BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

2018. 07. 19.

김동화

BERT

- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
  - 다양한 NLP task문제에서 Baseline 보다 5~6%의 성능개선을 가짐
- 요약
  - ELMo, GPT의 영향을 받은 방법
  - 모델이 굉장히 큼, 3억 4천만개 파라미터
  - Masked LM
    - Pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers
  - Pre-training이 되었을 때(2일) NLP Task 학습(fine-tunning)에 대해서 30분 (Batch size:32, epoch:3)돌리면 사람 수준의 성능을 가짐

Pre-trained language representation를 Task에 적용하는 두가지 방법

- Feature-based (Parameters are **fixed**)
  - **ELMo**: downstream tasks-specific architectures
    1. Token representation
    2. Bidirectional Language Model (biLM)
    3. **Freezed** 한 후 token embeddings(**x**)과 biLM output(**h**) 에 elmo embeddings를 병합한 supervised learning

Downstream task: Extrinsic Evaluation(such as application)

Pre-trained language representation를 Task에 적용하는 두가지 방법

- Feature-based (Parameters are **fixed**)
  - **ELMo**: downstream tasks-specific architectures
    1. Token representation
    2. Bidirectional Language Model (biLM)
    3. **Freezed** 한 후 token embeddings(**x**)과 biLM output(**h**) 에 elmo embeddings를 병합한 supervised learning
- Fine-tunning (Parameters are not **fixed**)
  - Generative Pre-trained Transformer (GPT)
    - Unidirectional language models

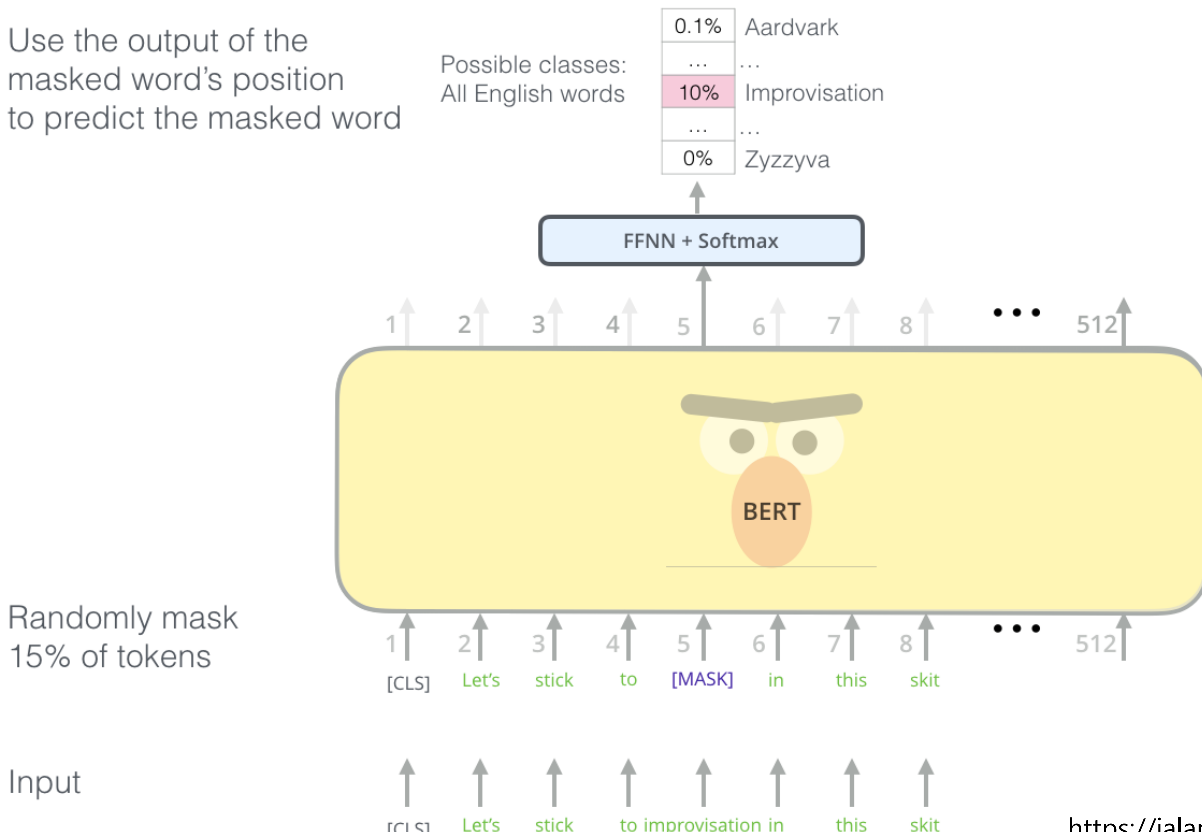
Downstream task: Extrinsic Evaluation(such as application)

Limitation

- Unidirectional LM
  - Left-to-right architecture
    - 이전 단어들을 가지고 다음 단어를 예측/attention
  - Fine-tunning을 적용할 때 Sub-optimal를 가지는 문제가 있음

New objective

- Masked language model(MLM)
  - 입력 token을 랜덤하게 masking 처리(15%) 한 후 masking 된 token id를 예측
  - 좌우 문맥을 결합시킬 수 있음(bidirectional)



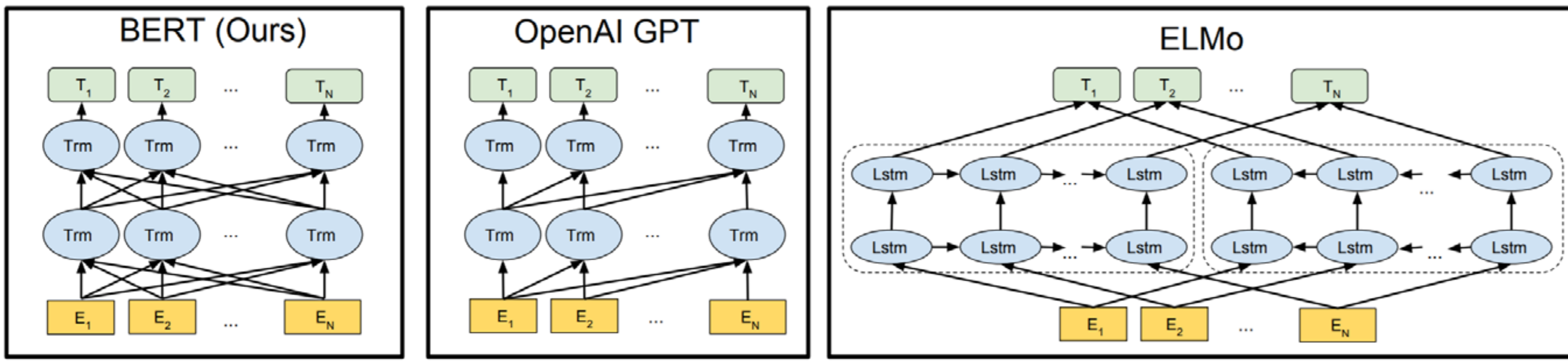
https://jalammar.github.io/illustrated-bert/

Contribution

- Masking을 사용하여 pre-trained deep bidirectional representations
  - 독립적인 Forward LM, Backward LM를 병합한 것과 다름
- 다양한 NLP Task에 문장, 단어를 표현할 수 있는 fine-tunning-based representation
- 높은 성능을 가짐

BERT

- 구조
  - Encoder: Bidirectional
  - Decoder: left-context-only



- 설명 전개
  - Input representation
  - Pre-training task
  - Pre-training & Fine-tunning procedure

