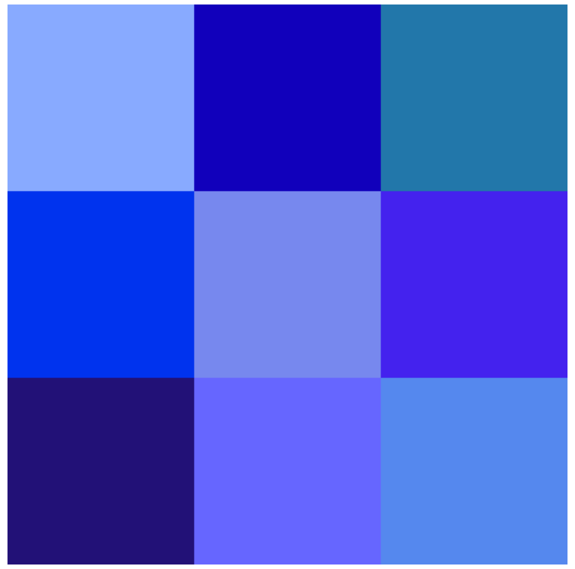
NLP분반 WEEK 7

BERT

Bidirectional Encoder Representations from Transformers

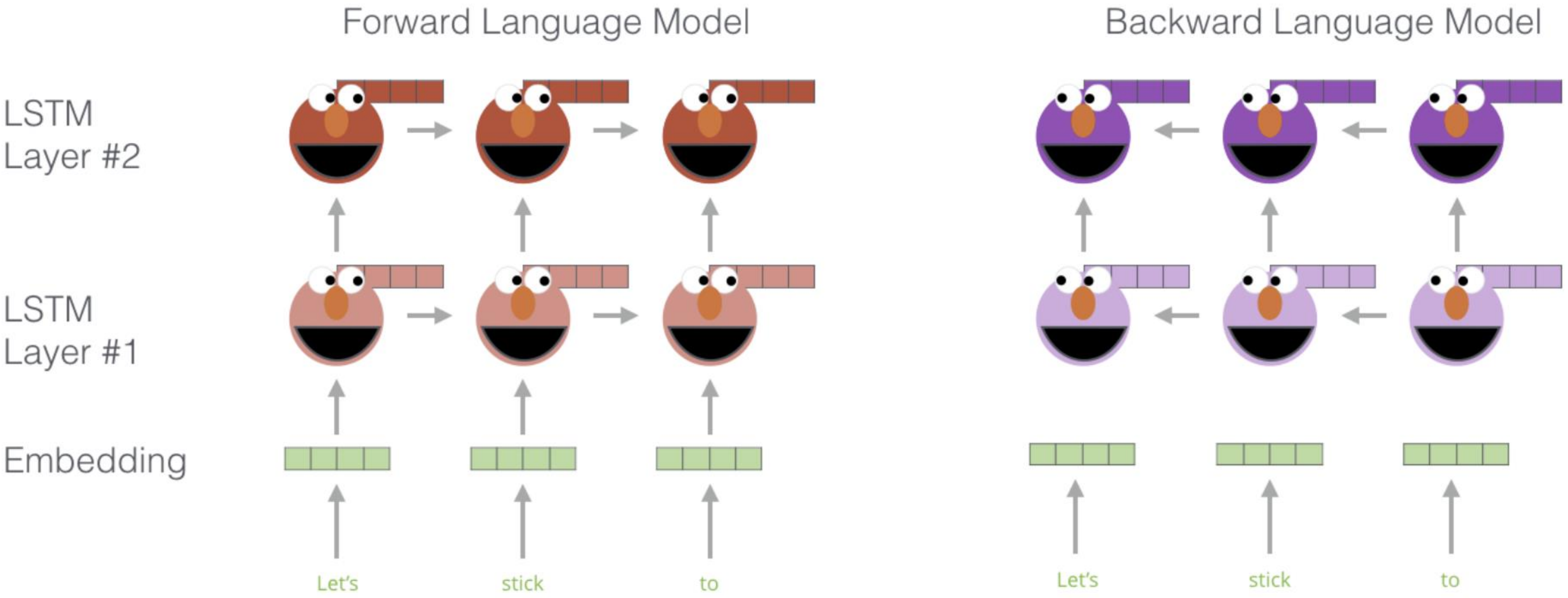
Pre-Training

Blue

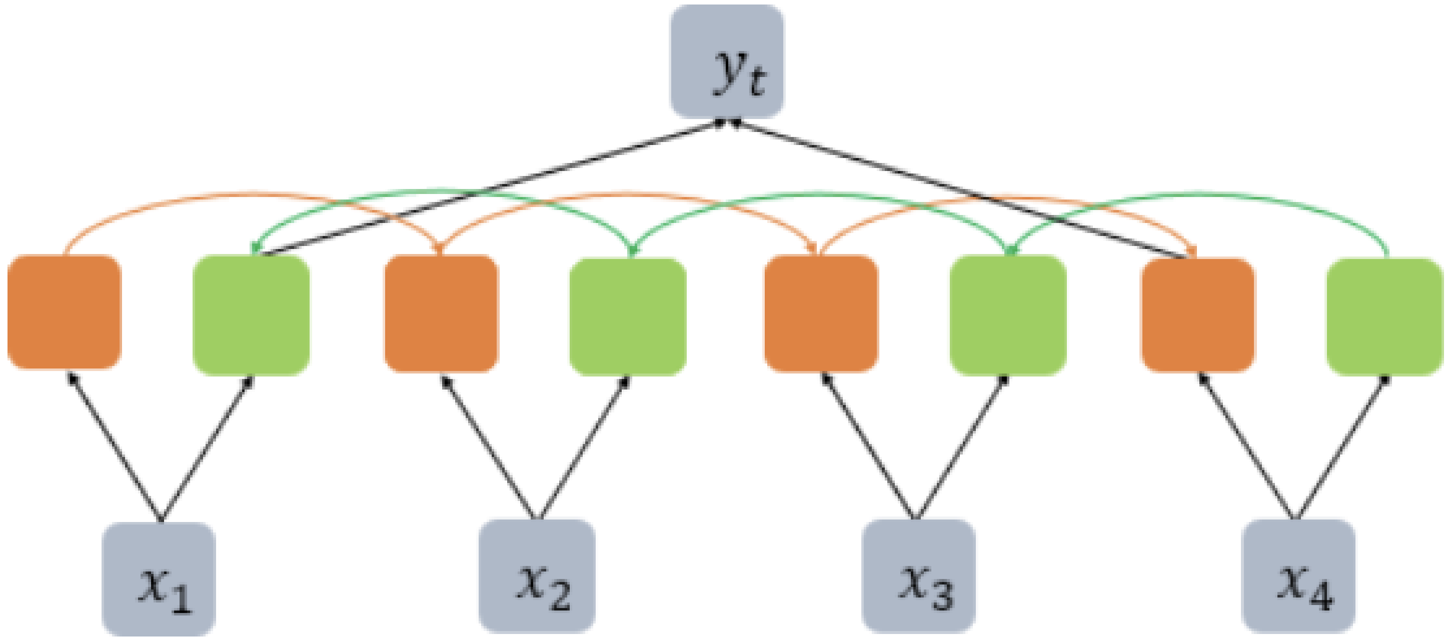


[0.2 0.8 -1.2]

