

transformer

practice 인공지능,머신러닝 · 2024. 9. 28. 14:56

transformer가 무엇인가요.

- multi head self attention을 사용한 model
- attention 만을 이용해서 RNN대체 (attention is all you need)

왜 나오게 되었는지요(장점)

- long term dependency 해결
- 병렬처리(동시처리) - 처리시간 줄임 - 입력문장 전체를 넣어서 동시에 처리, cf RNN은 token 1-> 처리, token2 --> 처리

multi head attention 이 뭔고, 역할이 뭔가요

- attention head를 여러개 사용
- 여러 관점에서 문장해석이 가능하다.(x-> q1,q2...)

=====세부정리=====

transformer가 나온이유

- long term dependency self attention으로 해결
- 병렬처리(동시처리) - 처리시간 줄임 - 입력문장 전체를 넣어서 동시에 처리, cf RNN은 token 1-> 처리, token2 --> 처리

- 한번에 문장을 처리해서 문장을 빠르게 처리

Transformer

attention

주요한

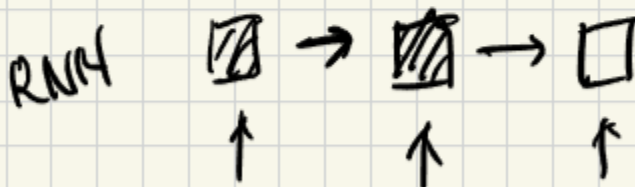
~~RNN~~ 525

attention is all you need

self attention = 일방적 연결

long term dependency → 해결

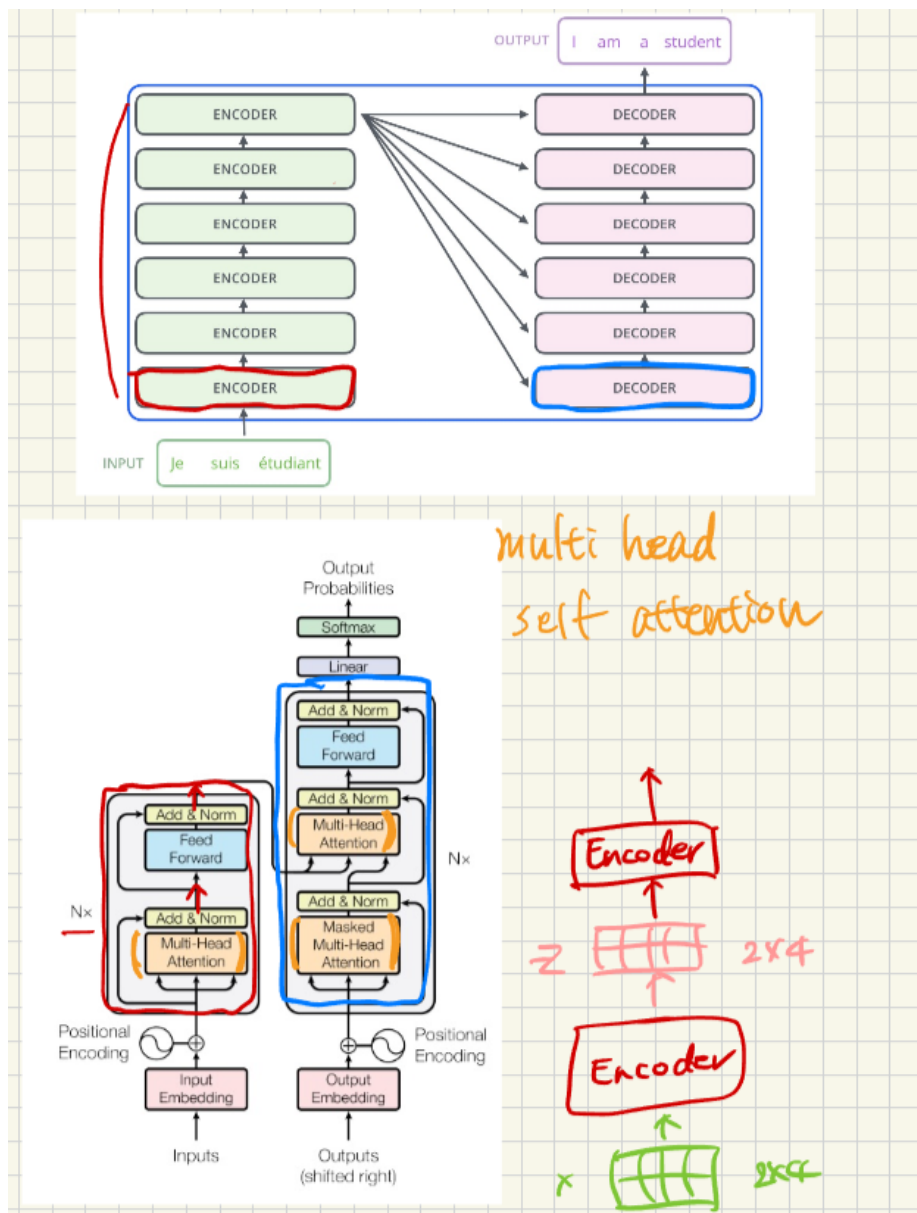
병렬 처리 → 속도가 빠름



token1 token2



token1 token2 ... tokenN



아래그림)

self attention

자기자신 문장을 봤을때 어느부분에 집중해야 하는지 학습하게 해주는 모듈

$q_1 \times k_1$ (내적) - 112 루트d로 크기줄임(scale) -> softmax :0.88

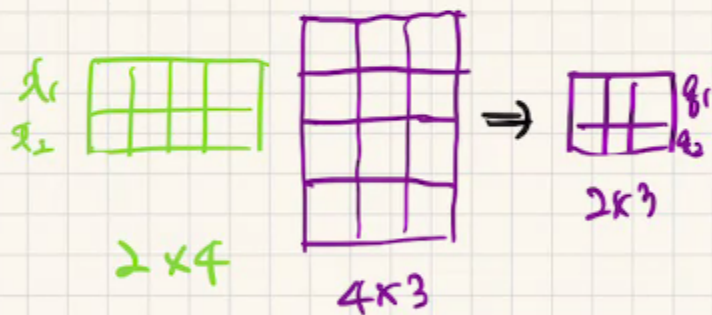
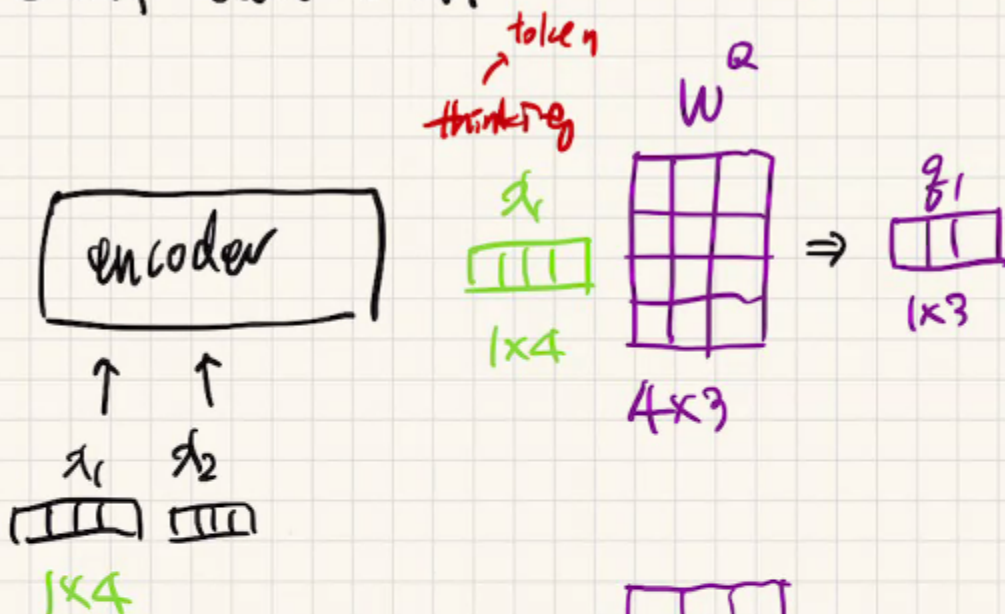
v_1, v_2 는 x 를 가중치 곱해서 변경