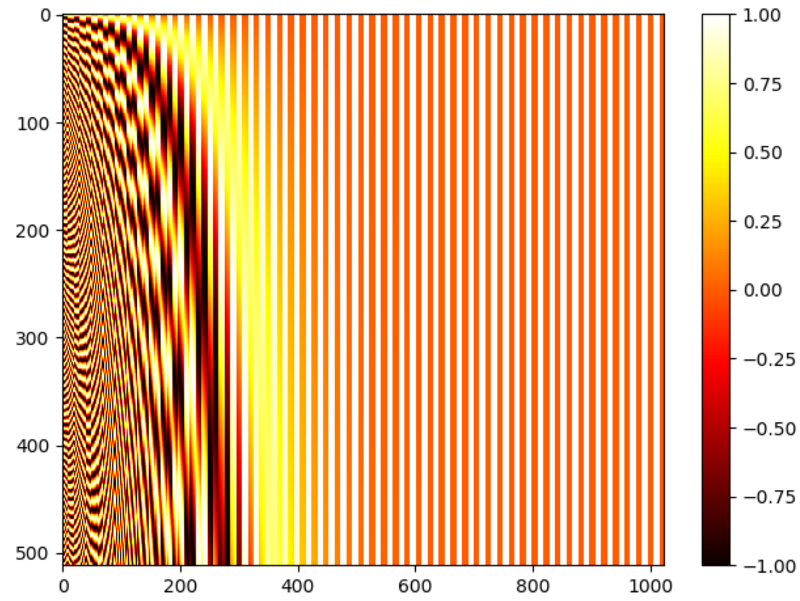
Input representation

- Position Encoding
  - Add some sequential structure
  - $PE_{(pos,2i)} = \sin(pos/10^{4 \cdot 2i/d_{model}})$
  - $PE_{(pos,2i+1)} = \cos(pos/10^{4 \cdot 2i/d_{model}})$
    - d_{model} : embedding size
    - i: embedding size의 인덱스(a feature)
    - Pos: max length의 인덱스(a word)

Input representation

- Position Encoding
  - 모델에게 문장의 길이를 추론시켜 학습된 문장보다 더 길게 생성시킬 수 있는 효과가 있음

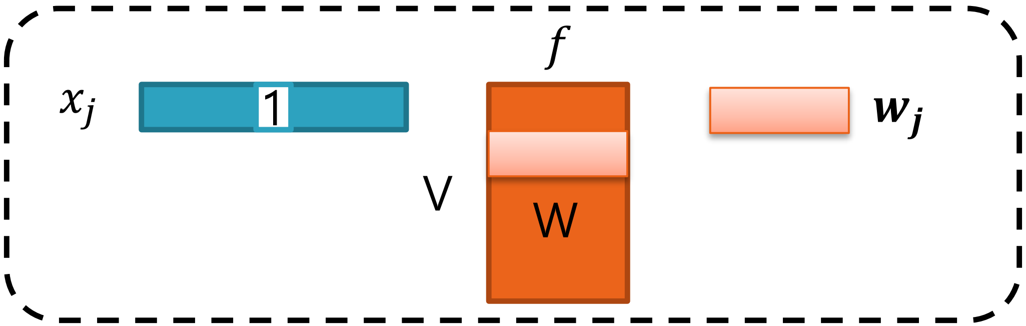
```python
import numpy as np
import matplotlib.pyplot as plt

max_length = 512
embed_size = 1024
position_enc = np.array([
    [pos / np.power(10000, 2*i/embed_size) for i in range(embed_size)]
    if pos != 0 else np.zeros(embed_size) for pos in range(max_length)])

position_enc[1:, 0::2] = np.sin(position_enc[1:, 0::2]) # dim 2i
position_enc[1:, 1::2] = np.cos(position_enc[1:, 1::2]) # dim 2i+1

img = plt.imshow(position_enc, cmap='hot', interpolation='nearest', aspect='auto')
cmap = plt.get_cmap('hot')
plt.colorbar(img, cmap=cmap)
plt.show()
```



Input representation

- Position Encoding

No temporal information



  - Add some sequential structure
    - $\boldsymbol{p_j} \in \mathbb{R}^f$

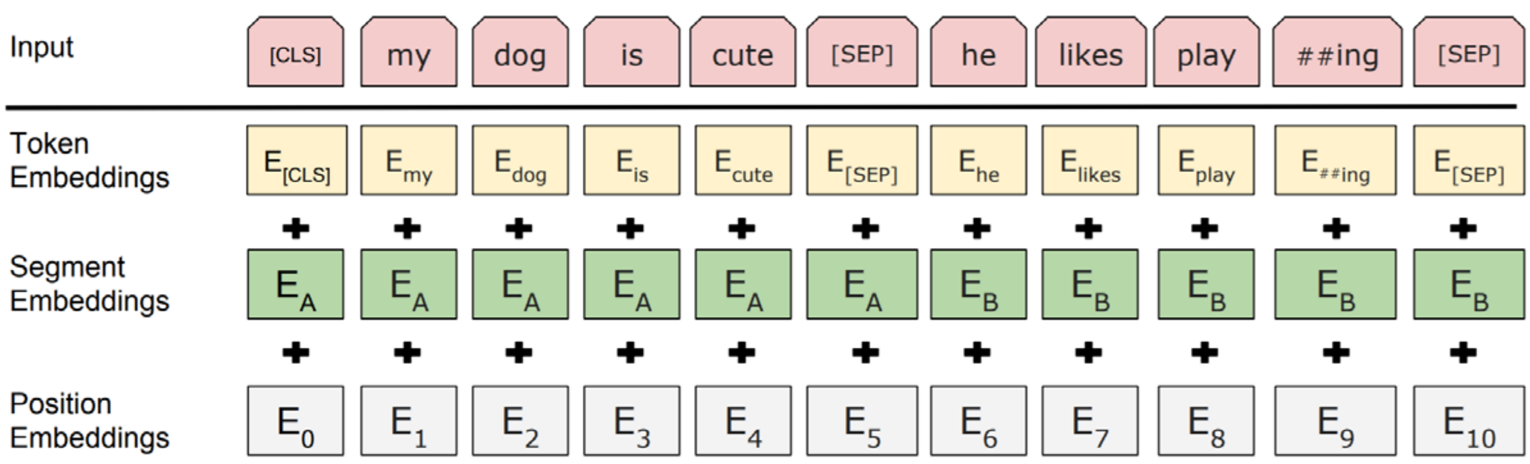
  - $\boldsymbol{e_j} = (\boldsymbol{w_j} + \boldsymbol{p_j})$

- Position Embeddings
  - Max length = sequence length = 512

Input representation

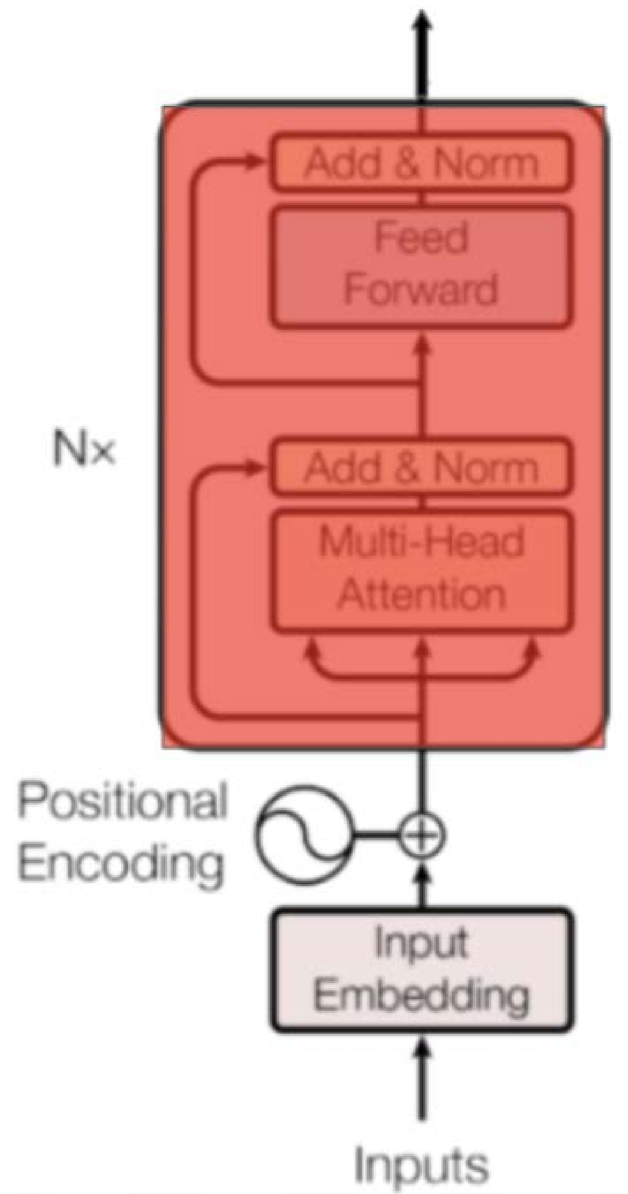
- Segment embedding
  - [SEP]이라는 token을 삽입해 sentence를 구분
    - A pair sentences: type ids, documents: segment ids
  - 각 문장사이의 숫자가 append (e.g. 0, 1, . . .)

```
#  tokens:   [CLS] is this jack ##son ##ville ? [SEP] no it is not . [SEP]
#  type_ids: 0     0  0    0    0     0       0 0     1  1  1  1   1 1
# (b) For single sequences:
#  tokens:   [CLS] the dog is hairy . [SEP]
#  type_ids: 0     0   0   0  0     0 0
```



