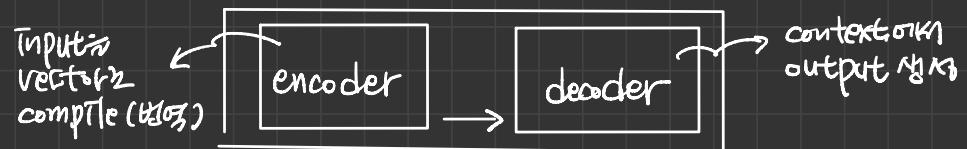
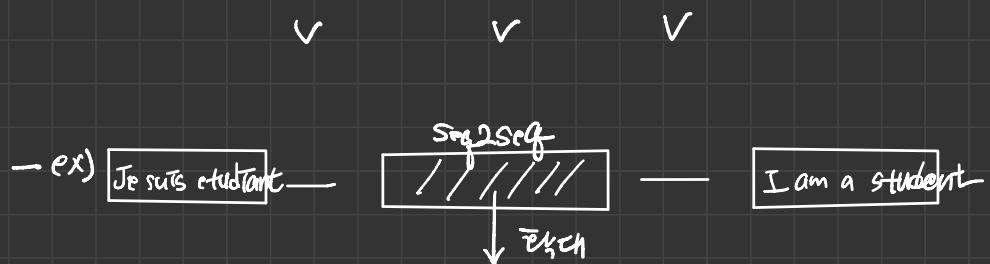


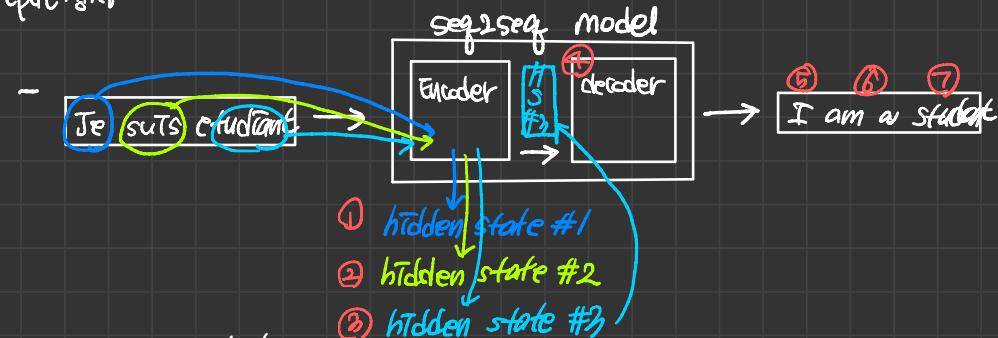
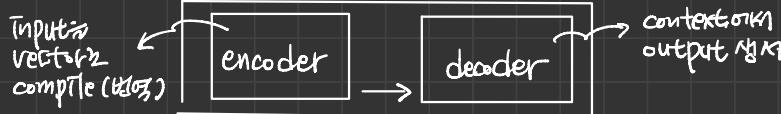
주제 : Transformer를 알기 위한 여정



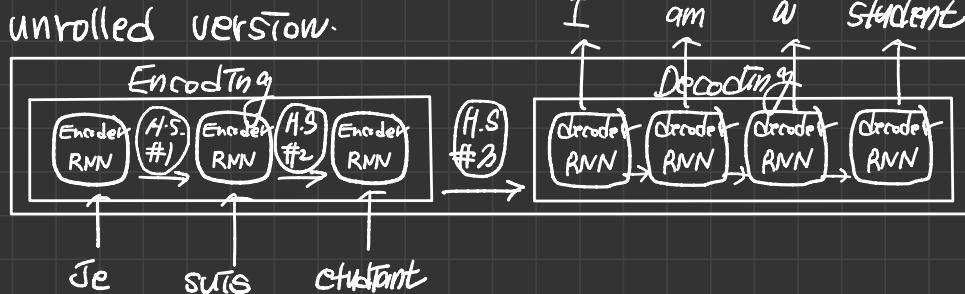
[Seq2Seq]



→ ex) [Je suis étudiant] —  — I am a student
↓
→ 텍스트



- unrolled version.