q_1, q_2 도 x 를 가중치 곱해서 변경

아래그림 출처 - <https://nlpinkorean.github.io/illustrated-transformer/>

Self attention

Q K V



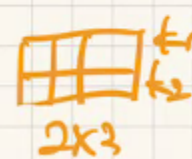
$q_1 \cdot k_1$ $q_1 \cdot k_2$

112 96

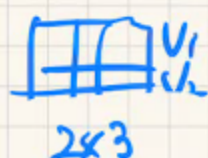
softmax

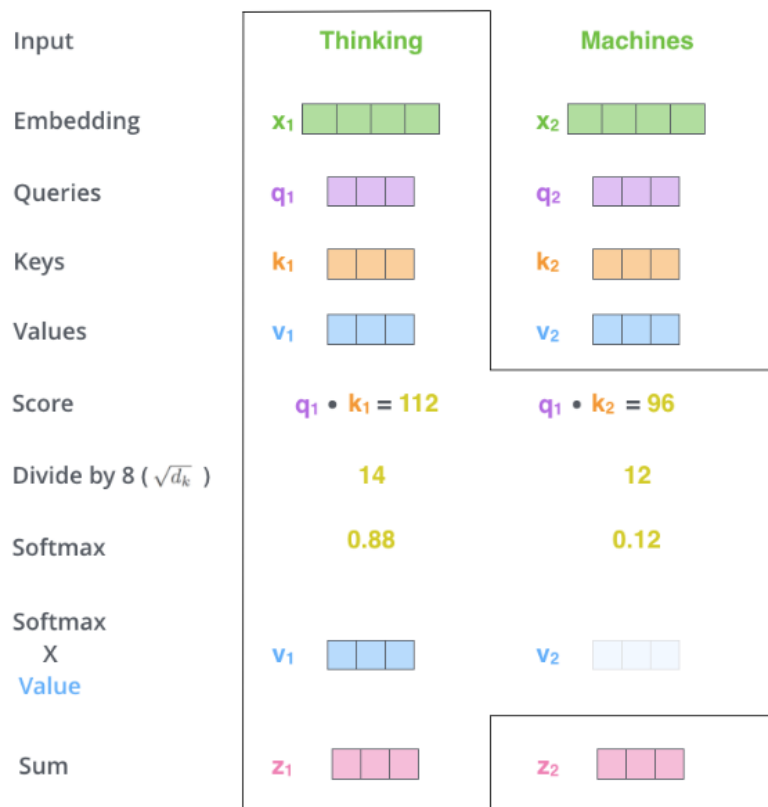
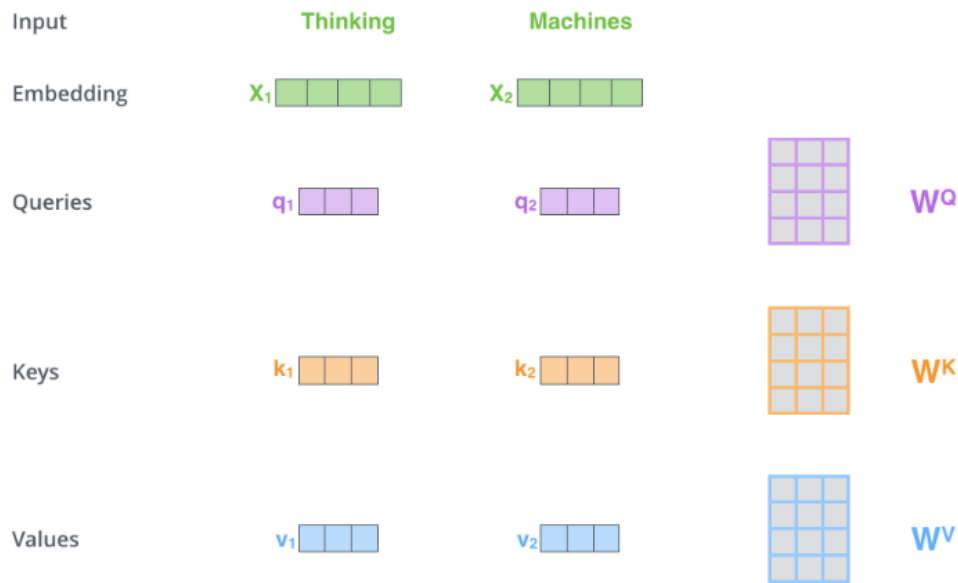
0.88 0.12

$$\frac{e^{14}}{e^{14} + e^{12}}$$



$$\vec{z}_1 = 0.88 \vec{v}_1 + 0.12 \vec{v}_2$$





아래그림

$x_1 \rightarrow q_1 \rightarrow z_1$ (x_1 의 입장에서 결과과 z_1)

$x_2 \rightarrow q_2 \rightarrow z_2$

x 를 가중치 W^Q, W^K, W^V 와 곱해서 q, q_2 행렬, k_1, k_2 행렬, v_1, v_2 행렬을 만든다.

$$\begin{array}{c} \text{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline \end{array} \end{array} \times \begin{array}{c} W^Q \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{array} = \begin{array}{c} Q \\ \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \begin{array}{l} q1 \\ q2 \end{array}$$

$$\begin{array}{c} \text{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline \end{array} \end{array} \times \begin{array}{c} W^K \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{array} = \begin{array}{c} K \\ \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \begin{array}{l} k1 \\ k2 \end{array}$$

$$\begin{array}{c} \text{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline \end{array} \end{array} \times \begin{array}{c} W^V \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{array} = \begin{array}{c} V \\ \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \begin{array}{l} v1 \\ v2 \end{array}$$