Encoder Block

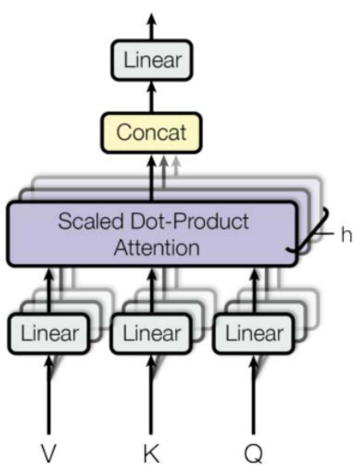
- Self attention head(A)
  - BERT base: 12개
  - BERT large: 24개
- Max length = 512
  - 512 x 24 = 12,288개
- 병렬 처리 관련
  - 병렬 처리가 아닌 블록들이 Recursive하게 반복 처리



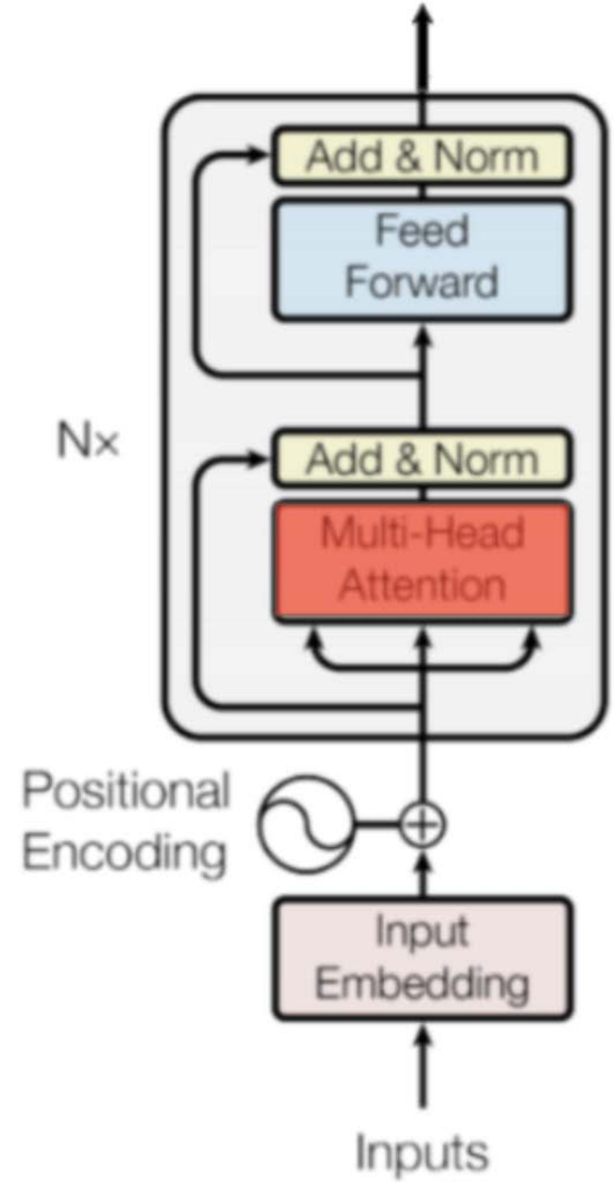
Encoder Block

- Multi-Head Attention
  - 서로 다른 가중치 행렬을 계산
  - 어텐션 head를 여러 번(h) 계산
  - Concatenation

$$\mathrm{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\mathrm{head}_1; \ldots ; \mathrm{head}_h]\mathbf{W}^O$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V)$$

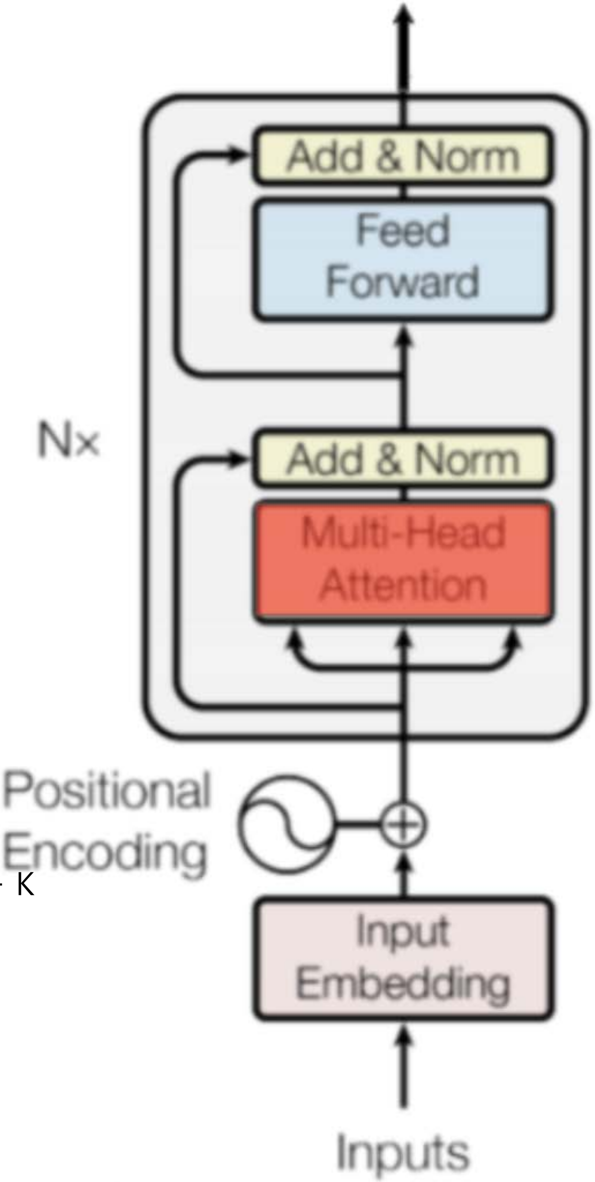


Encoder Block(t-1) Encoder Block(t) Encoder Block(t+1),.. ~ 24개



Encoder Block

- Multi-Head Attention
  - 서로 다른 가중치 행렬을 계산
  - 어텐션 head를 여러 번(h) 계산
  - Concatenation

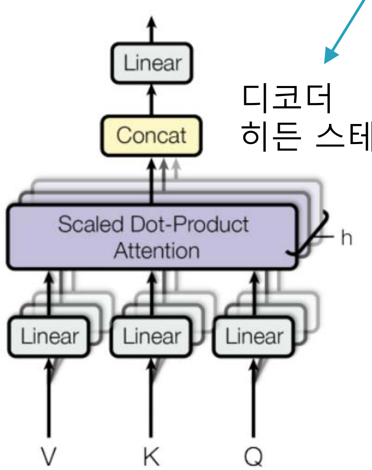


$$\mathrm{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\mathrm{head}_1;\ldots;\mathrm{head}_h]\mathbf{W}^O$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

