

TRANSFORMER MODELS

in NLP

조현국





목차

Contents

#1, Introduce PLMs

#2, NLP 기법

#3, Attention

#4, TRANSFOMER

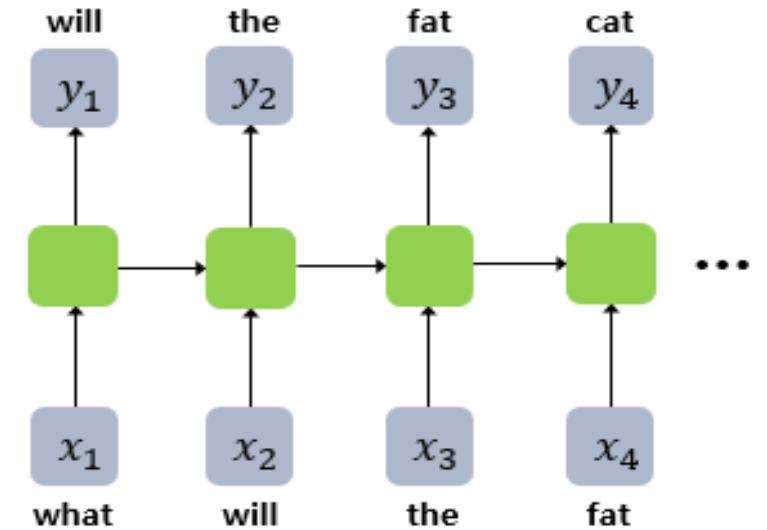
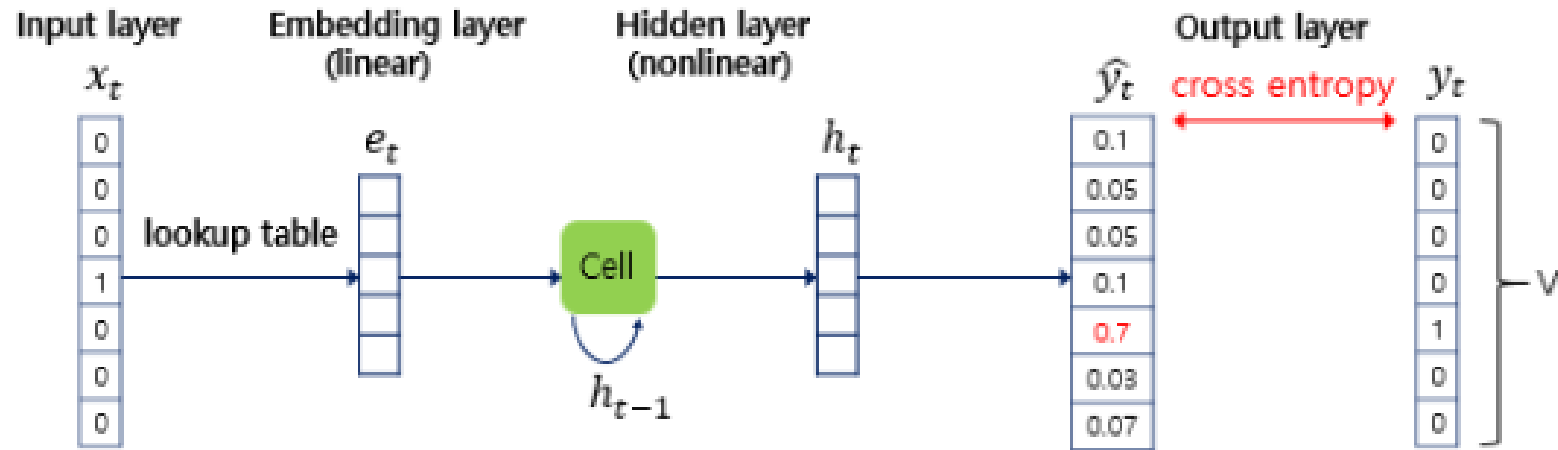


Part 1,

Introduce PLMs



RNN



- Recurrent Neural Network Language Model의 약자
- 시점(time step)의 개념이 도입
- 출력층 => SoftMax 함수
- 손실함수 => Cross-Entropy

PLM(Pretrained Language Model)

PLMs

(Pretrained
Language Models)

BERT

GPT-3

BART

RoBERTa

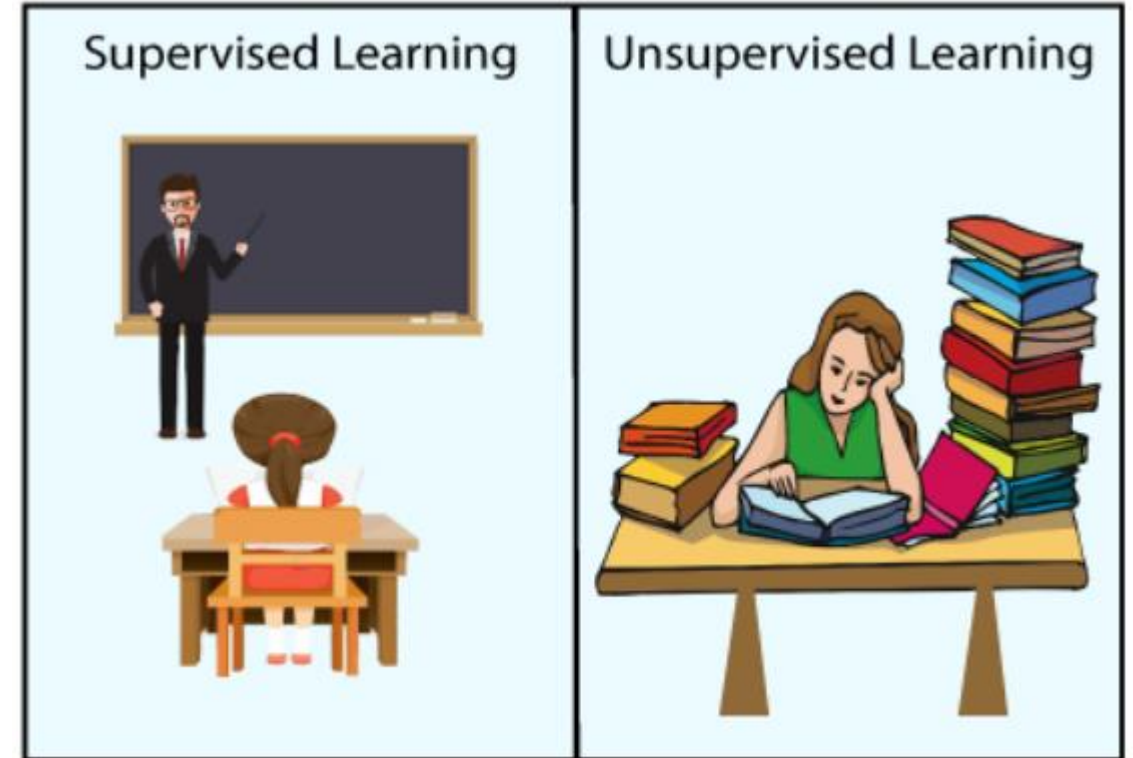
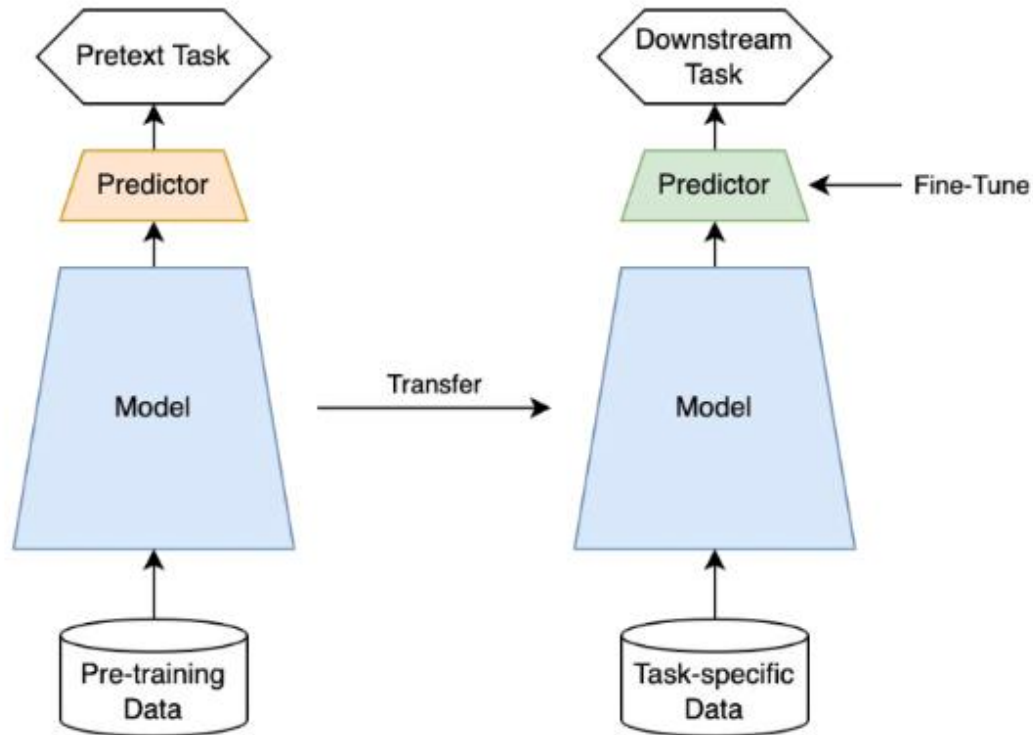
...

