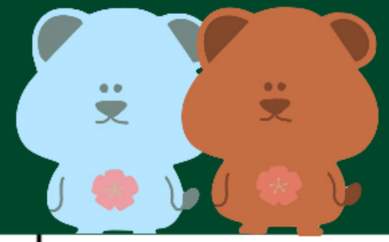
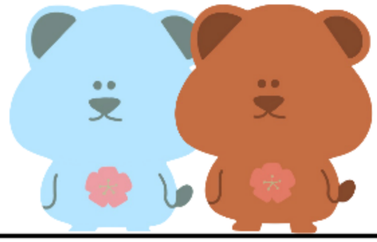


Week 10. Modeling contexts of use: Contextual Representations and Pretraining

발표자: 김소민, 김나현

EWHA EURON

목차



EWHA EURON

Reflections on Word Representations

EWHA EURON

Pretrained Word Vector

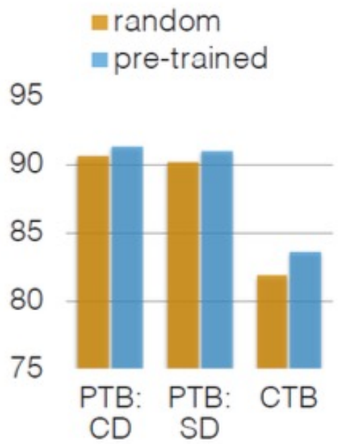
Representation for words before ~2011

- Rule Based Feature Extraction (State of the Art)

	POS WSJ (acc.)	NER CoNLL (F1)
State-of-the-art*	97.24	89.31
Supervised NN	**96.37**	**81.47**
Unsupervised pre-training followed by supervised NN**	**97.20**	**88.87**
+ hand-crafted features***	97.29	89.59

Pretrained Word Vector

- 최근 동향 (2014 ~)
- Word Vector with random initialization + Training on Task of Interest
- 성능이 random initialization 보다 훨씬 좋음



- Chen and Manning (2014)
  Dependency parsing
- Random: uniform(-0.01, 0.01)
- Pre-trained:
  - PTB (C & W): +0.7%
  - CTB (word2vec): +1.7%

Tips for unknown words with word vectors

Common Practice
- 학습 시 잘 등장하지 않는 (5회 미만)의 단어를 <UNK>으로 처리
- Test 할때에 Out-of-Vocab (OOV) 단어를 <UNK>으로 매칭

Con
- <UNK>으로 매칭된 단어가 중요한 의미를 가질 수 있음

Tips for unknown words with word vectors

Solution

1. Char-Level Embedding Model을 이용
2. Pre-trained 된 Word Vector 이용
3. Testing 시에 Random Vector를 부여하고 Vocab 에 추가

Word Embedding (Representation for words)

- 단어를 벡터로 표현함으로써 컴퓨터가 이해할 수 있도록 자연어를 적절히 변환할 수 있음
- 단어를 벡터로 표현하는 방법: Word2vec, GloVe, fastText 등

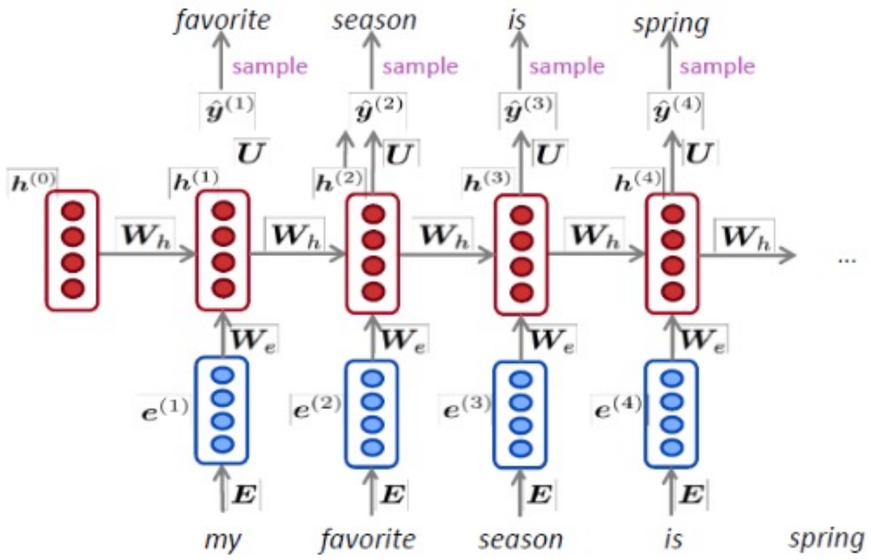
Con

- word token이 나타나는 context에 따라 달라지는 word type을 고려하지 못함
- word의 언어적, 사회적 의미에 따라 다른 측면을 고려하지 못함

ex) Star
Hollywood Star <-> Star (Galaxy)

Word Embedding (Representation for words)