(이전의 출력을 기반으로 예측)

★ ★ context vector라는 단위로 어떤 유형의 데이터가 문장의 모든 정보를 가지고 있음을 학습할
 가능성이. → DNA와 같은 고정된 데이터의 처리방법을 알았고 반복문은 어떤가?
 (어떤 GRU는 같은 데이터에 대해 다른 맵을 만들기 때문)

[Attention]

→ 현재 입력문장의 정보가

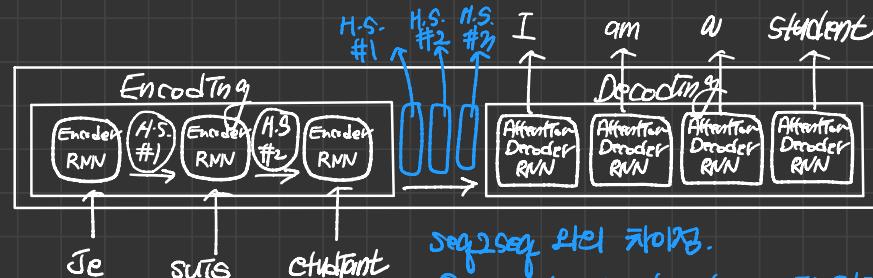
도입된 범위로 카운팅되거나

어디서 해당 사용된다

마지막으로 단어의 뜻과

앞의 단어 단어별로 합쳐

합성해서 알게됨



Sequence 와의 차이점.

- ① encoder가 decoder로 더 많은 데이터를 넘긴다. 다시 말해, *이전의 hidden state* 뿐만 아니라 모든 *hidden state*를 decoder로 넘긴다.
- ② attention decoder의 output 생성하기 전에, 몇 단계는 더 걸친다.

Attention Score

현재 디코더의 사용 단어를 예측하기 위해 인코더의 모든 은닉 상태가 디코더의 초기 시점의 은닉 상태 (*start* 단어) 유사성지를 평균화해 스코어로.

- 디코더는 매번 인코더의 모든 출력 중에서 어떤 정보가 중요한지를 계산합니다.

- i = 현재의 디코더가 처리 중인 인덱스

- j = 각각의 인코더 출력 인덱스

- 에너지(Energy) $e_{ij} = a(s_{i-1}, h_j)$

Softmax
(비율적으로 유사 확률화)

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

디코더가 이전에 출력했던 단어를 만들기 위해 사용했던
hidden state

인코더 각각의 hidden state

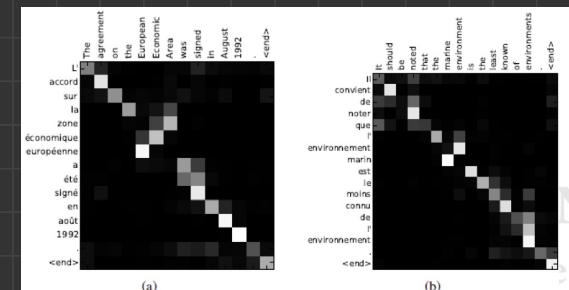
매번 디코더가 출력단어를 만들 때마다

모든 j (인코더 각각의 출력)을 고려하는 것

이전에 디코더가 출력했던 정보 s_{i-1} 를

인코더 모든 출력과 비교해서 에너지값을

(즉, 어떤 정부가 중요한지) 구해겠다..!



[Transformer]

• 높은 원칙: Attention is All you need.

• RNN이나 CNN을 정의 필요로 하지 않는다.