- 대규모의 데이터를 사전학습 (Pretrain)한 모델

-Fine-Tuning을 하여 보다 적은 양의 데이터로 높은 성능을 얻을 수 있다.

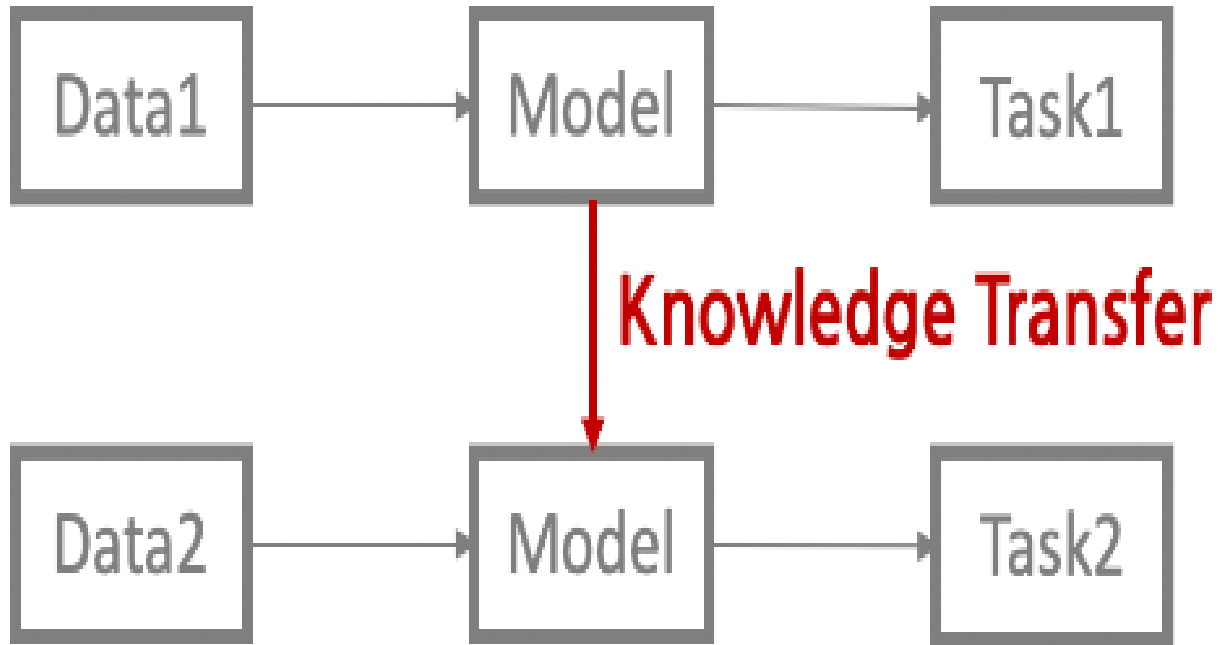
-대표적인 모델 GPT, BERT

Self-Supervised Learning



데이터 내부의 구조(Inner Structure)를 활용하여 Label(정답)이 있는 것 처럼 학습 샘플의 일부 정보를 활용해 나머지 정보를 Label (정답) 로 삼아 예측하도록 학습 이 과정에서 나온 모델을 다른 작업(Task)에 Transfer-Learning해 성능을 극대화

Transfer-Learning



- 특정 Task1을 학습한 모델을 다른 Task2 수행에 재사용

- 모델의 학습 속도가 빨라짐

- 학습 방식

1. Fine-Tuning

Downstream Task Data 전체사용
Downstream Data에 맞게 모델 전체 UPDATE

2. Prompt-Tuning

Downstream Task Data 전체사용
Downstream Data에 맞게 모델 일부 UPDATE

3. In-context Learning (GPT-3)

Downstream Task Data 일부사용
모델 UPDATE 없음

Part 1 UPSTEAM TASK(Pretrain - GPT)

티끌

모아



< 다음 단어 맞추기 >

문맥을 이해해 정답을 맞추는 모델

다음 단어의 정답인 태산이라는 단어의 확률을 높이고 나머지 단어들의 확률을 낮춘다.

Part 1 UPSTEAM TASK(Pretrain - BERT)

티끌  태산



< 빈칸 채우기 >

Masked Language Model(마스크 언어 모델)

빈칸에 들어갈 단어인 '모아'라는 단어의 확률을 높이고 나머지 단어들의 확률을 낮춘다.

DOWNSTREAM TASK(Fine-Tuning)

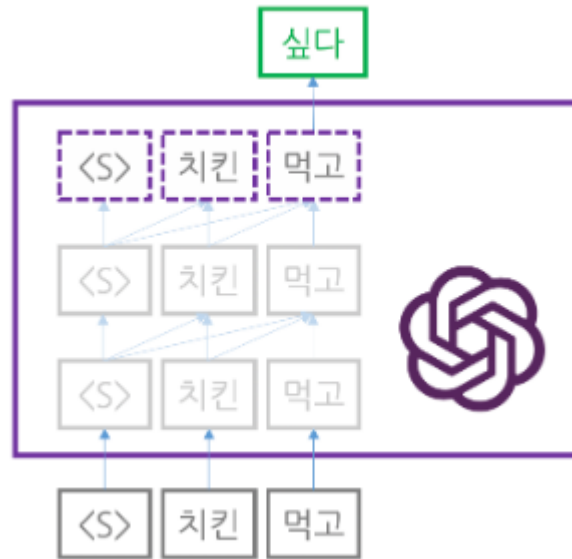
< 문서 분류 (BERT) >



< 개체명 인식 (BERT) >



< 문장 생성 (GPT) >



< 자연어 추론 (BERT) >



< 질의 응답 (BERT) >



Downstream Task의 본질은 분류(classification)