아래그림

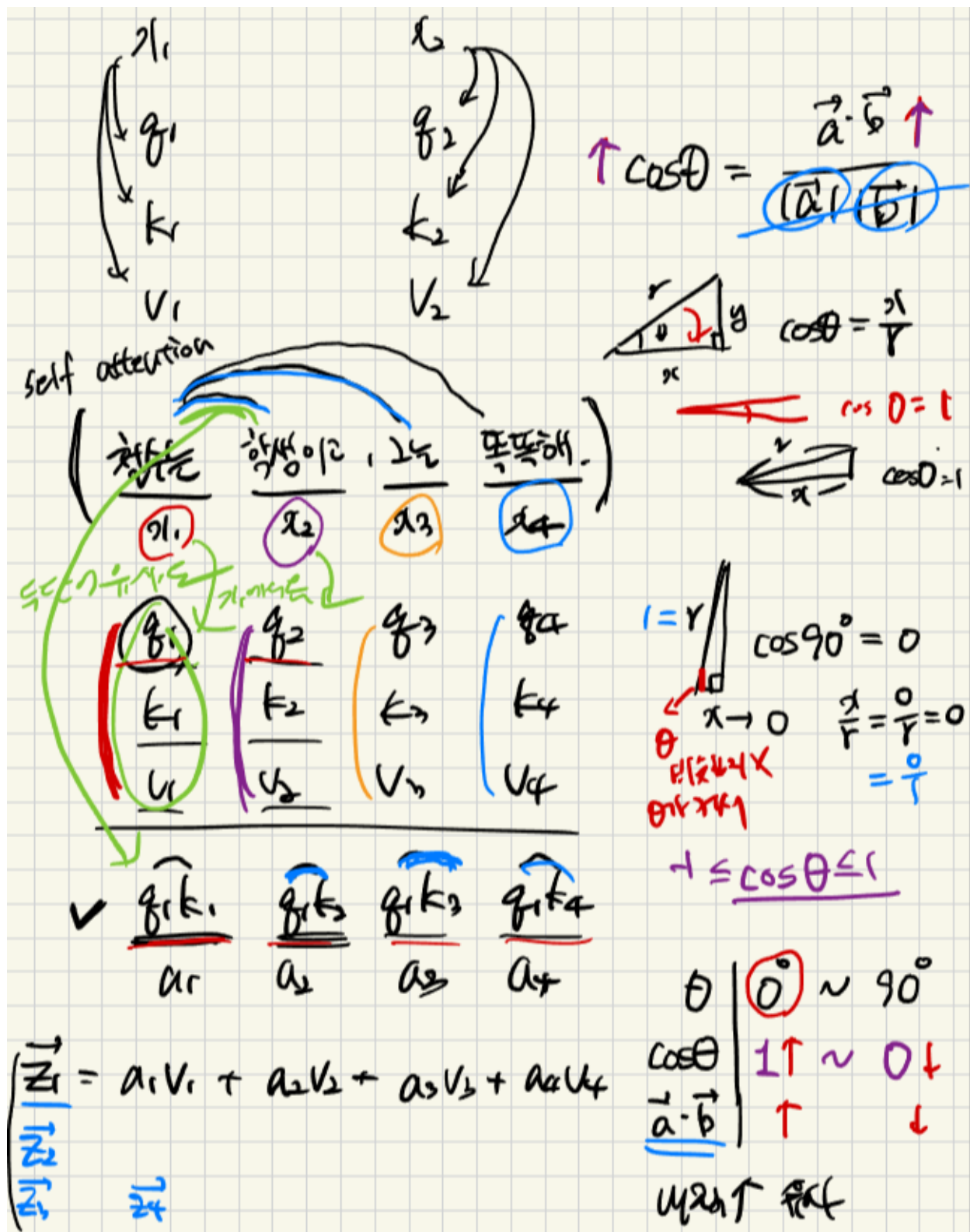
z를 아래 세부식으로 구한다.