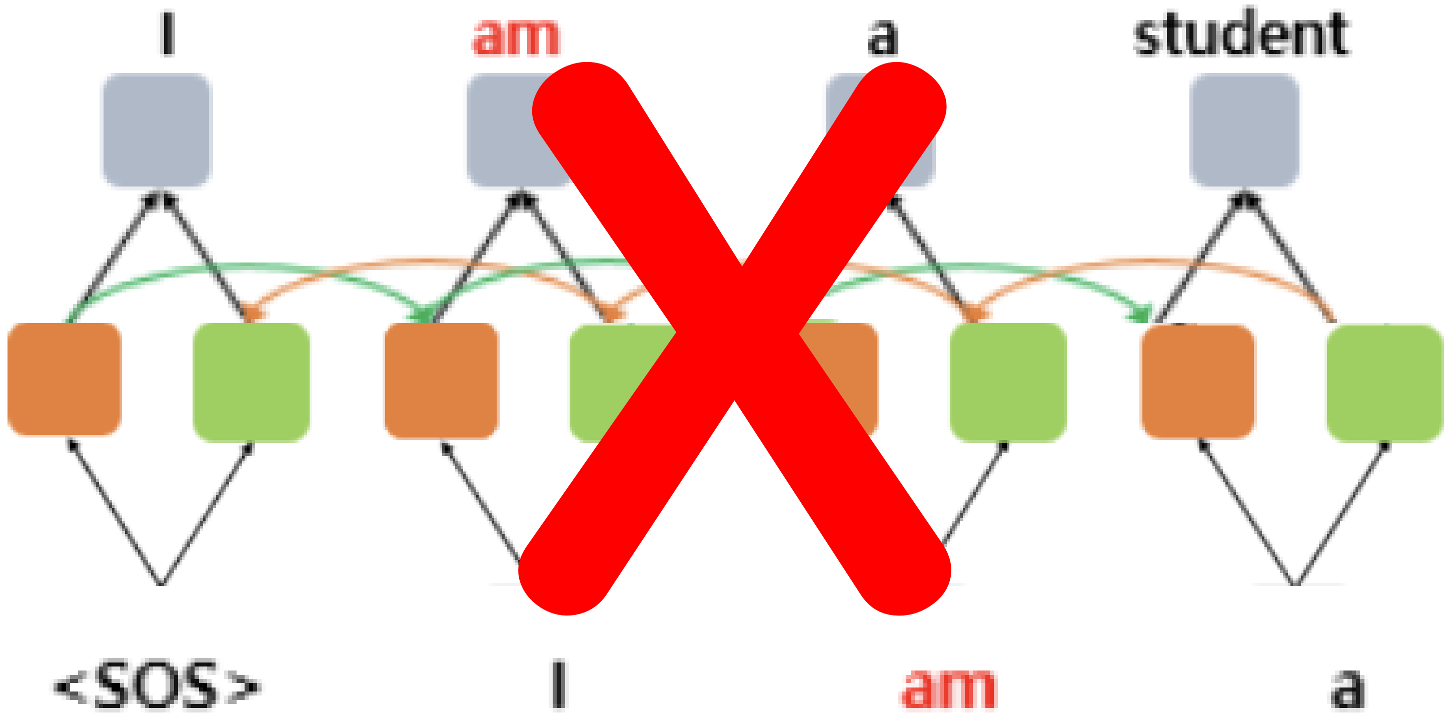
ELMo



Pre-Training :



양방향RNN



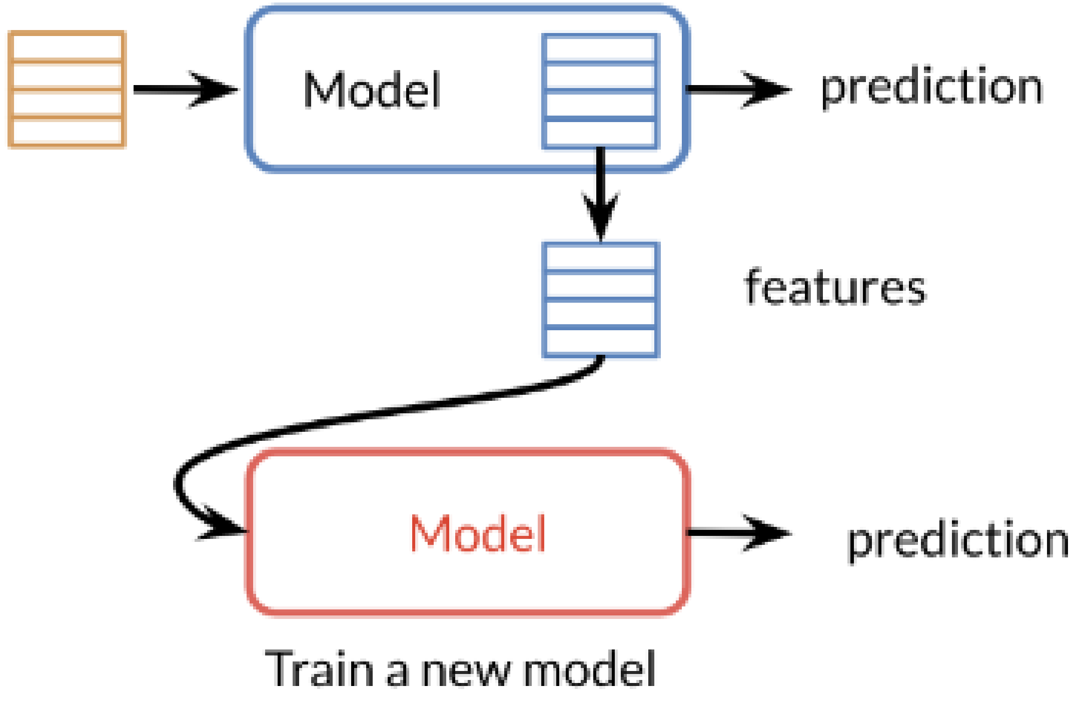
양방향RNNLM

양방향성을 유지하면서 LM로써 역할을 할 수 있는 모델

: For Contextual Consideration

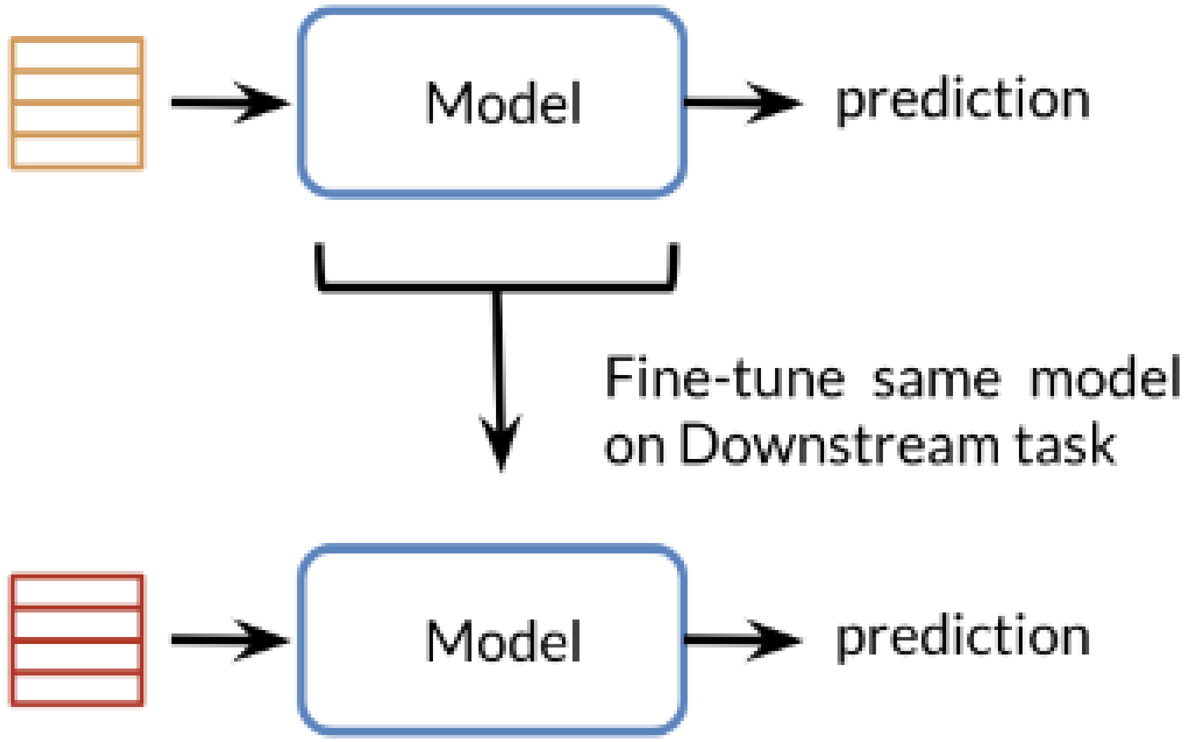
Pre-Training :

Feature Based



train word embeddings and then using those features (i.e. word vectors) on a different task.