입력 받은 자연어가 어떤 범주에 해당하는지 확률 형태로 반환

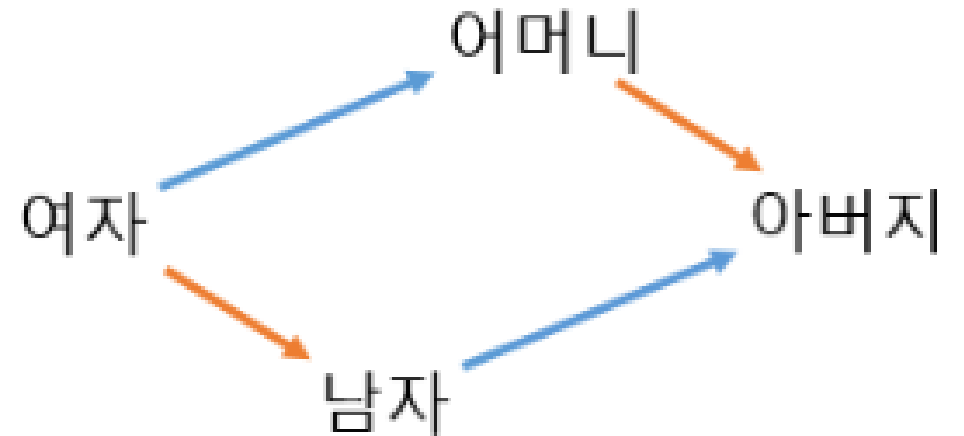
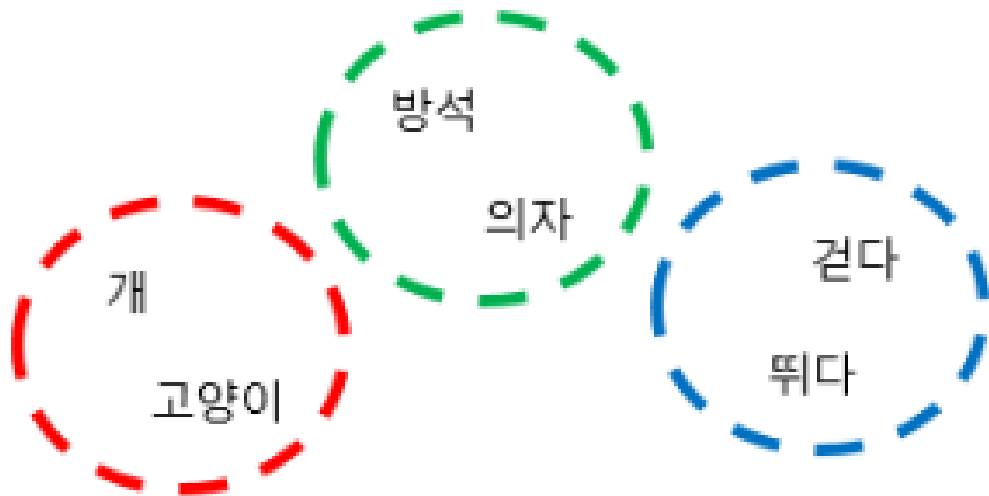
A close-up photograph of a brown leather bag with a strap, resting on a dark blue textured surface. Inside the bag, a black smartphone is placed on top of a brown notebook. A red pen with a silver clip is also visible, resting on the notebook. A pair of black-rimmed glasses is partially visible in the upper right corner of the bag.

Part 2

NLP 기법

Word-Embedding

“유사 의미의 벡터는 가까운 공간에 존재” “단어를 벡터화 하여 산술 연산”



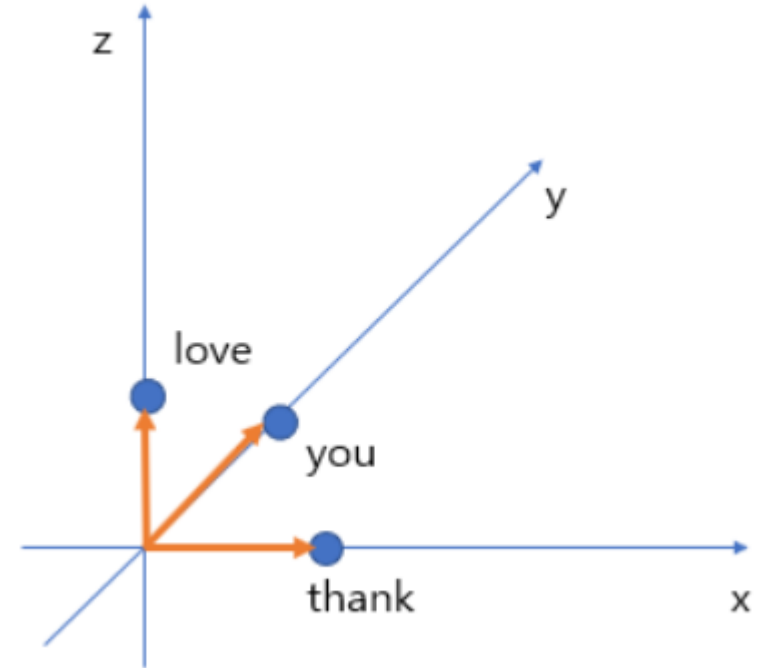
Word를 Dense Vector로 표현하는 방법(주변 단어)

Word2Vec에서 사용

유사도 검사 시 사용

One-Hot-Encoding

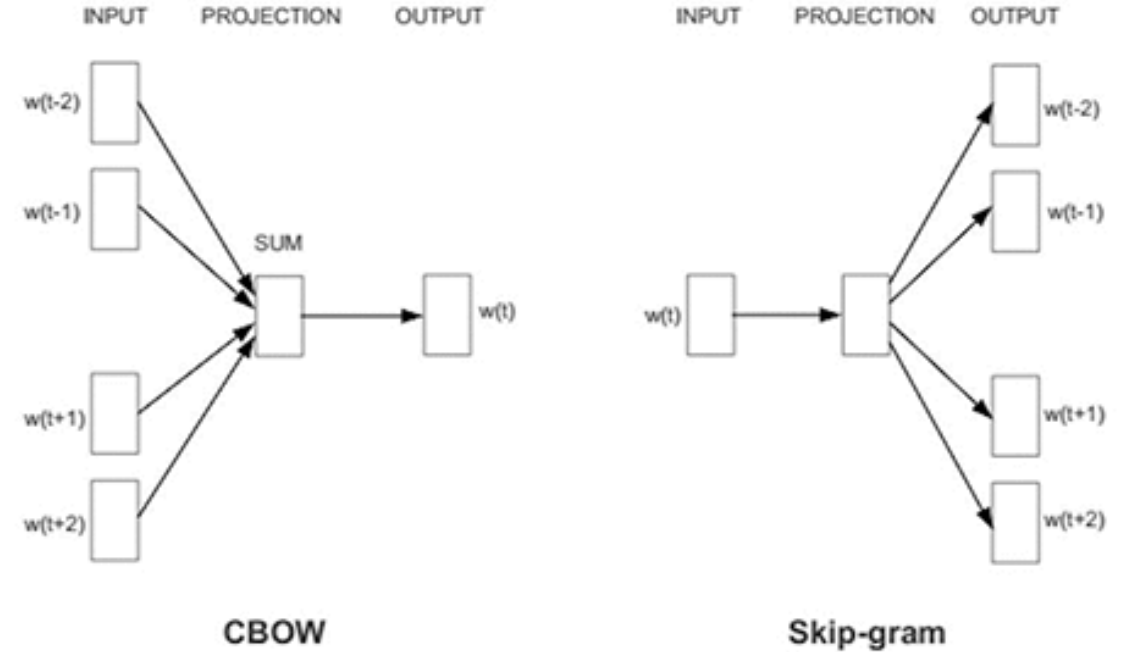
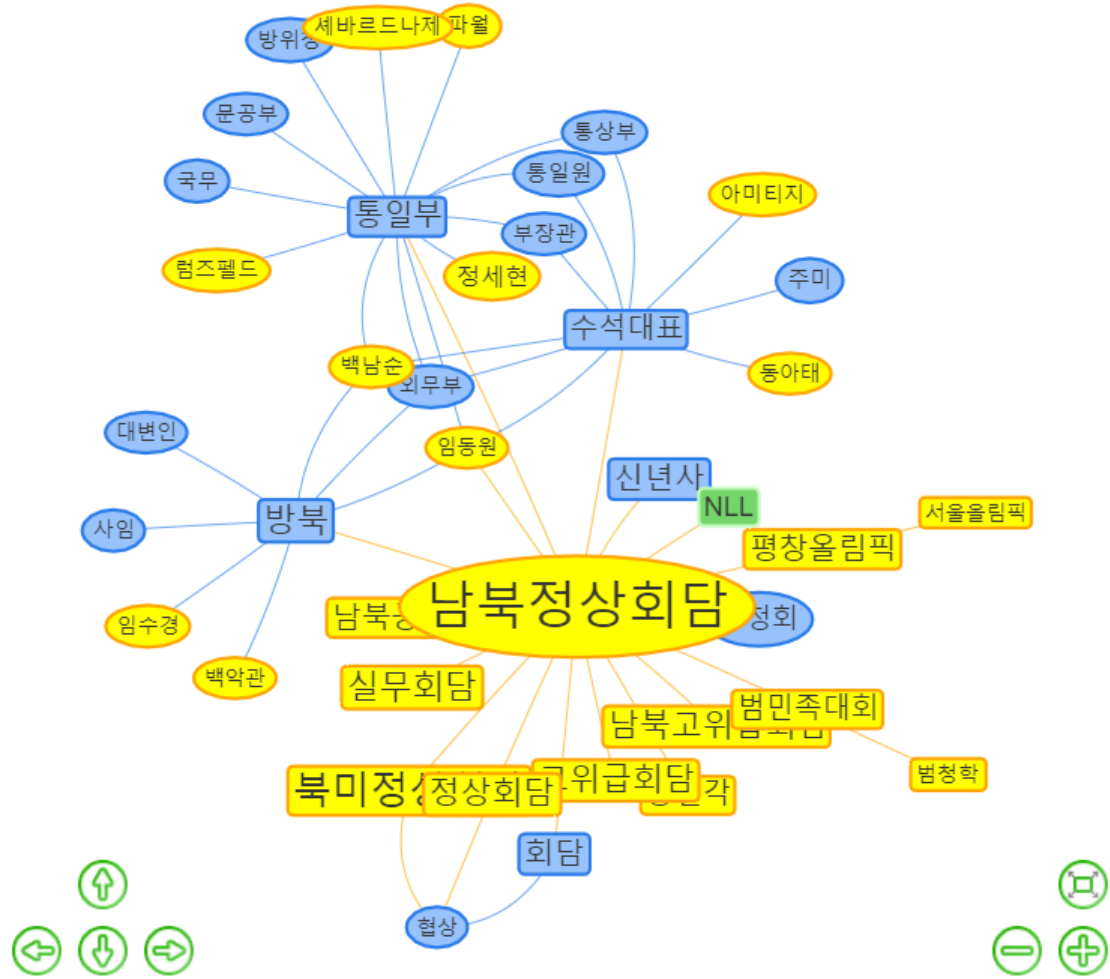
단어	단어 인덱스	원-핫 벡터
you	0	[1, 0, 0, 0, 0, 0, 0]
say	1	[0, 1, 0, 0, 0, 0, 0]
goodbye	2	[0, 0, 1, 0, 0, 0, 0]
and	3	[0, 0, 0, 1, 0, 0, 0]
I	4	[0, 0, 0, 0, 1, 0, 0]
say	5	[0, 0, 0, 0, 0, 1, 0]
hello	6	[0, 0, 0, 0, 0, 0, 1]