디코더
히든 스테이트

인코더
히든 스테이트

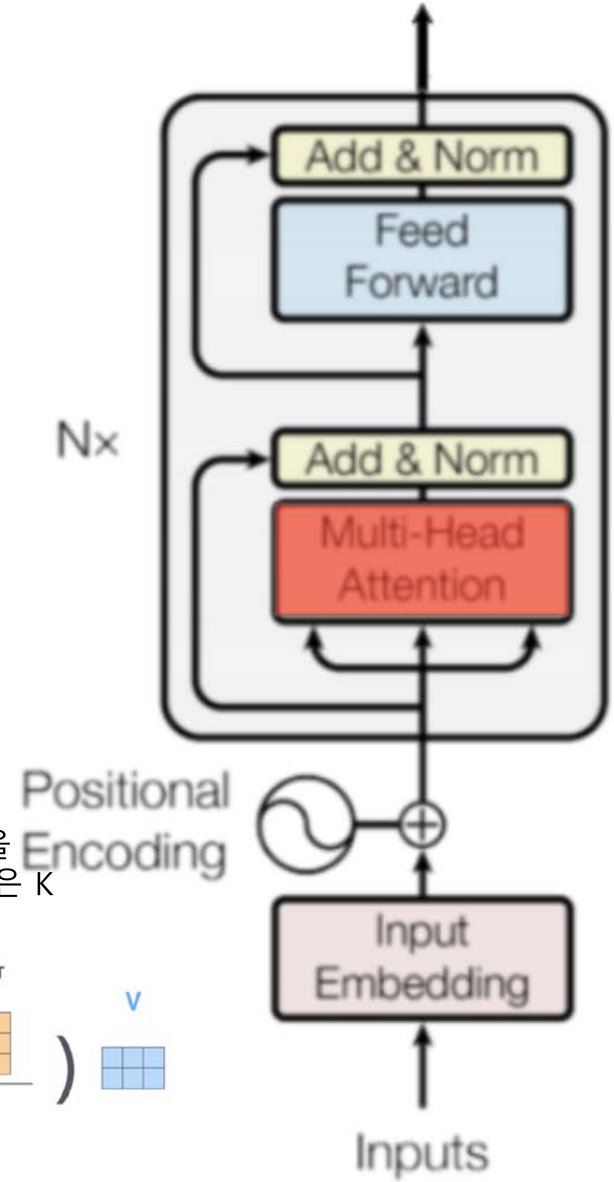
어텐션을
부여 받은 K



Encoder Block(t-1) Encoder Block(t) Encoder Block(t+1),.. ~ 24개

Encoder Block

- Multi-Head Attention
  - 서로 다른 가중치 행렬을 계산
  - 어텐션 head를 여러 번(h) 계산
  - Concatenation

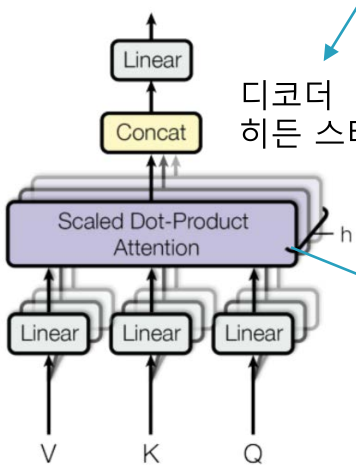


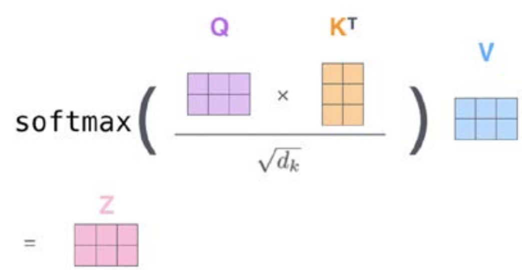
$$\mathrm{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\mathrm{head}_1; \ldots; \mathrm{head}_h]\mathbf{W}^O$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

