딥러닝/클라우드

Chapter 17

NLP 이론 (2)

오 세 종

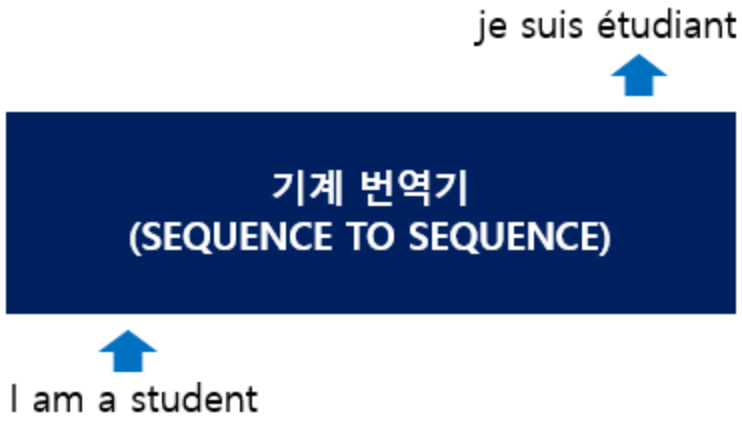
DANKOOK UNIVERSITY

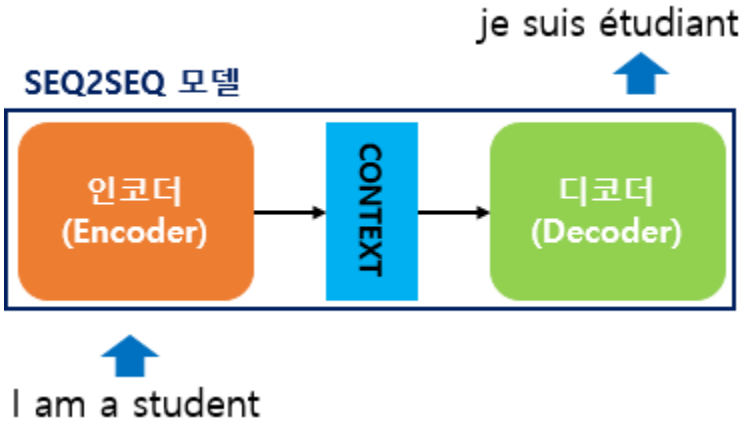
Contents

1. seq2seq
2. Attention mechanism
3. Transformer

1. seq2seq

- 기계 번역에 주로 사용

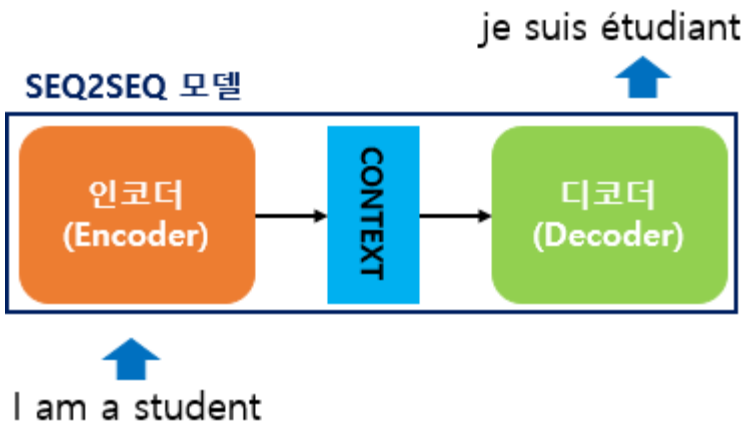


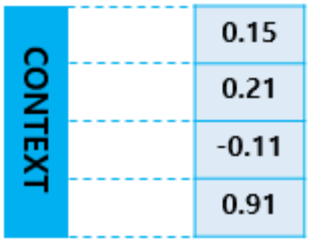


참조: https://wikidocs.net/24996

1. seq2seq

- seq2seq는 크게 인코더와 디코더라는 두 개의 모듈로 구성
- 인코더는 입력 문장의 모든 단어들을 순차적으로 입력받은 뒤에 마지막에 이 모든 단어 정보들을 압축해서 하나의 벡터로 만든다. -> 컨텍스트 벡터 (context vector)
- 입력 문장의 정보가 하나의 컨텍스트 벡터로 모두 압축되면 인코더는 컨텍스트 벡터를 디코더로 전송
- 디코더는 컨텍스트 벡터를 받아서 번역된 단어를 한 개씩 순차적으로 출력