표현하고 싶은 단어의 인덱스에 1의 값을 부여하고 그 외 인덱스에 0을 부여하는 방식

부여된 벡터들을 one-hot-vector라 한다.

Word2Vec



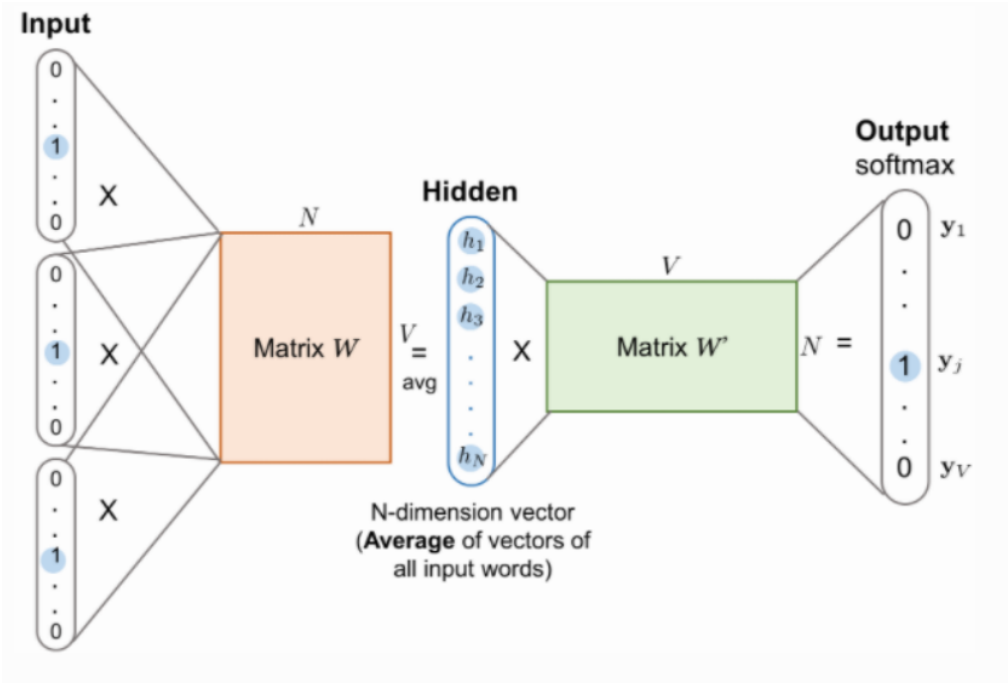
통계 기반 기법으로 단어 빈도수를 기반으로 벡터로 표현

NNLM(Neural Net Language Model) 기반
대량의 문서를 벡터공간에
고수준의 의미를 가진 벡터를 가지도록 하는 모델

CBOW 방식, Skip-gram 방식이 있다.

CBOW

You [?] goodbye and I say hello.
윈도우크기 = 1



중심 단어 주변 단어

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

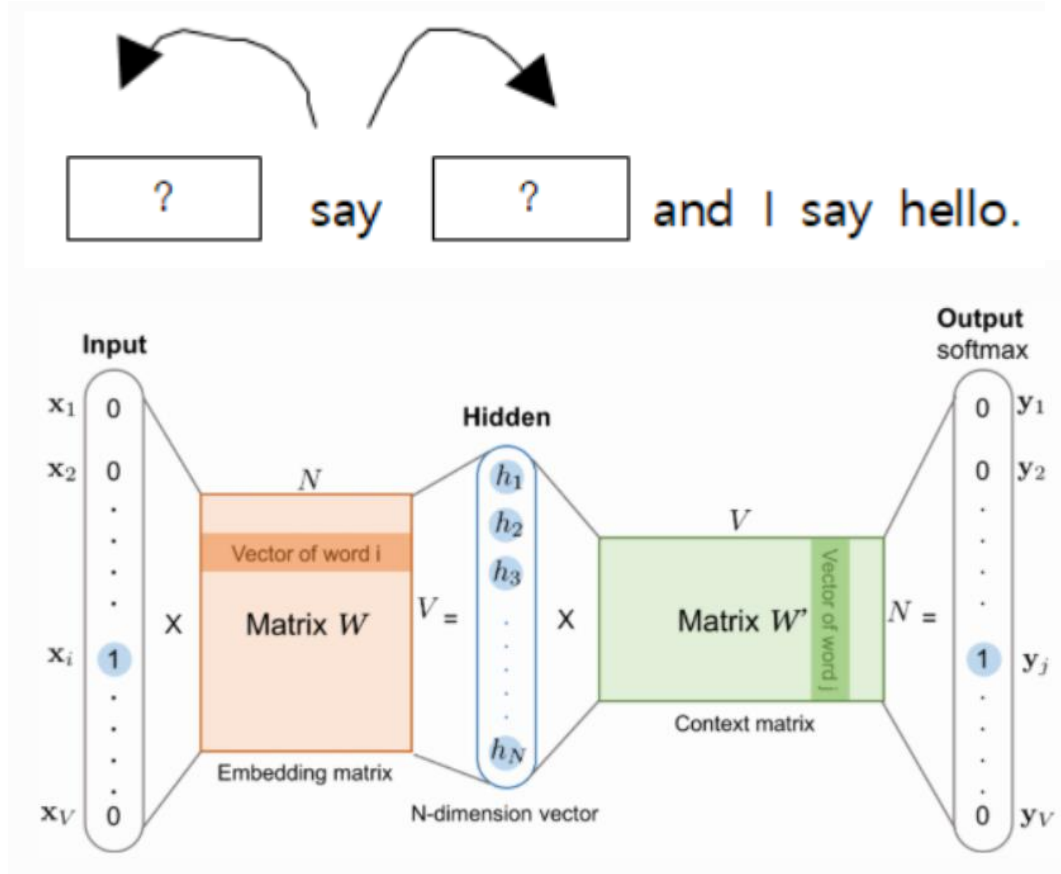
The fat cat sat on the mat

The fat cat sat on the mat

중심 단어	주변 단어
[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 1, 0, 0]	[0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0]	[0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]