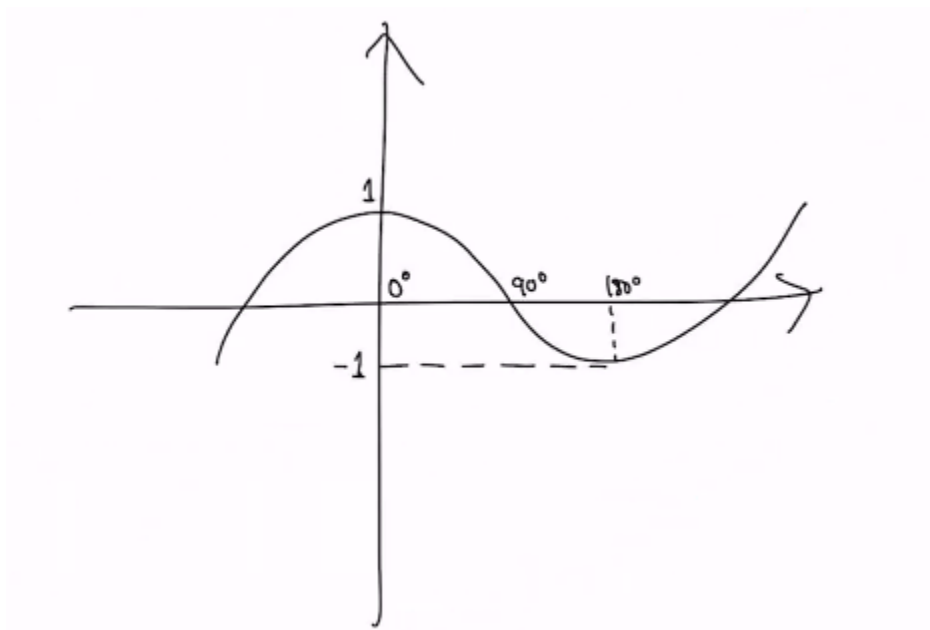
$$\text{softmax} \left(\frac{\begin{array}{c} Q \\ \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \times \begin{array}{c} K^T \\ \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \end{array}}{\sqrt{d_k}} \right) \begin{array}{c} V \\ \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \end{array}$$

$$= \begin{array}{c} Z \\ \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array}$$