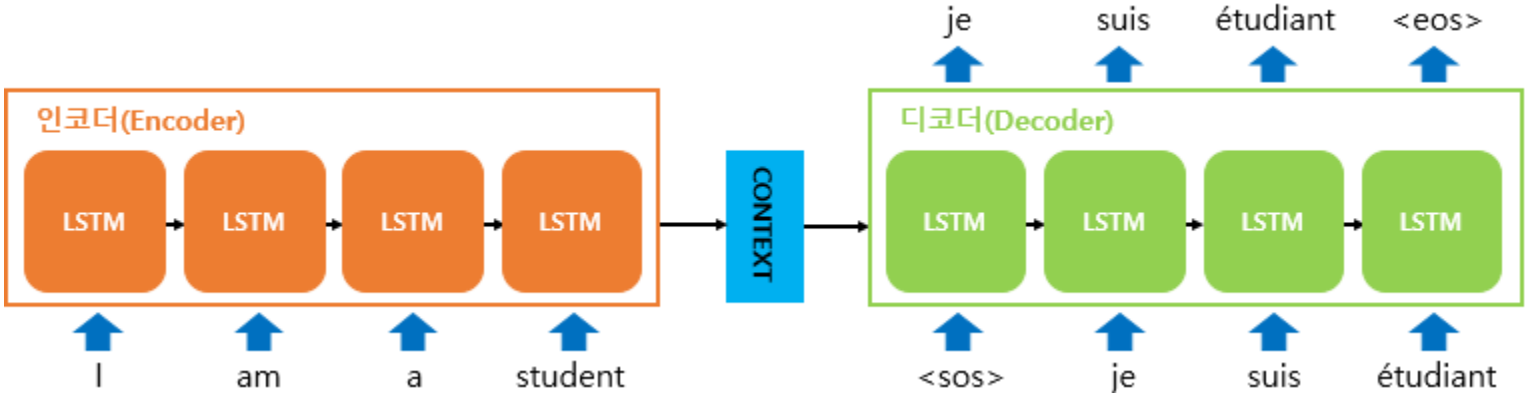




Context vector 의 예

1. seq2seq

- 인코더-디코더의 상세 구조와 예측(test)과정

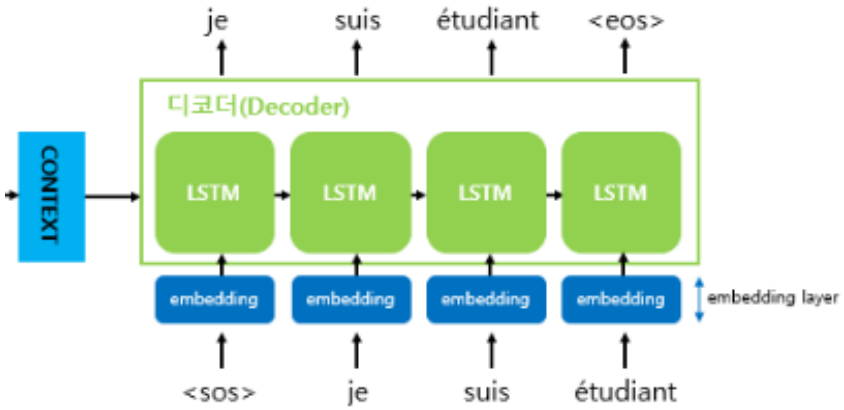


  - 인코더
    - 입력 문장은 단어 토큰화를 통해서 단어 단위로 쪼개지고 단어 토큰 각각은 RNN 셀의 각 시점의 입력
    - 인코더 RNN 셀의 마지막 시점의 은닉 상태를 디코더 RNN 셀로 넘겨주는데 이것이 컨텍스트 벡터
  - 디코더
    - 초기 입력으로 문장의 시작을 의미하는 심볼 <sos>
    - 기본적으로 다음에 올 단어를 예측하고, 그 예측한 단어를 다음 시점의 RNN 셀의 입력으로 넣는 행위를 반복
    - 이 행위는 문장의 끝을 의미하는 심볼인 <eos>가 다음 단어로 예측될 때까지 반복

https://blog.kakaocdn.net/dn/BKXJt/btqCd1I0OTv/DujVTXOZ54SwKomoSpjRCK/img.gif

1. seq2seq

- 디코더의 훈련
  - 디코더에게 인코더가 보낸 컨텍스트 벡터와 실제 정답인 상황인 <sos> je suis étudiant를 입력 받았을 때, je suis étudiant <eos>가 나와야 된다고 정답을 알려주면서 훈련



