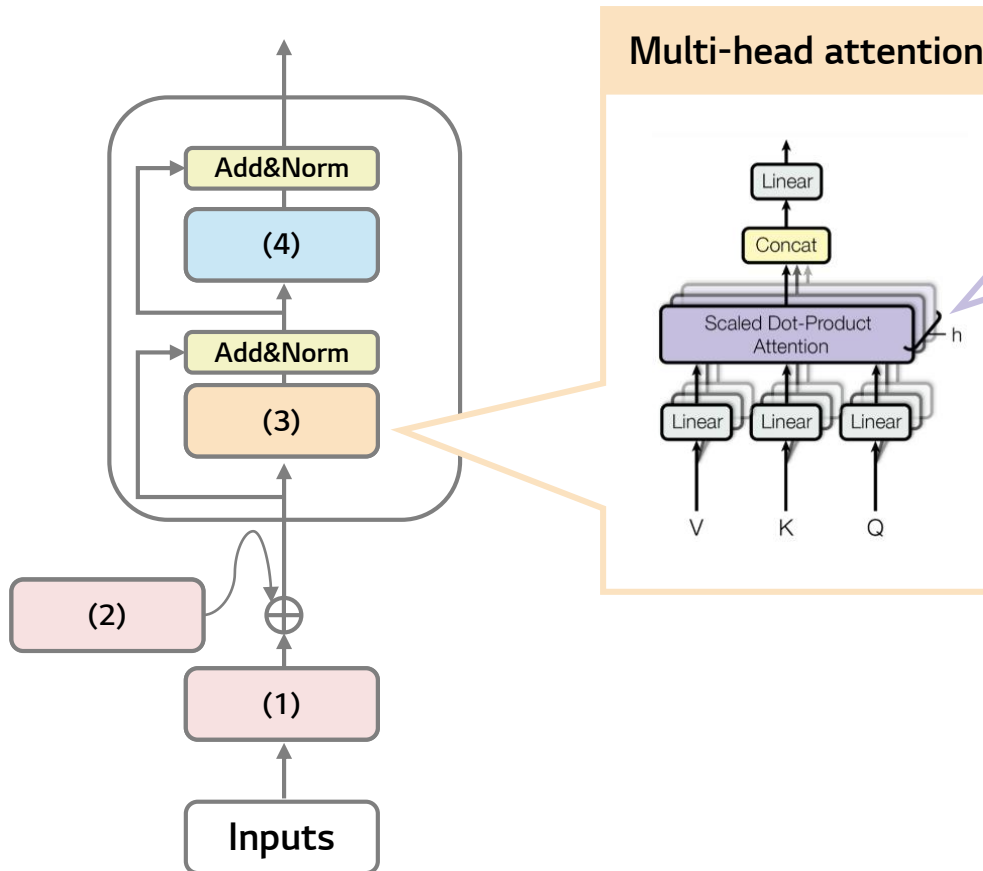


# M9. BERT

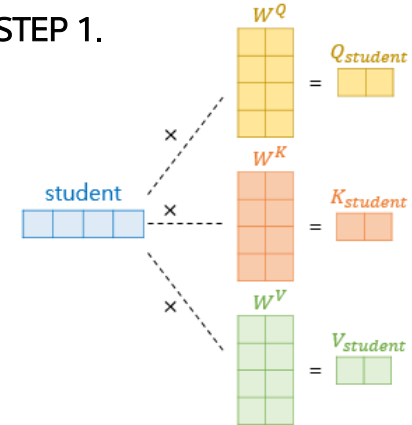
## Remember Transformer?

- Transformer 인코더 구조



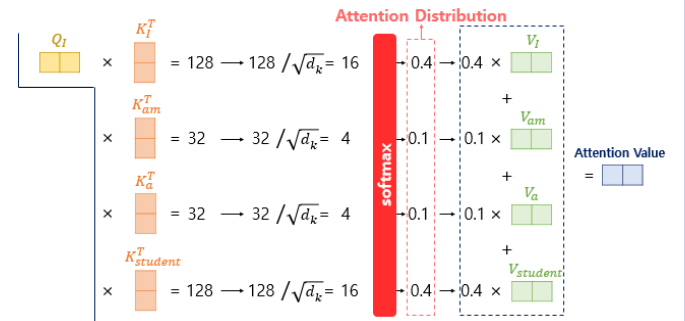
### Scaled dot-product attention

STEP 1.



STEP 2.

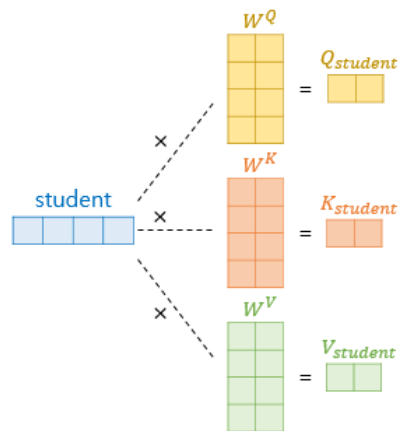
STEP 3.



## Scaled dot-product attention

Input : I am a student

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



**Query** : self attention의 주체.  
'student'라는 단어에 대한 representation이 생성됨

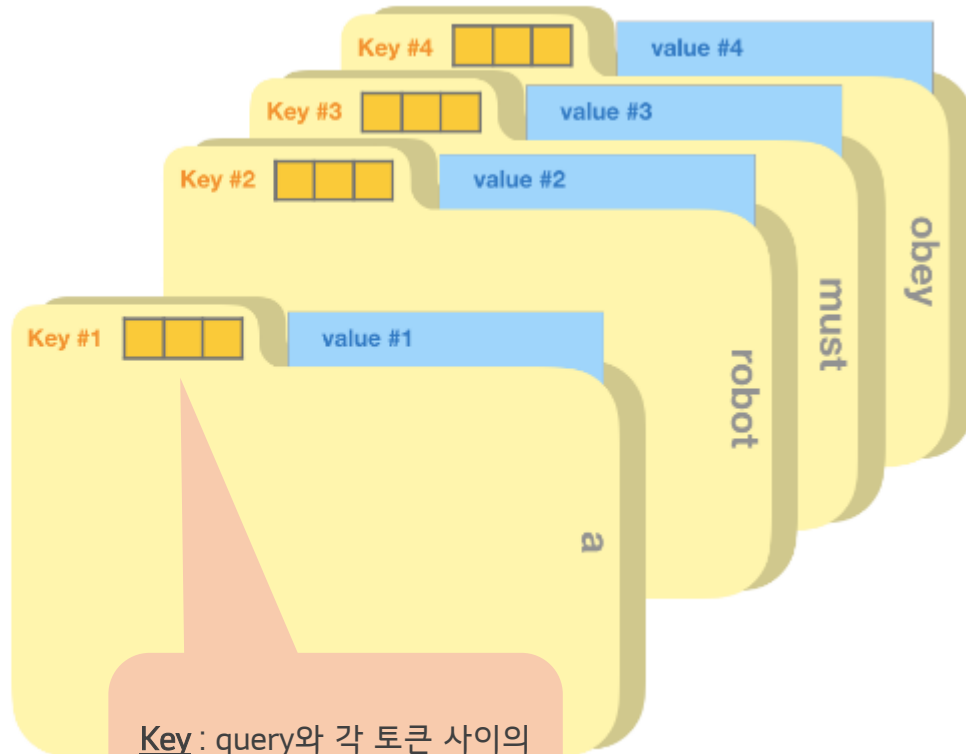
**Key** : query와 문장의 각 토큰 사이의 attention score을 계산하기 위한 벡터.

**Value** : 각 토큰에 대한 새로운 representation  
Attention output은 value에 attention score을 곱하여 계산됨.

## Scaled dot-product attention

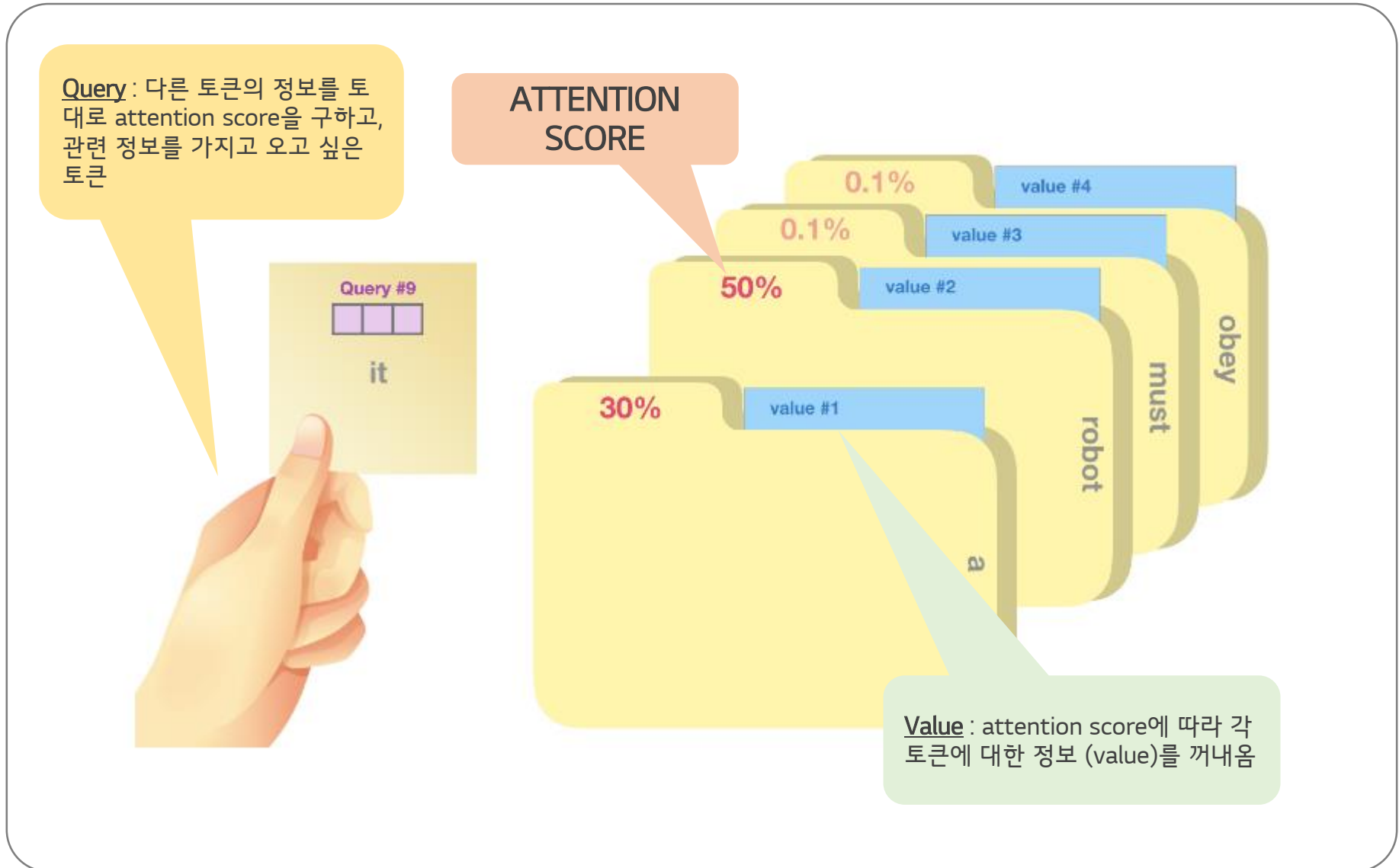
링크: <http://jalammar.github.io/illustrated-gpt2/>

Query : 다른 토큰의 정보를 토대로 attention score을 구하고, 관련 정보를 가지고 오고 싶은 토큰


















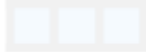



Key : query와 각 토큰 사이의 attention score을 계산하기 위한 벡터.