행렬 형태로 표현한 self-attention 계산

세부식

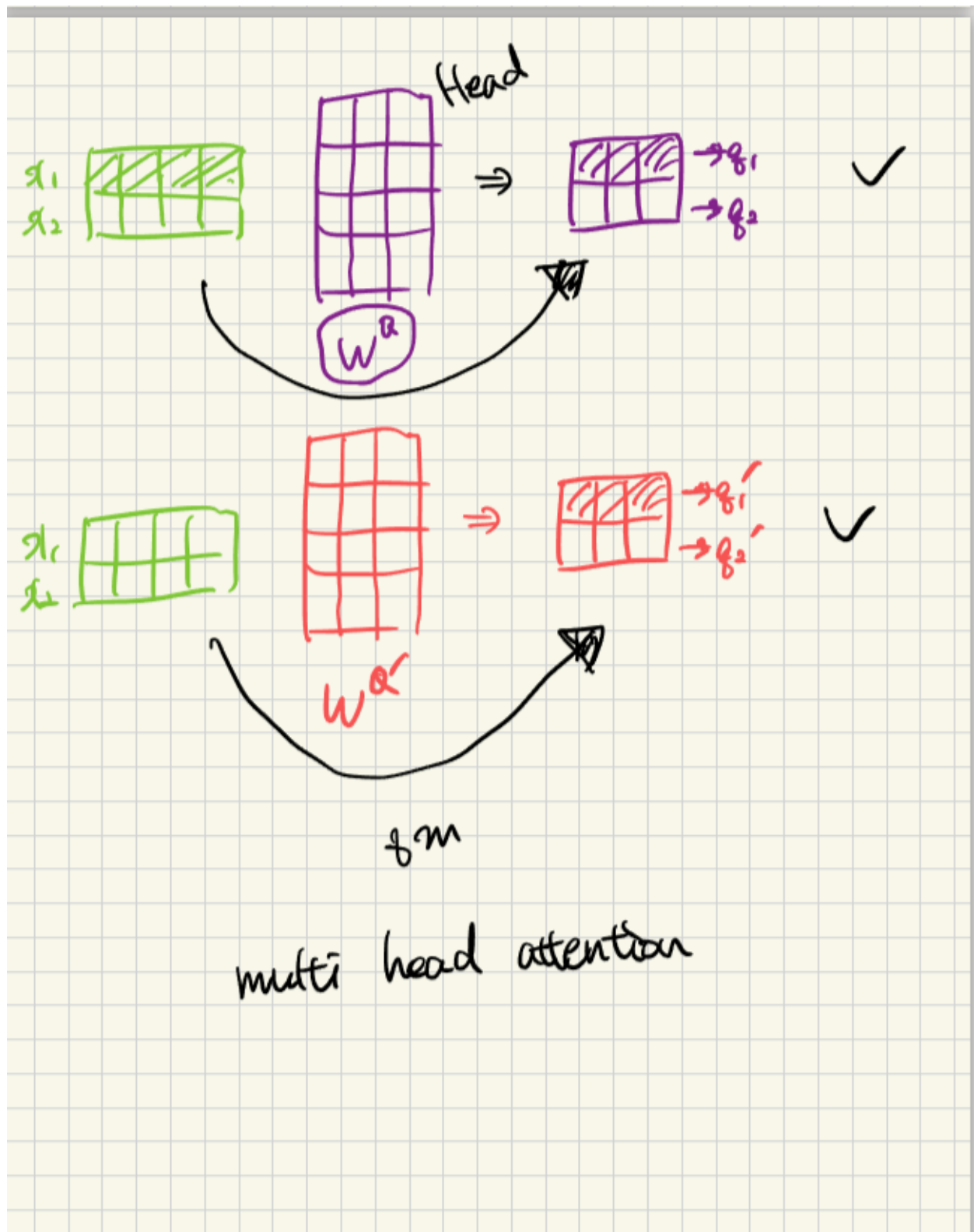




아래그림

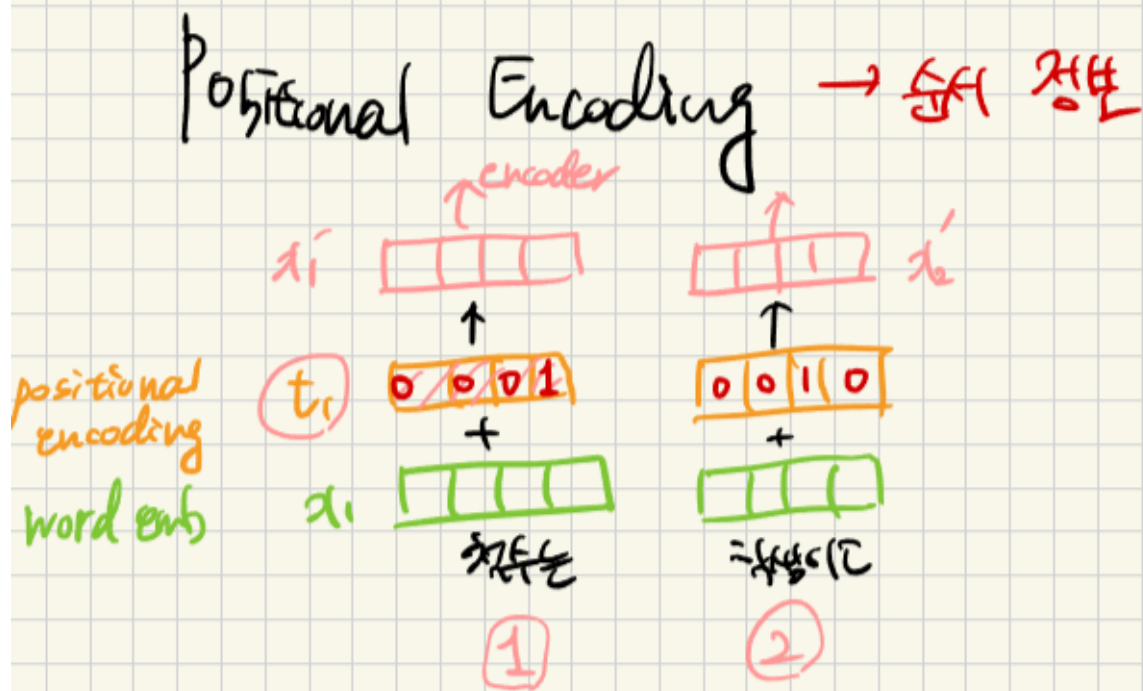
multi head attention

기존 가중치 W_q 대신 새로운 가중치 W_q' 를 곱해서 새로운 q_1' , q_2' 를 구해서 이를 반복해서 **multihead attention**을 구한다.



아래그림

입력값 x_1 에 positional encoding을 더해서 순서정보를 갖는 X_1' 를 구한다.