디코더
히든 스테이트

인코더
히든 스테이트

어텐션을
부여 받은 K

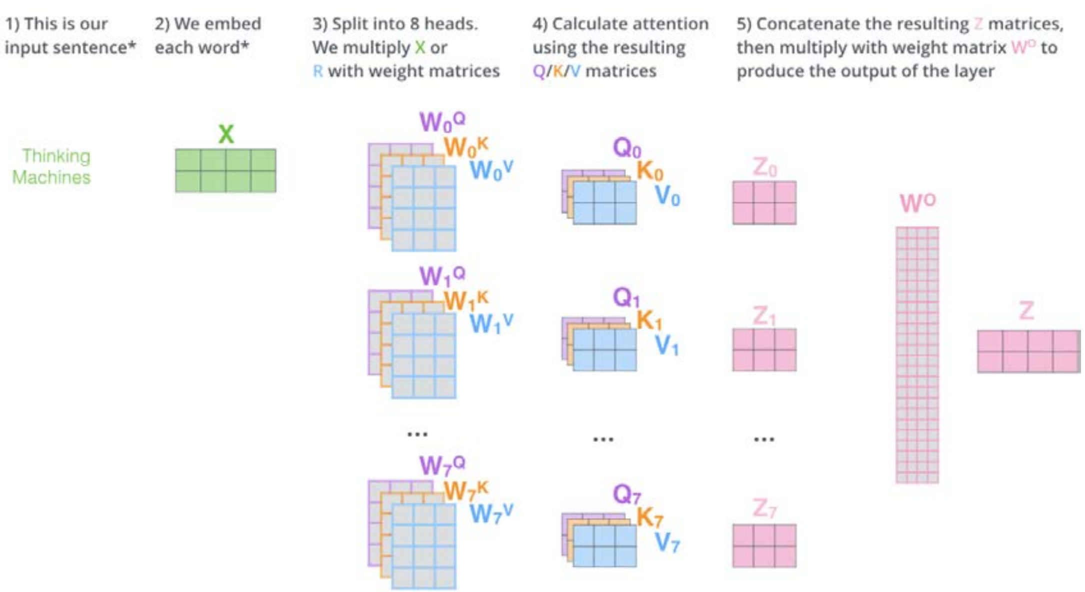


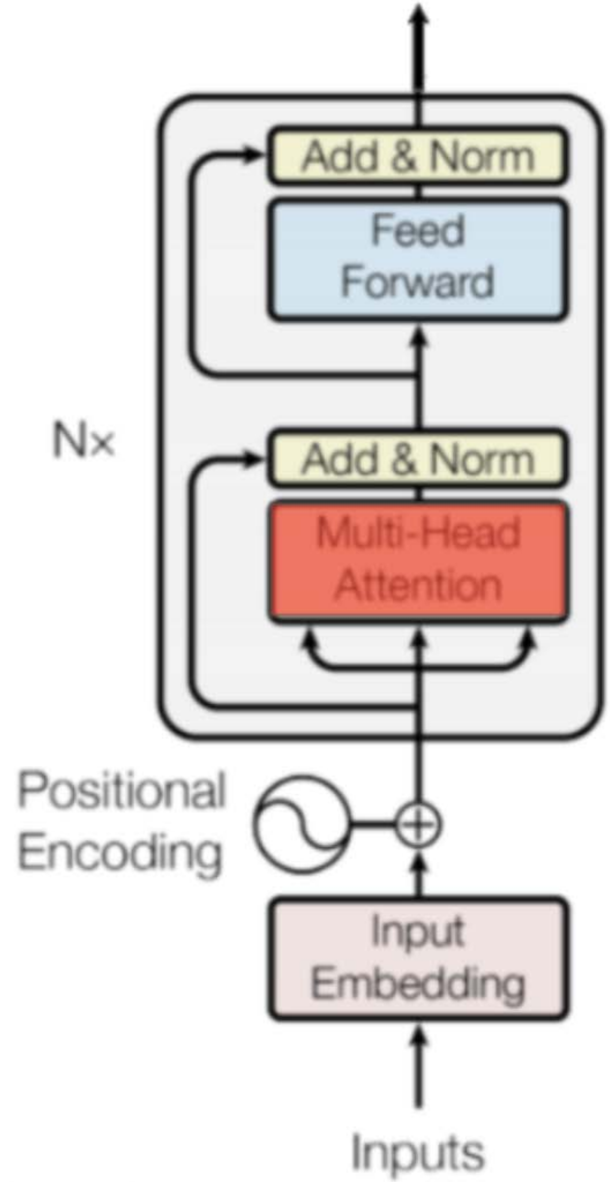


Encoder Block(t-1) Encoder Block(t) Encoder Block(t+1),.. ~ 24개

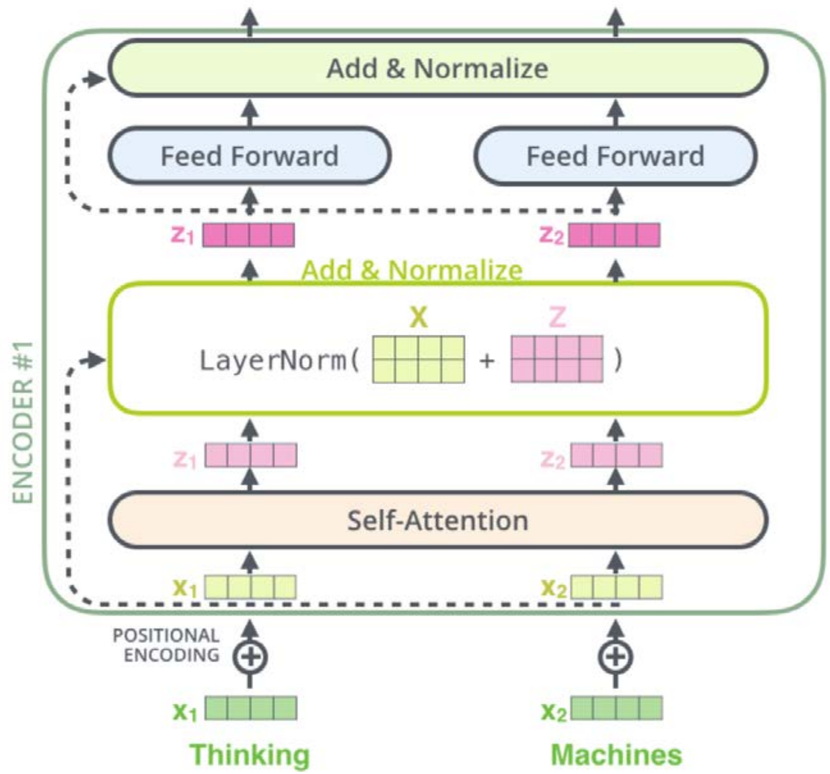
Encoder Block

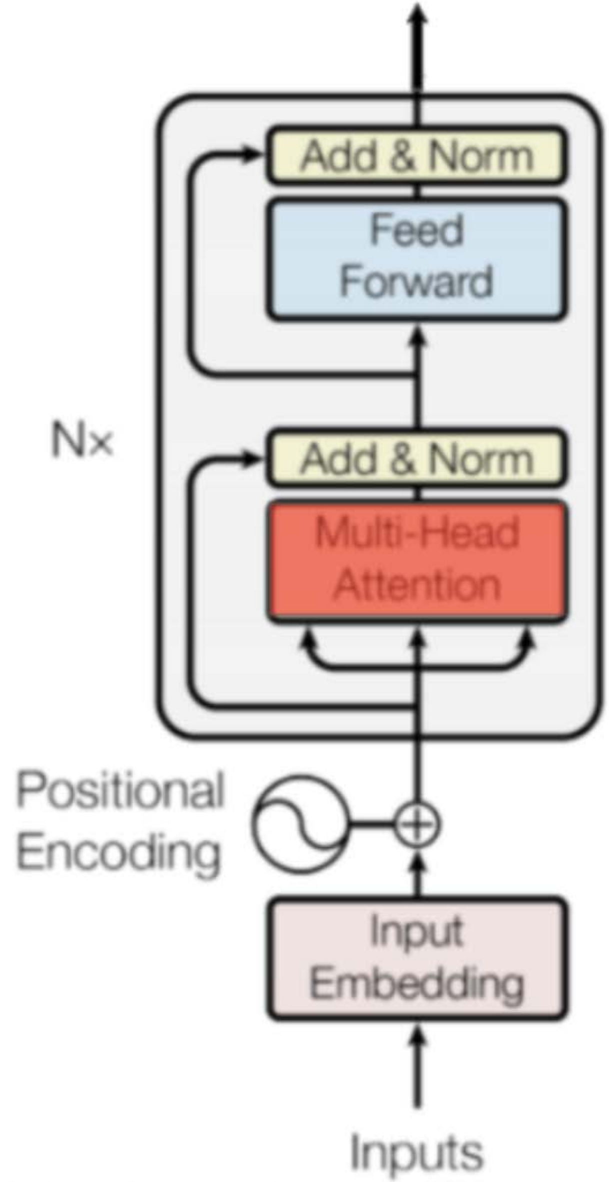
- Multi-Head Attention
  - 서로 다른 가중치 행렬을 계산
  - 어텐션 head를 여러 번(h) 계산
  - Concatenation