## Scaled dot-product attention

링크: <http://jalammar.github.io/illustrated-gpt2/>

## Scaled dot-product attention

링크: <http://jalammar.github.io/illustrated-gpt2/>

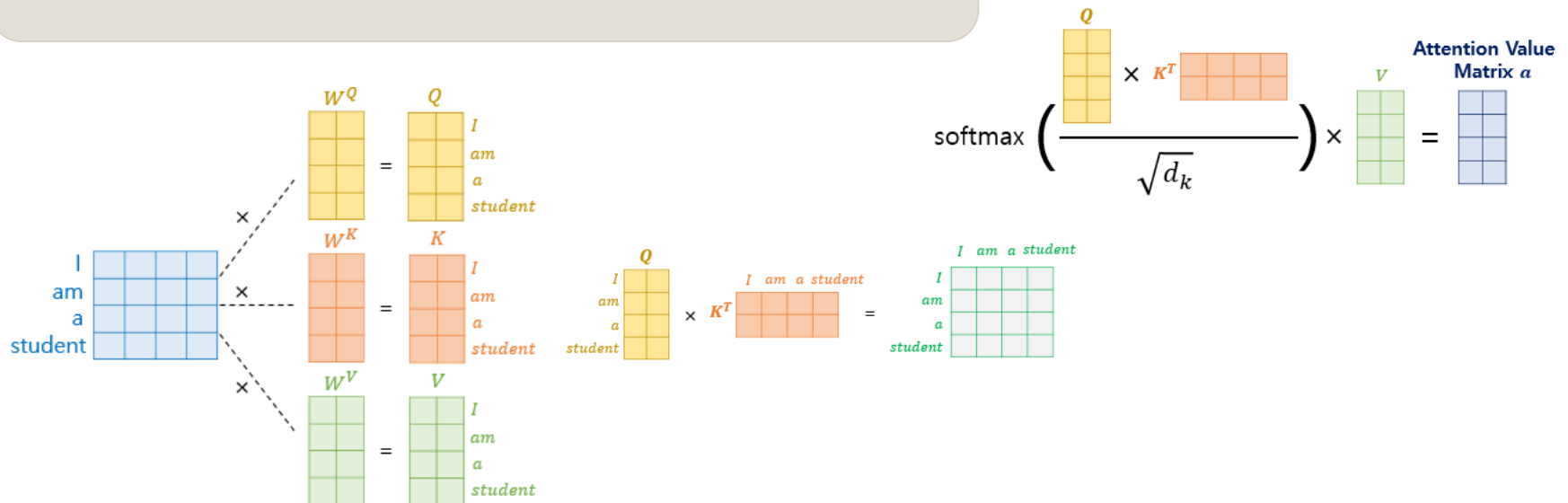
Word	Value vector	Score	Value X Score
<S>		0.001	
a		0.3	
robot		0.5	
must		0.002	
obey		0.001	
the		0.0003	
orders		0.005	
given		0.002	
it		0.19	
		Sum:	

Query 토큰이었던 'it'에 대해  
문맥 전체의 정보를 담아  
업데이트된 representation

## Self attention의 의미

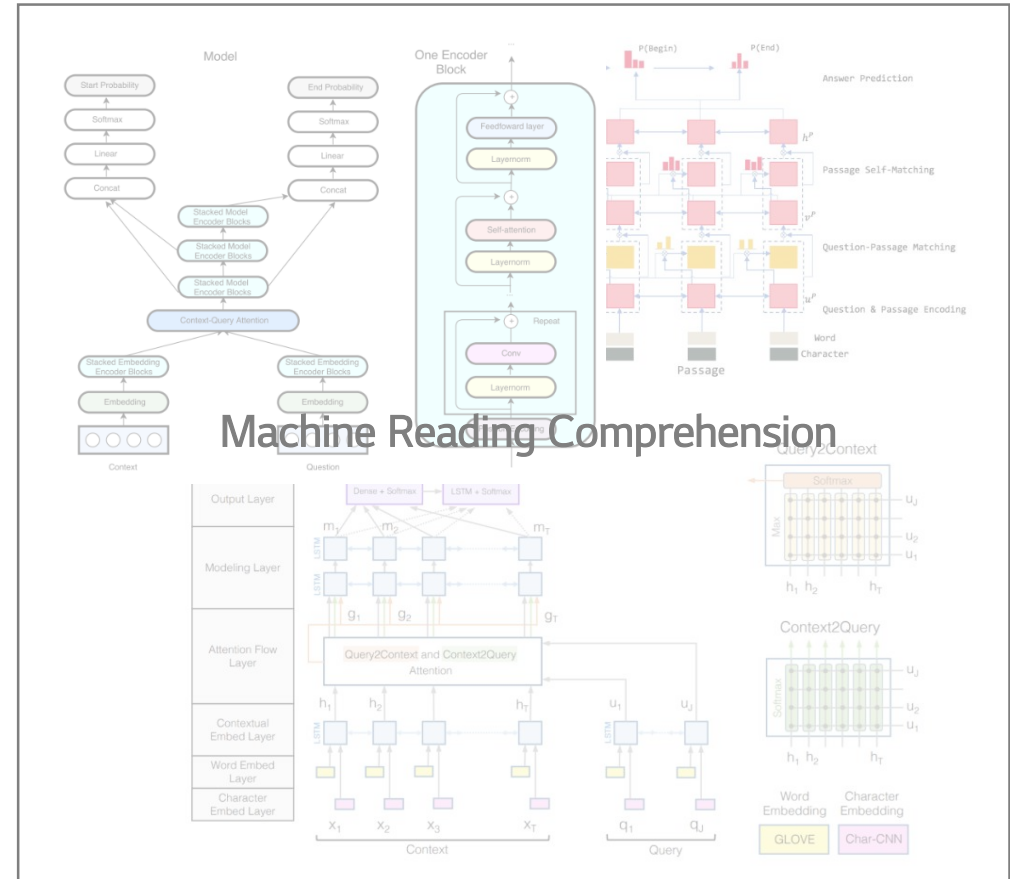
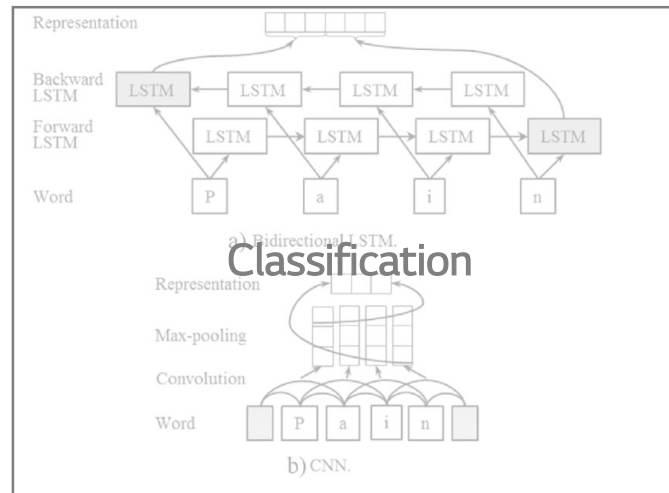
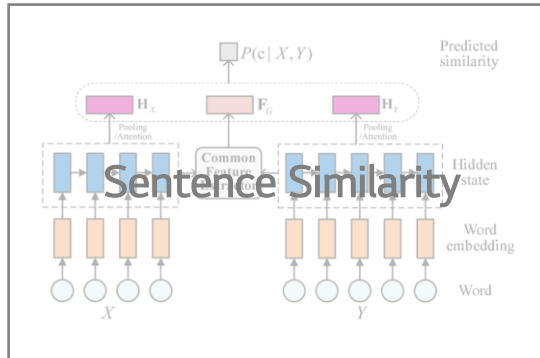
- Self attention은 인풋 시퀀스 전체에 대해 attention을 계산해 각 토큰의 representation을 만들어가는 과정으로, 업데이트된 representation은 **문맥 정보**를 가지고 있다.
- 예를 들어 “아이유는 1993년에 태어났다. 그녀는 최근에 드라마 호텔 델루나에 출연했다” 라는 인풋에 대해 self-attention을 적용하면 “그녀”에 해당하는 representation은 “아이유”에 대한 정보를 담게 된다.

Scaled dot-product attention은 matrix로 계산할 수 있기 때문에 RNN처럼 이전 토큰이 처리되길 기다릴 필요가 없음!



과거에는 각각의 NLU 태스크를 수행하기 위해

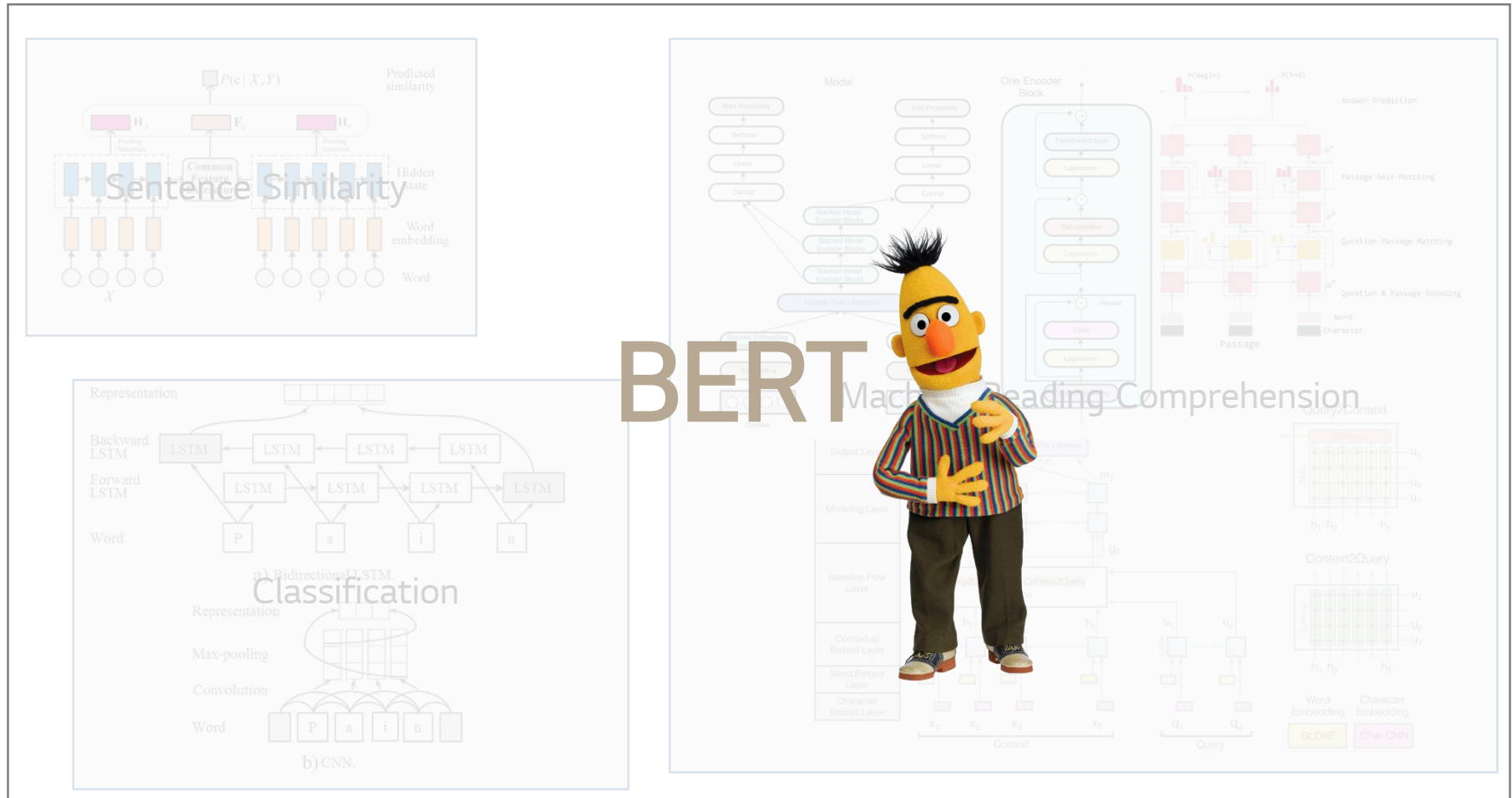
RNN, CNN, Transformer, Attention 등 다양한 알고리즘을 이용해 알맞은 아키텍처를 디자인했다.