주변단어를 기준으로 중심단어를 예측하는 모델

Skip-gram



중심 단어 (Center Word) and 주변 단어 (Context Word) example:

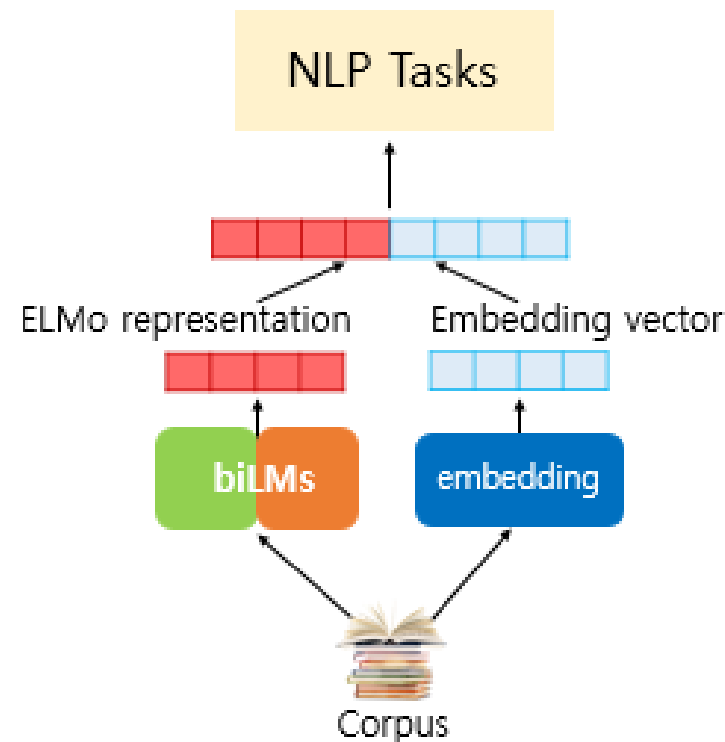
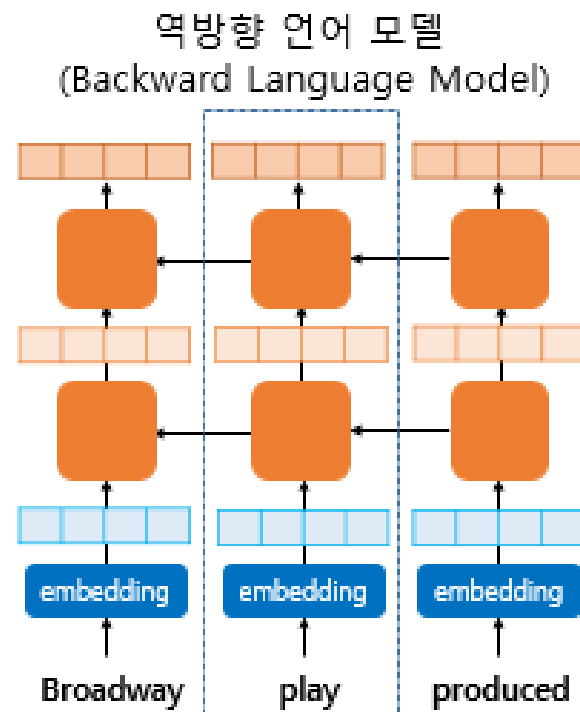
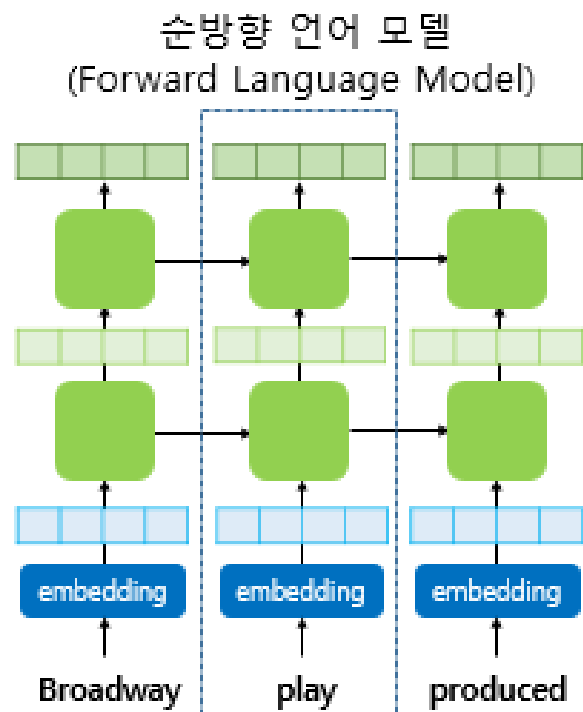
The fat **cat** sat on the mat

The **fat** cat sat on the mat

중심 단어	주변 단어
cat	The
cat	Fat
cat	sat
cat	on
sat	fat
sat	cat
sat	on
sat	the

중심단어를 기준으로 주변단어를 예측하는 모델

ELMo



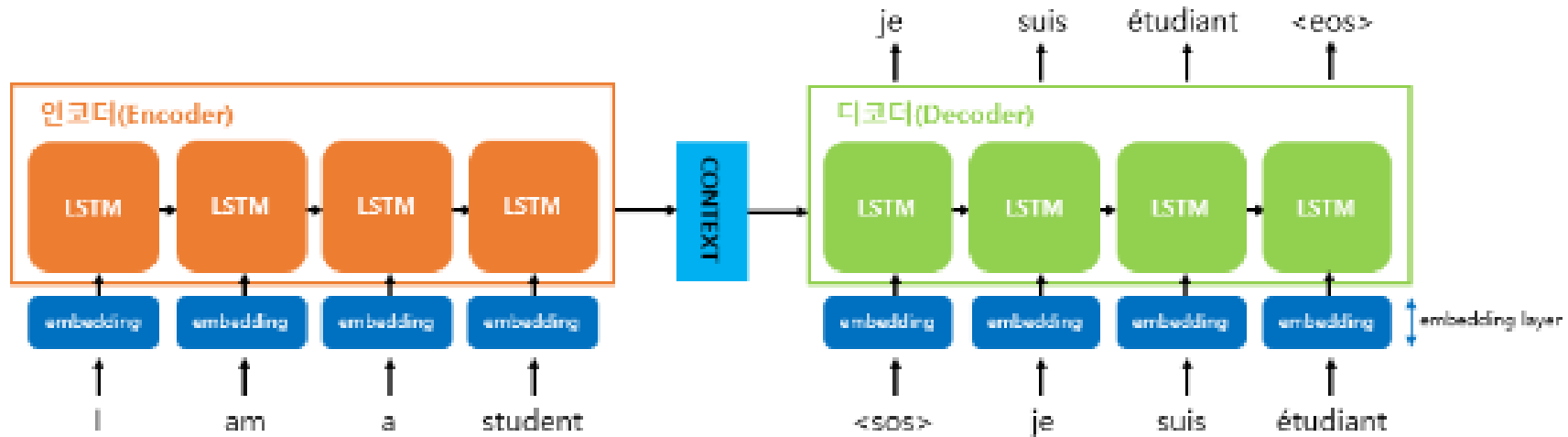
Pre-trained Language Model이며, Word-Embedding 시 문맥을 고려하여 성능을 극대화

Word2Vec이 단어 중심이었다면, ELMo는 문맥 즉, 문장을 고려한다.