Encoder Block

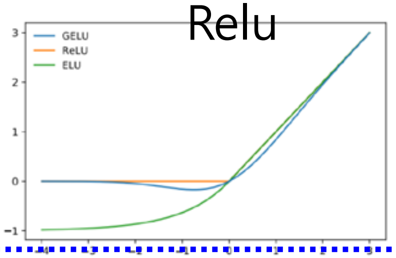


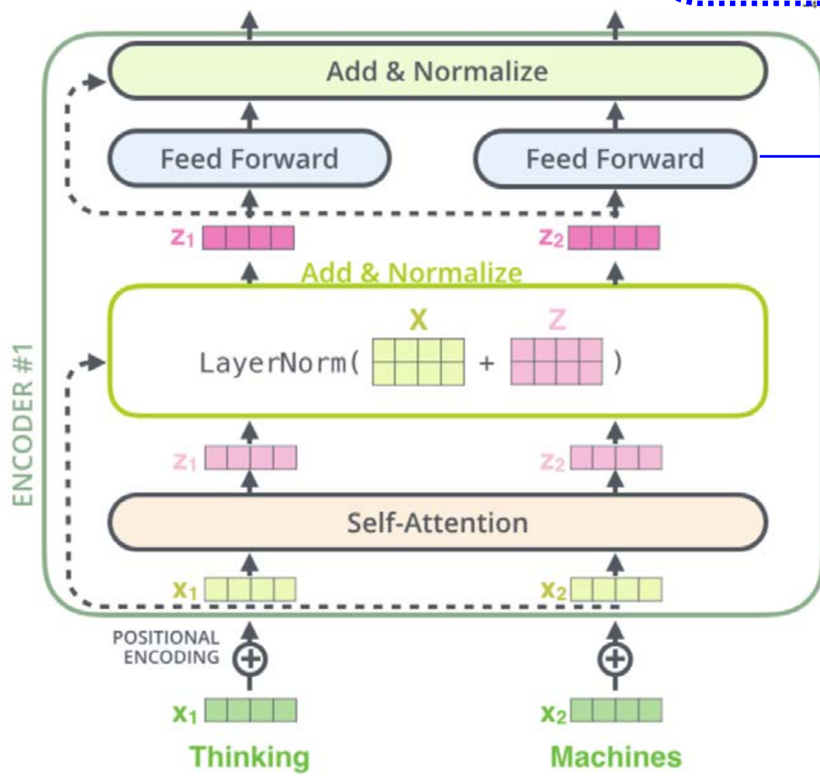


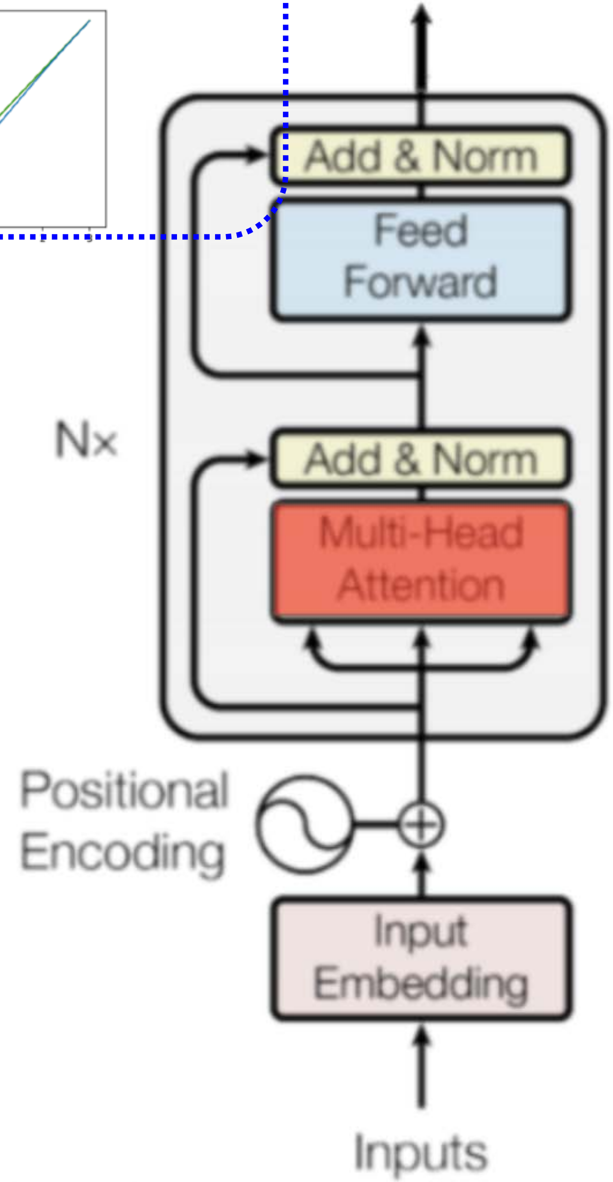
Encoder Block

$$FFN(x) = \boxed{\max(0, xW_1 + b_1)}\, W_2 + b_2$$

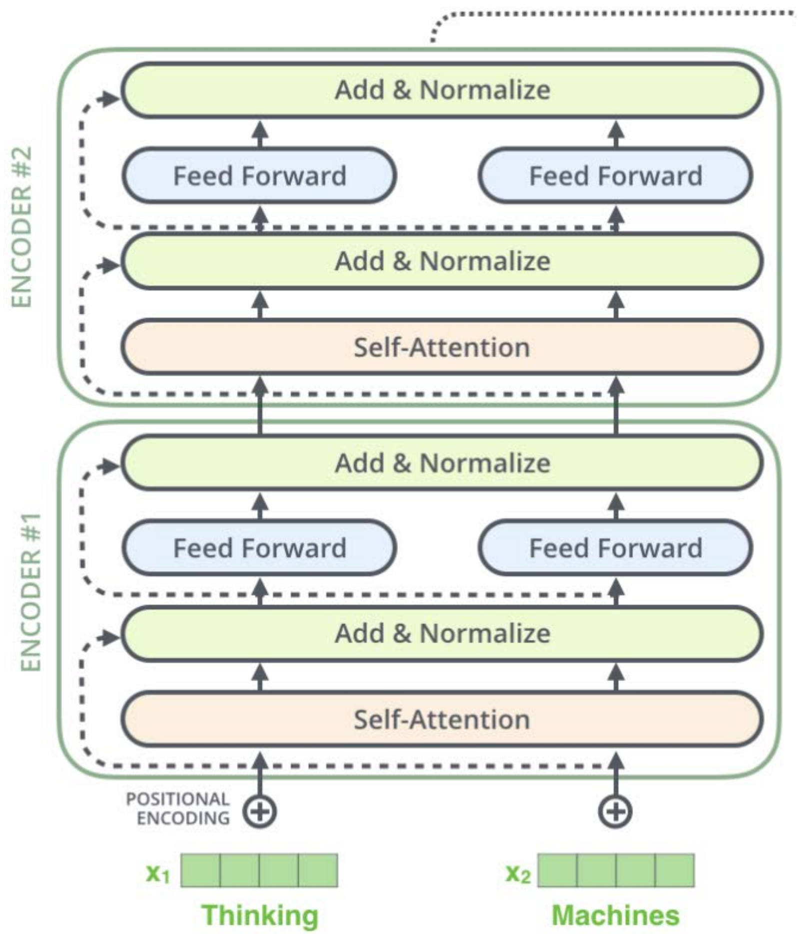
Relu

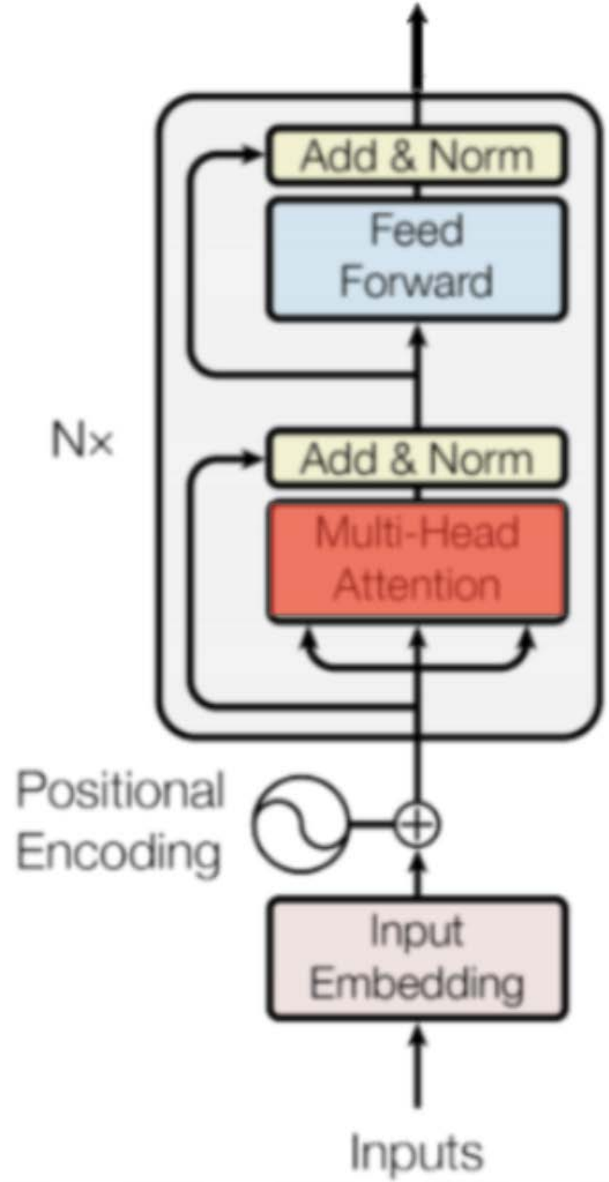




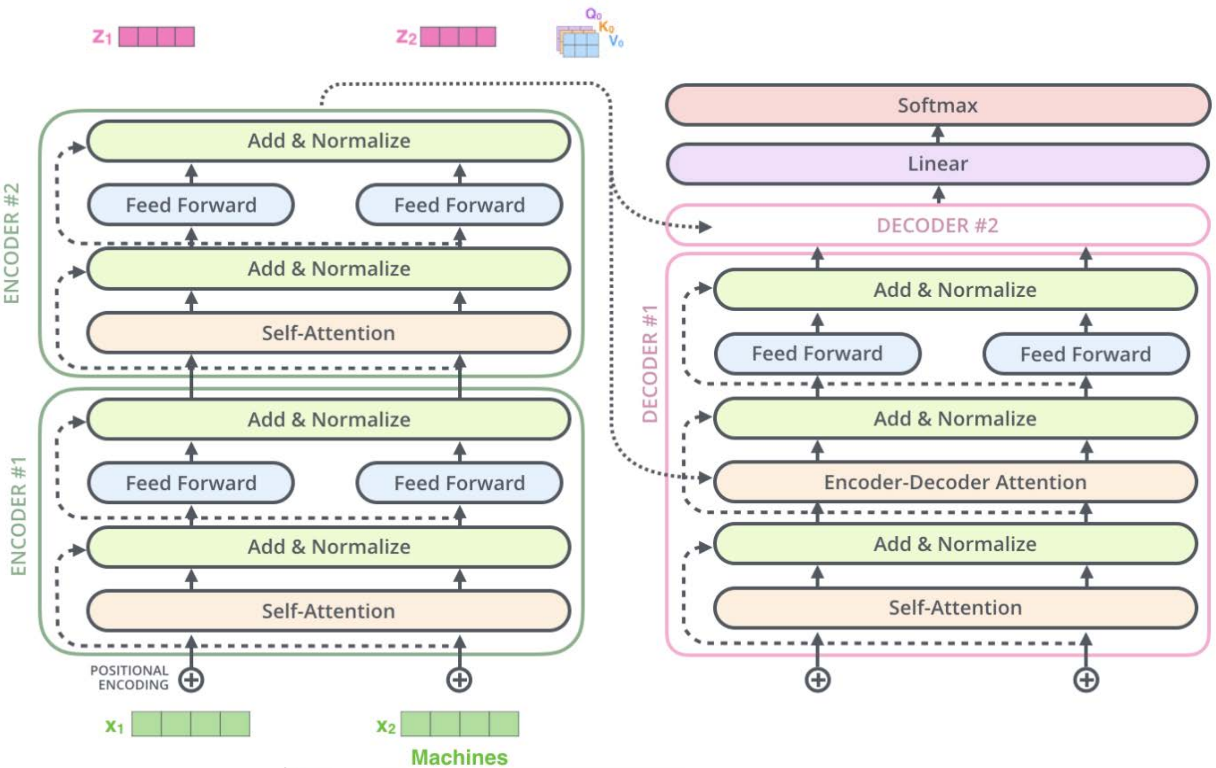


Encoder Block





Encoder & Decoder Block



- **Encoder-decoder attention**

$$Y = \mathrm{MultiHead}(V, K, Q) = \mathrm{MultiHead}(X_e, X_e, X_d)$$

- **Self-attention layers in the encoder**

$$Y = \mathrm{MultiHead}(V, K, Q) = \mathrm{MultiHead}(X_e, X_e, X_e)$$

- **Self-attention layers in the decoder**

$$Y = \mathrm{MultiHead}(V, K, Q) = \mathrm{MultiHead}(X_d, X_d, X_d)$$

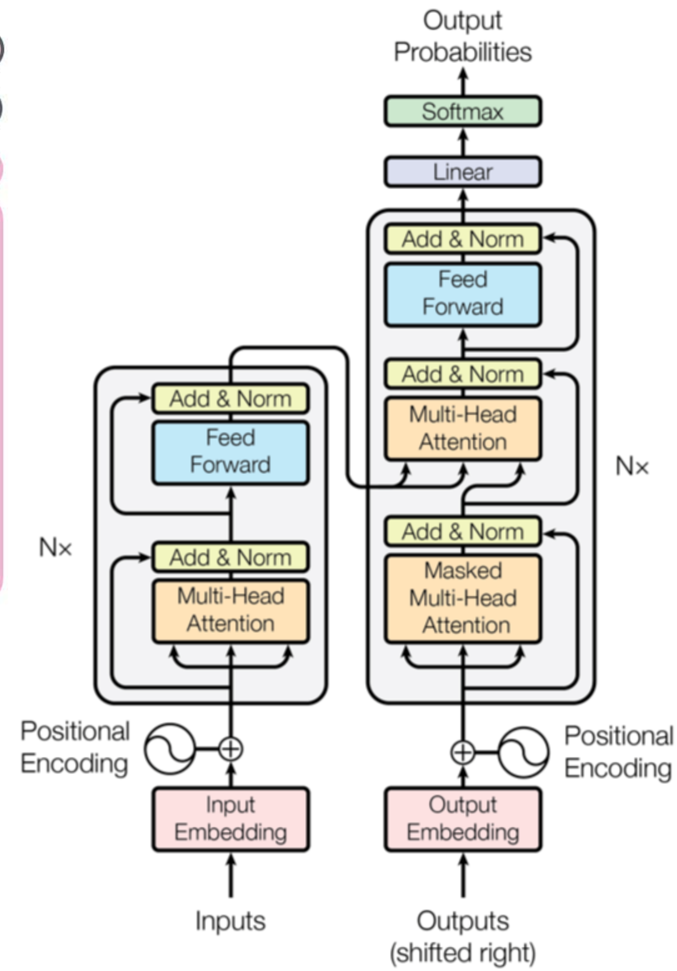
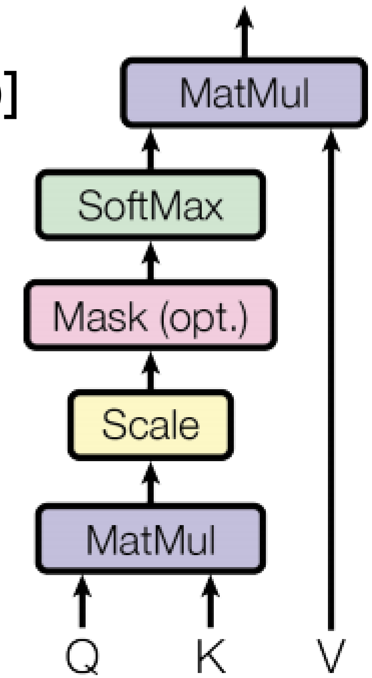