## 새로운 패러다임, BERT

: '언어'를 이해하는 거대한 모형을 만들어 놓고 다양한 태스크를 수행할 수 있게 한다.



## 새로운 패러다임, BERT



감성 분석하는 데이터 5만 건을  
통해 CNN으로 인간의 감정을  
이해하게 되었다.



MRC 데이터 10만 건을 공부하  
여 휴먼의 질문에 따른 답변을  
찾아내게 되었다.



의도는 100개가 넘는데...  
학습할 데이터가 너무 적다.  
의도 분류 어려움.



NLU 과제 고득점 비법이요?

음.. 책을 많이 읽었어요.

텍스트를 많이 읽다 보니  
단어가 어떤 의미인지도 자연  
스럽게 알게 되고...  
문맥이 자연스럽지 않다거나  
하는 것도 알겠더라고요.

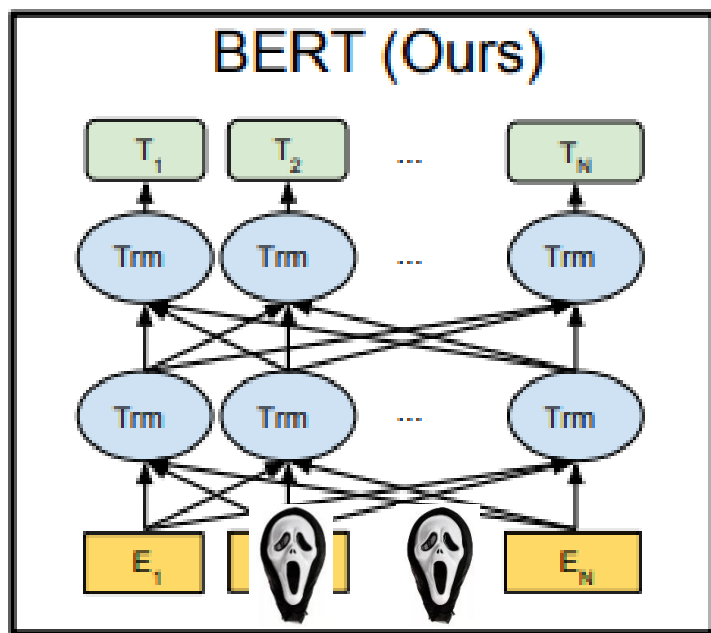


BERT: Bidirectional Encoder Representations from Transformers

- Model 특징
  - Bi-directional
  - Transformer Network 이용
- pre-train 과제 수행시킨 뒤 fine-tuning
  - Masked Language Model
  - Next Sentence Prediction

## BERT 사전학습 과제 1: Masked Language Model

: 가려진 단어를 맞추는 과제를 해결함으로써 주변 맥락에 따른 단어 의미 학습



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	$E_0$	$E_1$	$E_2$	$E_3$	$E_4$	$E_5$	$E_6$	$E_7$	$E_8$	$E_9$	$E_{10}$

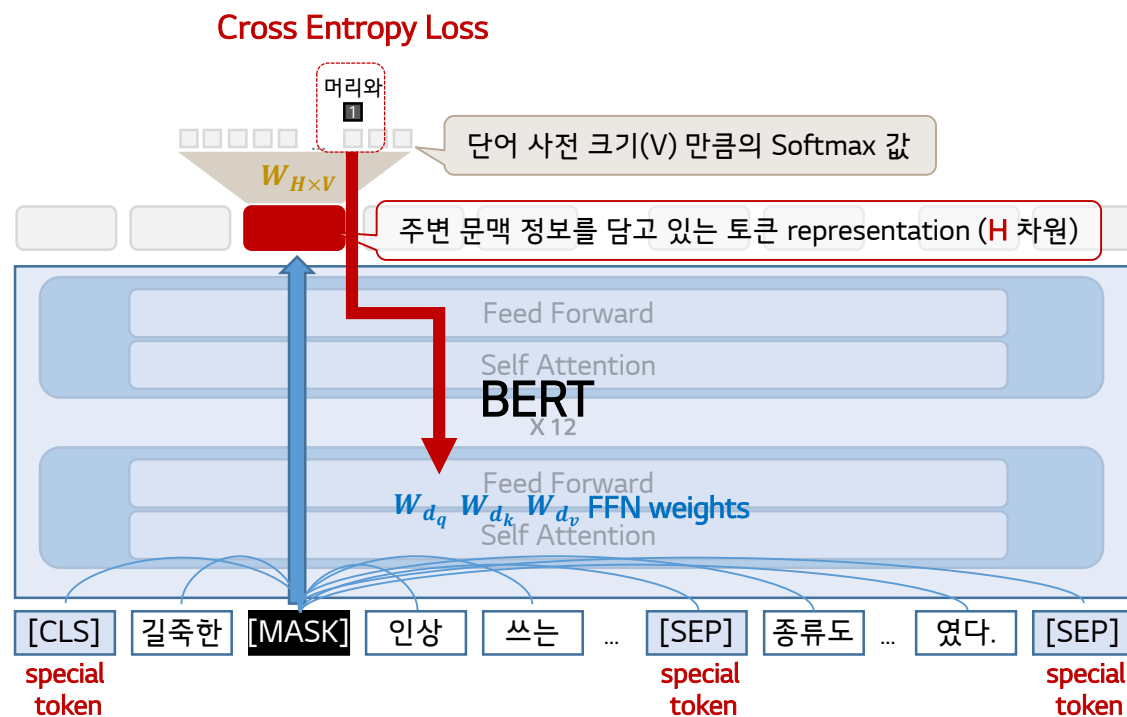
길쭉한 **[MASK]** 인상 쓰는 듯한 **[MASK]** 일자눈썹,  
날씬하고 뺏뺏한 몸통과 세로줄 스웨터, 딱 맞는  
면바지와 깔끔한 새들 슈즈, 위로 삐죽삐죽 솟은  
머리카락 등 어니와는 **[MASK]**의 디자인이다.



길쭉한 **머리와** 인상 쓰는 듯한 **두꺼운** 일자눈썹,  
날씬하고 뺏뺏한 몸통과 세로줄 스웨터, 딱 맞는  
면바지와 깔끔한 새들 슈즈, 위로 삐죽삐죽 솟은  
머리카락 등 어니와는 **정반대**의 디자인이다.

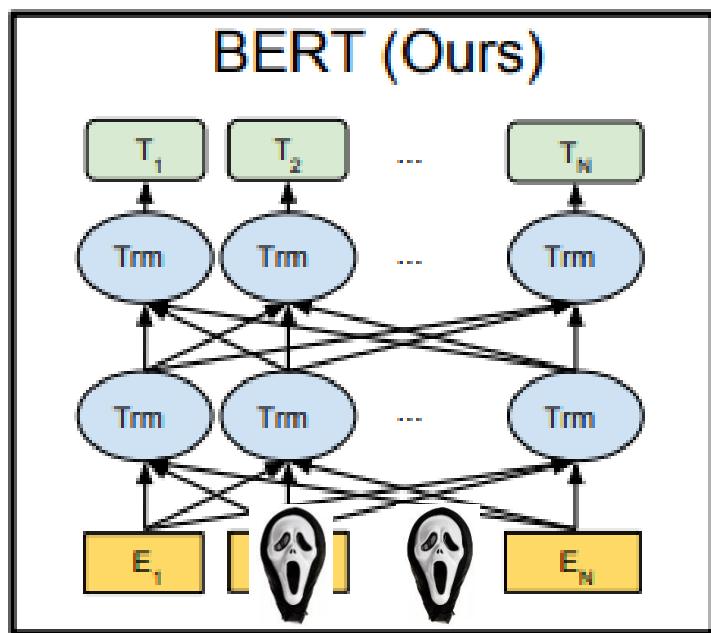
## BERT 사전학습 과제 1: Masked Language Model

: 가려진 단어를 맞추는 과제를 해결함으로써 주변 맥락에 따른 단어 의미 학습



## BERT 사전학습 과제 2: Next Sentence Prediction

: 제시된 두 문장이 이어진 문장인지 아닌지를 맞추는 과제를 수행



[1] 길쭉한 머리와 인상 쓰는 듯한 [MASK] 일자눈썹, 날씬하고 뺏뺏한 몸통과 세로줄 스웨터, 딱 맞는 면바지와 깔끔한 새들 슈즈, 위로 삐죽삐죽 솟은 머리카락 등 어니와는 [MASK]의 디자인이다.

[2] 제작 컨셉부터 똥똥이와 홀쭉이 콤비였다.

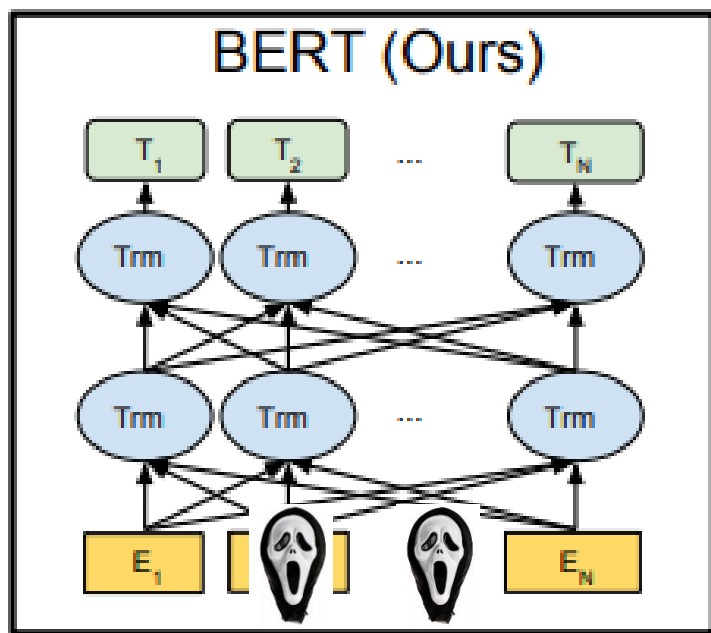


연결된 문단이다  
(Next)

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

## BERT 사전학습 과제 2: Next Sentence Prediction

: 제시된 두 문장이 이어진 문장인지 아닌지를 맞추는 과제를 수행



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	$E_A$	$E_A$	$E_A$	$E_A$	$E_A$	$E_B$	$E_B$	$E_B$	$E_B$	$E_B$	$E_B$
Position Embeddings	$E_0$	$E_1$	$E_2$	$E_3$	$E_4$	$E_5$	$E_6$	$E_7$	$E_8$	$E_9$	$E_{10}$

[1] 길쭉한 머리와 인상 쓰는 듯한 [MASK] 일자눈썹, 날씬하고 뺏뺏한 몸통과 세로줄 스웨터, 딱 맞는 면바지와 깔끔한 새들 슈즈, 위로 삐죽삐죽 솟은 머리카락 등 어니와는 [MASK]의 디자인이다.