Fine Tuning



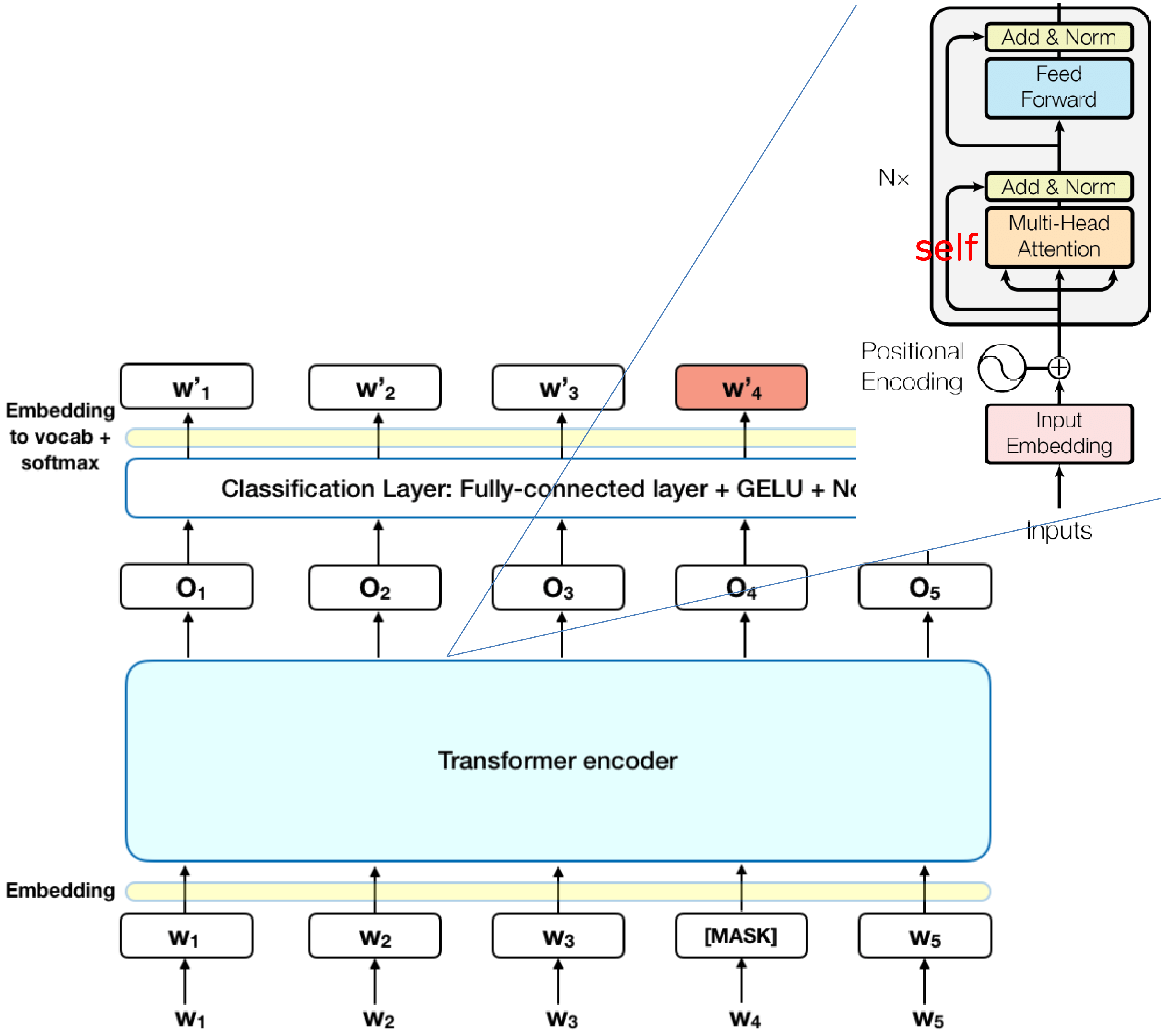
"update" the meaning of words based on your specific domain.

BERT :

0. Architecture

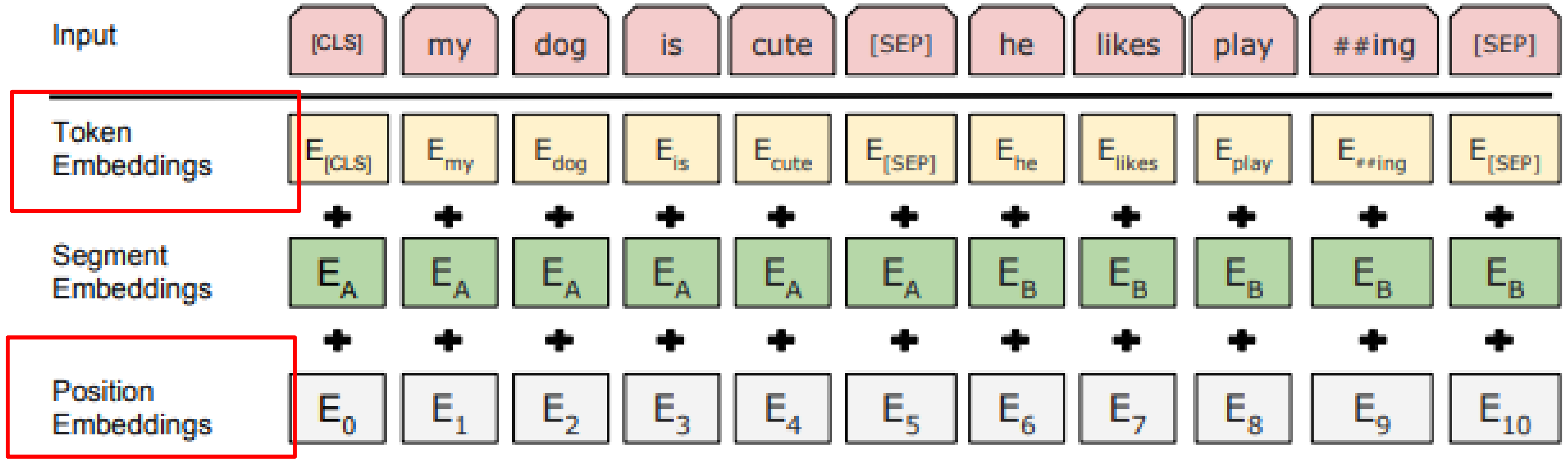
- 초기 트랜스포머 모델 : L = 6, H = 512, A = 8
- BERT-Base : L=12, H=768, A=12
- BERT-Large : L=24, H=1024, A=16

L : num_layers
H : hidden size (d_model)
A : num attention heads



BERT :

1. Embedding



Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{\#\#ing}$	$E_{[SEP]}$
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