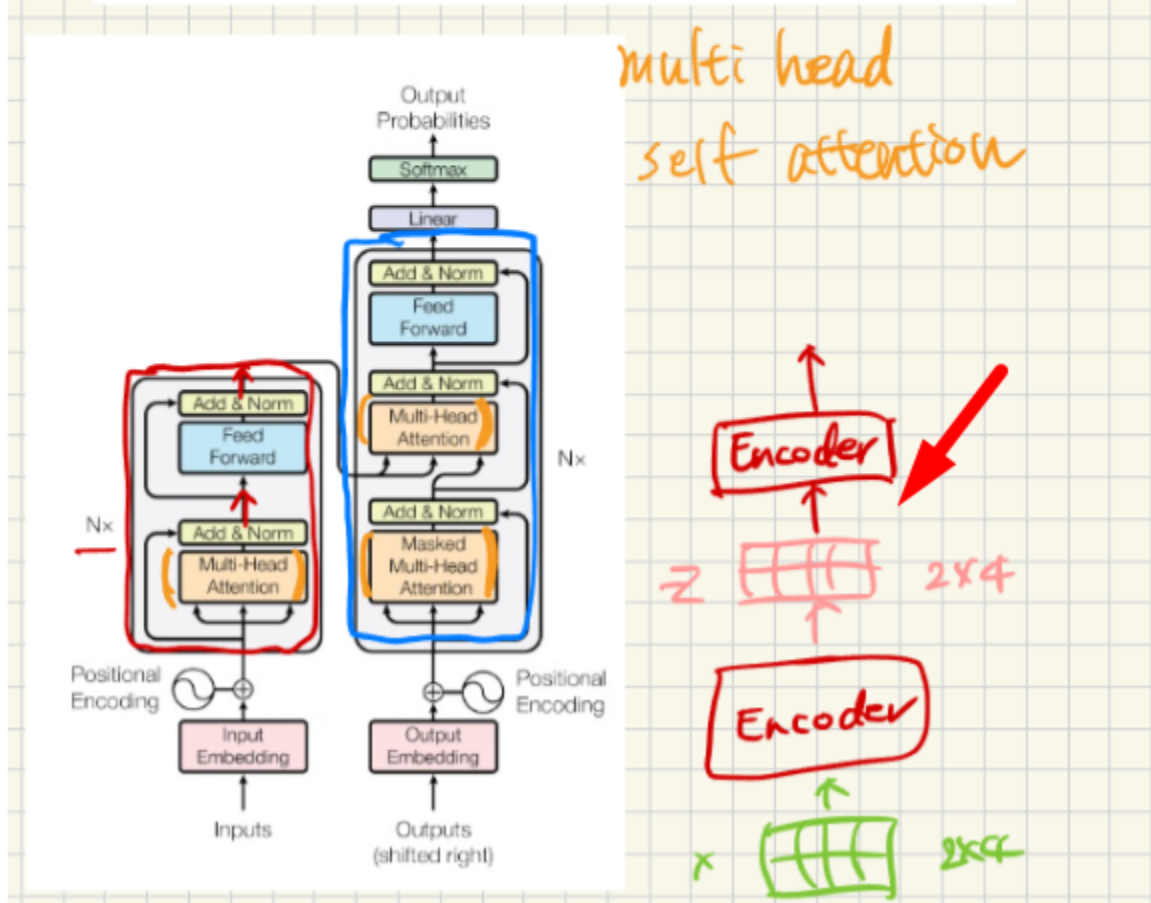
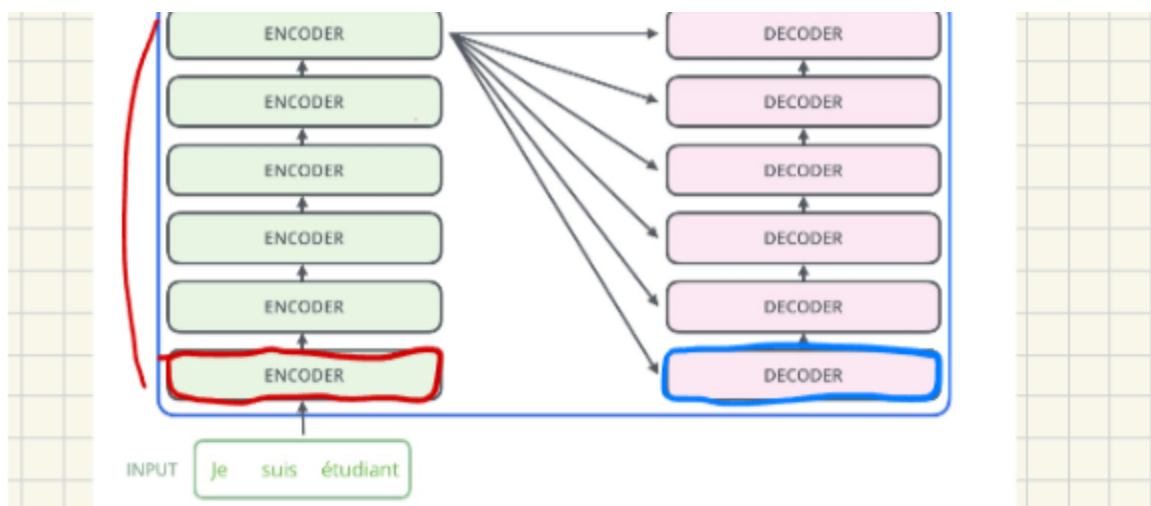