[2] 그 남자는 우유를 세 통 집어들었다.

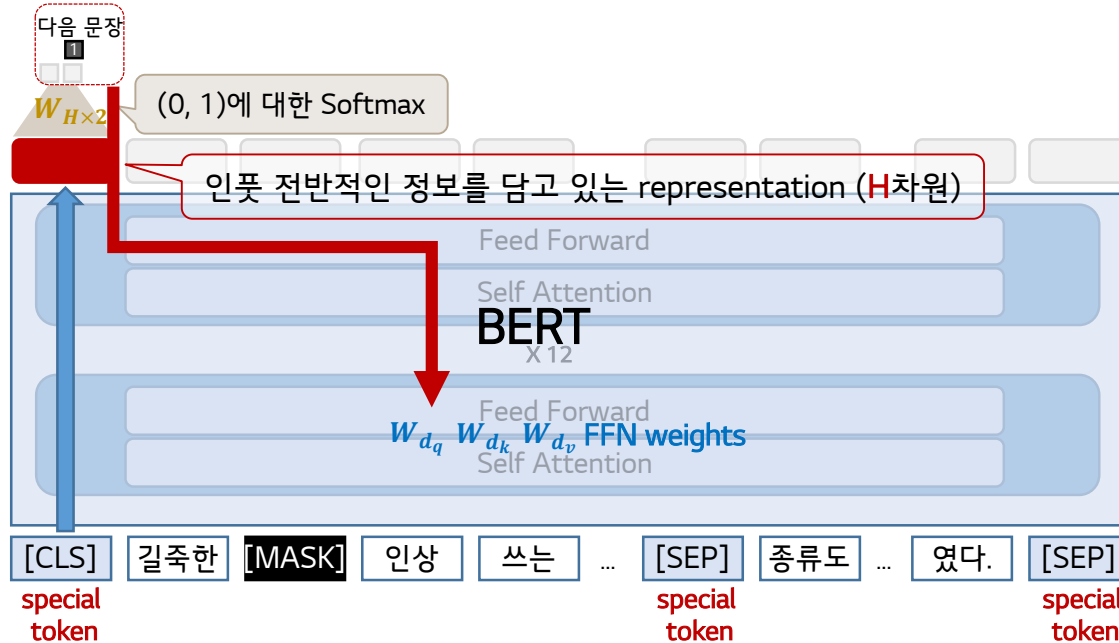


연결된 문단이 아님  
(Not Next)

## BERT 사전학습 과제 2: Next Sentence Prediction

: 제시된 두 문장이 이어진 문장인지 아닌지를 맞추는 과제를 수행

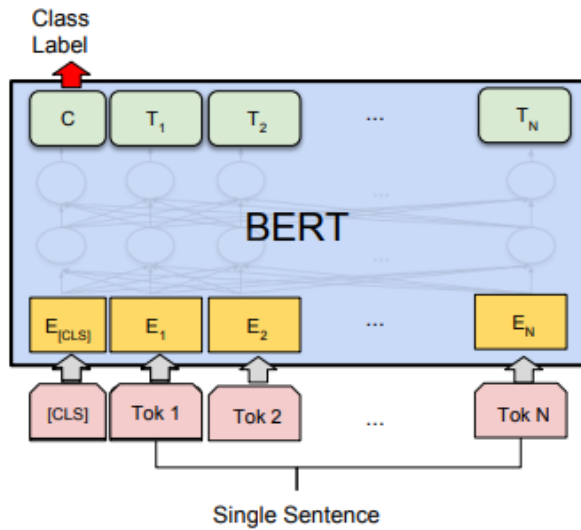
## Cross Entropy Loss



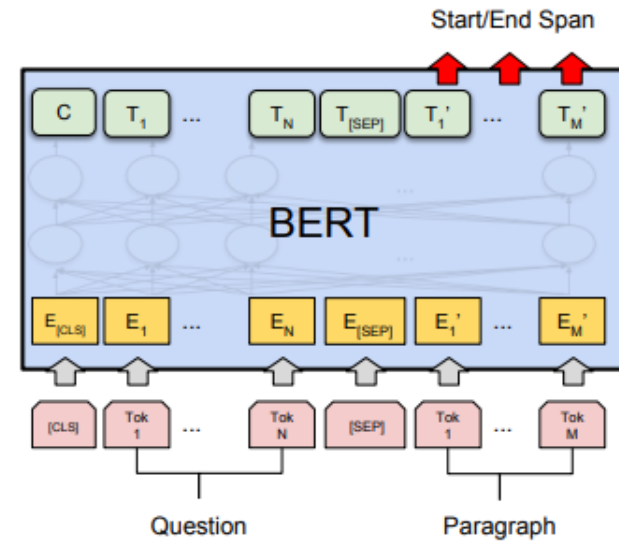


## BERT Fine-tuning

: 주어진 과제 유형에 따라서 마지막 output layer만 변경하여 간단하게 fine-tuning



(b) Single Sentence Classification Tasks:  
SST-2, CoLA



(c) Question Answering Tasks:  
SQuAD v1.1

방대한 양의 데이터로 수행한 사전학습 과제의 힘

- 사전 학습 과제

- 40 epoch, 1,000,000 iterations
- 30억 단어에 달하는 책 (위키피디아 & 소설)을 읽은 셈
- BERT\_base : 4 Cloud TPUs(=16 TPU chips)
- BERT\_large: 16 Cloud TPUs(= 64 TPU chips)
- 엄청난 자원을 사용하여 4일 내내 학습



**SELF  
SUPERVISED  
LEARNING**

- Fine-tuning

- 1~4 epoch만 추가 수행
- 추가학습은 조금만 수행해도 좋은 성능!!! → 사전학습의 위력



- 11개의 Natural Language Processing task에서 State-Of-The-Art 달성 !

## 방대한 양의 데이터로 수행한 사전학습 과제의 힘


### ■ SQuAD 1.1 leaderboard

Rank	Model	EM	F1
	Human Performance Stanford University (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) Google AI Language <a href="https://arxiv.org/abs/1810.04805">https://arxiv.org/abs/1810.04805</a>	87.433	93.160
2 Feb 14, 2019	Knowledge-enhanced BERT (single model) Anonymous	85.944	92.425
2 Sep 26, 2018	nlnet (ensemble) Microsoft Research Asia	85.954	91.677
3 Sep 09, 2018	nlnet (ensemble) Microsoft Research Asia	85.356	91.202

### ■ SQuAD 2.0 leaderboard

Rank	Model	EM	F1
	Human Performance Stanford University (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Mar 20, 2019	BERT + DAE + AoA (ensemble) Joint Laboratory of HIT and iFLYTEK Research	87.147	89.474
2 Mar 15, 2019	BERT + ConvLSTM + MTL + Verifier (ensemble) Layer 6 AI	86.730	89.286
3 Mar 05, 2019	BERT + N-Gram Masking + Synthetic Self-Training (ensemble) Google AI Language <a href="https://github.com/google-research/bert">https://github.com/google-research/bert</a>	86.673	89.147

### ■ KorQuAD 1.0 leaderboard

KorQuAD					1.0 (ENG)	1.0 (한국어)
 The Korean Question Answering Dataset						
Rank	Reg. Date	Model	EM	F1		
-	2018.10.17	Human Performance	80.17	91.20		
1	2019.06.26	LaRva-Kor-Large+ + CLaF (single) Clova AI LaRva Team	86.84	94.75		
2	2019.06.04	BERT-CLKT-MIDDLE (single model) Anonymous	86.71	94.55		
3	2019.06.01			94.37		
4	2019.03.15			94.08		
5	2019.07.10			94.02		
6	2019.05.01	Clova AI LaRva Team (LPT)	85.35	93.96		
7	2019.04.24	LaRva-Kor+ (single) Clova AI LaRva Team (LPT)	85.25	93.94		
8	2019.07.25	Bert-Base-Kor-LEN (ensemble) ChangWook Jun	85.51	93.46		
9	2019.06.29	BERT-DAL-Masking-Morp (single) JunSeok Kim	85.15	93.20		
10	2019.09.20	ETRI BERT (single model) deepfine	84.56	92.91		

53명의 리더보드 참가자 중  
BERT를 이용하지 않은 모델은  
51, 52, 53위 3건



(소개) CNS\_BERT - AI 기술팀에서 자체적으로 사전 학습한 BERT 모델을 확보함

- 인풋 코퍼스

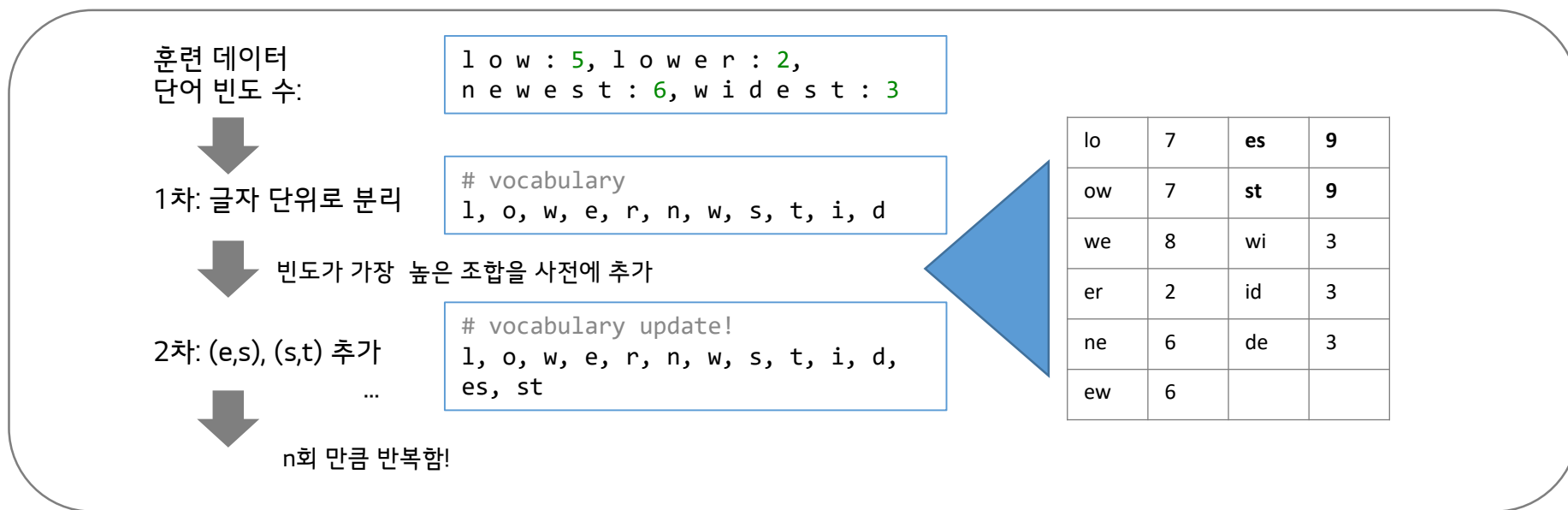
한국어 코퍼스	설명	길이 (문장 단위)
AI HUB 데이터	AI HUB에서 제공하는 한국어 위키 문서	519,006
News	2018년도 뉴스 기사 모음	138,438,851
Wiki	KorQuAD1.0 데이터	4,032,810
Namu Wiki	나무위키 데이터베이스 덤프	19,241,204