WordPiece Embedding

- 토큰이 단어 집합에 존재 => 분리하지 않음

ex) 그, 나, 너, is, want, here,

- 토큰이 단어 집합에 없음 => 토큰을 서브워드로 분리

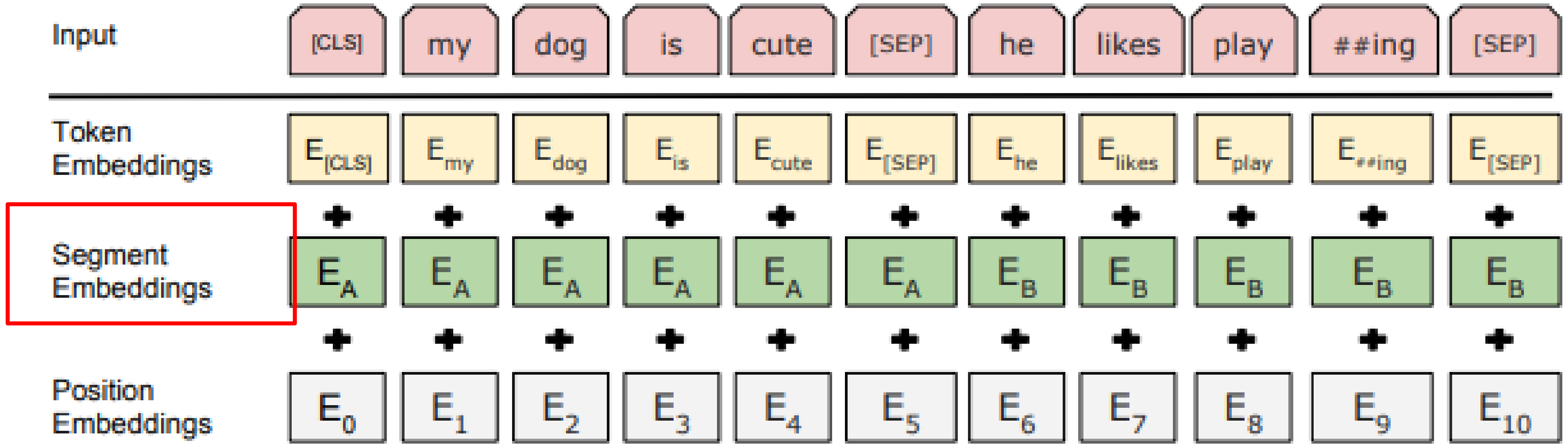
개막공연 : '개막' '##공연 ' , embeddings : 'em','##bed','##ding','##s'

Position Embedding

- Transformer와 유사
- pos을 sin이나 cos함수을 이용하지 않고 별도의 임베딩 층을 만들어 학습을 통해 위치 정보 만든다

BERT

1. Embedding



Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{\#\#ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

Segment Embedding

- 2개의 문장을 구분하기 위한 임베딩

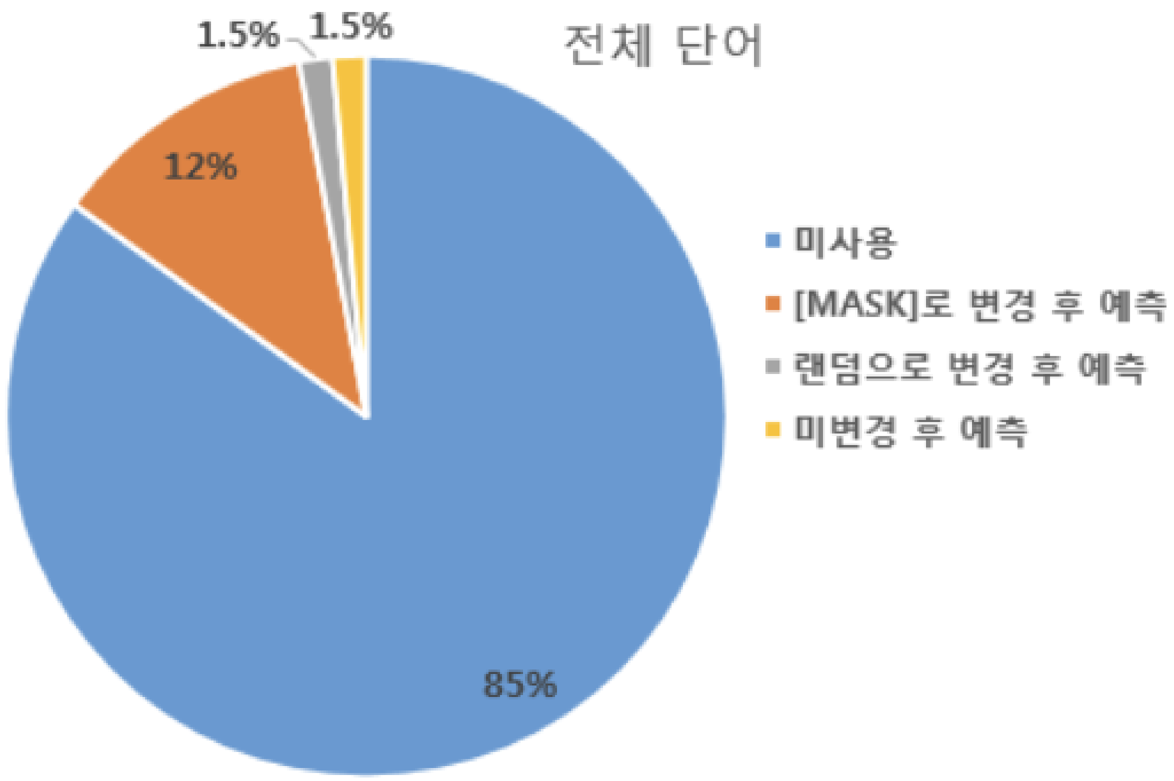
[SEP]에 의해 구분됨

a “sentence” can be an arbitrary span of contiguous text, rather than an actual linguistic sentence

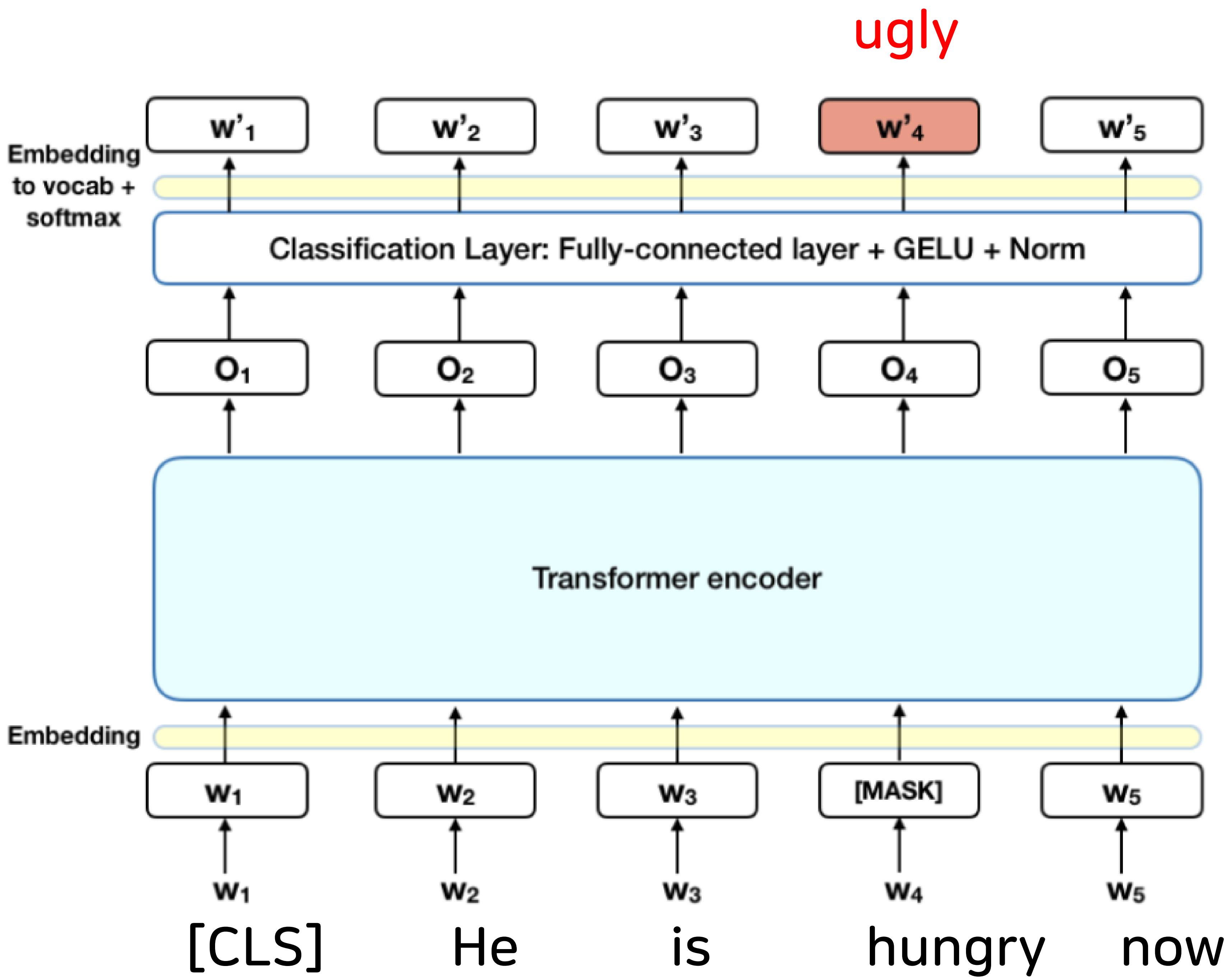
BERT :