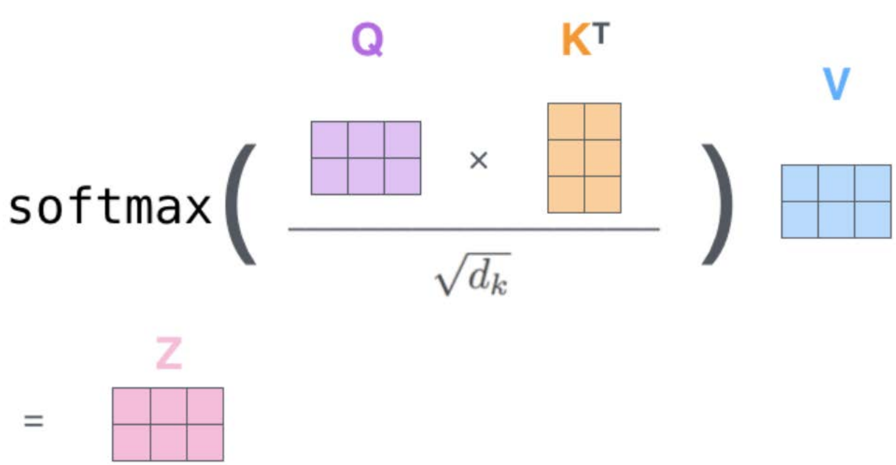
Figure 1: The Transformer - model architecture.

Masked Attention

- Mask 3D Tensor(Attention mask)
  - [batch_size, seq_length(encoder), seq_length(decoder)]
  - [batch_size, 1, seq_length(encoder), seq_length(decoder)]
  - 1.0: 어텐션 하고 싶은 위치(0)
  - 0.0: mask 위치(-10000.0)
  - 제로 패딩은 항상 마스킹 처리해 패널티 부과

```
adder = (1.0 - tf.cast(attention_mask, tf.float32)) * -10000.0
```

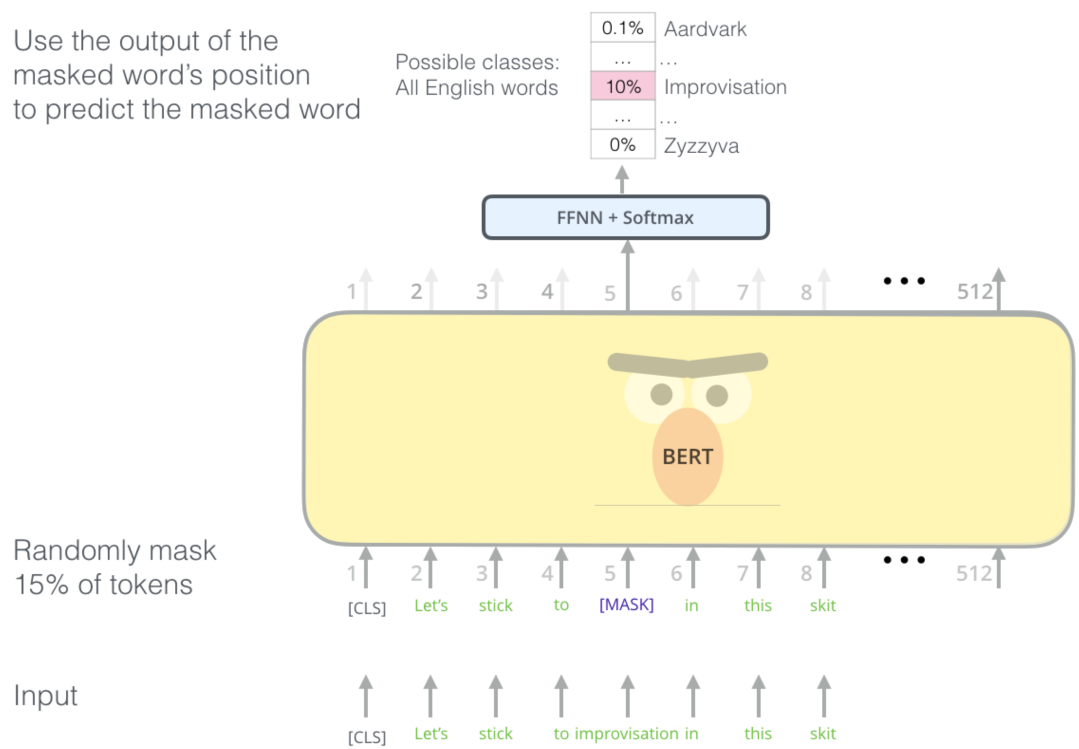




Training

- Masked Language Model
  - 문장의 다음 단어를 예측하는 것이 아니라 문장내 랜덤한 단어를 마스킹하고 이를 예측
  - 너무 Mask token만 예측하려고 함(수렴...); 주변 단어들도 잘 학습되게 correct word/incorrect word를 섞음
  - MLM 방식은 Q&A task에 충분하지 않는 성질을 가짐

- Rather than *always* replacing the chosen words with `[MASK]`, the data generator will do the following:
- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.



