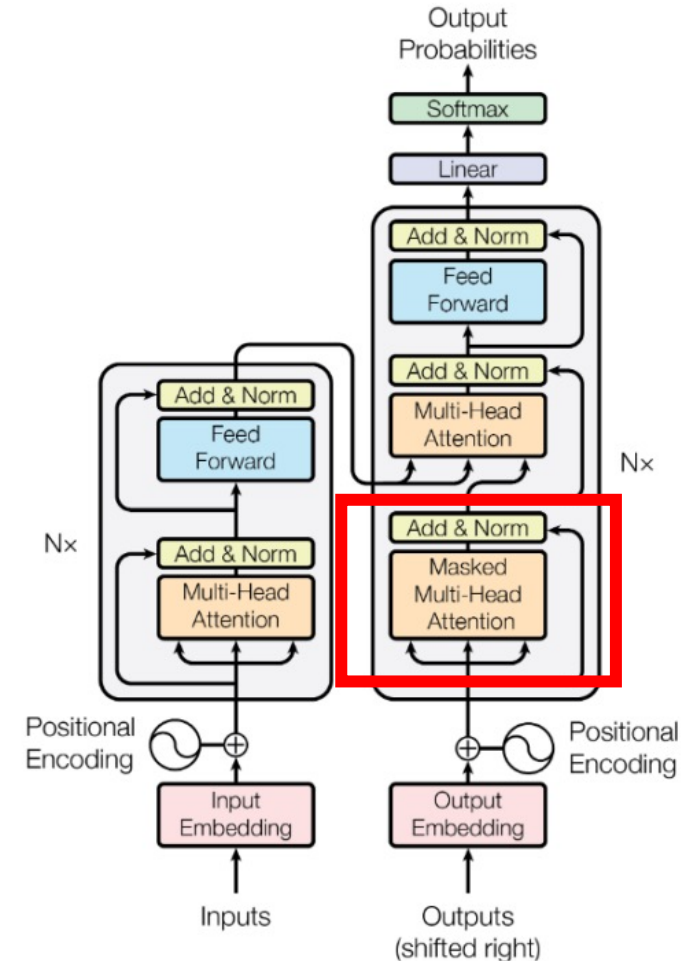
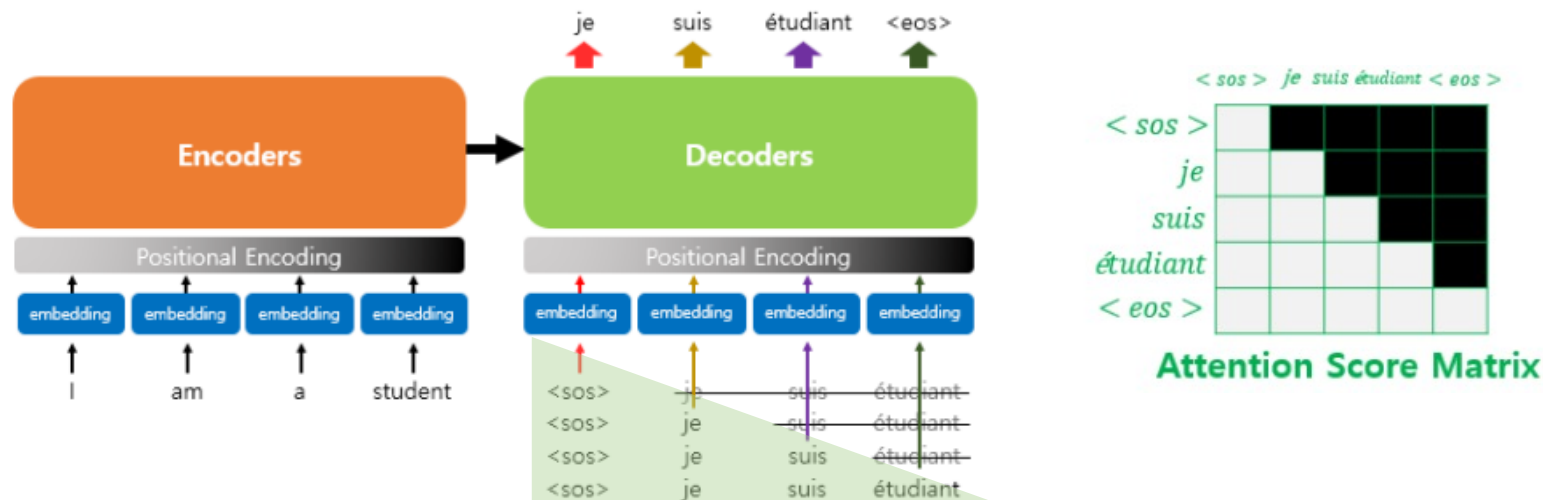


Figure 1: The Transformer - model architecture.