아래그림)

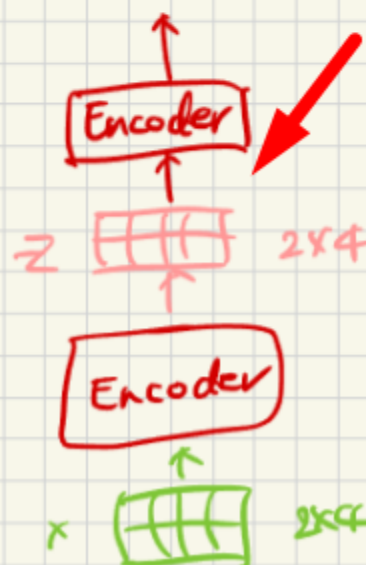
$x(2 \times 4) \rightarrow \text{encoder} \rightarrow z(2 \times 4)$ 해서 size가 encoder지나도 변하지 않음.

그래서 encoder를 여러개 쌓는다(z 를 다시 encoder통과).

decoder는 self attention하는게 똑같은데 뒷쪽단어는 볼수없게 마스킹한다.



multi head
self attention



Transformer 가 뭔지?

- multi head self attention 을 사용한 model
- attention 만큼 이동해서 RNN 대체

왜 나에 나왔는지?

- long term dependency 해결
- 병행 처리 → 처리 시간 ↓
(동시)

• multi head attention 무슨 말!

- Attention head는 여러개 사용
- 여러 관점에서 문장 처리 가능

♡ 1



...

구독하기

'practice 인공지능,머신러닝' 카테고리의 다른 글

교차검증 (0)

2024.10.11

전이학습 (Transfer Learning). (0)

2024.10.11

[하이퍼파라미터 튜닝 \(Hyperparameter tuning\): 학습률, 배치 크기, Grid search](#) (0)

2024.10.05

[Autoencoder](#) (0)

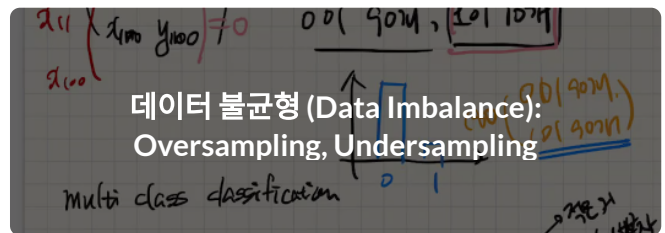
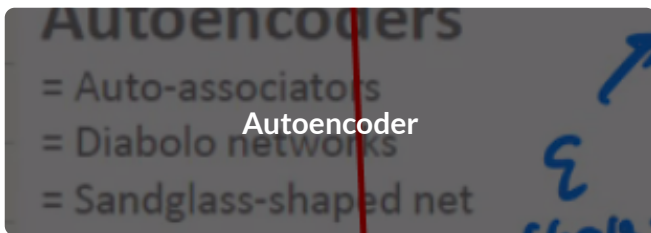
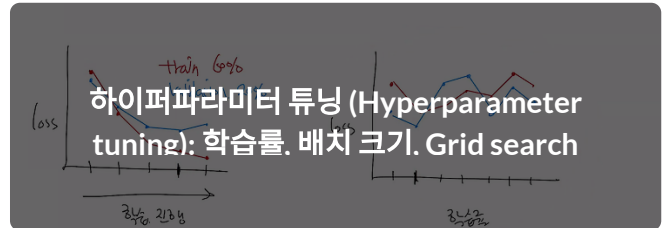
2024.10.04

[데이터 불균형 \(Data Imbalance\): Oversampling, Undersampling](#) (1)

2024.09.28

관련글

[관련글 더보기](#)



자연어(NLP)

네이쳐2024 님의 블로그입니다.

구독하기 +

댓글 1



익명

비밀댓글입니다.

2024. 9. 28. 14:59



이름

비밀번호

내용을 입력하세요.



완료