PLM과 Word2Vec, ELMo의 차이점

- Word2Vec과 ELMo의 방식
 - Feature를 사전 학습하는 방식
Embedding Layer에 가중치(weight)만 사전학습
(Fine-Tuning을 할 수 없다.)
- 추후 PLM방식
 - Model Parameter 자체를 사전 학습하는 방식
(Fine-Tuning이 가능하다.)

Seq2seq



Encoder 와 Decoder라는 두개의 모듈로 나뉜다.

Encoder

입력 문장의 모든 단어들을 순차적으로 입력 받은 뒤에 모든 단어들을 압축하여 하나의 Vector로 만든다.
(이 Vector를 context vector라 부른다.)

- Attention(Key) => Hidden state

Decoder

Encoder에서 압축된 context vector를 받아 번역된 단어를 하나씩 순차적으로 출력한다.

- Attention(Query) => Hidden State

Seq2seq의 문제점

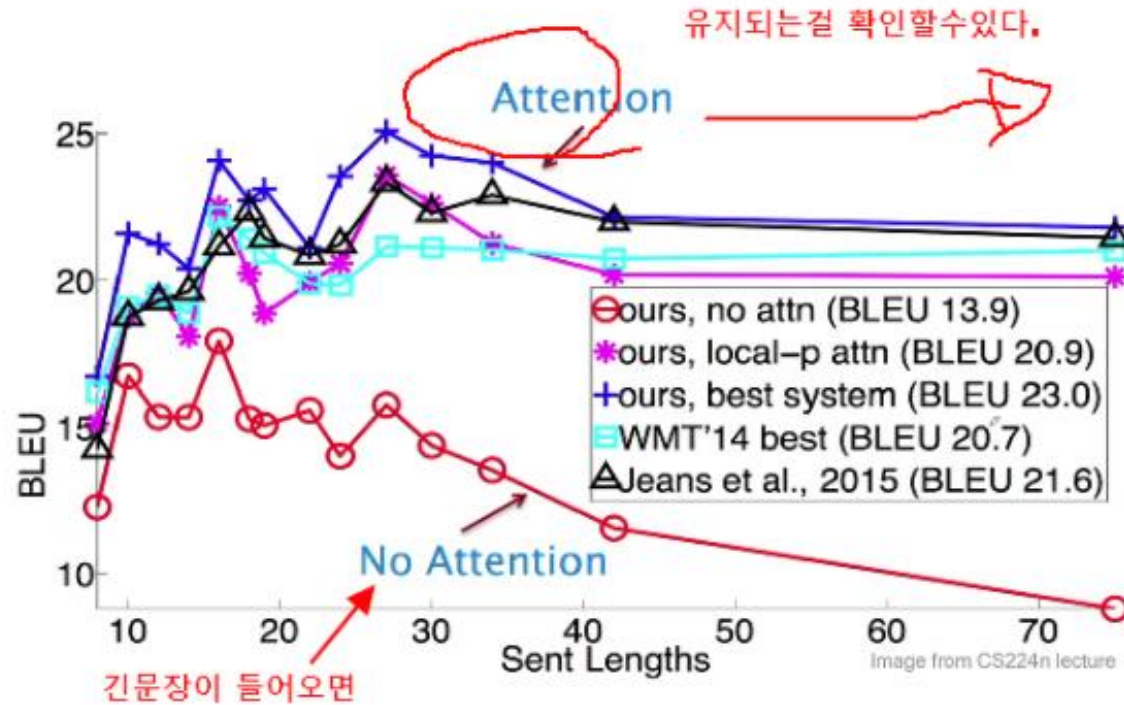
- 하나의 고정된 context vector에 모든 정보를 압축
⇒ 정보 손실 발생
 - RNN의 고질적인 문제
=> Vanishing Gradient(기울기 소실) 발생
- 즉, 번역 및 입력한 문장의 길이가 길어질 수록
번역 품질(성능)이 떨어지는 현상이 발생

A person is working at a desk with a white Apple monitor, a white keyboard, and a white mouse. The person's hands are visible, typing on the keyboard. A blue overlay box is in the foreground, containing the text "Part 3" and "Attention".

Part 3

Attention

Attention Mechanism

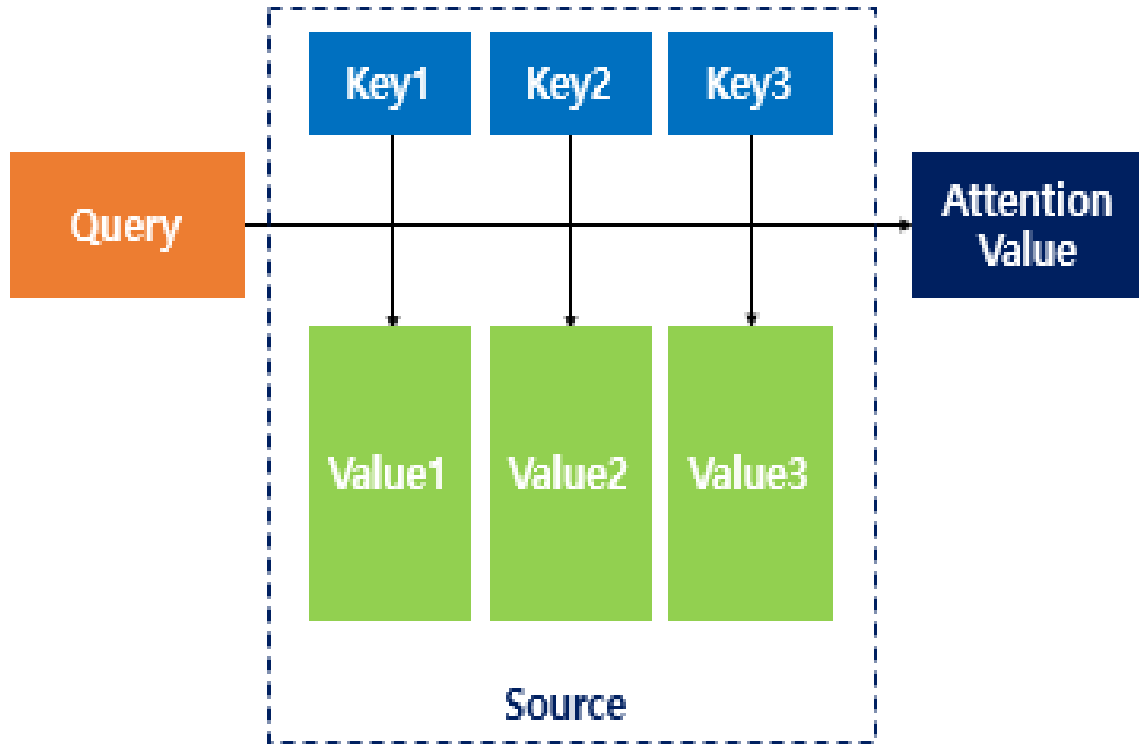


Seq2seq의 문제점을 보완하고 긴 문장의 정확도를 보정하는 기법

Decoder에서 출력할 단어를 예측하는 time-step마다 Encoder에서 입력 받은 시퀀스를 다시 참고한다.

(이때 모든 단어를 참고하지 않고 예측 단어와 관련이 있는 입력단어를 치중해서 본다.)

Attention 입력



Attention 함수의 입력은 Query, Key, Value이다.