Solution: Neural Language Model
-> Utilizing Context



Input: (pre-trained) Word Vectors
Output: next word

LM: Context-Specific Word Representation

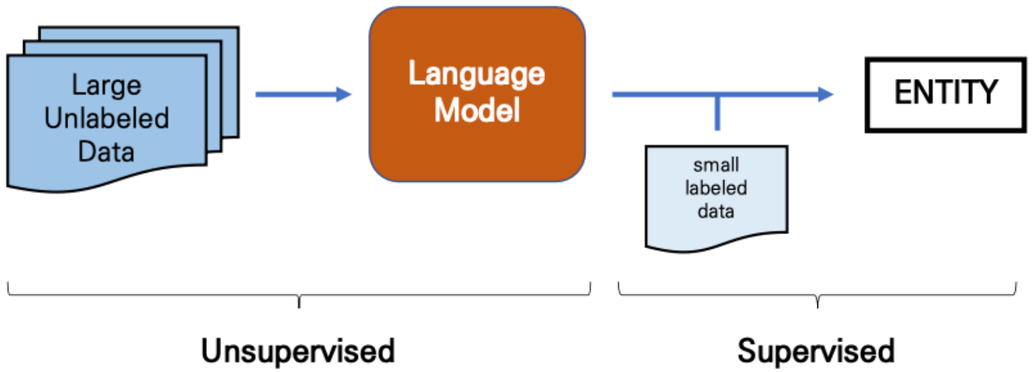
Pre-ELMo and ELMO

EWHA EURON

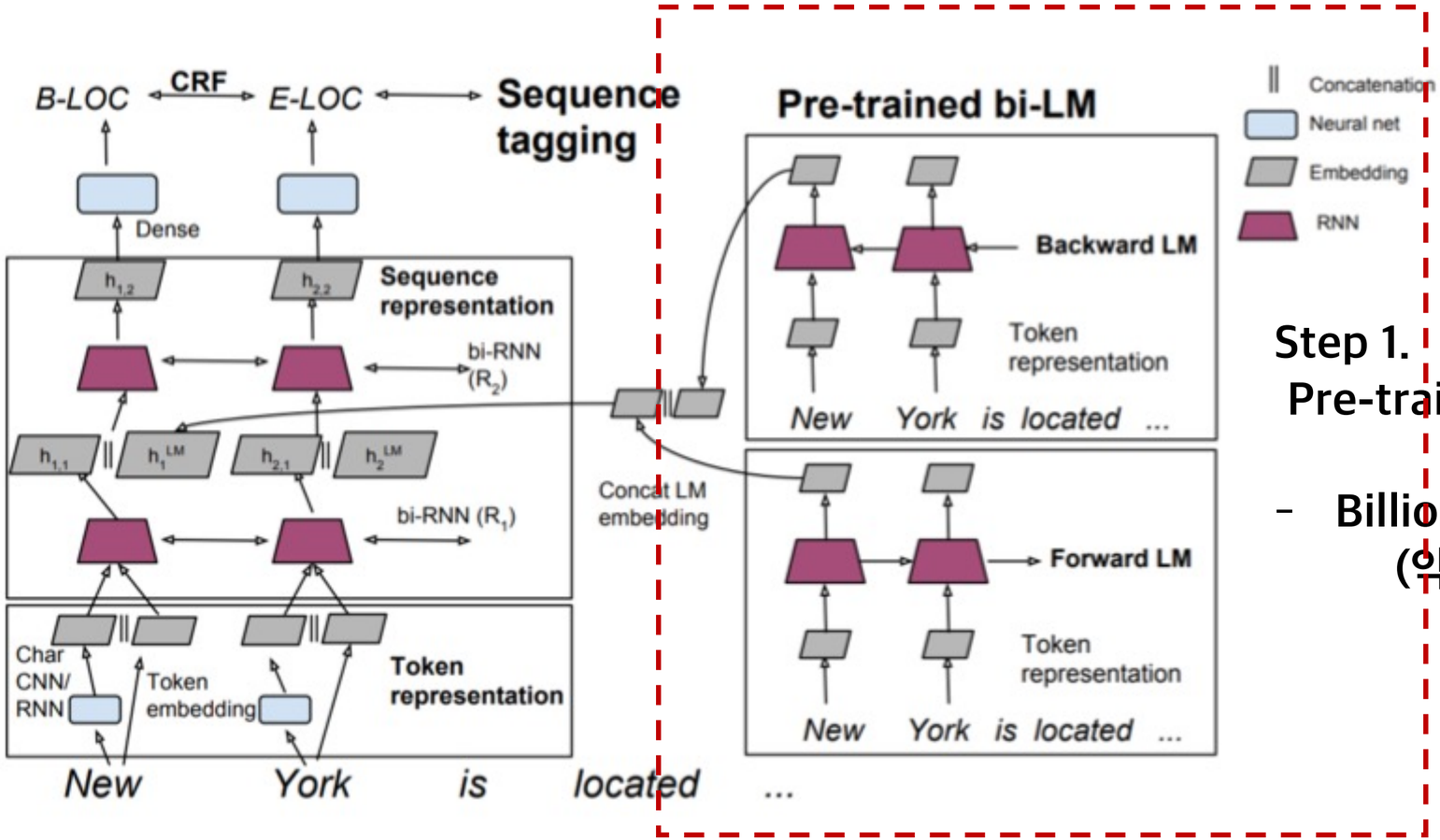
Pre-ELMO

KEY- Idea

- RNN으로 context에 담긴 뜻을 학습
- 일반적으로 학습에 이용되는 데이터는 small task-labeled data (ex. NER)
- Large unlabeled corpus로 먼저 학습을 시키는 semi-supervised approach를 이용



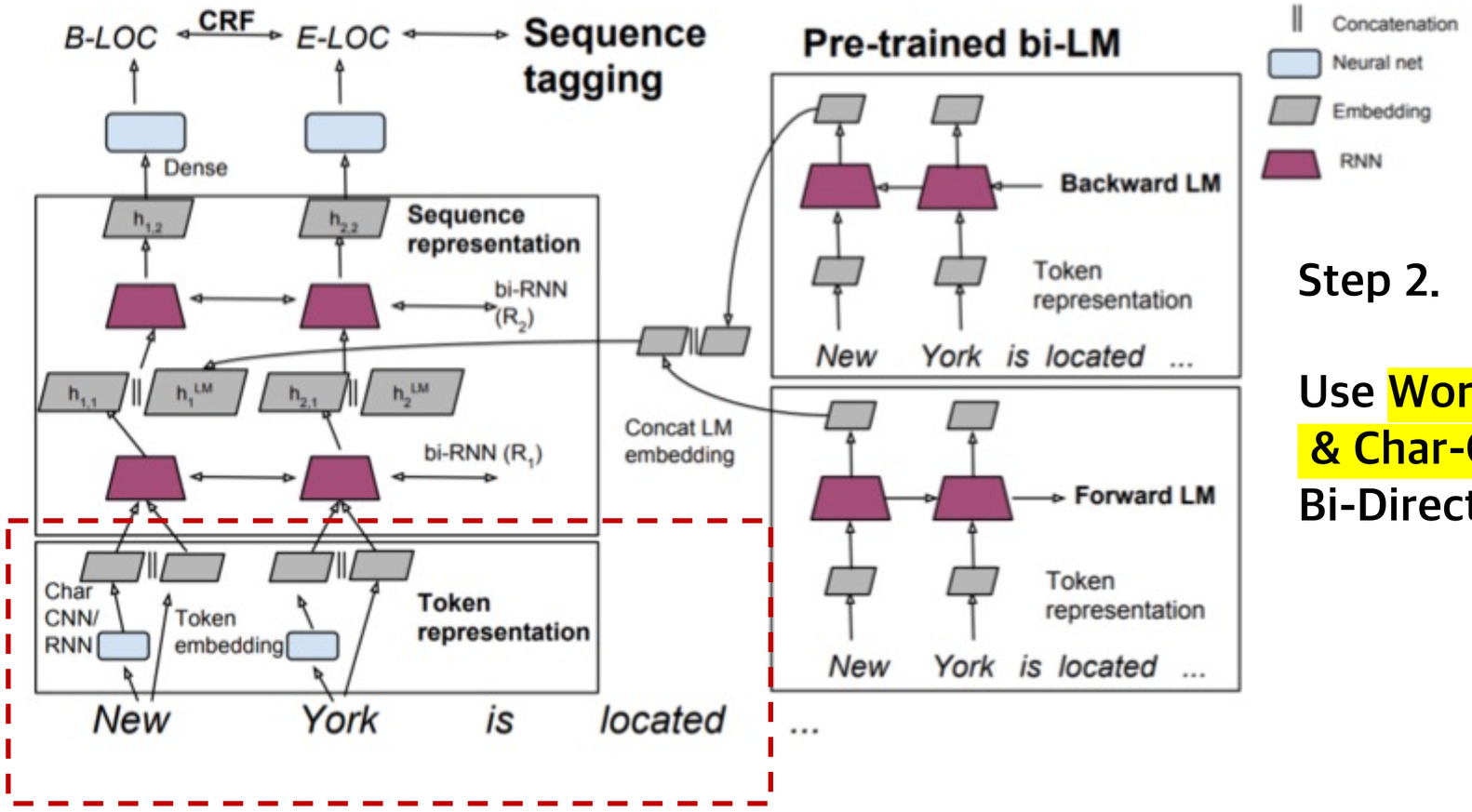
Pre-ELMO



Step 1.
Pre-train Bidirectional Language Model

- Billion Word Benchmark 이용
  (약 8억개의 단어가 존재)