2. Pre-training

Masked Language Model



Pre-training과 fine tuning 간의 간극을 해소시키기 위해



W'1
W'2
W'3
W'4
W'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
W1
W2
W3
[MASK]
W5
W1
W2
W3
W4
W5

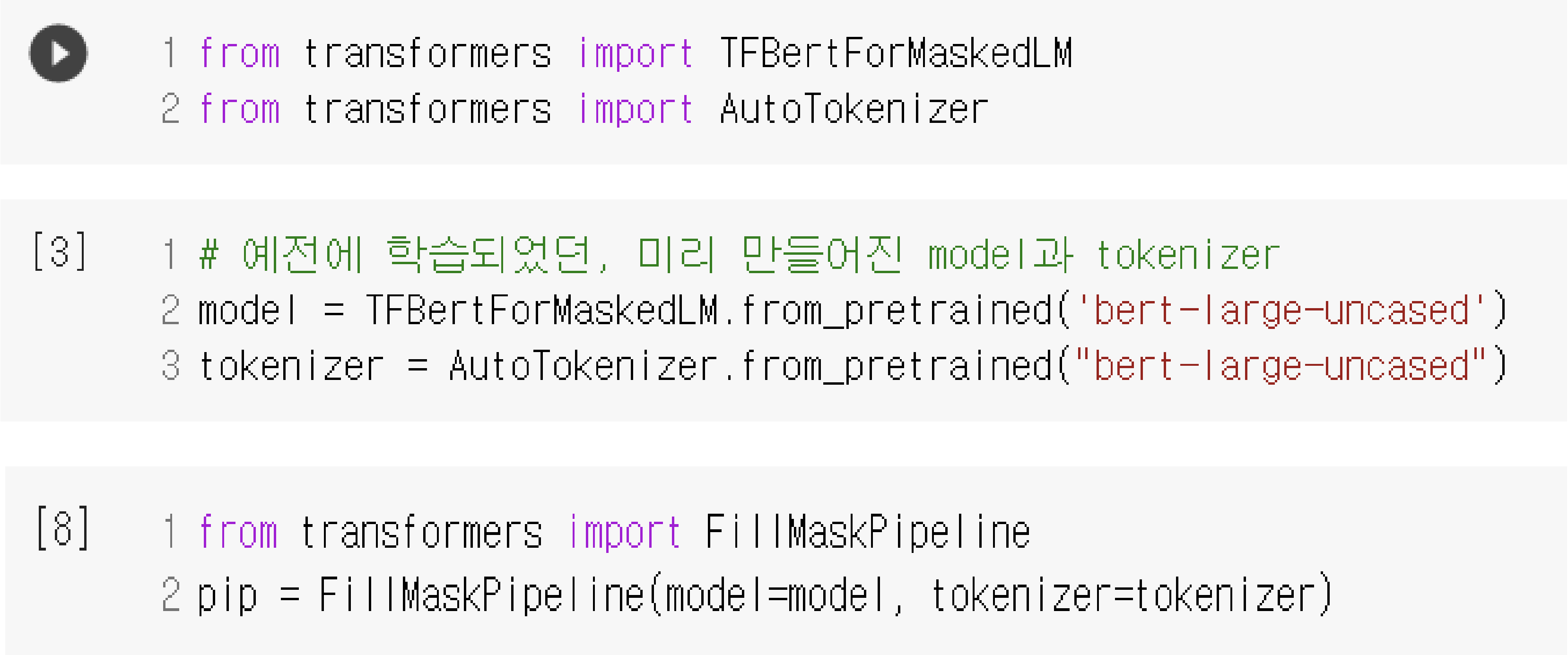
ugly

[CLS] He is hungry now

BERT :

2. Pre-training

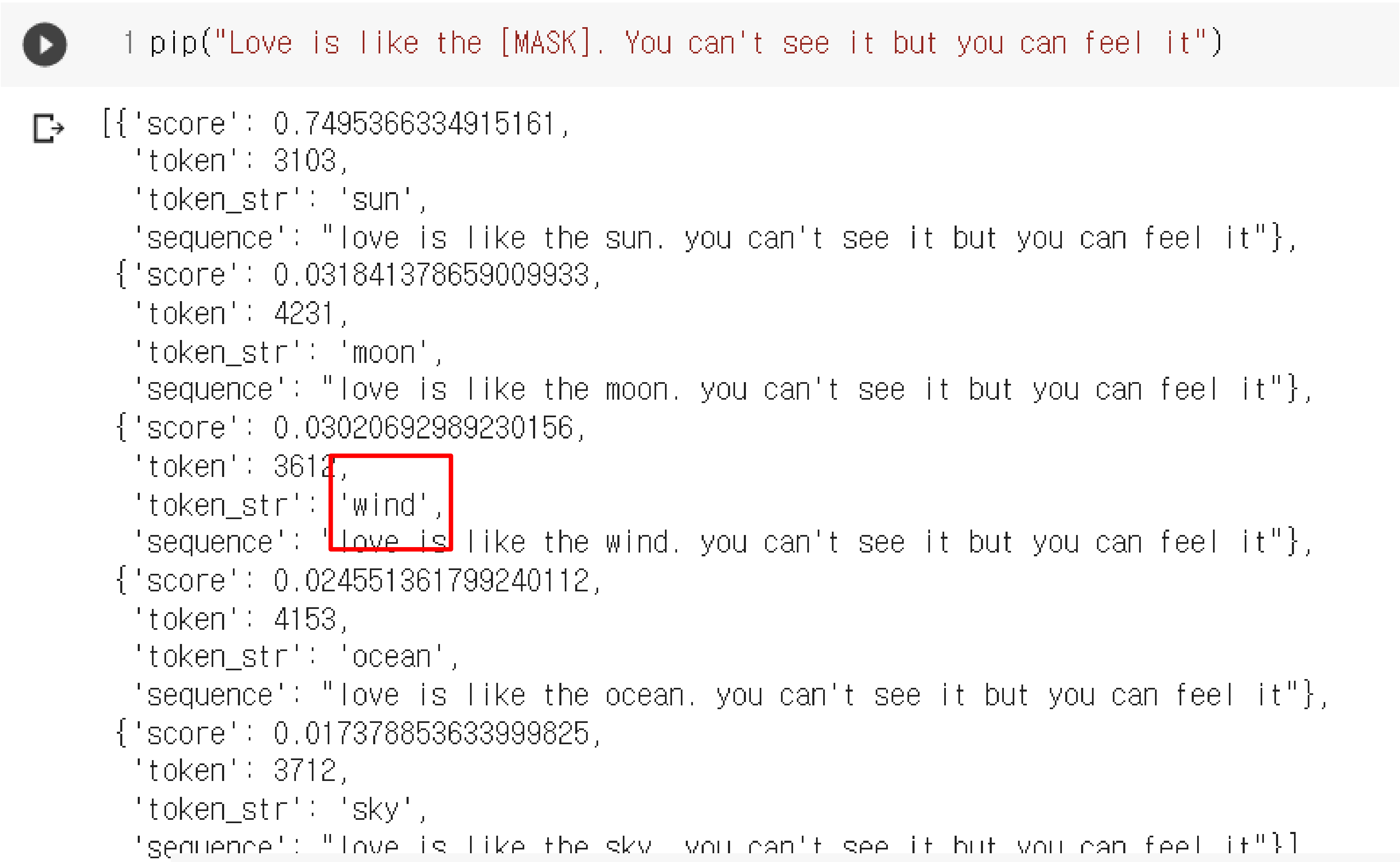
Masked Language Model



```
from transformers import TFBertForMaskedLM
from transformers import AutoTokenizer
```