- 학습 디테일

- 32 core TPU 16개로 1,000,000 스텝 동안 사전학습
- 10,000 스텝마다 모델을 저장하며 performance를 체크함
- 기타 하이퍼 파라미터는 기본 BERT 논문의 설정에 따름

-> Google이 공개한 모델 대비 의도분류, 기계독해 등 다양한 태스크에서 우월한 성능을 보임

## Byte Pair Encoding



<https://wikidocs.net/22592>

- 빈도수가 높은 문자의 조합으로 parsing하는 방법
- OOV의 발생 빈도가 비교적 낮아짐.

## WordPiece Tokenization

ORIGINAL TOKENIZATION	WORDPIECES
Leicestershire	Leicester ##shire
beat	beat
Somerset	Somerset
by	by
an	an
innings	innings

- **likelihood가 높은 문자의 조합**으로 parsing하는 방법
- ## / @@ 와 같은 기호를 사용하여 띄어쓰기 단위로 단어를 구분할 수 있음
- 한국어의 경우 형태소 분석 후 WordPiece 토큰나이징을 수행하는 것이 더 좋은 것으로 알려져 있음.

## BERT Vocab 사전 소개

## WordPiece Tokenization

- Google에서 공개한 한국어가 지원되는 multi-lingual 모델에는 119,547개 단어가 포함되어 있음

1	[PAD]	0번 index는 패딩에 사용	100	[unused99]	119541	##。
2	[unused1]		101	[UNK]	119542	##「
3	[unused2]		102	[CLS]	119543	##」
4	[unused3]		103	[SEP]	119544	##、
5	[unused4]		104	[MASK]	119545	##·
6	[unused5]		105	<S>	119546	##鯨
7	[unused6]		106	<T>	119547	##鰻

- 1~99번 자리는 [unused] 토큰 자리로 비워두어 모델에 자유도를 줌
- > 도메인에 따라 특정 단어를 추가해야 할 때 이 자리에 추가하면 fine-tuning 과정에서 학습됨
- 다국어 모델이기 때문에 한국어 뿐만 아니라 한자, 일본어 등 다양한 언어 문자가 포함되어 있음.

## BERT Vocab 사전 소개

Byte Pair Encoding

- CNS\_BERT : 99,694개 단어 포함

1	[PAD]	0번 index는 패딩에 사용	100	[unused0]	200	[UNK]	사전에 없는 단어 처리
2	<html>		101	[unused1]	201	[CLS]	} Special 토큰들
3	</html>		102	[unused2]	202	[SEP]	
4	<strong>		103	[unused3]	203	[MASK]	
5	</strong>		104	[unused4]	204	<S>	
6	<head>		105	[unused5]	205	<T>	
7	</head>		106	[unused6]	206	워 커	
					207	의협	

- 웹 문서에 대한 적용을 위해 HTML 태그를 포함하여 사전학습을 진행함
- 99~199번 자리는 unused 토큰 자리로 남겨두어 자유도를 줌.
- 한국어 코퍼스에 대해 BPE를 통해 구축한 사전으로, 한국어 자연어처리에 더욱 최적화되어 있음.



## BERT 토큰나이징 결과 예시

원본 문장

"티라노사오로스와 트라케토사이루스 싸우면 누가 이김?"



MeCab

['티라', '노사', '오로스', '랑', '트라', '케토사이루스', '싸우', '면', '누가', '이김', '?']

BERT  
(Google)['티', '##라', '노', '##사', '오', '##로', '##스', '랑', '트', '##라', '[UNK]',  
'싸', '##우', '면', '누', '##가', '이', '##김', '?']

## BERT 토큰나이징 결과 예시

원본 문장

"티라노사오로스랑 트라켄토사이루스 싸우면 누가 이김?"



MeCab

['티라', '노사', '오로스', '랑', '트라', '켄토사이루스', '싸우', '면', '누가', '이김', '?']

BERT  
(CNS\_BERT)['티라', '노사', '오@@', '로스', '랑', '트라', '켄@@', '토@@', '사이@@', '루스',  
'싸우', '면', '누가', '이@@', '김', '?']

## 사전학습 모델

미니\_실습4\_BERT\_토큰나이저\_실습.ipynb

: 구글은 사전 학습한 BERT 모델과 코드를 깃허브를 통해 오픈 소스로 공개함

영문 모델

- **BERT-Large, Uncased (Whole Word Masking)** : 24-layer, 1024-hidden, 16-heads, 340M parameters
- **BERT-Large, Cased (Whole Word Masking)** : 24-layer, 1024-hidden, 16-heads, 340M parameters

다국어 모델

- **BERT-Base, Multilingual Cased** : 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters

중국어 모델

- **BERT-Base, Chinese** : Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-heads, 110M parameters

Smaller BERT 모델

	H=128	H=256	H=512	H=768
L=2	2/128 (BERT-Tiny)	2/256	2/512	2/768
L=4	4/128	4/256 (BERT-Mini)	4/512 (BERT-Small)	4/768
L=6	6/128	6/256	6/512	6/768
L=8	8/128	8/256	8/512 (BERT-Medium)	8/768
L=10	10/128	10/256	10/512	10/768
L=12	12/128	12/256	12/512	12/768 (BERT-Base)

## 사전학습 모델

: TensorFlow Hub에서 사전 학습된 모델을 쉽게 가져다 사용할 수 있다.

The screenshot displays the TensorFlow Hub interface with the search term 'bert'. The left sidebar contains filters for Problem domain, Model format (TF.js, TFLite, Coral), TF Version (TF1, TF2), Fine tunable, Architecture, Publisher, Dataset, and Language. The main area shows a grid of nine pre-trained BERT models, each with a description, publication date, and associated datasets.

Model Name	Task	Published by	Updated	Dataset
bert-uncased-tf2-qa	Text question answering	See--	02/25/2020	Natural Questions
bert_zh_L-12_H-768_A-12	Text embedding	TensorFlow	05/02/2020	Wikipedia
bert_multi_cased_L-12_H-768_A-12	Text embedding	TensorFlow	05/02/2020	Multilingual Wikipedia
bert_en_cased_L-12_H-768_A-12	Text embedding	TensorFlow	05/02/2020	Wikipedia and BooksCorpus
bert_en_cased_L-24_H-1024_A-16	Text embedding	TensorFlow	05/01/2020	Wikipedia and BooksCorpus
bert_en_uncased_L-12_H-768_A-12	Text embedding	TensorFlow	05/02/2020	Wikipedia and BooksCorpus
bert_en_uncased_L-24_H-1024_A-16	Text embedding	TensorFlow	05/02/2020	Wikipedia and BooksCorpus
bert_en_wwm_cased_L-24_H-1024_A-16	Text embedding	TensorFlow	04/30/2020	Wikipedia and BooksCorpus
bert_en_wwm_uncased_L-24_H-1024_A-16	Text embedding	TensorFlow	05/01/2020	Wikipedia and BooksCorpus

## BERT 인풋 살펴보기

: BERT 모델은 다음과 같은 3개의 인풋을 받음.

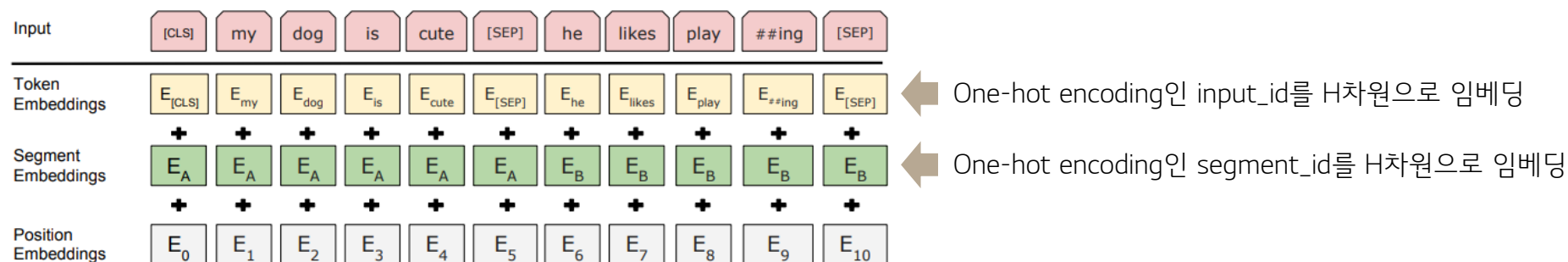
Input	input_ids	[CLS]    문장 1    [SEP]    문장 2    [SEP]		
	segment_ids	0, 0, ...,	0 ,	1, 1 ,    ... , 1
	input_mask	1, 1, ... ,	1 ,	1, 1 , ...    , 0

- Input IDs : 인풋 텍스트를 단어사전을 이용해 id로 변환한 리스트
  - 텍스트는 스페셜 토큰 [CLS] 에 해당하는 id로 시작함
  - 속성이 다른 텍스트 사이 혹은 인풋이 끝났을 때는 [SEP] 토큰 사용
- Segment IDs : 텍스트의 속성을 0 혹은 1로 나타내주기 위한 id
  - 예를 들어 두 개의 문장의 유사도를 측정하는 태스크라면 첫 번째 문장 부분은 segment\_id 0, 두 번째 문장 부분은 segment\_id 1의 값을 가짐
- Input Masks : padding으로 처리한 부분은 0의 값을 가지도록 마스킹
  - Self-attention을 계산할 때 Mask = 0인 부분에는 가중치를 주지 않음.

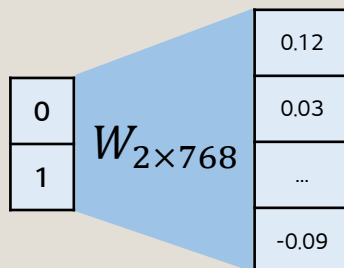
## BERT 인풋 살펴보기

: BERT 내부적으로 input\_id와 segment\_id는 임베딩을 통해 변환됨.

추가적으로 토큰의 위치 정보를 담은 positional embedding이 생성됨



Quiz) BERT의 hidden dimension을 768차원으로 설정했다면, segment embedding을 생성하는 weight matrix의 차원은?