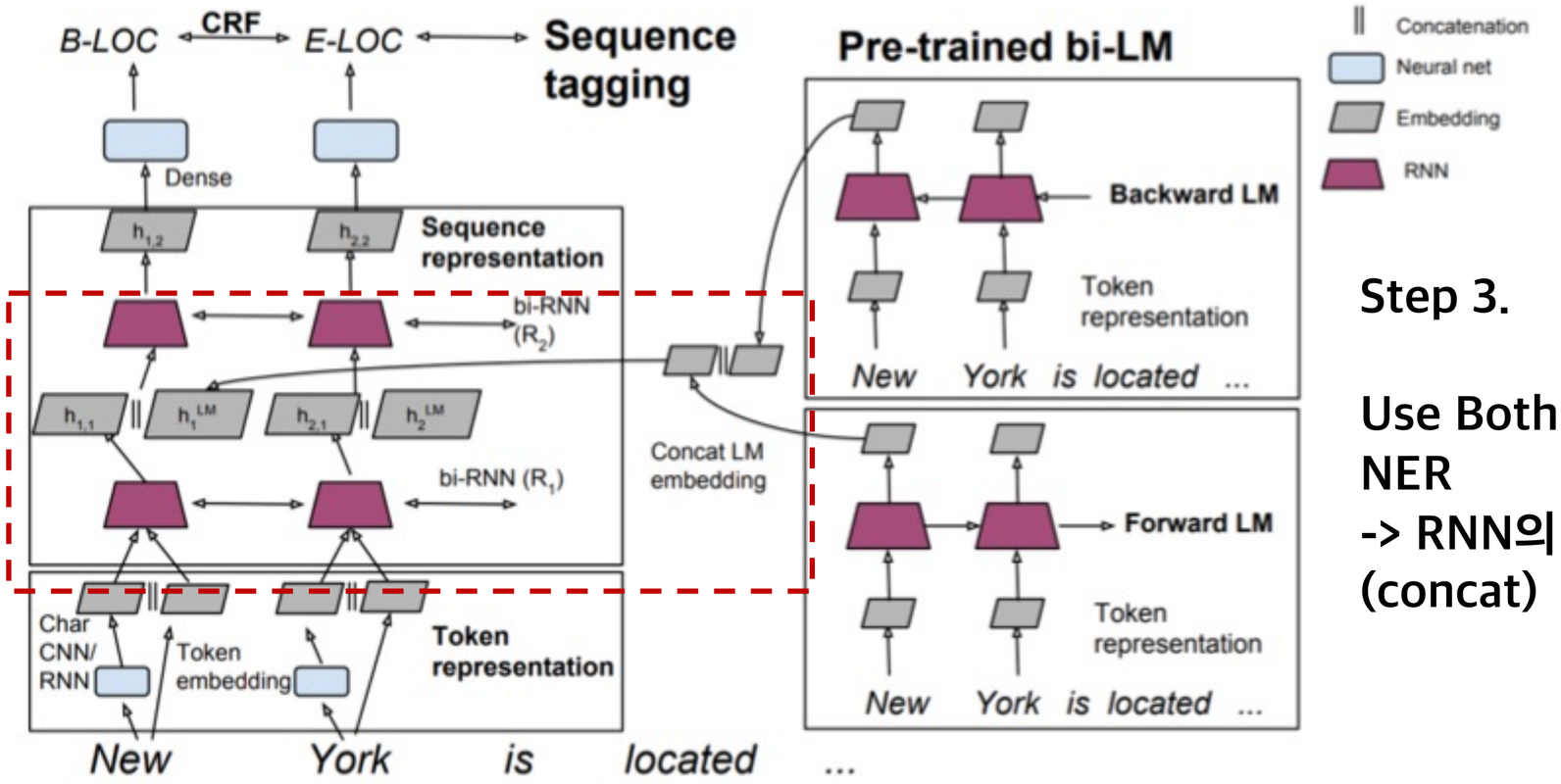
Pre-ELMO



Step 2.

Use Word (Token) Embedding & Char-CNN for the input of 2-layer Bi-Directional RNN

Pre-ELMO



Step 3.

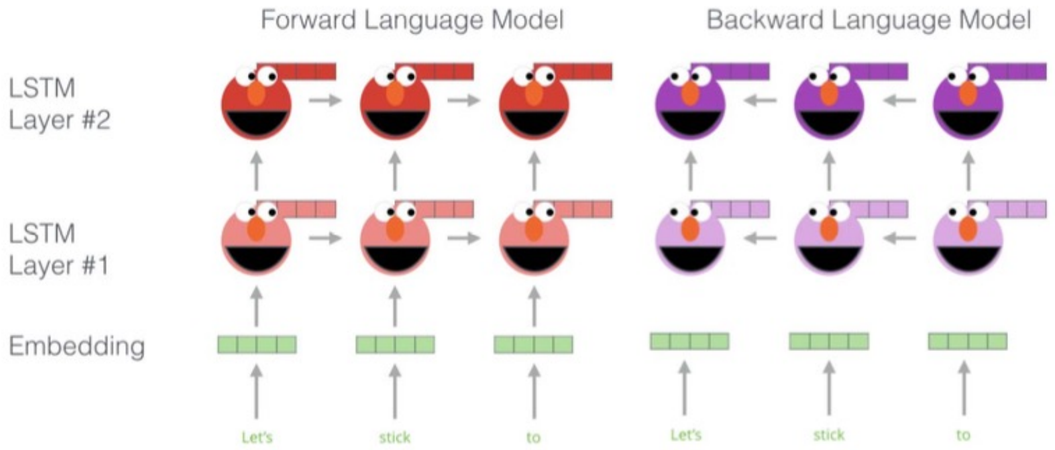
Use Both word Embeddings and LM for NER

-> RNN의 1st layer output 과 LM을 합침 (concat)