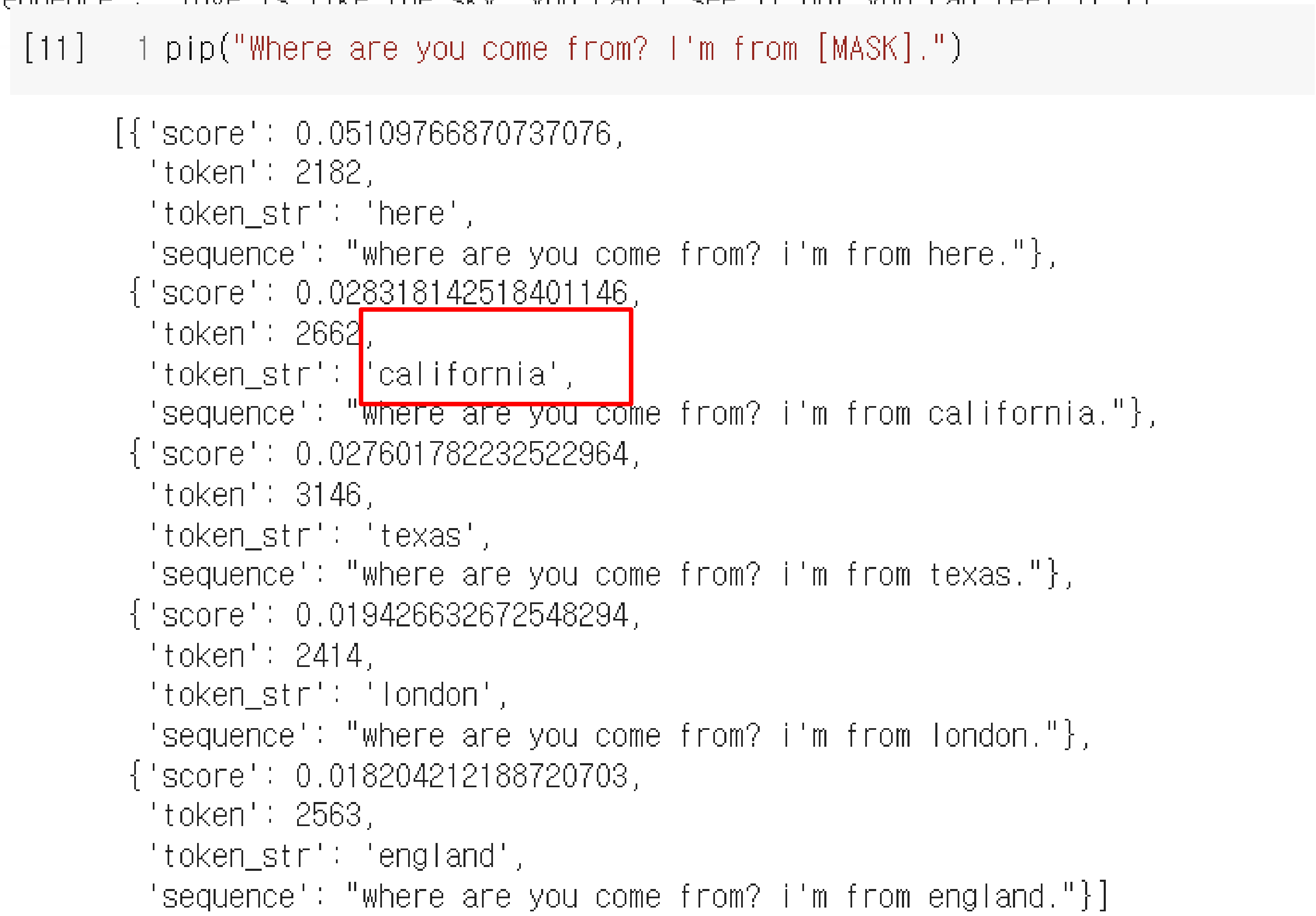
```
[3]  # 예전에 학습되었던, 미리 만들어진 model과 tokenizer
     model = TFBertForMaskedLM.from_pretrained('bert-large-uncased')
     tokenizer = AutoTokenizer.from_pretrained("bert-large-uncased")
```

```
[8]  from transformers import FillMaskPipeline
     pip = FillMaskPipeline(model=model, tokenizer=tokenizer)
```



```
pip("Love is like the [MASK]. You can't see it but you can feel it")
```

```
[{'score': 0.7495366334915161,
  'token': 3103,
  'token_str': 'sun',
  'sequence': "love is like the sun. you can't see it but you can feel it"},
 {'score': 0.031841378659009933,
  'token': 4231,
  'token_str': 'moon',
  'sequence': "love is like the moon. you can't see it but you can feel it"},
 {'score': 0.03020692989230156,
  'token': 3612,
  'token_str': 'wind',
  'sequence': "love is like the wind. you can't see it but you can feel it"},
 {'score': 0.024551361799240112,
  'token': 4153,
  'token_str': 'ocean',
  'sequence': "love is like the ocean. you can't see it but you can feel it"},
 {'score': 0.017378853633999825,
  'token': 3712,
  'token_str': 'sky',
  'sequence': "love is like the sky. you can't see it but you can feel it"}]
```



```
[11]  pip("Where are you come from? I'm from [MASK].")
```

```
[{'score': 0.05109766870737076,
  'token': 2182,
  'token_str': 'here',
  'sequence': "where are you come from? i'm from here."},
 {'score': 0.028318142518401146,
  'token': 2662,
  'token_str': 'california',
  'sequence': "where are you come from? i'm from california."},
 {'score': 0.027601782232522964,
  'token': 3146,
  'token_str': 'texas',
  'sequence': "where are you come from? i'm from texas."},
 {'score': 0.019426632672548294,
  'token': 2414,
  'token_str': 'london',
  'sequence': "where are you come from? i'm from london."},
 {'score': 0.018204212188720703,
  'token': 2563,
  'token_str': 'england',
  'sequence': "where are you come from? i'm from england."}]
```

BERT :