1. seq2seq

- Bart example

17_Bart_example.py

```
from transformers import BartForConditionalGeneration, BartTokenizer

# load model & tokenizer
model = BartForConditionalGeneration.from_pretrained("facebook/bart-base",
                                                     forced_bos_token_id=0)
tokenizer = BartTokenizer.from_pretrained("facebook/bart-base")

# prepare input
example_sentence = "UN Chief Says There Is No <mask> in Syria"
inputs = tokenizer(example_sentence, return_tensors="pt")
print(inputs)

# generate output
generated = model.generate(inputs["input_ids"],max_new_tokens=50)
print(generated)
outputs = tokenizer.decode(generated[0], skip_special_tokens=True)
print(outputs)
```

- Transformers 패키지 설치 필요
- BART : seq2seq 모델의 일종

1. seq2seq

```
model.safetensors: 100%|████████████████████████████████████████| 558M/558M [00:48<00:00, 11.6MB/s]
>>>
>>> tokenizer = BartTokenizer.from_pretrained("facebook/bart-base")
vocab.json: 100%|████████████████████████████████████████| 899k/899k [00:00<00:00, 2.29MB/s]
merges.txt: 100%|████████████████████████████████████████| 456k/456k [00:00<00:00, 5.26MB/s]
tokenizer.json: 100%|████████████████████████████████████████| 1.36M/1.36M [00:00<00:00, 5.13MB/s]
```

```
>>> print(inputs)
{'input_ids': tensor([[    0,  4154,  1231, 15674,   345,  1534,   440, 50264,    11,  1854,
             2]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])}
>>>
>>> generated = model.generate(inputs["input_ids"],max_new_tokens=50)
>>> print(generated)
tensor([[    2,     0,  4154,  1231, 15674,   345,  1534,   440,  1771,    11,
          1854,     2]])
>>> outputs = tokenizer.decode(generated[0], skip_special_tokens=True)
>>> print(outputs)
UN Chief Says There Is No War in Syria
```

2. Attention Mechanism

- seq2seq 모델은 인코더에서 입력 시퀀스를 컨텍스트 벡터라는 하나의 고정된 크기의 벡터 표현으로 압축하고, 디코더는 이 컨텍스트 벡터를 통해서 출력 시퀀스를 생성

- RNN에 기반한 seq2seq 모델의 문제
  - 1) 하나의 고정된 크기의 벡터에 모든 정보를 압축하려고 하다보니 정보 손실이 발생
  - 2) RNN의 고질적인 문제인 기울기 소실(vanishing gradient) 문제가 존재

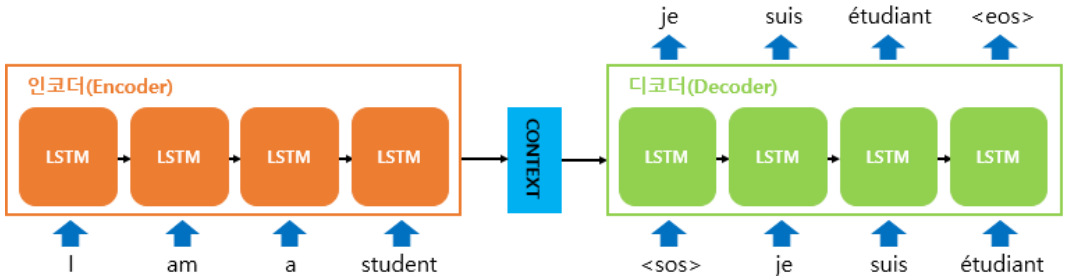
- 이에 대한 대안으로 Attention 등장
  - 기본 아이디어는 디코더에서 출력 단어를 예측하는 매 시점(time step)마다, 인코더에서의 전체 입력 문장을 다시 한 번 참고
  - 단, 전체 입력 문장을 전부 다 동일한 비율로 참고하는 것이 아니라, 해당 시점에서 예측해야 할 단어와 연관이 있는 입력 단어 부분을 좀 더 집중(attention)해서 본다.