ELMO

KEY- Idea

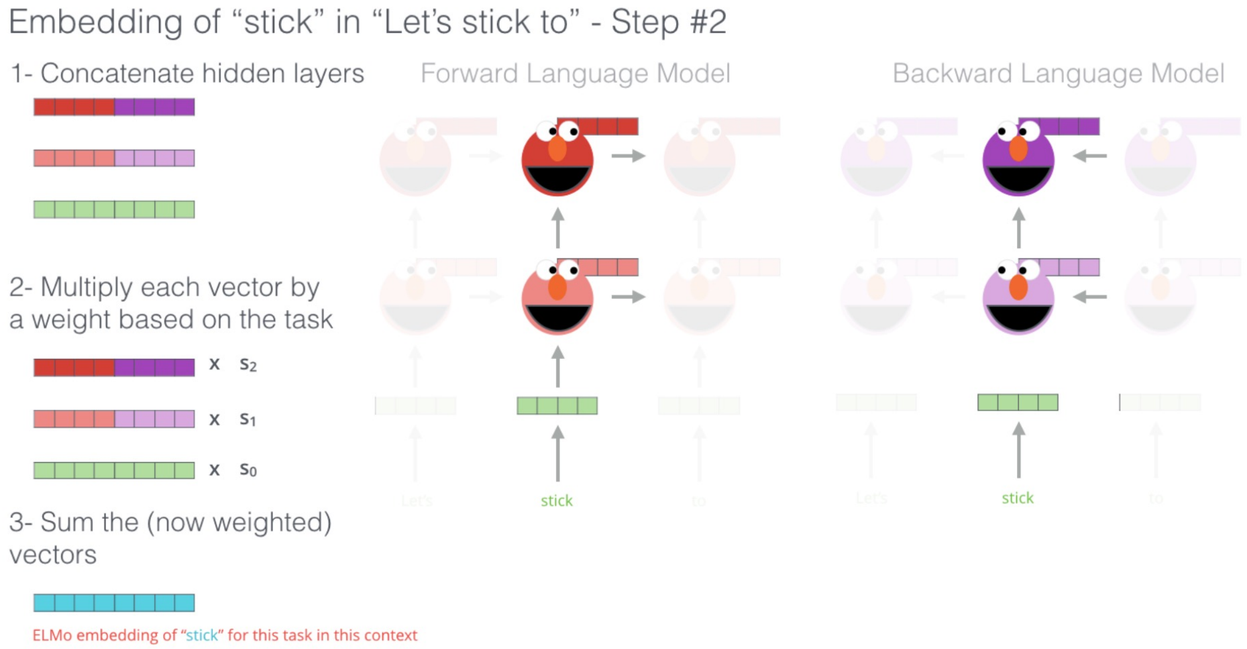
- 전체적인 구조는 pre-ELMO와 비슷
- 모든 문장을 이용해 Contextualized Word Vector를 학습
- 기존의 word embedding이 window를 이용해 주변 context를 이용한것과 대조
- 단어 Embedding은 Char CNN만을 이용



ELMO - Architecture

- 학습한 LM들의 Layer들을 concat한 후, task에 맞게 linear combination해서 이용

Cf) 기존의 비슷한 방법론 ; LSTM의 top layer만을 이용했던 것과 대조됨



ELMO - 성능

CoNLL 2003 Named Entity Recognition (en news testb)

Name	Description	Year	F1
ELMo	ELMo in BiLSTM	2018	92.22
TagLM Peters	LSTM BiLM in BiLSTM tagger	2017	91.93
Ma + Hovy	BiLSTM + char CNN + CRF layer	2016	91.21
Tagger Peters	BiLSTM + char CNN + CRF layer	2017	90.87
Ratinov + Roth	Categorical CRF+Wikipeda+word cls	2009	90.80
Finkel et al.	Categorical feature CRF	2005	86.86
IBM Florian	Linear/softmax/TBL/HMM ensemble, gazettes++	2003	88.76
Stanford	MEMM softmax markov model	2003	86.07

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

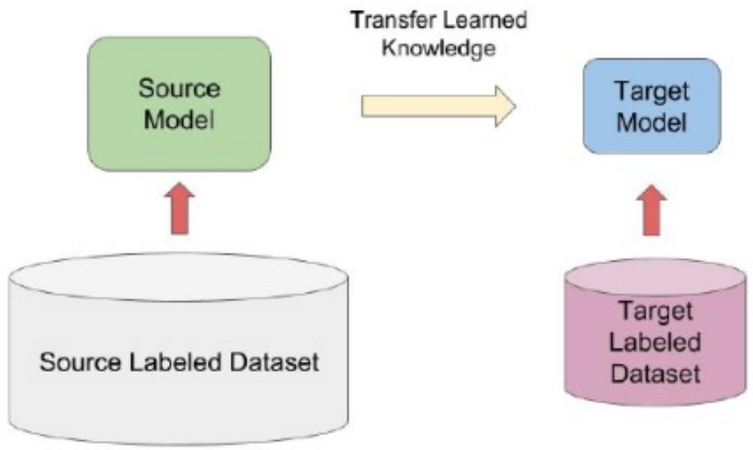
- Pre-ELMO와 비교하여 NER task에서의 성능 향상 (0.3% 정도)
- NER뿐만 아니라 NLP 대부분의 Task에서도 좋은 성능 보임 “Pixie Dust”

ULMfit and onward

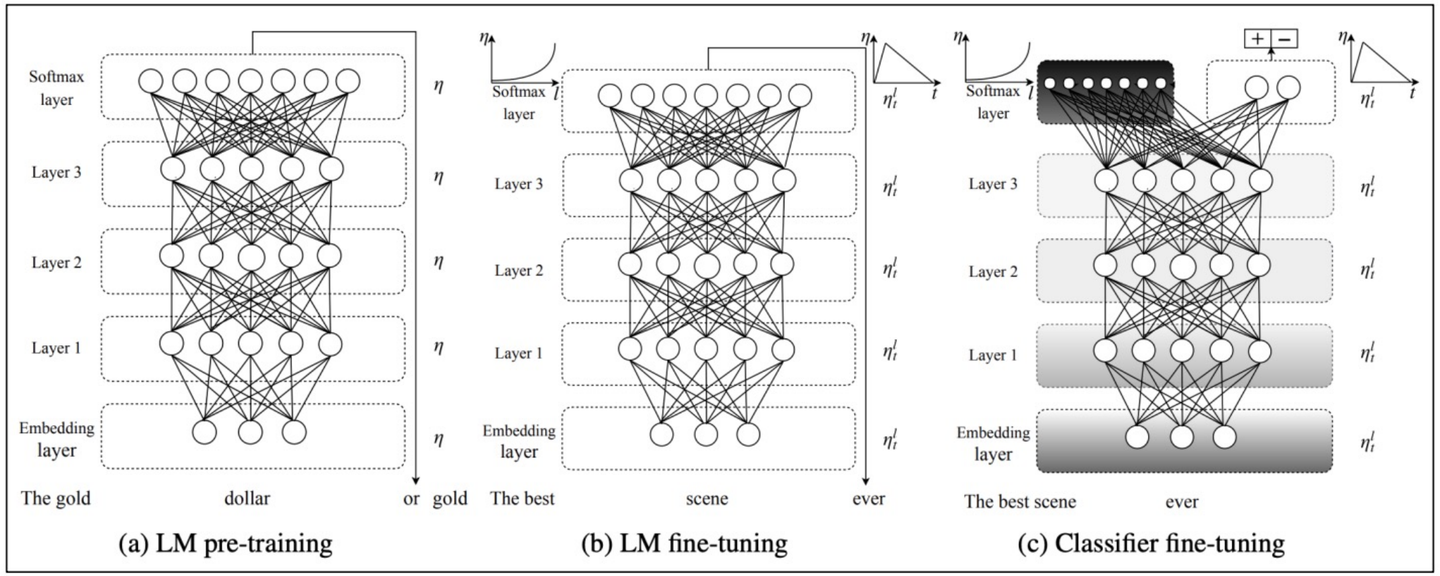
EWHA EURON

ULMFit

- NLP에서 본격적으로 Transfer Learning을 도입
- 1개의 GPU로 학습할 수 있는 정도의 사이즈 (Pretraining with small dataset)