Unit 03 | Transformer



20기 정규세션
TOBIG'S 19기 임승섭

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 맞추는지 학습하는 과정

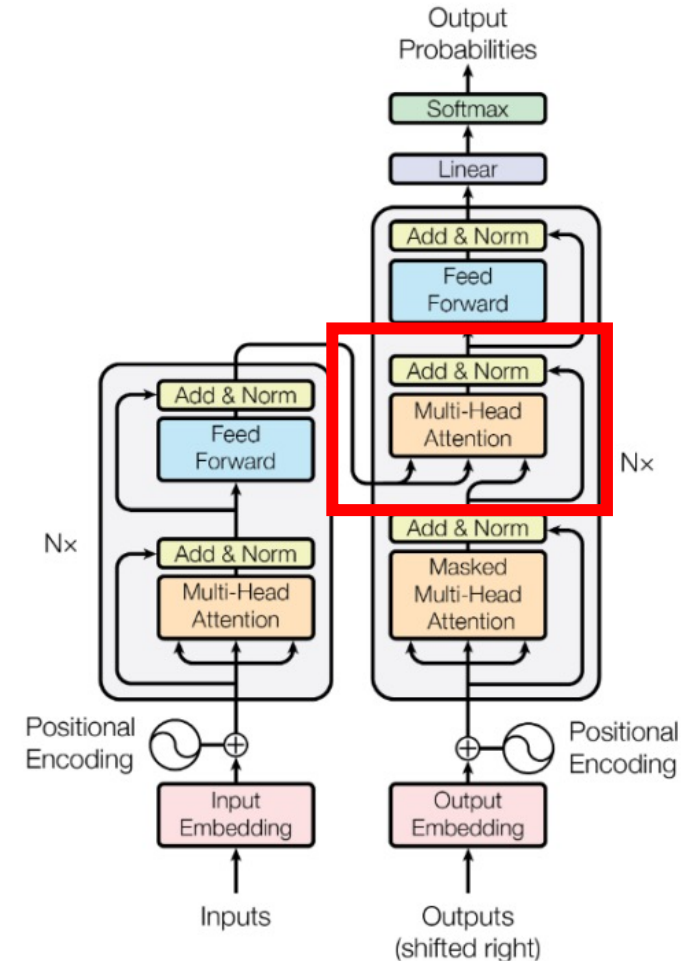
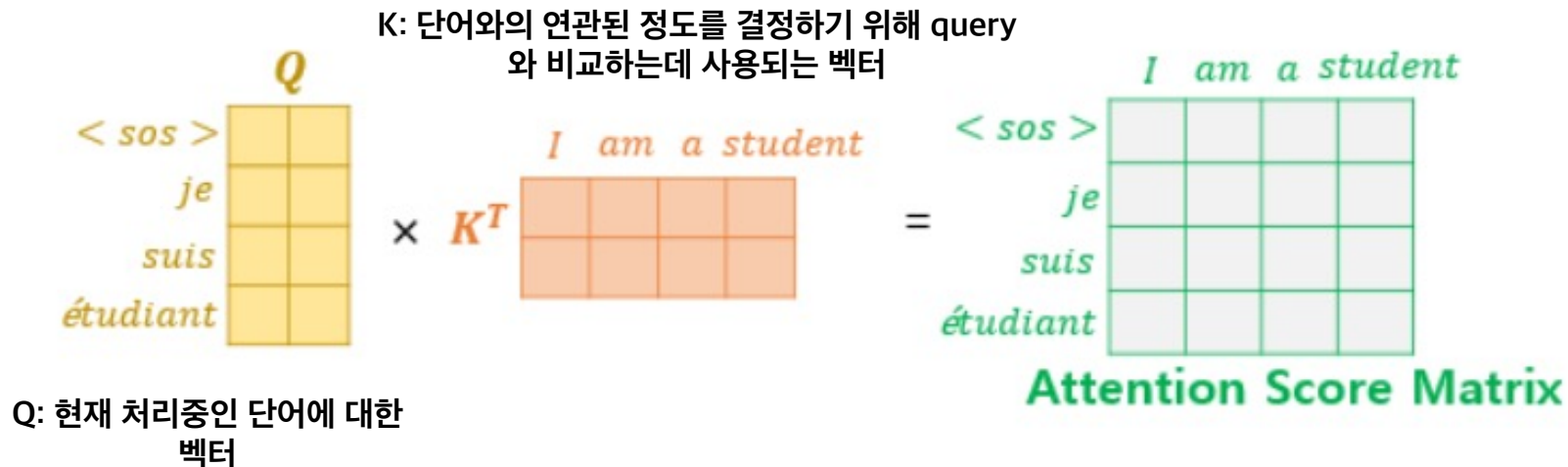


Figure 1: The Transformer - model architecture.



20기 정규세션

TOBIG'S 19기 임승섭

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 정도)

과제 2: Transformer 기반 후속 논문(ex: BERT, GPT 등) 1개를 읽고 정리해주세요!

- 자연어처리 계열 논문만 가능합니다. (ViT 등과 같은 Vision 계열 X)
- 후속 연구 관련 논문 1개를 선택한 다음 정리해주세요(노션 및 개인 블로그 링크를 올려도 됩니다.)

과제 1 or 과제 2 둘 중 하나 선택 !

Reference



20기 정규세션

TOBIG'S 19기 임승섭

<https://wikidocs.net/31379>

<https://heekangpark.github.io/nlp/attention>

[Attention Is All You Need](#)

[NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE](#)

[Sequence to Sequence Learning with Neural Networks](#)

19기 NLP Advanced 강의자료