참고: https://glee1228.tistory.com/3

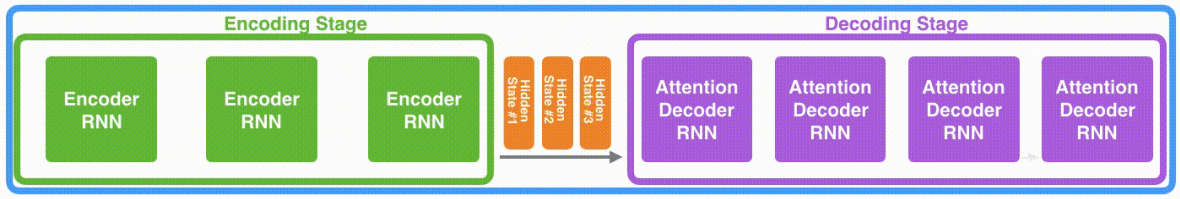
2. Attention Mechanism

- Attention의 기본 idea
  - 디코더에서 출력 단어를 예측하는 매 시점(time step)마다, 인코더에서의 전체 입력 문장을 다시 한 번 참고
  - 단, 전체 입력 문장을 전부 다 동일한 비율로 참고하는 것이 아니라, 해당 시점에서 예측해야 할 단어와 연관이 있는 입력 단어 부분을 좀 더 집중(attention)해서 본다.