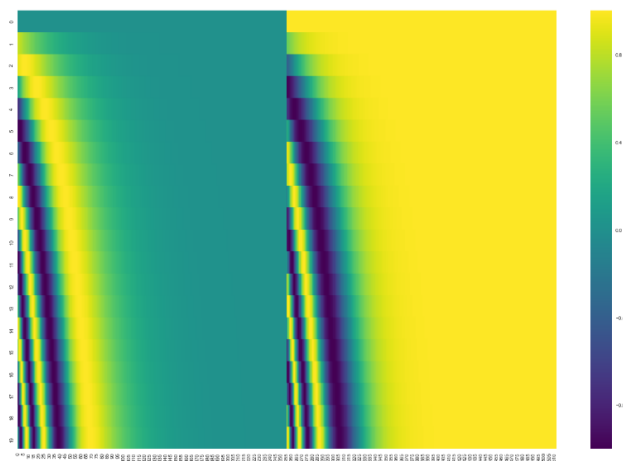


## Position Embedding

: Transformer 논문에서 sinusoidal encoding을 사용한 것과 달리  
BERT에서는 position embedding도 “**학습**” 함.

### Transformer

sin & cos 함수를 사용해  
위치에 따른 인코딩을 만들어냄

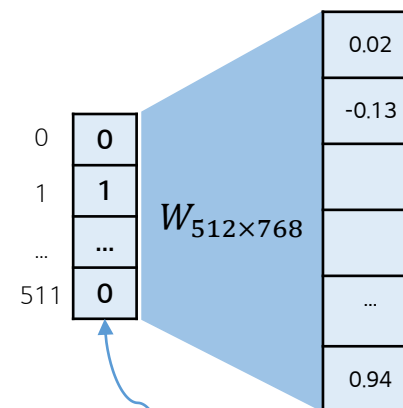


$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

### BERT

학습을 통해  
위치에 대한 임베딩을 만들어냄



One-hot encoded position vector

## BERT fine-tuning

- 사전 학습된 weight를 transfer learning을 이용해 downstream task에 fine-tuning

Downstream task에 대해  
fine-tuning할 작은 부분

인풋 전반적인 정보를 담고 있는  
representation ( $H$ 차원)

문맥의 의미를 고려한  
Tok'2에 대한 representation ( $H$ 차원)

C T1 T2 [ ] T<sub>[SEP]</sub> T'1 T'2 [ ] T'<sub>[SEP]</sub>

사전학습을 통해  
input 시퀀스를  
잘 represent할 수 있는 큰 모델 학습

Input_ids	[CLS]	Tok 1	Tok 2	...	[SEP]	Tok 1	Tok 2	...	[SEP]
Segment_ids	0	0	0	0	0	1	1	...	1
Input_masks	1	1	1	1	1	1	1	1	1



## BERT fine-tuning

- 사전 학습된 weight를 transfer learning을 이용해 downstream task에 fine-tuning

해결하고자 하는 태스크에 따라  
Fully Connected Layer을 디자인하여  
BERT가 만들어낸 representation을 예측 값으로 매핑

인풋 전반적인 정보를 담고 있는  
representation ( $H$ 차원)

문맥의 의미를 고려한  
Tok'2에 대한 representation ( $H$ 차원)

C T1 T2 [SEP] T'1 T'2 T'[SEP]

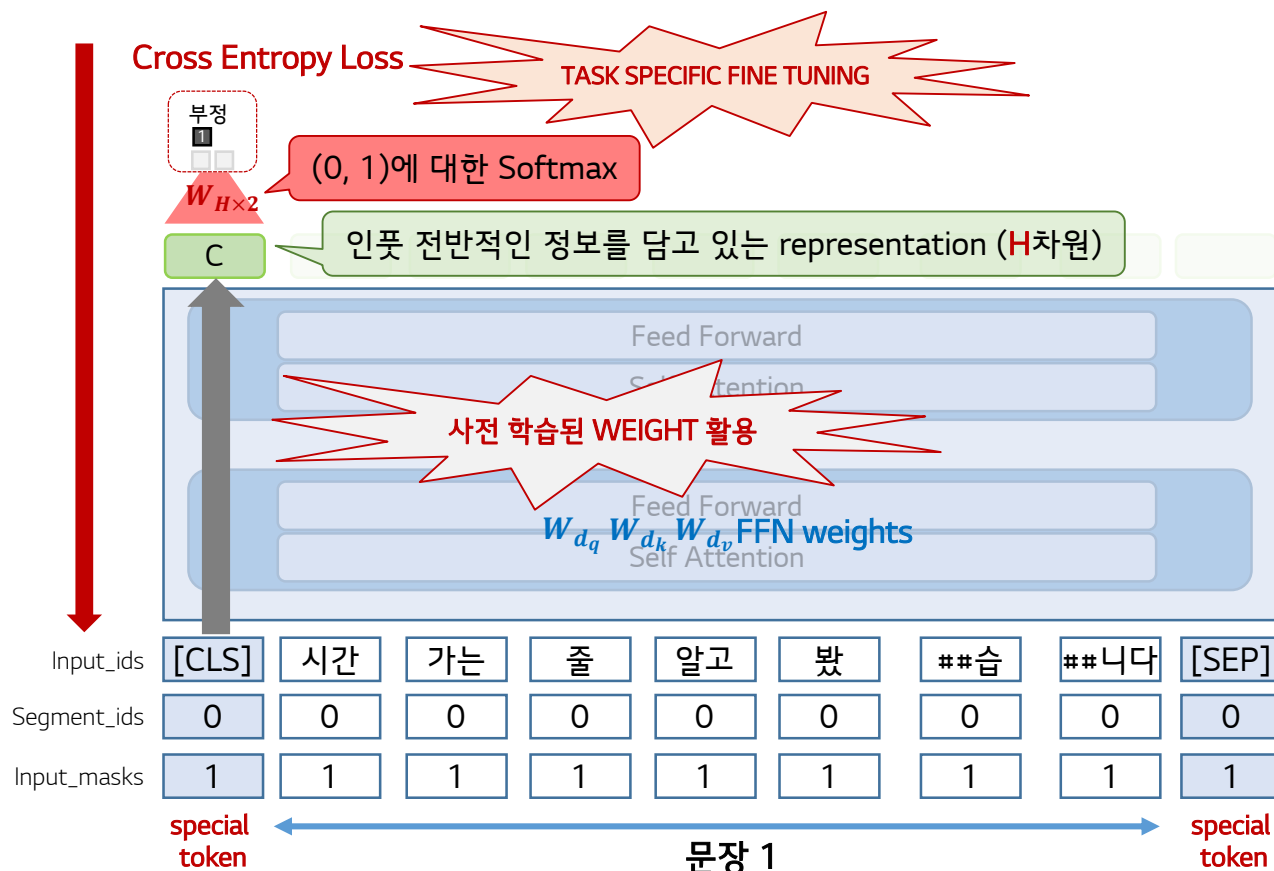
사전학습을 통해  
input 시퀀스를  
잘 represent할 수 있는 큰 모델 학습

Input_ids	[CLS]	Tok 1	Tok 2	...	[SEP]	Tok 1	Tok 2	...	[SEP]
Segment_ids	0	0	0	0	0	1	1	...	1
Input_masks	1	1	1	1	1	1	1	1	1

## Fine-tuning 예제

## 1) 감성분석

- 인풋 : 단일문장
- 아웃풋 : [긍정 / 부정]에 해당하는 확률 값

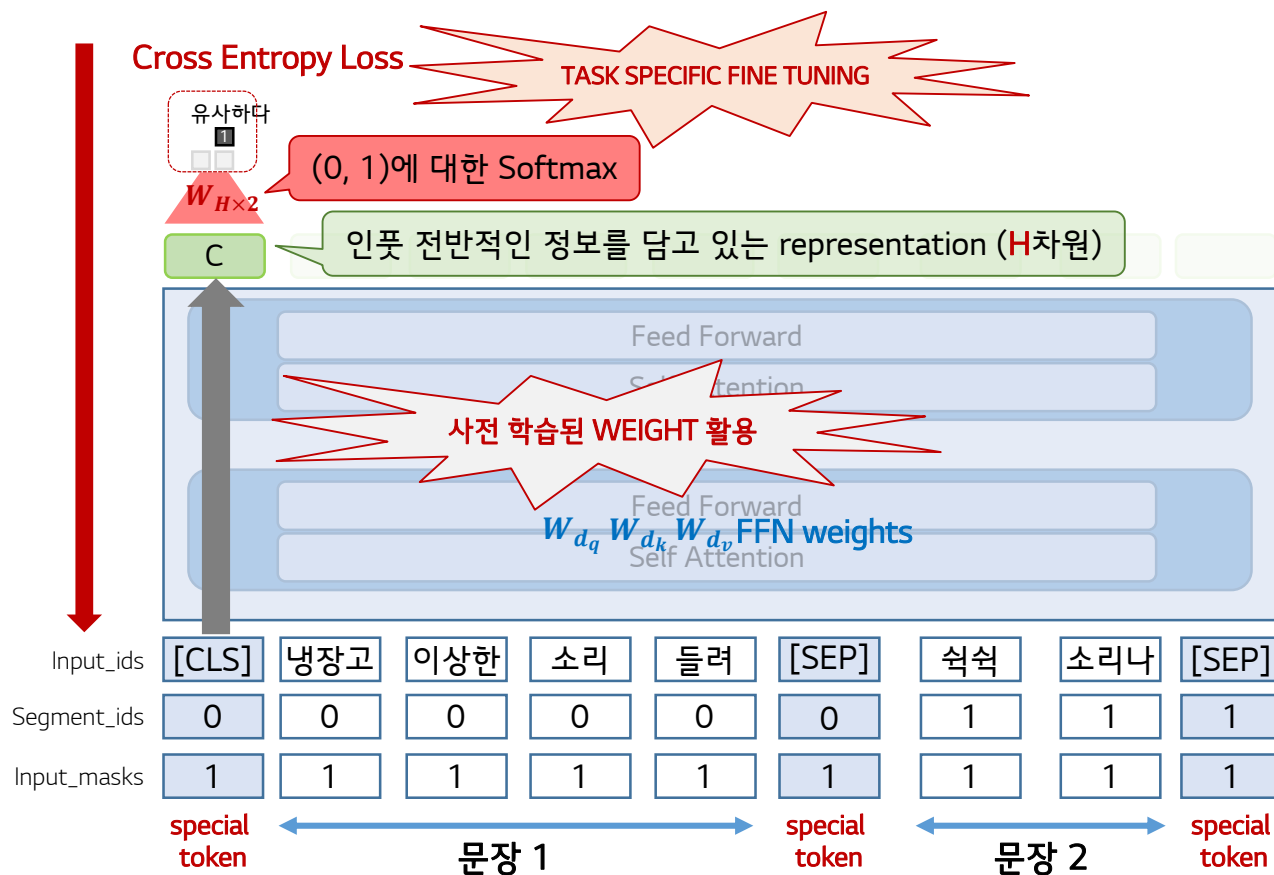


\*\* 이 때 Loss에 대해 전체 구조에 있는 weight가 fine-tuning된다.