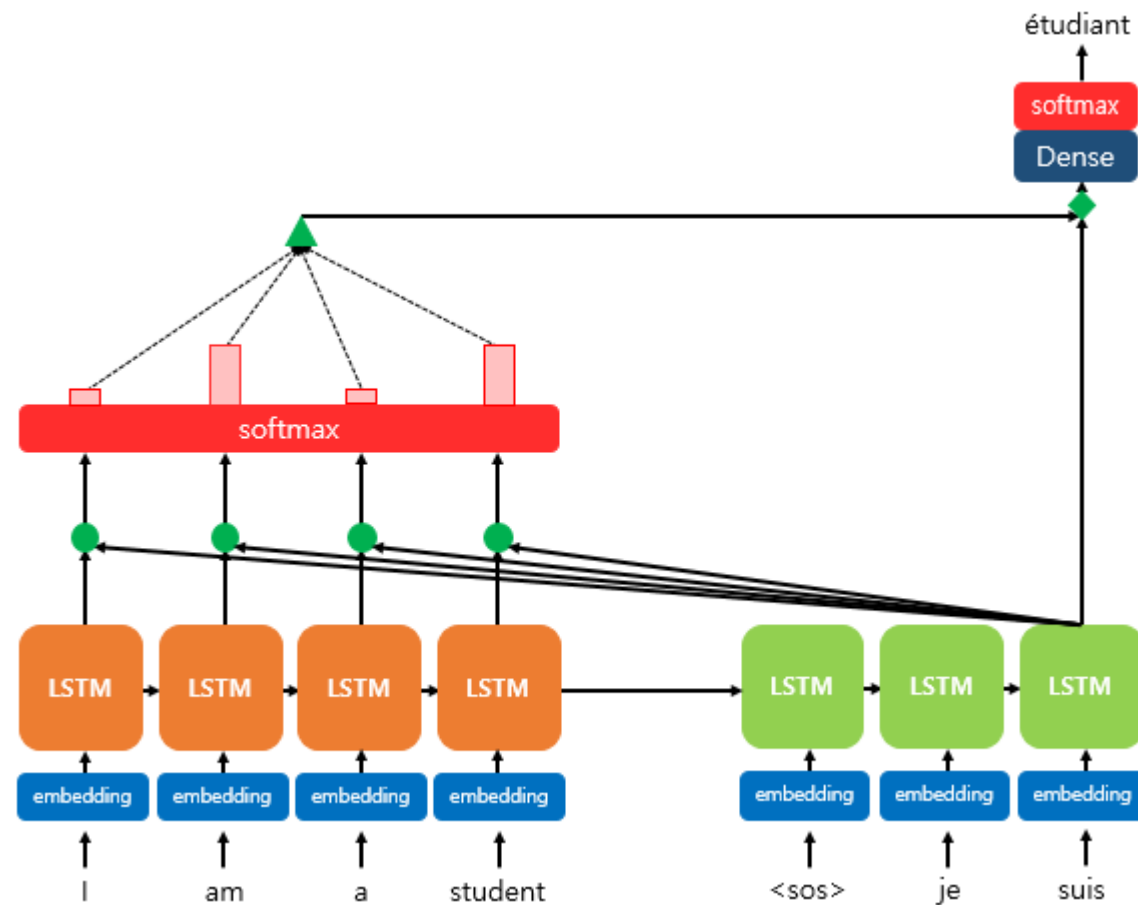
$$\text{Attention}(Q, K, V) = \text{Attention Value}$$

Q = t 시점의 Decoder cell의 은닉상태
K, V = 모든 시점의 Encoder Cell의 은닉상태들

-- 진행 --

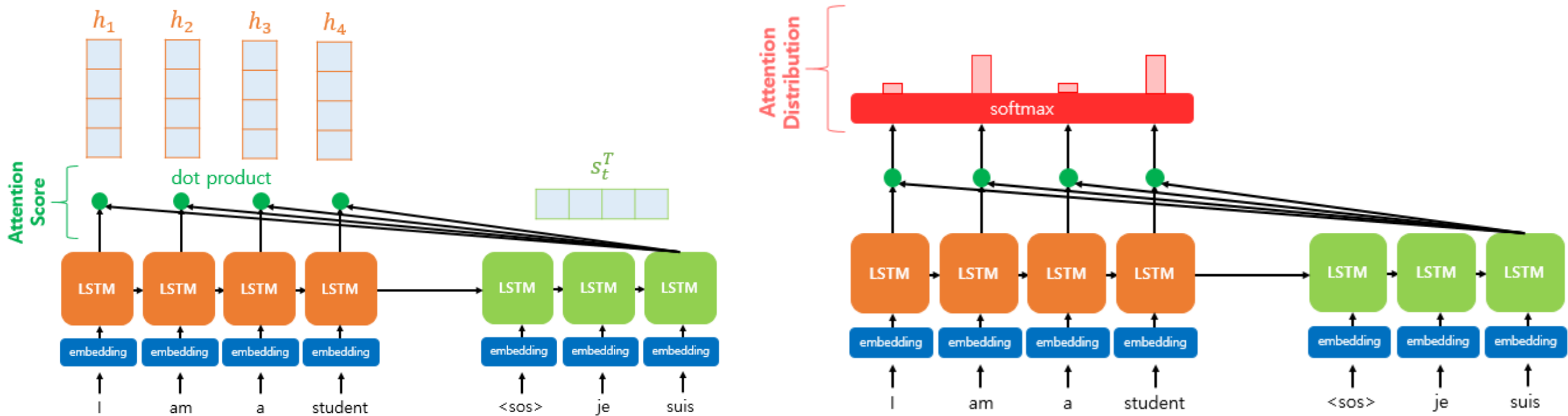
1. Q에 대해 모든 K와의 유사도를 구한다.
2. 해당 유사도를 K와 매핑 되어있는 각각의 V에 반영한다.
3. 유사도가 반영된 값을 모두 더해 Return한다.

Dot-Product Attention



Attention의 종류 중 이해가 쉬운 Dot-Product Attention으로 설명
Decoder의 세번째 LSTM에서 연결된 선이 바로 Attention이다

Dot-Product Attention



1.왼쪽 그림

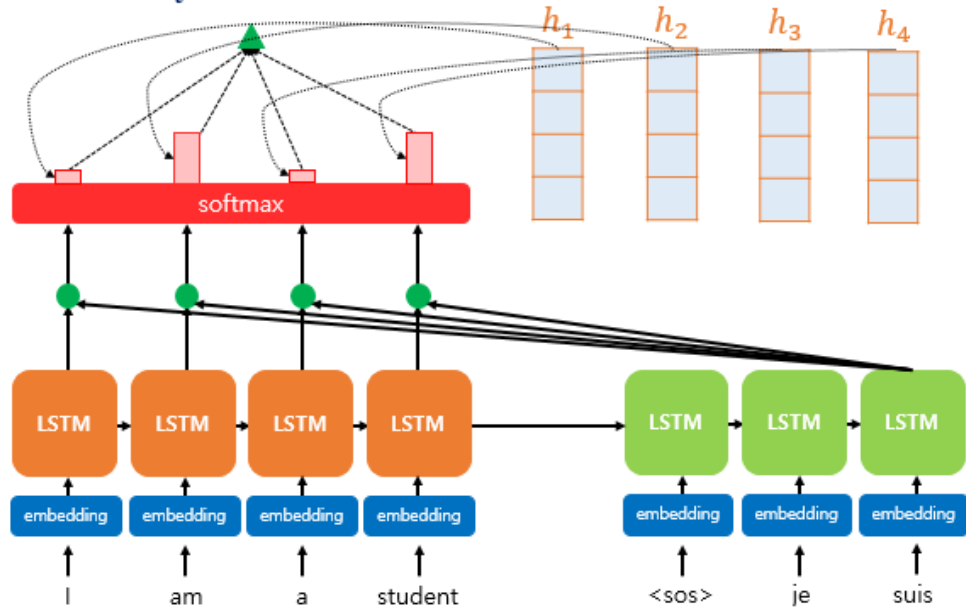
Attention Score를 구하는 것 현재 Decoder시점 t에서 단어를 예측하기 위해 Encoder의 모든 은닉상태 각각 Decoder의 현시점의 은닉상태 s_t 와 얼마나 유사한지를 판단하는 스코어 값이다.

2. 오른쪽 그림

활성화 함수인 Softmax함수를 통해 Attention 분포를 구한다.
Encoder의 은닉상태에서의 Attention Weight의 크기를 직사각형으로 시각화 하였다.