seq-2seq



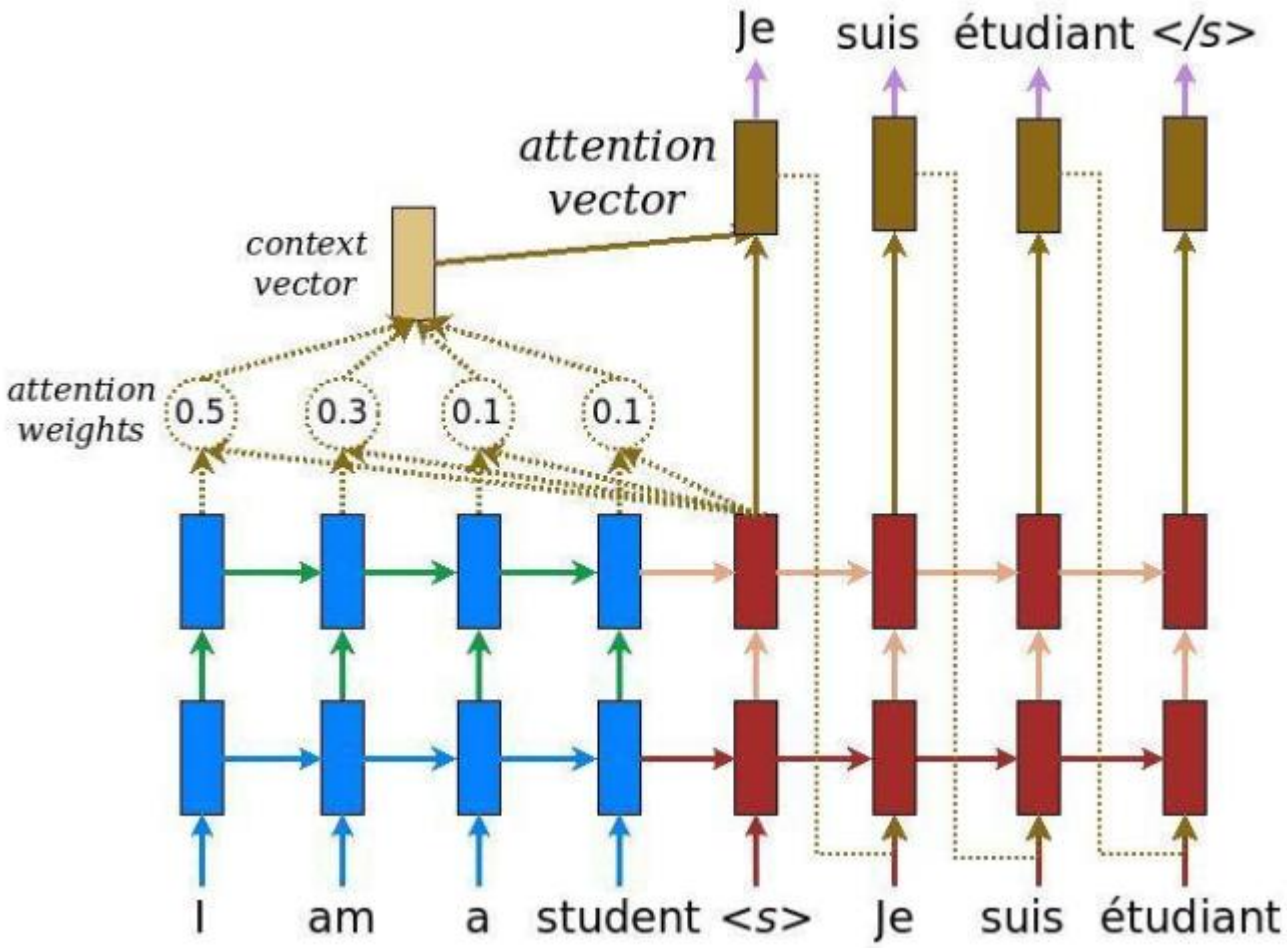
Attention



https://blog.kakaocdn.net/dn/bkL0yX/btqCdgGnHVl/9PYdhz6L6xspMpke7zjBHk/img.gif

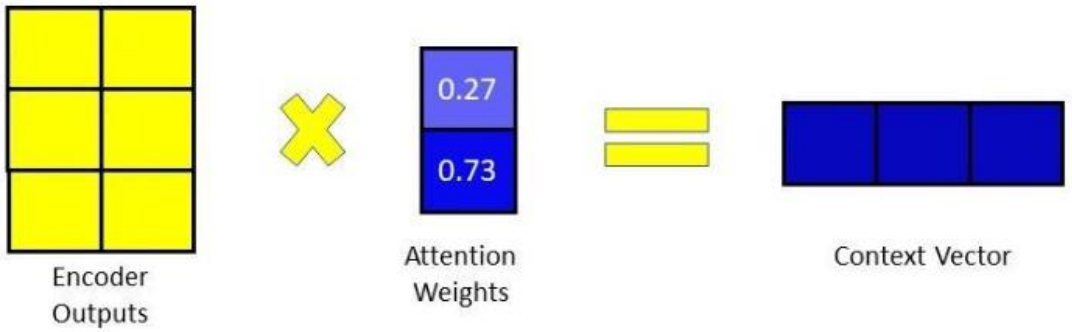
2. Attention Mechanism

- 집중할 단어를 찾는 방법



2. Attention Mechanism

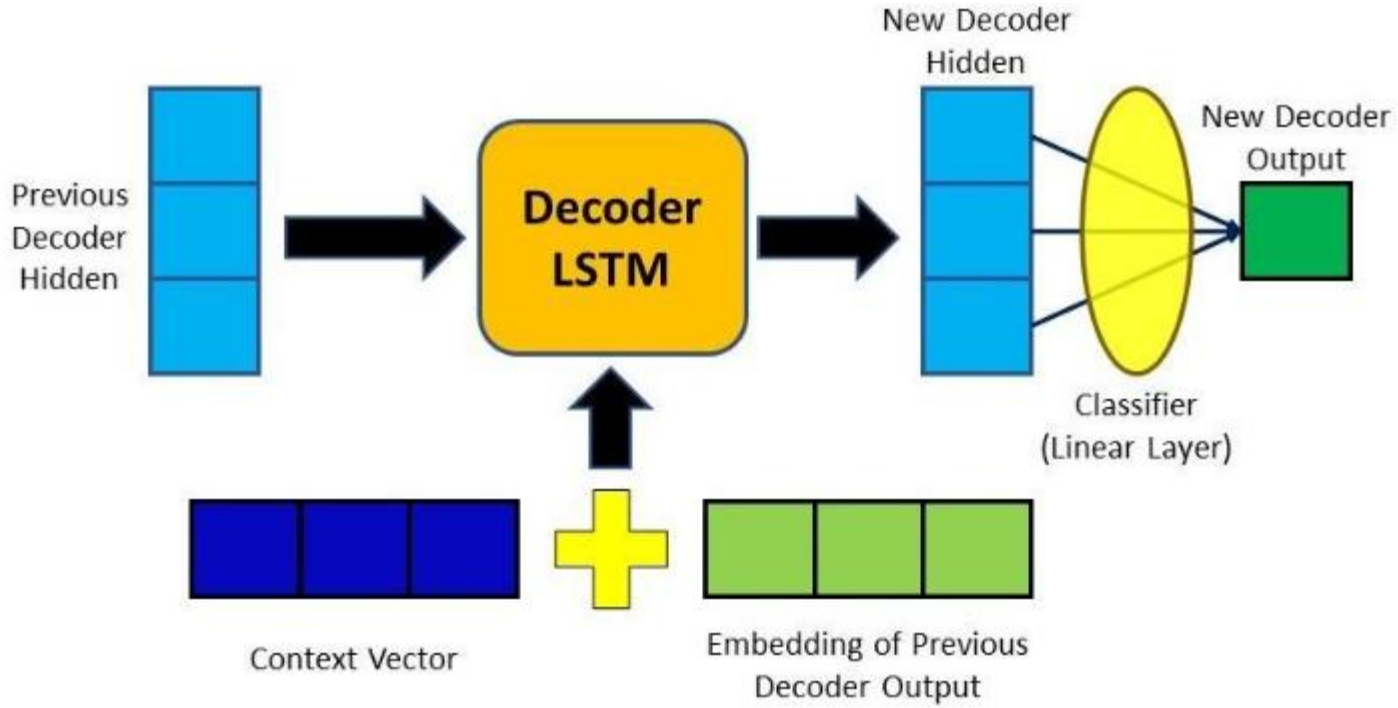
- Attention weight (score)의 계산
  - 디코더의 현재 hidden state 와 인코더의 모든 hidden states 간의 유사성을 계산
  - 내적(dot product)이나 다른 유사도 척도를 사용
  - 이렇게 계산된 attention score 에 softmax 함수를 적용하여 0과 1사이로 정규화 (모든 가중치의 합은 1)
  - 정규화된 attention weight 와 인코더의 hidden states를 사용하여 가중 평균 계산 -> 문맥 벡터(context vector) 생성
    - Seq2seq의 문맥 벡터는 인코더의 마지막 hidden state
    - Attention 의 문맥 벡터는 디코더의 현재 시점과 가장 관련이 있는 hidden state 반영