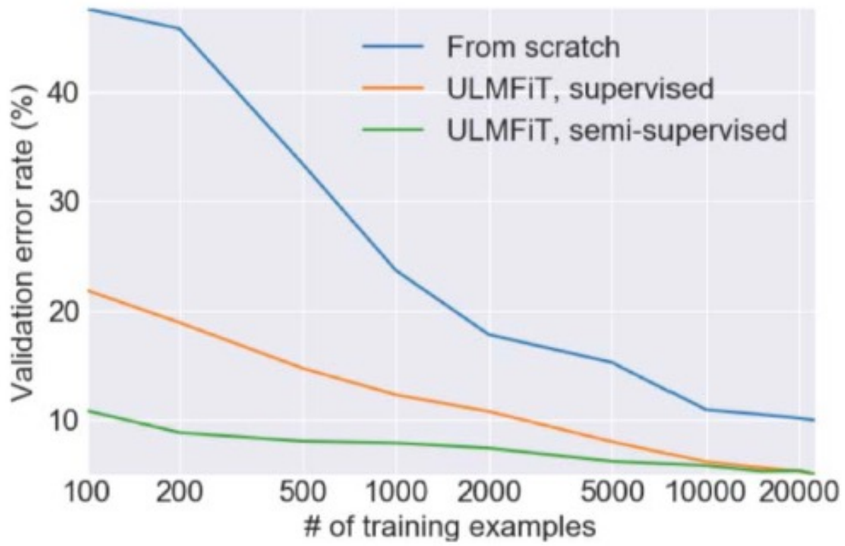
ULMFit



(a) 3-layer bi-LSTM LM pre-training
(b) Target Task에 맞춰 LM fine-tuning
(c) Target task의 Classifier fine-tuning

ULMFit

- Transfer learning을 적용하면 훨씬 더 효율적인 결과를 볼 수 있음
- From scratch <-> Supervised/ Semi Supervised 비교

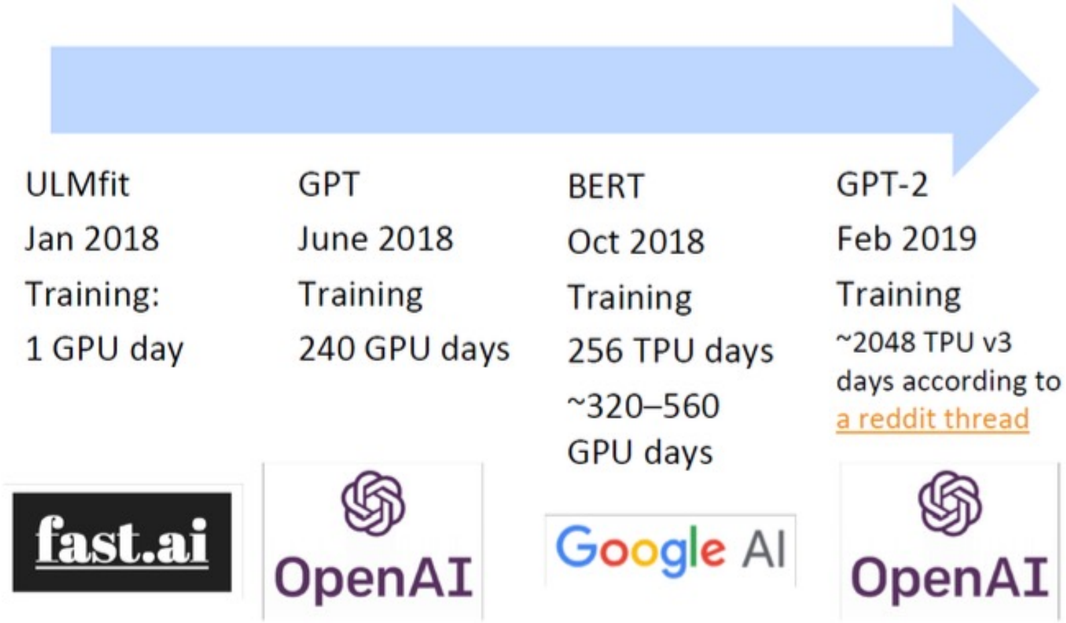


	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

ULMFit and Onwards

Scaling UP

- ULMfit 이후 모델의 파라미터를 늘려 LM pre-training을 하는 모델들이 등장
- GPU 1개로 학습할 수 있던 ULMfit에 비해 기하급수적으로 필요한 리소스가 증가
- ULMfit 이후로는 모두 Tranformer 기반 pre-trained model



Transformer architectures

EWHA EURON

The motivation for transformers

👉 Transformer : 2017년 구글이 발표한 논문인 "Attention is all you need"에서 나온 모델

기존의 seq2seq의 구조인 인코더-디코더를 따르면서도, Attention만으로 구현함.

👉 RNN: sequential한 특징 고려 → 병렬적 계산 (parallelization) 힘듦. long range dependencies 문제

👉 Attention: RNN의 한계를 보완하기 위해 사용

If attention gives us access to any state,
Maybe we can just use attention
and don't need the RNN?