2. Pre-training

Next Sentence Prediction

I am going outside

S1

I will be back after 6

S2

: IsNextSentence

I am going outside

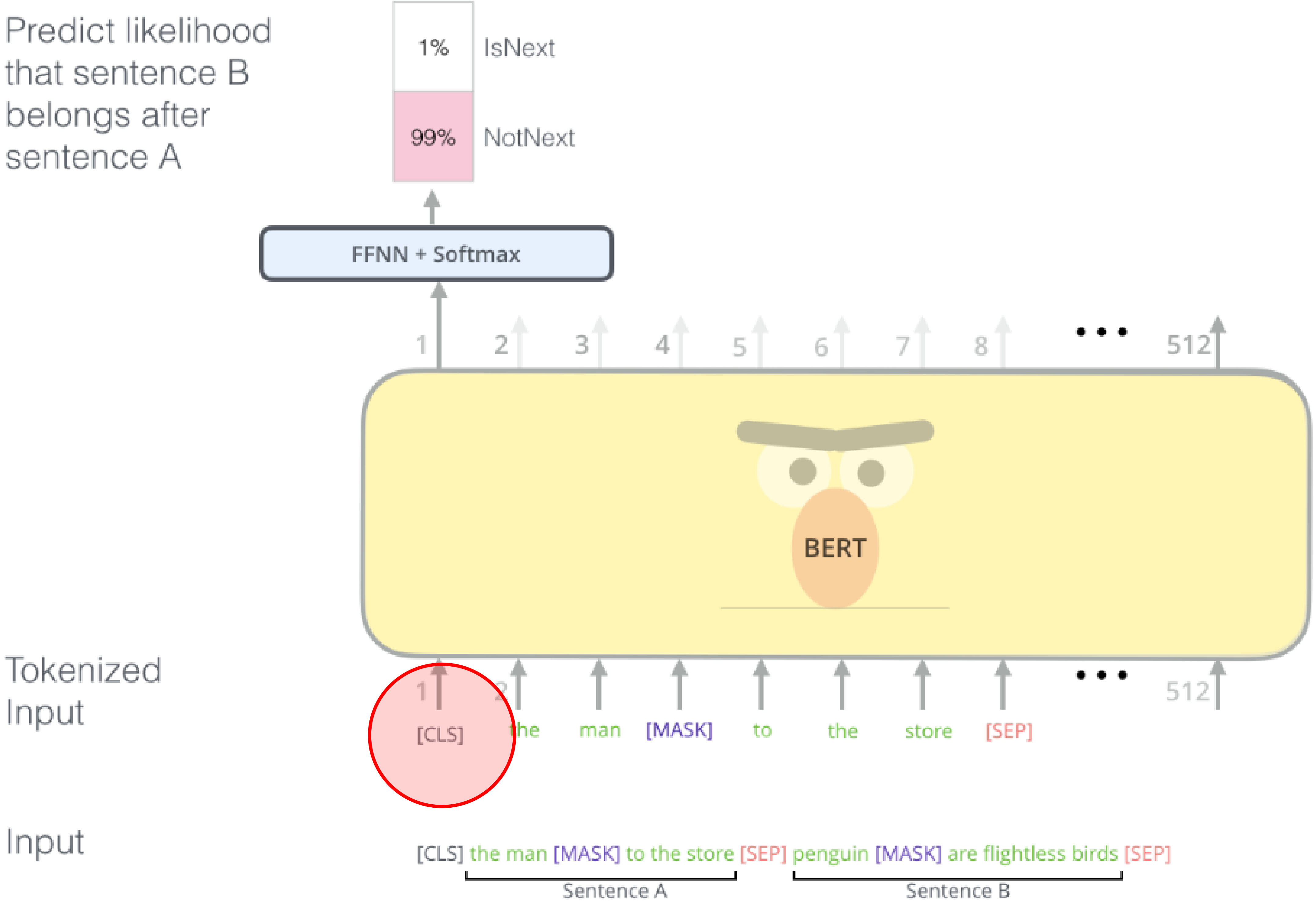
S1

Where are you come from?

S6

: NotNextSentence

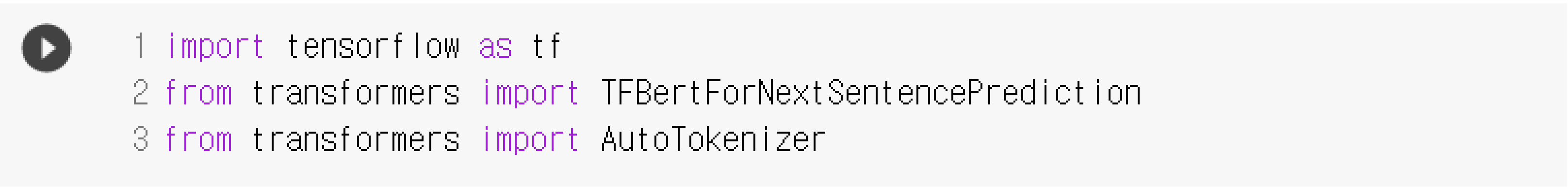
-앞뒤가 이어지는 문장 : 아닌 문장의 비율을 5:5로 설정


Predict likelihood that sentence B belongs after sentence A
1%
IsNext
99%
NotNext
FFNN + Softmax
1
2
3
4
5
6
7
8
512
BERT
Tokenized Input
1
2
512
[CLS]
the
man
[MASK]
to
the
store
[SEP]
Input
[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flightless birds [SEP]
Sentence A
Sentence B

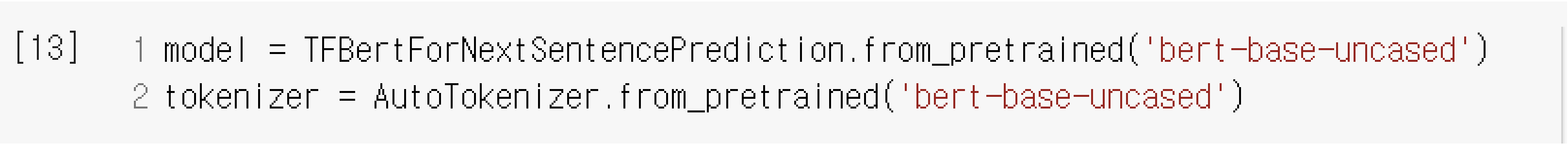
BERT :

2. Pre-training

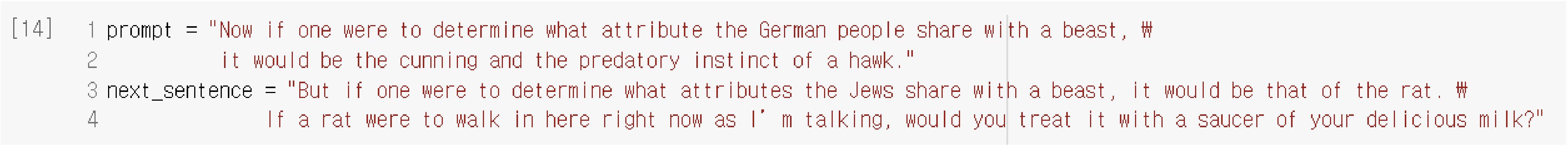
Next Sentence Prediction



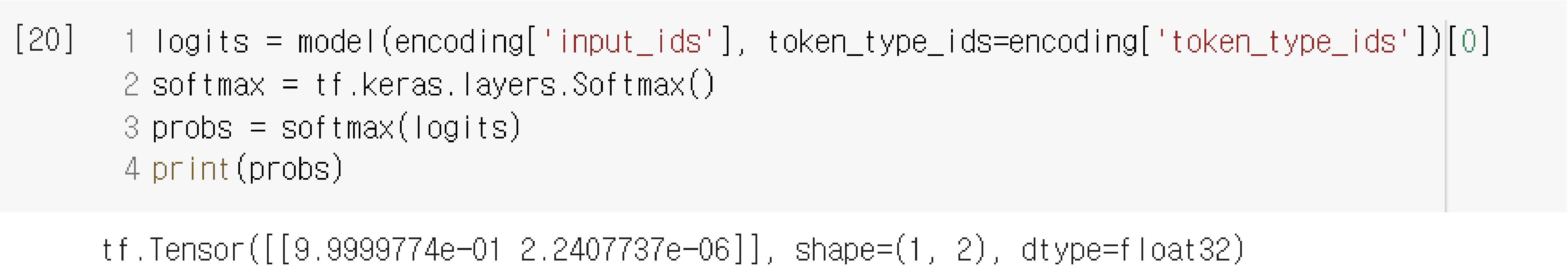
```
import tensorflow as tf
from transformers import TFBertForNextSentencePrediction
from transformers import AutoTokenizer
```



```
[13]
model = TFBertForNextSentencePrediction.from_pretrained('bert-base-uncased')
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')
```

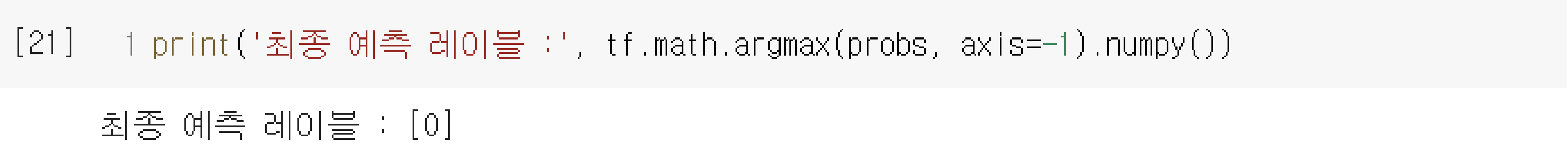


```
[14]
prompt = "Now if one were to determine what attribute the German people share with a beast, ₩
          it would be the cunning and the predatory instinct of a hawk."
next_sentence = "But if one were to determine what attributes the Jews share with a beast, it would be that of the rat. ₩
                 If a rat were to walk in here right now as I’ m talking, would you treat it with a saucer of your delicious milk?"
```