https://brunch.co.kr/@harryban0917/279

2. Attention Mechanism

- 새로운 디코더 output 생성



https://brunch.co.kr/@harryban0917/279

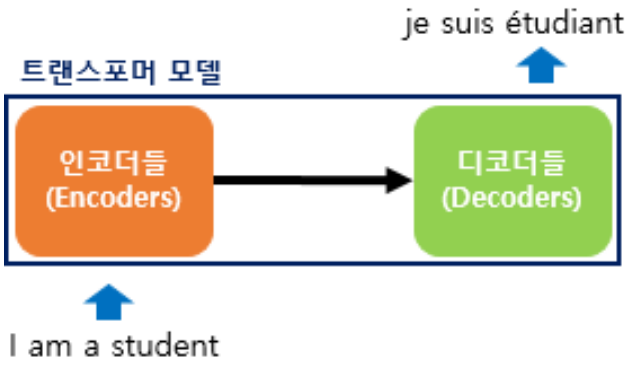
3. Transformer

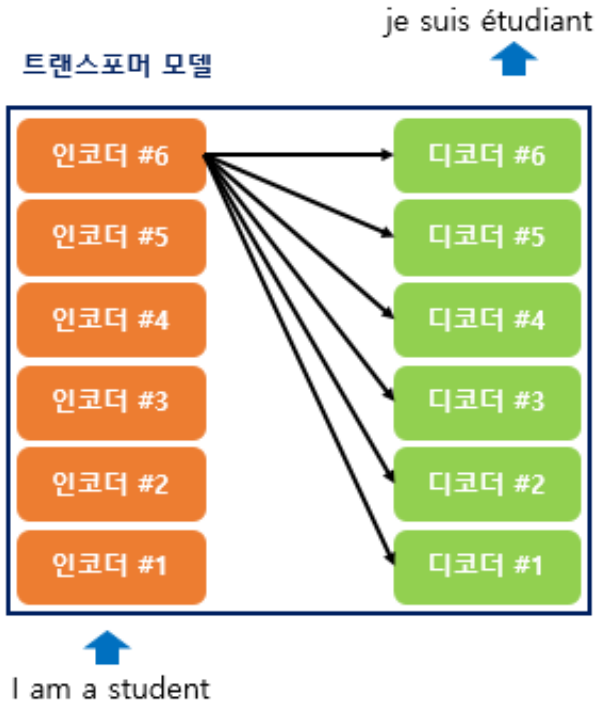
- 개요
  - 구글(Google)의 2017년 논문에 처음 등장
  - 트랜스포머 모델은 문장 속 단어와 같은 순차 데이터 내의 관계를 추적해 맥락과 의미를 학습하는 신경망
  - 어텐션(attention) 또는 셀프어텐션(self-attention)이라 불리며 진화를 거듭하는 수학적 기법을 응용해 서로 떨어져 있는 데이터 요소들의 의미가 관계에 따라 미묘하게 달라지는 부분까지 감지
  - 순차적 텍스트나 이미지, 비디오 데이터를 사용하는 애플리케이션은 무엇이든 트랜스포머 모델이 될 수 있음
  - CNN과 RNN을 이제는 트랜스포머가 대체
  - 트랜스포머가 사용하는 연산은 병렬 프로세싱에 적합하기 때문에 모델의 실행 속도 또한 빨라짐
  - AI의 패러다임 변화를 견인