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

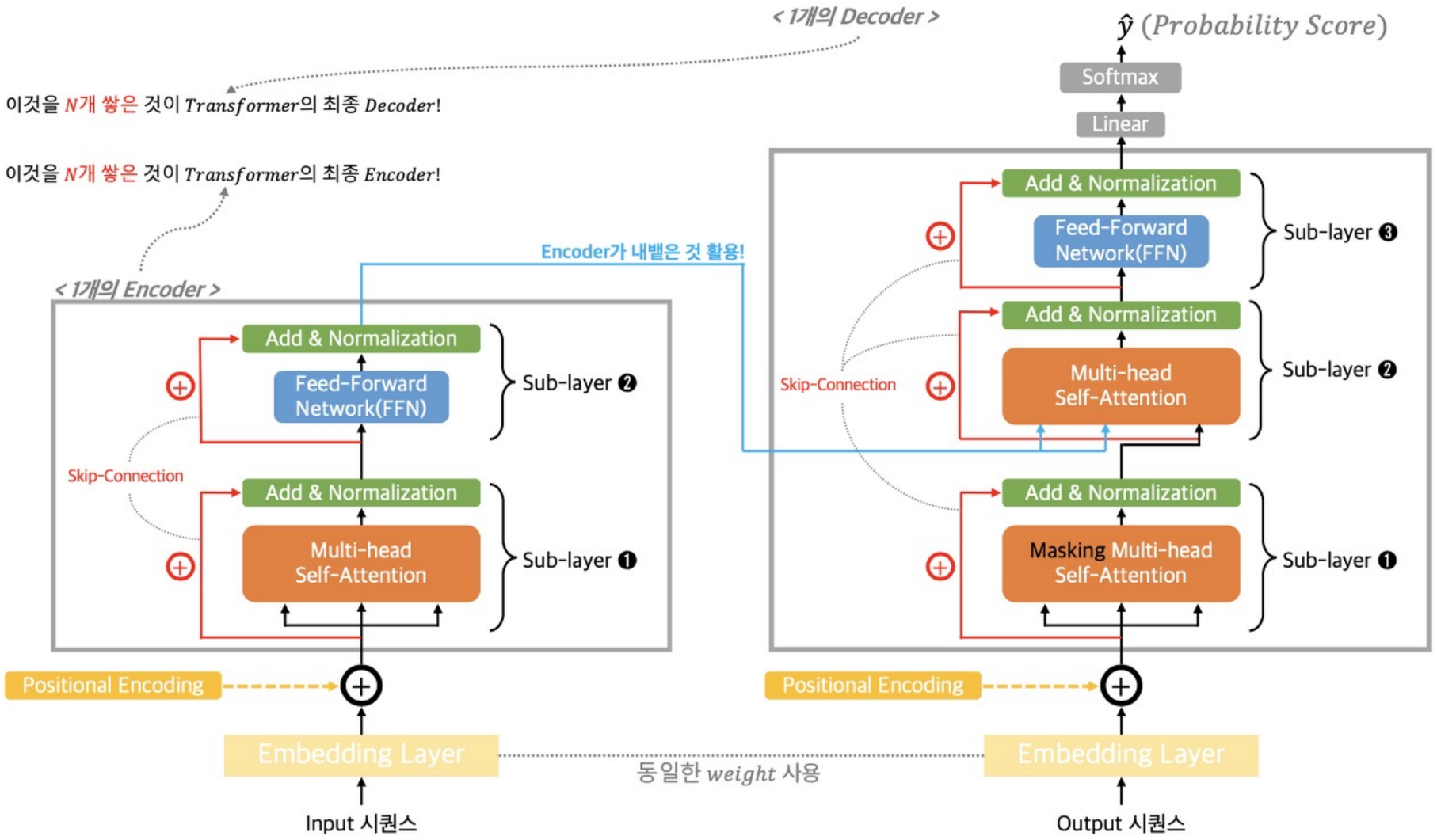
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

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

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

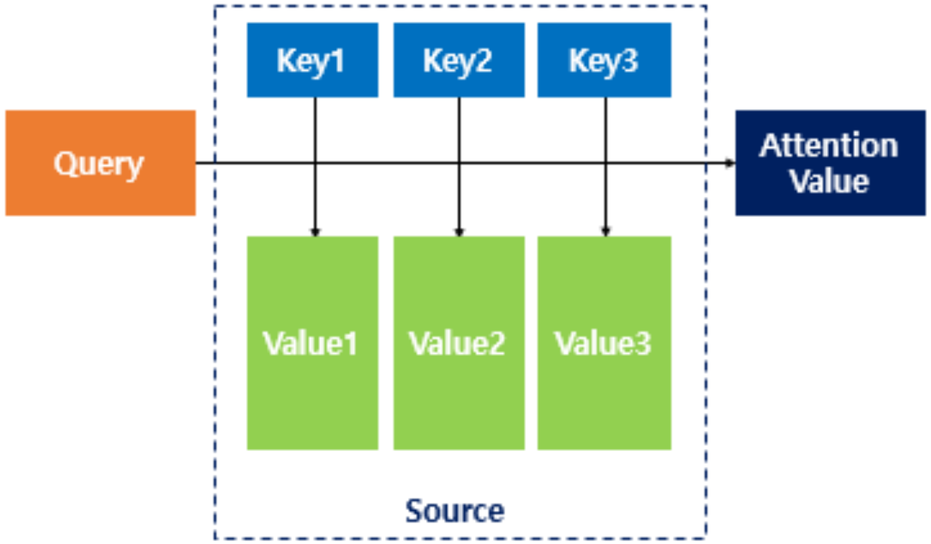
Transformer Overview



https://techblog-history-younghunjo1.tistory.com/497

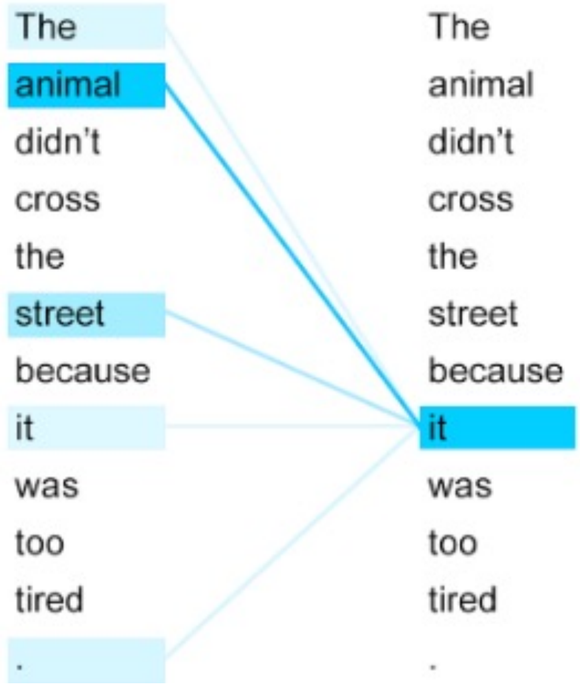
Attention Mechanism

Attention(Q, K, V) = Attention Value



주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 유사도 계산.
→ 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영
→ 유사도가 반영된 '값(Value)'을 모두 더해서 리턴
→ Attention Value!