## Fine-tuning 예제

## 2) 문장 유사도 분석

- 인풋 : 문장1, 문장2
- 아웃풋 : [유사하다 / 유사하지 않다]에 해당하는 확률 값



## Fine-tuning 예제

## 3) 질의응답 (MRC)

- 인풋 : 문단, 질문
- 아웃풋 : 답변 영역 (시작 지점, 끝 지점)

START INDEX  
END INDEX

{'context': '한편 1840년부터 바그너와 알고 지내던 리스트가 잊혀져 있던 1악장을 부활시켜 1852년에 바이마르에서 연주했다. 이것을 계기로 바그너도 이 작품에 다시 관심을 갖게 되었고, 그 해 9월에는 총보의 반환을 요구하여 이를 서곡으로 간추린 다음 수정을 했고 브라이트코프흐 & 헤르텔 출판사에서 출판할 개정판도 준비했다. 1853년 5월에는 리스트가 이 작품이 수정되었다는 것을 인정했지만, 끝내 바그너의 출판 계획은 무산되고 말았다. 이후 1855년에 리스트가 자신의 작품 파우스트 교향곡을 거의 완성하여 그 사실을 바그너에게 알렸고, 바그너는 다시 개정된 총보를 리스트에게 보내고 브라이트코프흐 & 헤르텔 출판사에는 20루이의 금을 받고 팔았다. 또한 그의 작품을 “하나하나의 음표가 시인의 피로 쓰여졌다”며 극찬했던 한스 폰 뷔로가 그것을 피아노 독주용으로 편곡했는데, 리스트는 그것을 약간 변형되었을 뿐이라고 지적했다. 이 서곡의 총보 첫머리에는 파우스트 1부의 내용 중 한 구절을 인용하고 있다.',

{'answers': [{ 'answer\_start': 402, 'text': '한스 폰 뷔로' }],  
'id': '5917067-1-1',  
'question': '파우스트 교향곡에 감탄하여 피아노곡으로 편곡한 사람은?'},

## Fine-tuning 예제

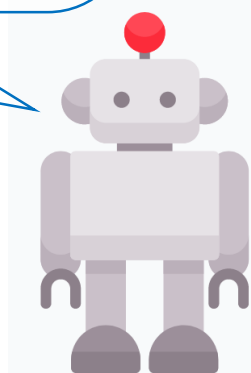
## 3) 질의응답 (MRC)

- 인풋 : 문단, 질문
- 아웃풋 : 답변 영역 (시작 지점, 끝 지점)

## 답변 시작 인덱스 예측

{ 'context': '0% 0% 0% 바그너와 알고 지내던 리스트가 잊혀져 있던 1악장을 부활시켜 1852년에 바이마르에서 연주했다. 이것을 계기로 바그너도 이 작품에 다시 관심을 갖게 되었고, 그 해 9월에는 총보의 반환을 요구하여 이를 서곡으로 간추린 다음 수정을 했고 브라이트코프흐 & 헤르텔 출판사에서 출판할 개정판도 준비했다. 1853년 5월에는 리스트가 이 작품이 수정되었다는 것을 인정했지만, 끝내 바그너의 출판 계획은 무산되고 말았다. 이후 1855년에 리스트가 자신의 작품 파우스트 교향곡을 거의 완성하여 그 사실을 바그너에게 알렸고, 바그너는 다시 개정된 총보를 리스트에게 보내고 브라이트코프흐 & 헤르텔 출판사에는 20루이의 금을 받고 팔았다. 또한 그의 작품을 "하나하나의 음표가 시인의 피로 쓰여졌다"며 2% 2% 87% 9% 뵐로가 그것을 피아노 독주용으로 편곡했는데, 리스트는 그것을 약간 변형되었을 뿐이라고 지적했다. 이 서곡의 총보 첫머리에는 파우스트 1부의 내용 중 한 구절을 인용하고 있다.',

```
{ 'answers': [{ 'answer_start': 402, 'text': '한스 폰 뵐로' }],
  'id': '5917067-1-1',
  'question': '파우스트 교향곡에 감탄하여 피아노곡으로 편곡한 사람은?' }
```



## Fine-tuning 예제

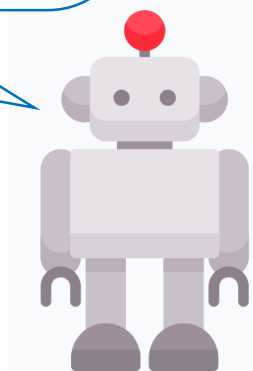
## 3) 질의응답 (MRC)

- 인풋 : 문단, 질문
- 아웃풋 : 답변 영역 (시작 지점, 끝 지점)

## 답변 끝 인덱스 예측

{ 'context': '0% 0% 0% 바그너와 알고 지내던 리스트가 잊혀져 있던 1악장을 부활시켜 1852년에 바이마르에서 연주했다. 이것을 계기로 바그너도 이 작품에 다시 관심을 갖게 되었고, 그 해 9월에는 총보의 반환을 요구하여 이를 서곡으로 간추린 다음 수정을 했고 브라이트코프흐 & 헤르텔 출판사에서 출판할 개정판도 준비했다. 1853년 5월에는 리스트가 이 작품이 수정되었다는 것을 인정했지만, 끝내 바그너의 출판 계획은 무산되고 말았다. 이후 1855년에 리스트가 자신의 작품 파우스트 교향곡을 거의 완성하여 그 사실을 바그너에게 알렸고, 바그너는 다시 개정된 총보를 리스트에게 보내고 브라이트코프흐 & 헤르텔 출판사에는 20루이의 금을 받고 팔았다. 또한 그의 작품을 "하나하나의 음표가 시인의 피로 쓰여졌다"며 0% 1% 한스 10% 71% 2% 을 파아노 독주용으로 편곡했는데, 리스트는 그것을 약간 변형되었을 뿐이라고 지적했다. 이 서곡의 총보 첫머리에는 파우스트 1부의 내용 중 한 구절을 인용하고 있다.',

```
{ 'answers': [{ 'answer_start': 402, 'text': '한스 폰 뷔로' }],
  'id': '5917067-1-1',
  'question': '파우스트 교향곡에 감탄하여 피아노곡으로 편곡한 사람은?' }
```



## Fine-tuning 예제

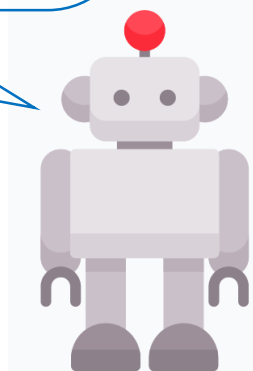
## 3) 질의응답 (MRC)

- 인풋 : 문단, 질문
- 아웃풋 : 답변 영역 (시작 지점, 끝 지점)

시작 **argmax** ~ 끝 **argmax** → MRC 정답 텍스트

{'context': '한편 1840년부터 바그너와 알고 지내던 리스트가 잊혀져 있던 1악장을 부활시켜 1852년에 바이마르에서 연주했다. 이것을 계기로 바그너도 이 작품에 다시 관심을 갖게 되었고, 그 해 9월에는 총보의 반환을 요구하여 이를 서곡으로 간추린 다음 수정을 했고 브라이트코프흐 & 헤르텔 출판사에서 출판할 개정판도 준비했다. 1853년 5월에는 리스트가 이 작품이 수정되었다는 것을 인정했지만, 끝내 바그너의 출판 계획은 무산되고 말았다. 이후 1855년에 리스트가 자신의 작품 파우스트 교향곡을 거의 완성하여 그 사실을 바그너에게 알렸고, 바그너는 다시 개정된 총보를 리스트에게 보내고 브라이트코프흐 & 헤르텔 출판사에는 20루이의 금을 받고 팔았다. 또한 그의 작품을 "하나하나의 음표가 시인의 피로 쓰여졌다"며 극찬했던 **한스 폰 뷔로**가 그것을 피아노 독주용으로 편곡했는데, 리스트는 그것을 약간 변형되었을 뿐이라고 지적했다. 이 서곡의 총보 첫머리에는 파우스트 1부의 내용 중 한 구절을 인용하고 있다.'

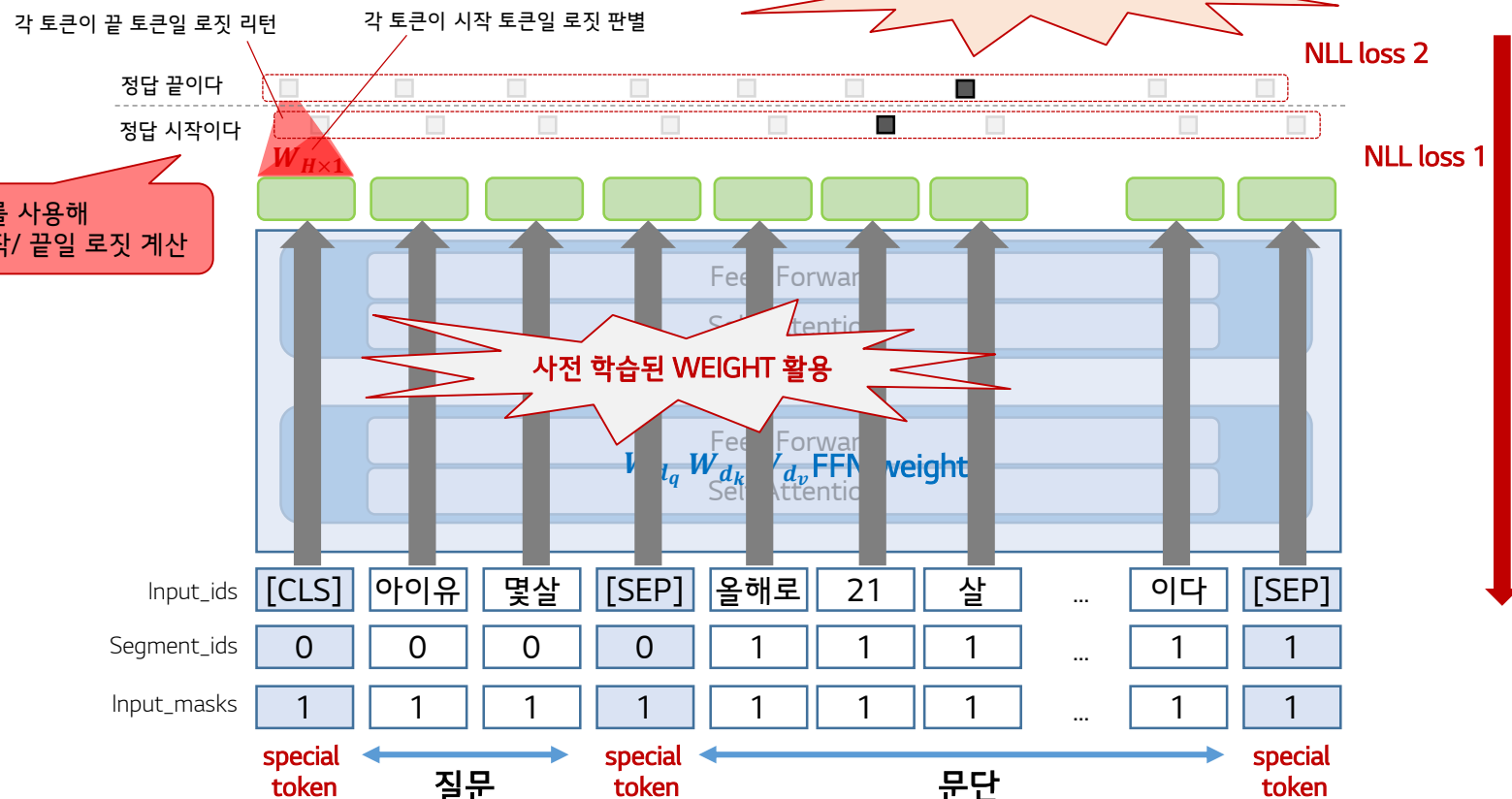
```
{'answers': [{'answer_start': 402, 'text': '한스 폰 뷔로'}],
'id': '5917067-1-1',
'question': '파우스트 교향곡에 감탄하여 피아노곡으로 편곡한 사람은?'},
```



## Fine-tuning 예제

## 3) 질의응답 (MRC)

- 인풋 : 문단, 질문
- 아웃풋 : 답변 영역 (시작 지점, 끝 지점)





(Quiz)

BERT를 이용해 문장의 의도를 분류하려고 합니다.  
모델을 어떻게 fine-tuning할 수 있을까요?