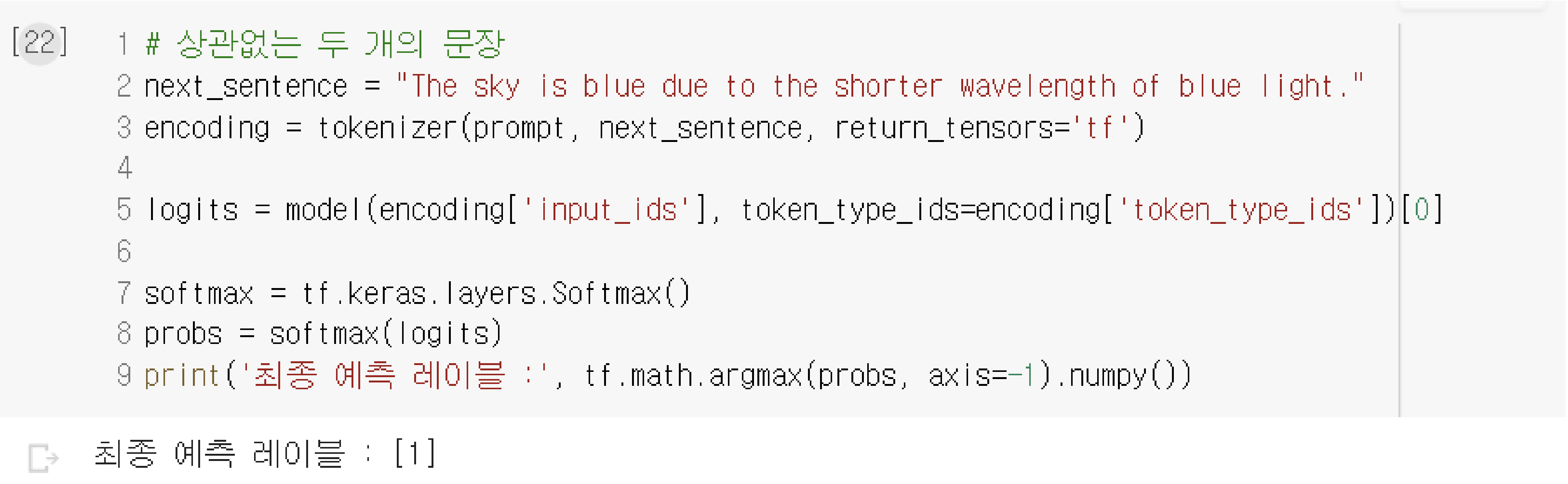
```
[20]
logits = model(encoding['input_ids'], token_type_ids=encoding['token_type_ids'])[0]
softmax = tf.keras.layers.Softmax()
probs = softmax(logits)
print(probs)
```

```
tf.Tensor([[9.9999774e-01 2.2407737e-06]], shape=(1, 2), dtype=float32)
```



```
[21]
print('최종 예측 레이블 :', tf.math.argmax(probs, axis=-1).numpy())
```

```
최종 예측 레이블 : [0]
```



```
[22]
# 상관없는 두 개의 문장
next_sentence = "The sky is blue due to the shorter wavelength of blue light."
encoding = tokenizer(prompt, next_sentence, return_tensors='tf')

logits = model(encoding['input_ids'], token_type_ids=encoding['token_type_ids'])[0]

softmax = tf.keras.layers.Softmax()
probs = softmax(logits)
print('최종 예측 레이블 :', tf.math.argmax(probs, axis=-1).numpy())
```

```
최종 예측 레이블 : [1]
```

BERT :

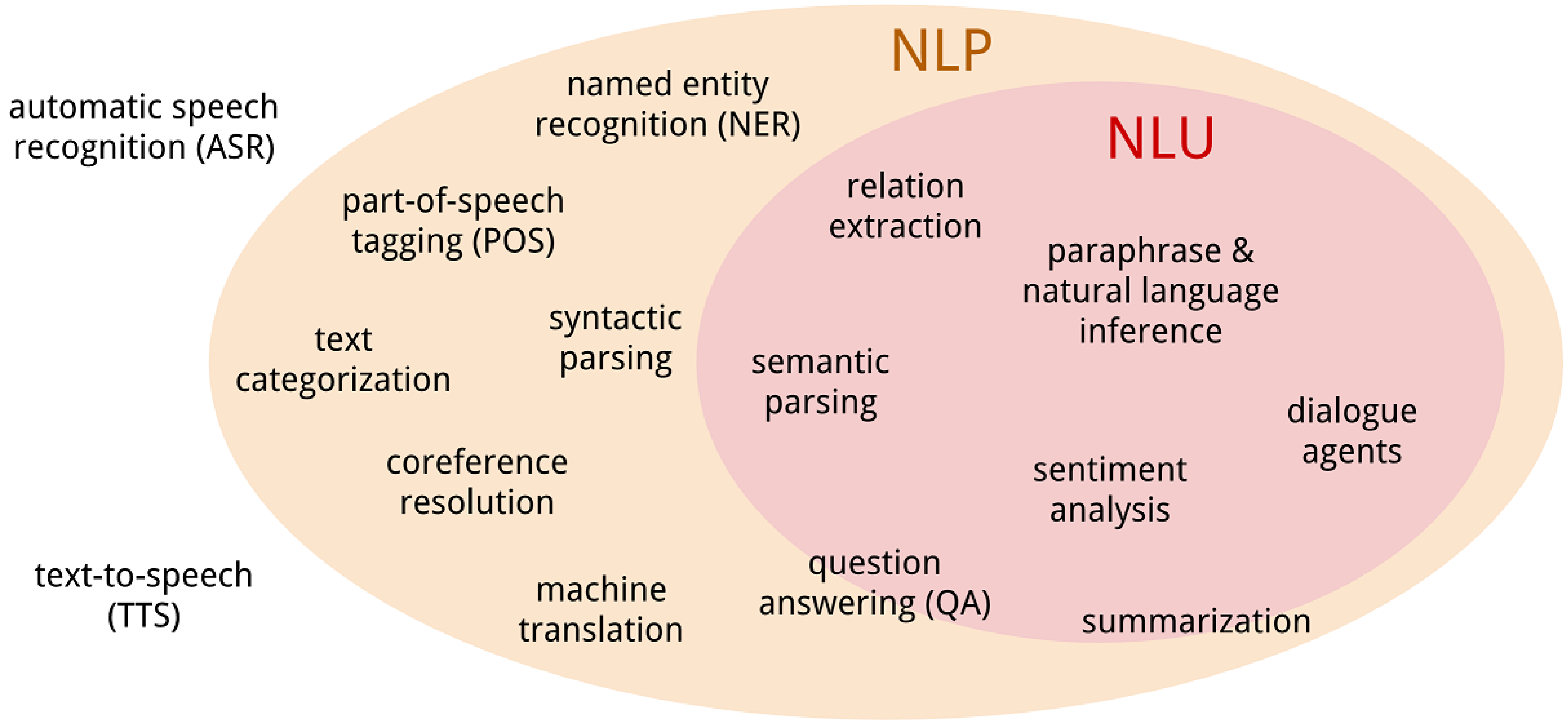
3. Fine-tuning

QA : Question and Answer 질의응답

NER : Named Entity Recognition 개체명 인식

MNLI : Multi-Genre Natural Language Inference

Qa:



NLP

NLU

automatic speech recognition (ASR)

named entity recognition (NER)

part-of-speech tagging (POS)

text categorization

syntactic parsing

coreference resolution

text-to-speech (TTS)

machine translation

relation extraction

paraphrase & natural language inference

semantic parsing

dialogue agents

sentiment analysis

question answering (QA)

summarization