Ref: https://medium.com/@hugmanskj/transformer%EC%9D%98-%ED%81%B0-%EA%B7%B8%EB%A6%BC-%EC%9D%B4%ED%95%B4-%EA%B8%B0%EC%88%A0%EC%A0%81-%EB%B3%B5%EC%9E%A1%ED%95%A8-%EC%97%86%EC%9D%B4-%ED%95%B5%EC%8B%AC-%EC%95%84%EC%9D%B4%EB%94%94%EC%96%B4-%ED%8C%8C%EC%95%85%ED%95%98%EA%B8%B0-5e182a40459d

3. Transformer

- 어텐션을 RNN의 보정을 위한 용도로서 사용하는 것이 아니라 어텐션만으로 인코더와 디코더를 구성
- 기존의 seq2seq처럼 인코더에서 입력 시퀀스를 입력받고, 디코더에서 출력 시퀀스를 출력하는 인코더-디코더 구조를 유지
- seq2seq 구조에서는 인코더와 디코더에서 각각 하나의 RNN이 t개의 시점(time step)을 가지는 구조였다면 transforme에서는 인코더와 디코더라는 단위가 N개(논문에서는 6개)로 구성되는 구조

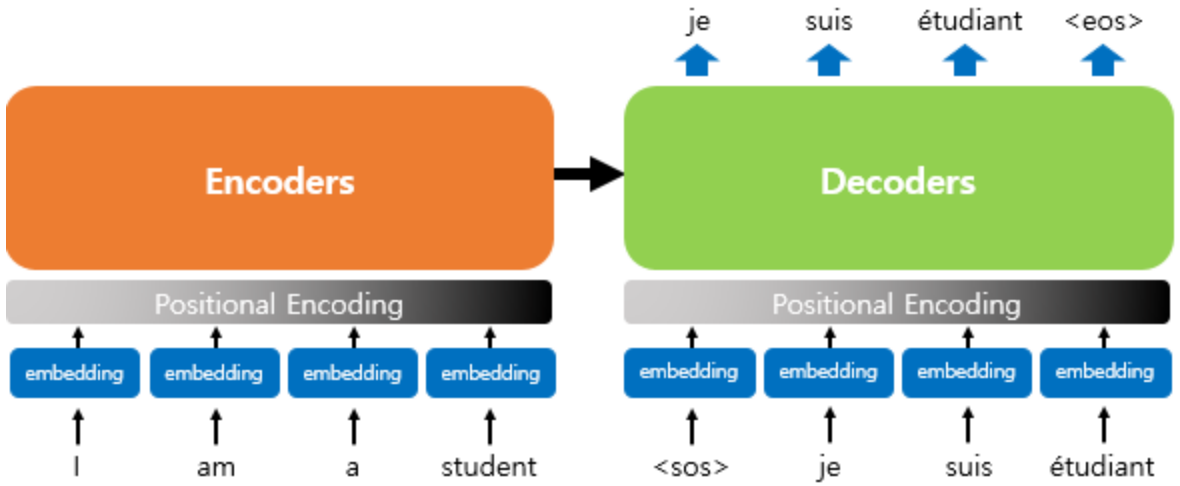




https://wikidocs.net/31379

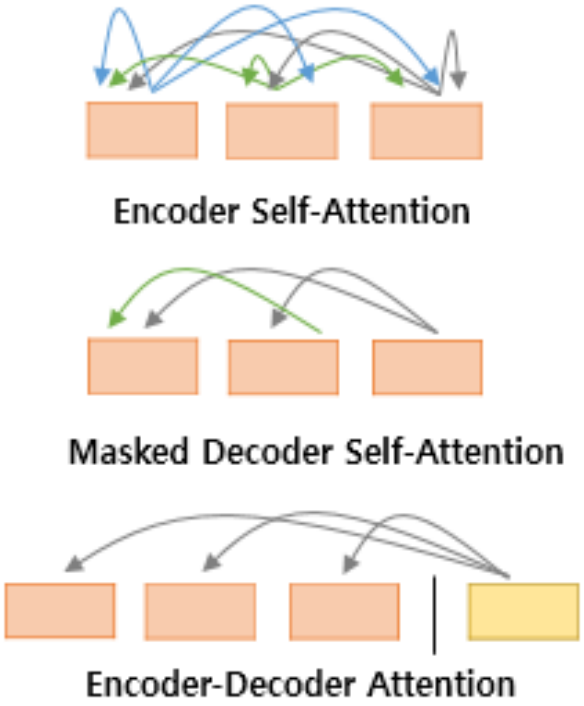
3. Transformer

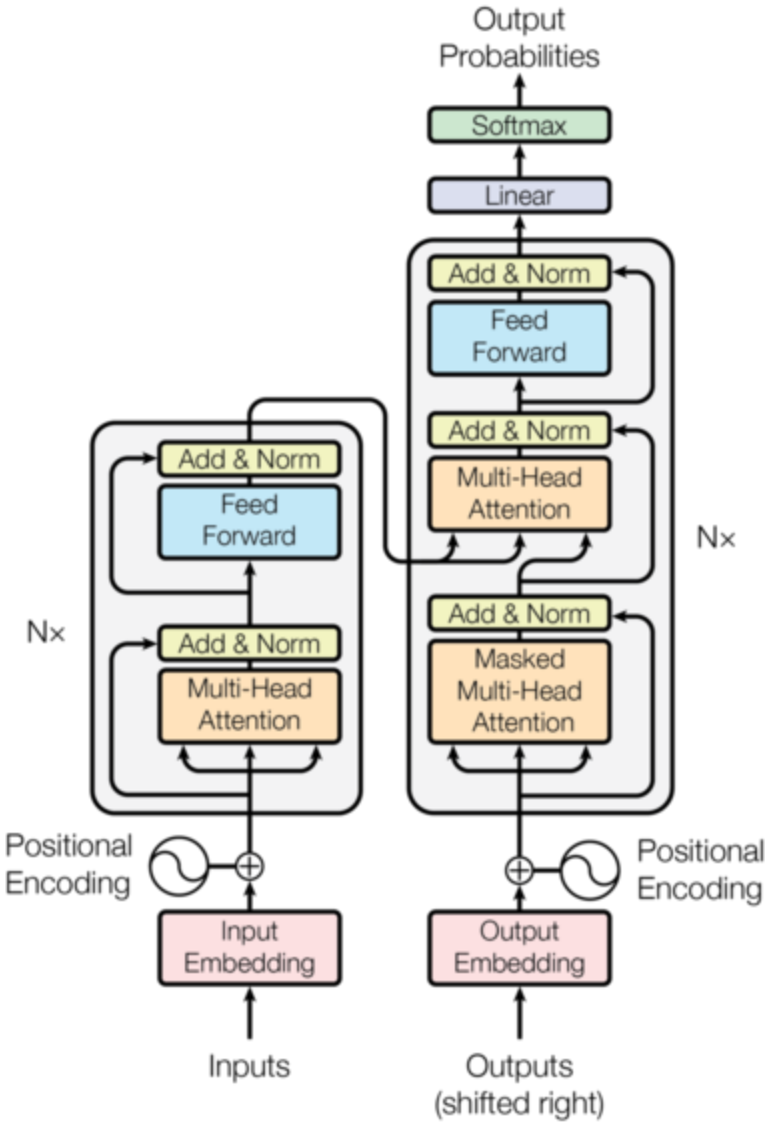
- 트랜스포머는 단어 입력을 순차적으로 받는 방식이 아닌 일괄 입력 방식
  - 단어의 위치(순서) 정보를 알려줄 필요가 있음
  - 트랜스포머는 단어의 위치 정보를 얻기 위해서 각 단어의 임베딩 벡터에 위치 정보들을 더하여 모델의 입력으로 사용하는데, 이를 포지셔널 인코딩(positional encoding)이라고 한다.



3. Transformer

- Attention in transformer





Multi-head = parallel

3. Transformer

- Transformer의 encoder와 decoder의 각 부분은 task에 따라 독립적으로 사용될 수 있다.

- Encoder 전용 모델
  - 응용분야: 문장 분류(sentence classification), 명명된 개체 인식(named entity recognition), 질문 답변(question answering)
  - ALBERT, BERT, DistillBERT, ELECTRA, RoBERTa

- Decoder 모델
  - 응용분야: 텍스트 생성(text generation)
  - GPT, GPT-2, Transformer XL

- Encoder-Decoder 모델
  - 응용분야: 번역(translation) 또는 요약(summarization)
  - BART, mBART, Marian, T5

[Note]

- LLM 관련 platform