→ RNN은 인과성을 토론 간의 관계를 계산하기 어려운데
어느 허위한 계산성이 때문에 그의 상상력을 무너지
마련을 가능

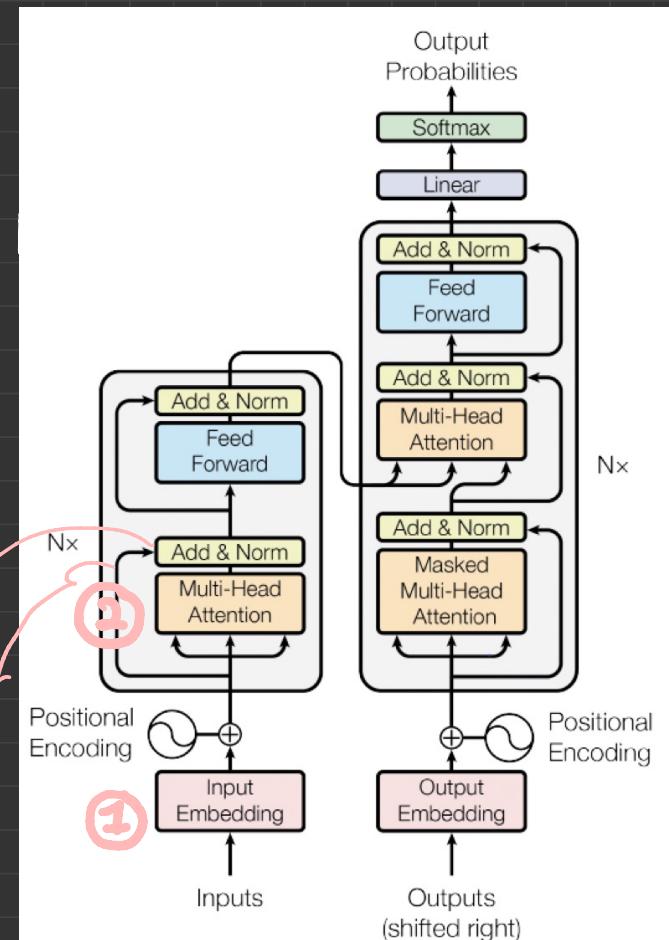
→ 대신 **positional encoding** 사용
(문장 내 각 단어에 따른 순서정보를 알려주기 위함)

• 인코더와 디코더 구조 (Attention 라인을
여기 확인하면서 봄)

(정수화
단어
시도
장르)

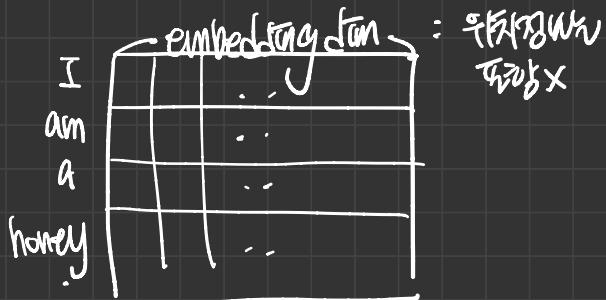
Attention(?)
단어별 관계 분석

(ResNet
장르)



① Input Embedding

- 단어별로 임베딩

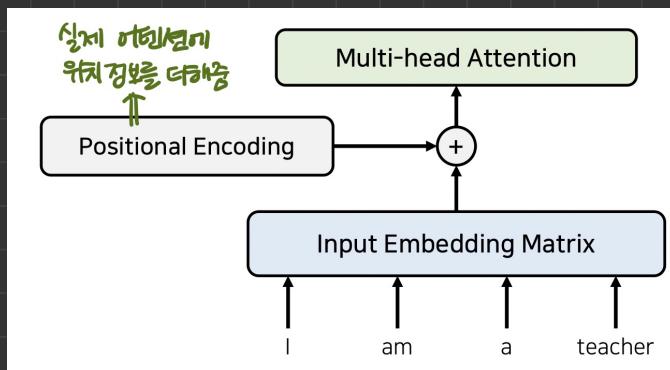


RNN는 시계열에 있는 위치 정보를 표현하는 임베딩 필요
→ transformer에서는 Positional Encoding 사용

Input embedding matrix는
글의 차례를 가지는 벡터로 위치 정보를
가지고 있는 인코딩 정보

② Multi-Head Attention

- 인코더에서 단어 단어간 상호작용을 사용하는데,
각 단어가 다른에게 어떤 영향을 미치고 있는지,
서로에 대한 영향을 스코어로 정하고 있다.



Attention Score

현재 단어의 시점에서 단어를 선택해
위해 인코더의 모든 hidden state 각각이
다른의 현재 위치에 대한 score를
유사한지를 판단하는 스코어입니다