Self Attention

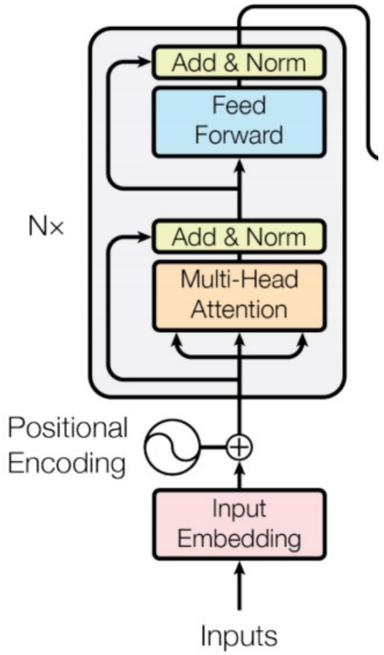


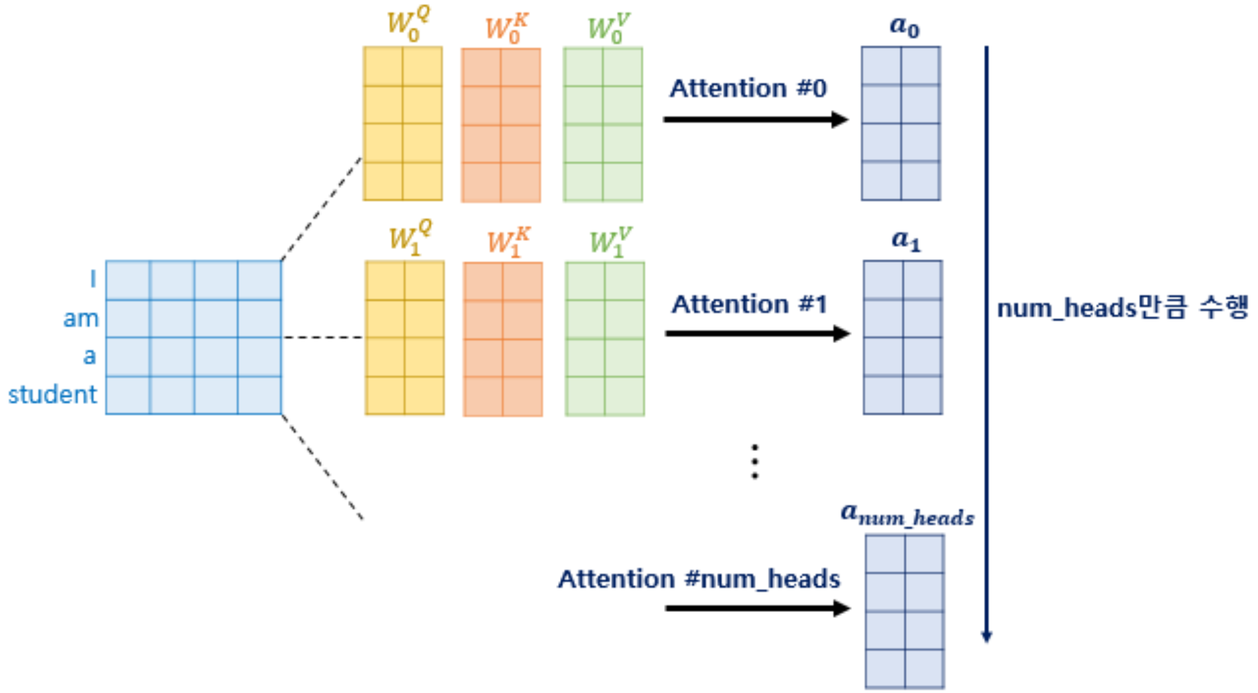
→ 어텐션을 자기 자신에게 수행하는 것.
입력 문장 내의 단어들끼리 유사도를 구함으로써 it에 해당하는 단어가 'animal'일 확률이 높다는 것을 찾아냄.

Transformer architectures - encoder

transformer의 기본적인 block 구조
👉 두 개의 sublayers

1. Multihead attention
2. 2-layer feed-forward Nnet (with ReLU)





👉 Multihead attention
: 어텐션을 병렬로 수행하여 여러 헤드들이 각기 다른 시각으로 정보들을 수집
→ 높은 성능

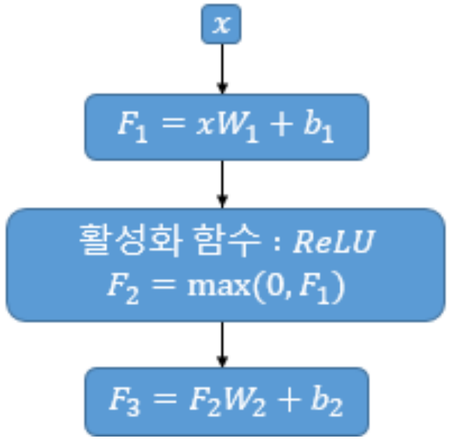
인코더의 입력으로 들어왔던 행렬의 크기는 유지됨.
(트랜스포머는 동일한 구조의 인코더를 쌓은 구조이므로 입력의 크기가 출력에서도 동일해야 함.)

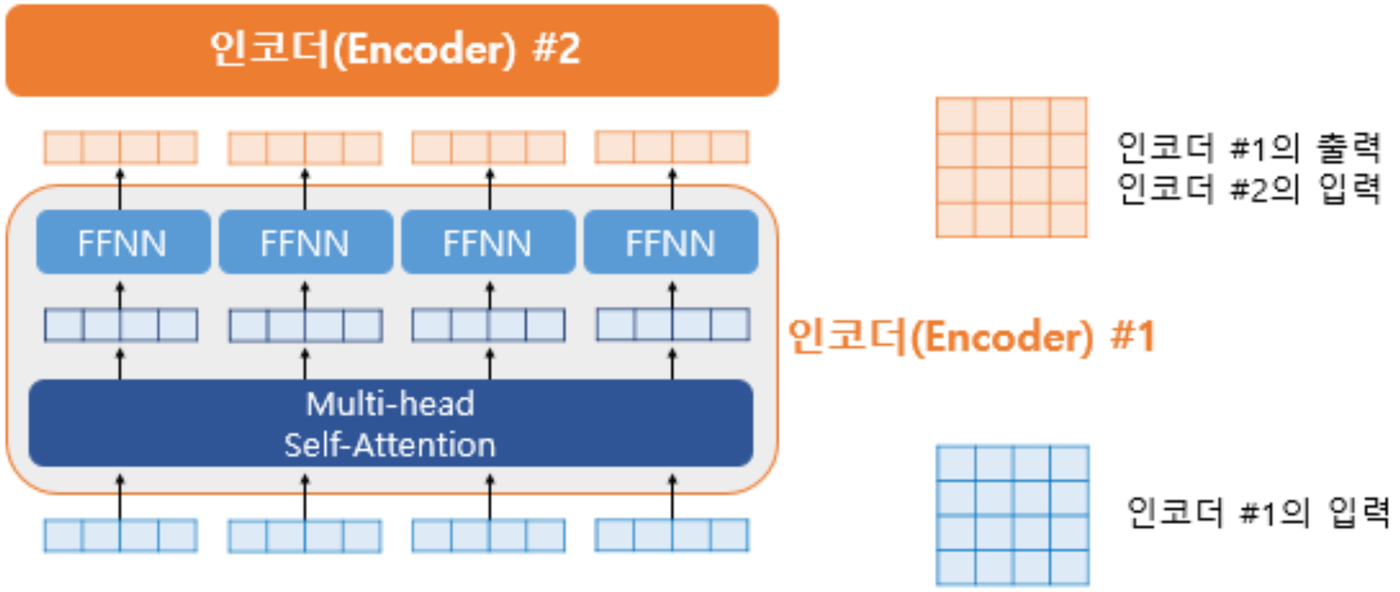
Transformer architectures - encoder

👉 2-layer feed-forward Nnet (with ReLU)