Dot-Product Attention

Attention Value a_t

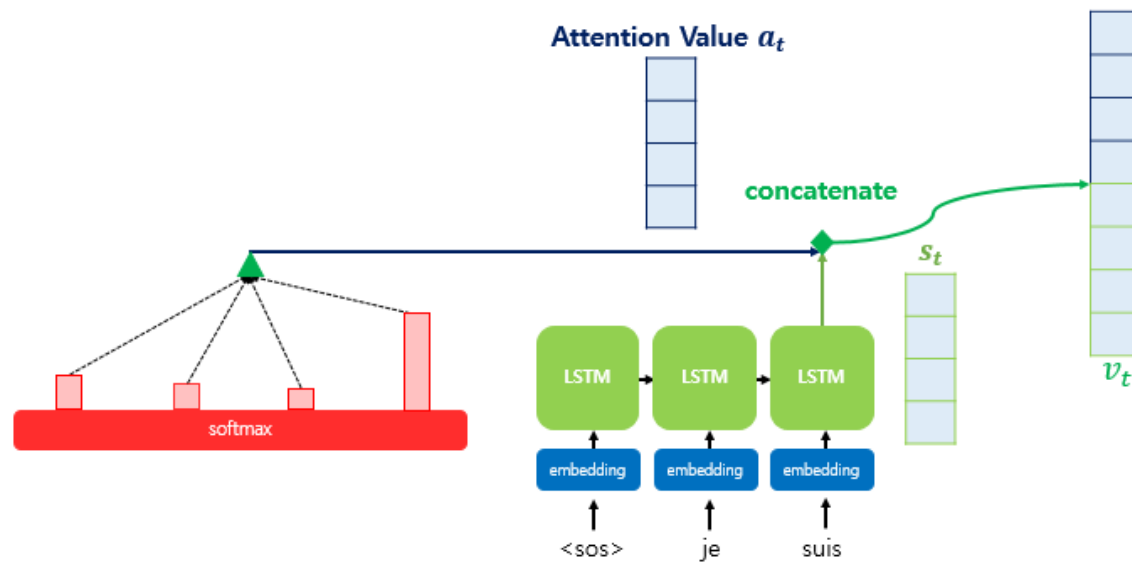


왼쪽 그림

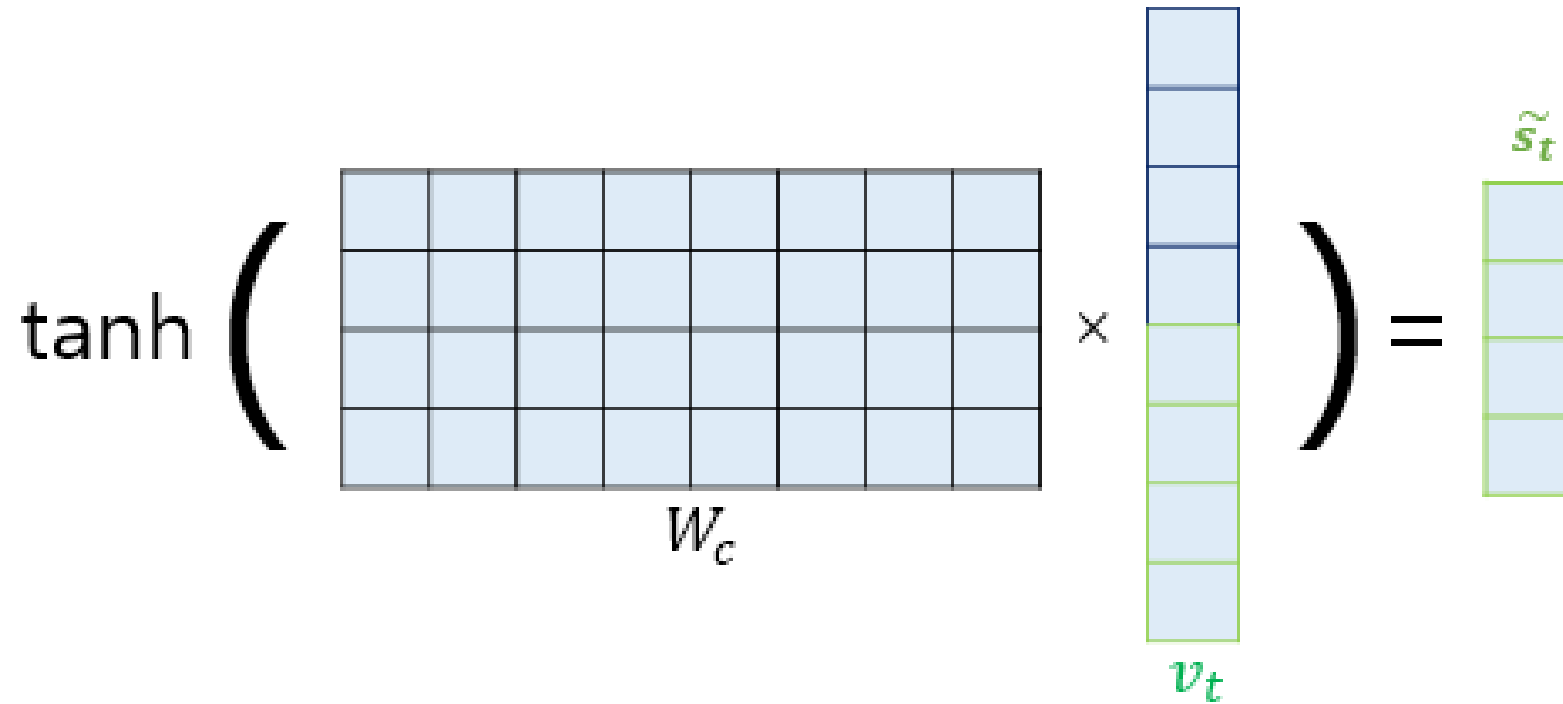
각 Encoder의 Attention Weight와 은닉 상태를 합쳐 Attention값(a_t)을 구한다.

오른쪽 그림

Attention 값과 Decoder의 t 시점의 은닉상태를 연결한다.(Concatenate)
 이때 v_t 를 \hat{y} 예측 연산의 입력으로 사용하여 Encoder의 정보를 활용해 \hat{y} 를
 좀 더 잘 예측 할 수 있게 된다. 이것이 바로 Attention의 핵심이다.



Dot-Product Attention

$$\tanh \left(W_c \times v_t \right) = \tilde{s}_t$$


가중치 행렬과 곱한 후에 하이퍼볼릭탄젠트 함수를 지나 출력층 연산을 위한 새로운 Vector \tilde{s}_t 를 가진다.
Seq2seq의 출력층 입력이 t 시점의 은닉상태인 s_t 였으나, Attention에서는 \tilde{s}_t 가 되는 것이다.

Attention의 다른 종류

이름	스코어 함수	Defined by	
<i>dot</i>	$score(s_t, h_i) = s_t^T h_i$	Luong et al. (2015)	
<i>scaled dot</i>	$score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$	Vaswani et al. (2017)	
<i>general</i>	$score(s_t, h_i) = s_t^T W_a h_i$ // 단, W_a 는 학습 가능한 가중치 행렬	Luong et al. (2015)	
<i>concat</i>	$score(s_t, h_i) = W_a^T \tanh(W_b[s_t; h_i])$ $score(s_t, h_i) = W_a^T \tanh(W_b s_t + W_c h_i)$	Bahdanau et al. (2015)	
<i>location - base</i>	$\alpha_t = softmax(W_a s_t)$ // α_t 산출 시에 s_t 만 사용하는 방법.	Luong et al. (2015)	

st => Query
hj => Keys
Wa, Wb => Weight

A stack of five books with yellow, blue, and white covers is shown. A dark blue graduation cap with a yellow tassel is placed on top of the books. The background is a plain, light gray surface.

Part 4

TRANSFOMER

Part 5

BERT



Part 6

GPT-2



Part 7

GPT-3





감사합니다.