인풋 :

참

잘

하는

짓

이다



칭찬

✓ 비난

질문

인사

감사

실습 1 - BERT fine-tuning으로 감성분석 모델 구현하기

실습 7\_BERT\_Classification.ipynb



## 실습 1 – BERT fine-tuning으로 감성분석 모델 구현하기

- 데이터 : 네이버 영화리뷰 (<https://github.com/e9t/nsmc>)

### 개봉영화 평점

저 산 너머	★★★★★ 8.41
 <b>어벤저스: 인피니티 워</b> 그렇다. 토르는 신이었다... 내년까지 살아가야 할 이유가 생겼습니다 이 영화의 최고의 빌런은 번역가다.	★★★★★ 8.96
1917	★★★★★ 8.88
트롤: 월드 투어	★★★★★ 8.91

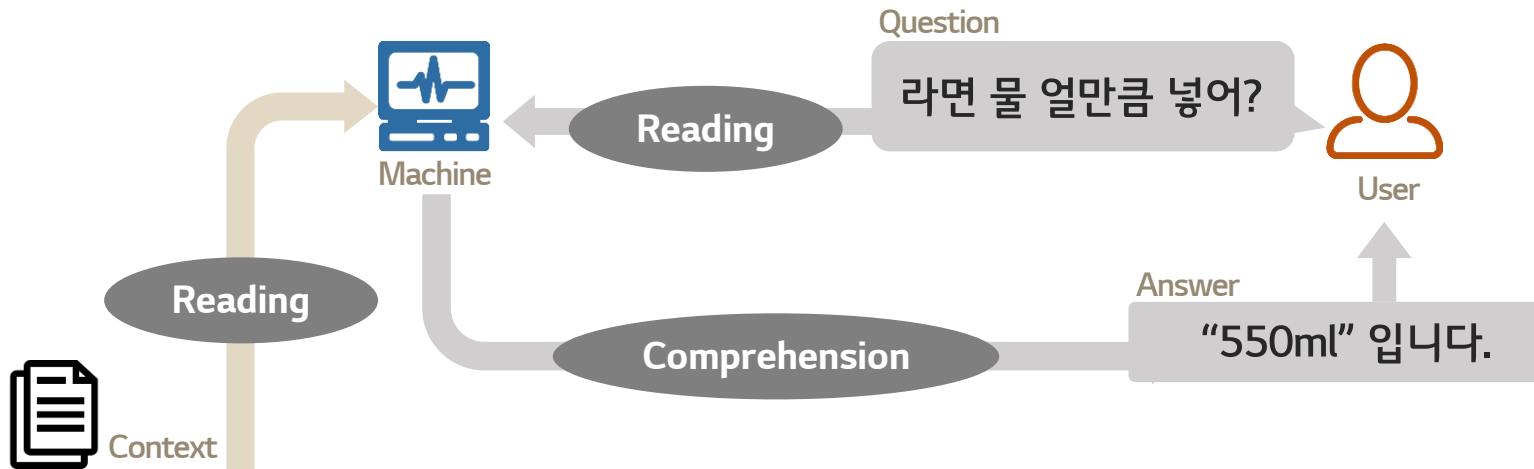
- 네이버 영화 평점 크롤링 데이터 20만 건
- 평점 1-4 댓글 = 부정
- 평점 9-10 댓글 = 긍정 으로 라벨링

- 학습 목표 :
  - BERT 토큰나이징을 이해한다.
  - BERT fine-tuning 코드를 구현하여 감성분석 모델을 훈련할 수 있다.
  - keras.Model을 이용해 원하는 모델을 만들고, compile하여 훈련할 수 있다.
  - 자연어 문장을 인풋으로 받아 감정을 판별하는 함수를 구현하고, 추론할 수 있다.

## 실습 2 – BERT fine-tuning으로 기계독해 모델 학습하기

실습 8\_BERT\_MRC.ipynb

- MRC: 기계가 자연어로 된 문서를 읽고 이해하여 답변을 해 주는 기술



## 라면 맛있게 끓이는 방법

물 550ml를 넣고 끓이기 시작합니다. 물이 끓기 시작하면 후레이크와 스프를 먼저 넣어주고, 면은 쪄개지 않고 통째로 넣습니다. 계란을 반숙으로 먹고 싶다면 면을 넣는 시점에 계란을 함께 투하합니다. 면이 익을 때까지 3~4분 가량 더 끓여주고 마지막에 대파를 송송 썰어 넣습니다.

## 실습 2 – BERT fine-tuning으로 기계독해 모델 학습하기

- 데이터 : KorQuAD 1.0 ([https://korquad.github.io/category/1.0\\_KOR.html](https://korquad.github.io/category/1.0_KOR.html))

**What is KorQuAD 1.0?**

KorQuAD 1.0은 한국어 Machine Reading Comprehension을 위해 만든 데이터셋입니다. 모든 질의에 대한 답변은 해당 Wikipedia article 문단의 일부 하위 영역으로 이루어집니다. Stanford Question Answering Dataset(SQuAD) v1.0과 동일한 방식으로 구성되었습니다.

[KORQUAD 1.0 소개 \(SLIDE\)](#)

[KORQUAD 1.0 소개 \(PAPER\)](#)

**Leaderboard**

KorQuAD 1.0의 Test set으로 평가한 Exact Match(EM) 및 F1 score입니다.

Rank	Reg. Date	Model	EM	F1
-	2018.10.17	Human Performance	80.17	91.20
1	2020.01.08	SkERT-Large (single model) Skelter Labs	87.66	95.15

- 학습 목표 :
  - BERT를 이용한 기계독해 모델 학습 및 추론을 체험해본다.

## 실습 2 – BERT fine-tuning으로 기계독해 모델 학습하기

- (참고) 512자가 넘는 긴 문단에서는 답을 어떻게 찾나요?

## 동숲 섬에 몇 명까지 살 수 있음?

이미 존재하던 마을의 주인이 되어 살아가는 기존 작품들에서 나아가, 아무것도 없는 무인도에 초기 주민 둘과 함께 이주해 처음부터 섬을 개척해 나가는 게임이다. 벌과 나비부터 다량어와 상어에 이르는 다양한 생물을 채집하여 박물관을 완성해 나가고, 직접 모은 재료를 가공해 만든 도구로 실내와 섬의 이곳저곳을 아름답게 꾸미고, 개성이 확실한 수많은 동물 주민들과 온라인으로 만난 다른 플레이어들과 교감하며 유유자적한 삶을 보내는 것이 주 콘텐츠다. 게임 진행에 따라서는 도로나 다리와 같은 큰 규모의 시설을 건설하는 것은 물론 지형까지 마음대로 수정할 수 있게 되어 **마인크래프트나 심즈 시리즈와 각종 건설 경영 시뮬레이션** 게임이 부럽지 않은 나만의 공간을 만들어낼 수도 있다.

시리즈 전통에 따라 플레이어가 거주하는 섬은 게임 계정과 무관하게 본체 1개당 하나로 제한되어 있으며, 하나의 섬에서 최대 8명까지 함께 생활할 수 있다. 컨트롤러 여러 개를 사용하면 화면을 조작할 수 있는 권한이 있는 한 명을 중심으로 최대 4명까지 동시에 본체 하나로 멀티플레이가 가능하다. 다른 이의 소프트웨어를 자신의 본체에 삽입해도 다른 이의 섬으로 바뀌지는 않는다.<sup>[1]</sup> 이는 Nintendo Switch가 세이브 데이터를 본체에 저장하는 특성을 지녔기 때문이다. 자신의 섬에 주민을 추가하는 것은 본체에 등록된 다른 계정으로 게임을 실행하는 방식으로 가능하며, 해당 본체에 이미 생성되어 있는 섬이 있는 상태로 다른 새로운 섬을 생성하는 것은 불가능하다.

지원 언어는 기존의 한국어, 영어, 일본어, 스페인어, 프랑스어, 독일어, 이탈리아어에 더해 네덜란드어, 포르투갈어, 러시아어, 중국어(간체, 번체)가 처음으로 추가되었다. 특히 포르투갈어는 이식작이 아닌 신작으로서는 이례적인 추가인데, **슈퍼 스매시브라더스 얼티밋이나 슈퍼 마리오 메이커 2**처럼 전작에 있었던 포르투갈어도 신작이 나오면서 빠지는 추세였기 때문이다. 언어는 본체 설정을 바꾸면 바꿀 수 있고, 언어 변경 시 명칭이 바뀐다는 경고문이 나온다.

통신을 통해 친구의 섬에 놀러가거나 친구를 자신의 섬으로 초대하는 멀티플레이 기능도 존재한다. 로컬 통신이나 인터넷 접속을 통해 최대 동시 8명까지 하나의 섬에서 모여 놀 수 있다. 인터넷 멀티플레이를 위해서는 **Nintendo Switch Online** 서비스(유료)에 가입해야 한다.

플레이어가 생활하게 될 무인도는 게임 시작 시 북반구와 남반구 중 선택할 수 있다. 선택에 따라 계절의 차이가 발생하여, 출시일 3월 20일 기준 북반구는 봄, 남반구는 가을부터 계절이 시작된다. 또한 섬의 지형은 4개가 주어져 그 중 하나를 선택할 수 있으며, 섬의 이름은 게임 초반 퀘스트 도중에 《~도》, 《~섬》 중 하나를 선택해 자유롭게 정할 수 있다.

긴 문단을 stride를 주며  
512자 안에 들어가게 자름

Start + End logit이 가장 높은  
후보를 최종 정답으로 채택

[CLS] 동 ##숲 섬 에 몇 명 ##  
까지 살 수 있음 ? [SEP] 이미 존  
재 ##하던 마을의 ...

BERT 게임 (logit = 0.04)

[CLS] 동 ##숲 섬 에 몇 명 ##  
까지 살 수 있음 ? [SEP] 수정할  
수 있게 되어 ...

BERT **8명 (logit = 12.98)**

[CLS] 동 ##숲 섬 에 몇 명 ##  
까지 살 수 있음 ? [SEP] 수정할  
수 있게 되어 ...

BERT 2 (logit = 2.31)

[CLS] 동 ##숲 섬 에 몇 명 ##  
까지 살 수 있음 ? [SEP] 나오면  
서 빠지는 추세였기 때문...

BERT 8명 (logit = 9.13)