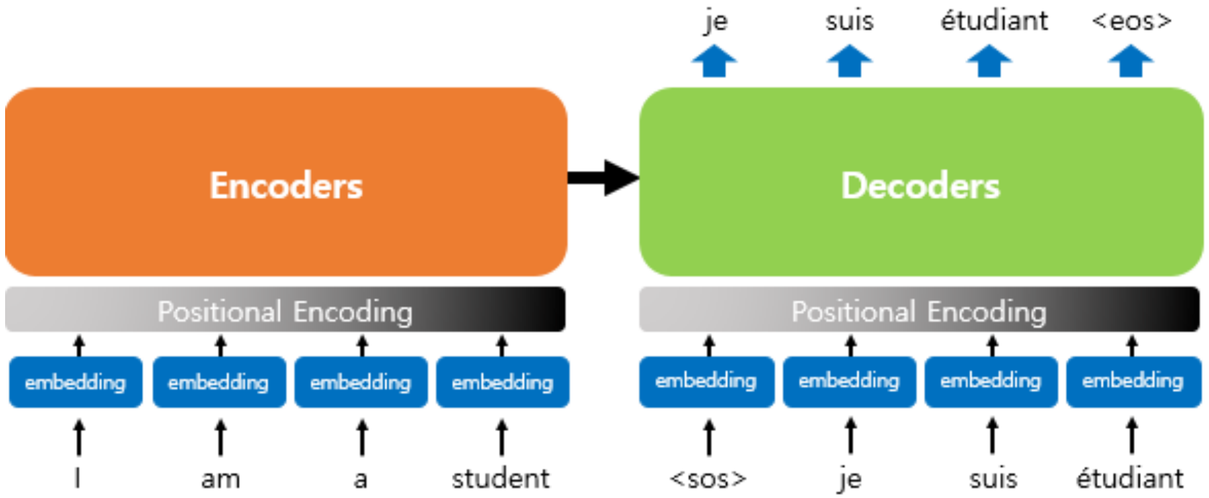
Position-wide Feed-forward 신경망
: 인코더와 디코더에서 공통적으로 가지고 있는 서브층.

$$FFNN(x) = MAX(0, xW_1 + b_1)W_2 + b_2$$

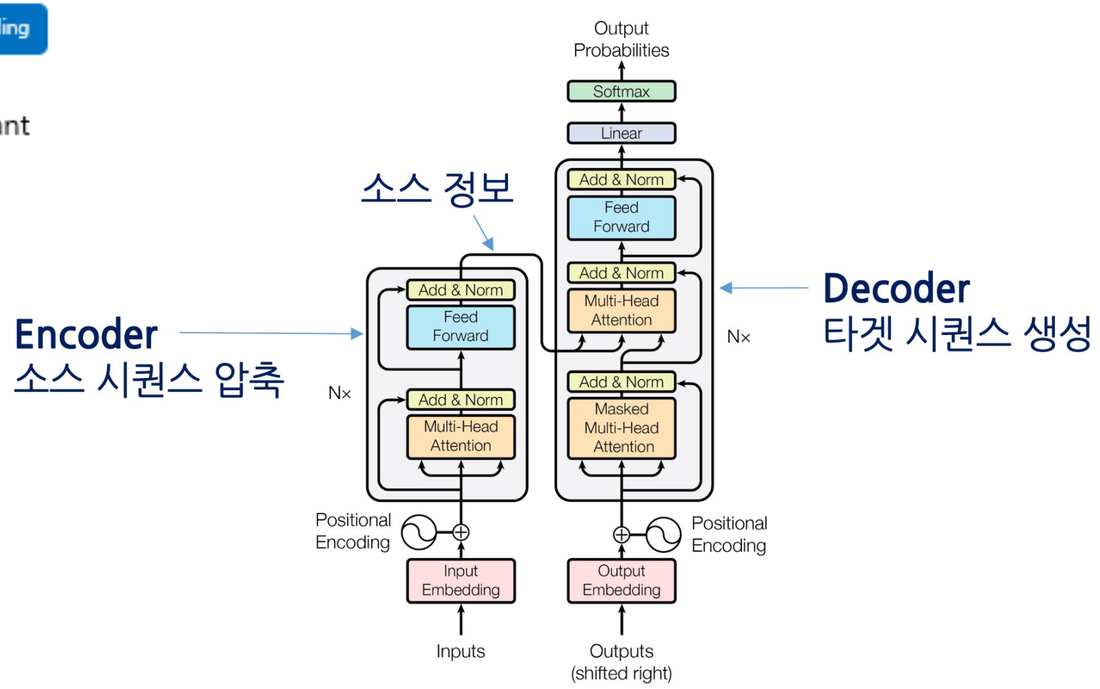




Transformer architectures



디코더는 기존의 seq2seq 구조처럼 시작 symbol <sos>를
입력으로 받아 종료 symbol <eos>가 나올 때까지 연산을 진행함.



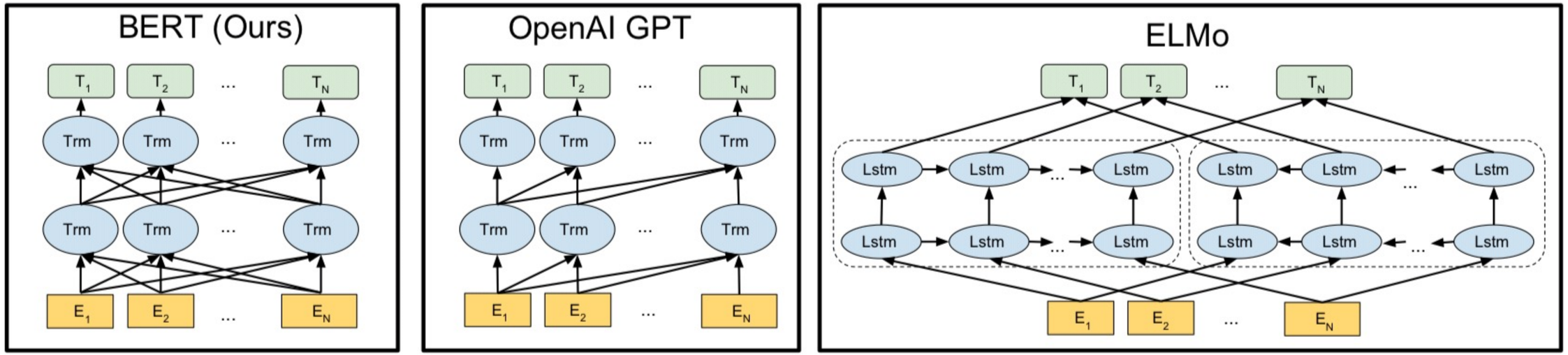
Encoder
소스 시퀀스 압축

Decoder
타겟 시퀀스 생성

BERT

BERT(Bidirectional Encoder Representations from Transformers)
2018년에 구글이 공개한 사전 훈련된 모델
(* 트랜스포머의 인코더를 쌓아올린 구조)

레이블이 없는 방대한 데이터로 사전 훈련된 모델
→ (Fine Tuning) 레이블이 있는 다른 Task에서 추가 훈련과 함께
하이퍼파라미터를 재조정
→ 성능 향상.



Masked Language Model 통해서 양방향성 얻음.

BERT

중간에 단어들을 masking하고, 빈칸에 들어갈 단어들을 예측하게 하는 방식