Training

- Next Sentence Prediction
  - 50% 비율로 참인 문장과 랜덤하게 추출되어 거짓인 문장의 비율로 구성
  - 98%의 정확성

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

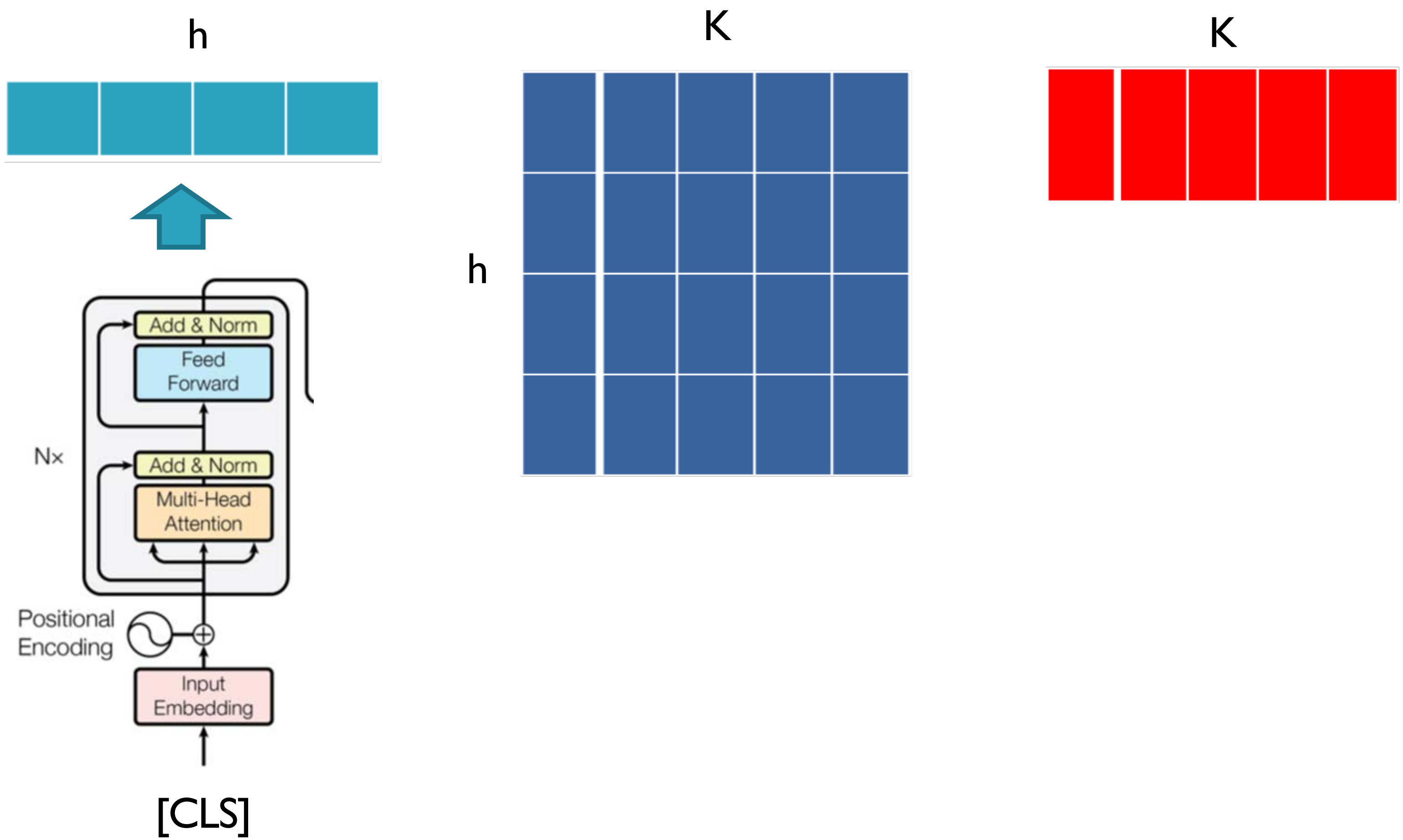
Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

Pre-training Procedure

- BooksCorpus
- English Wikipedia (60 GB)
  - sampled such that the combined length is ≤ 512 tokens
  - Batch size: 256
  - L2 decay: 0.01
  - Learning rate: 0.0001 and decay after 10,000 steps
  - Batch norm params: scale=0.9, shift=0.999
- Object function
  - Mean masked LM likelihood + Mean next sentence prediction
- GPU 몇 백개 사용

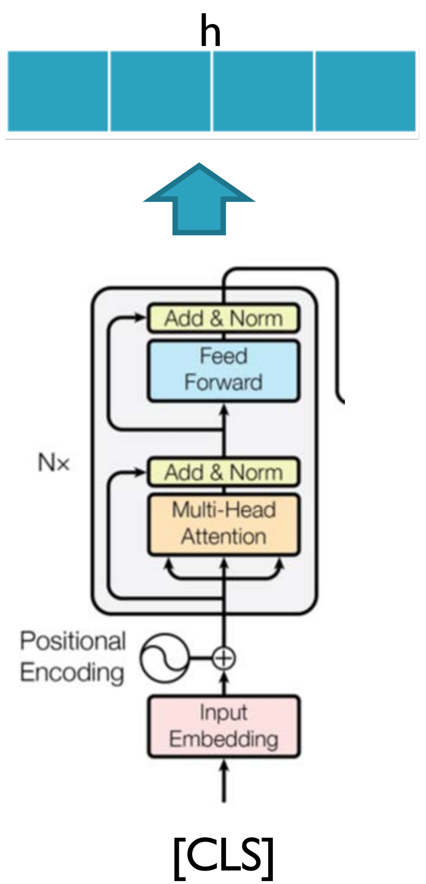
Fine-tunning Procedure

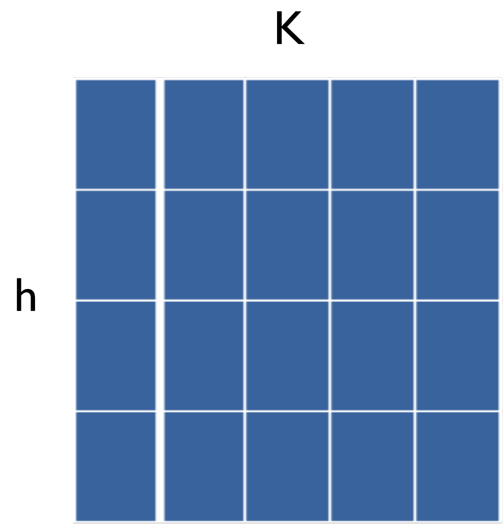
- First token([CLS])의 대한 final hidden state(output of the Transformer)

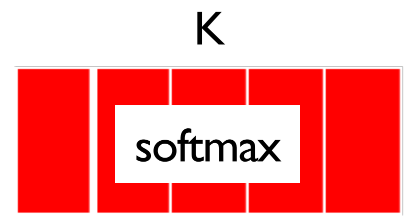


Fine-tunning Procedure

- First token([CLS])의 대한 final hidden state(output of the Transformer)
  - Pre-training에서 설정했던 파라미터와 같게 설정
  - 빠르다 3 epoch 30분 …





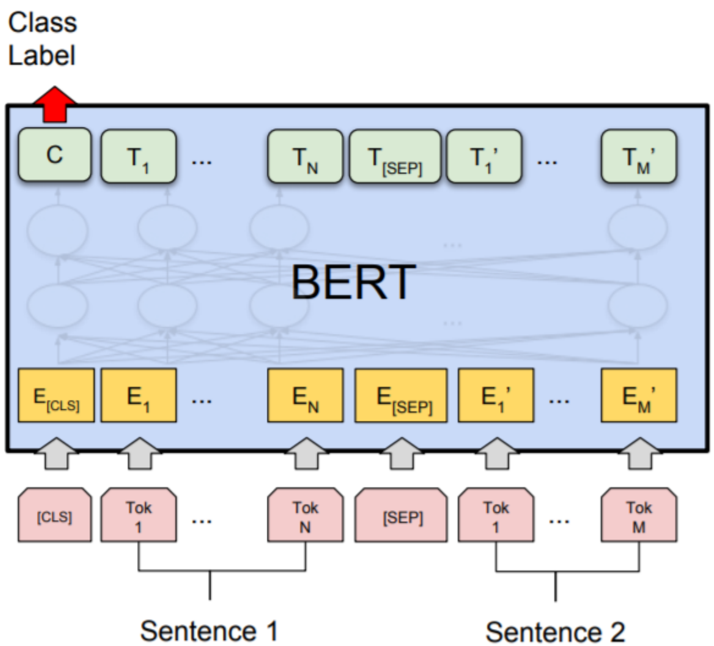


성능비교

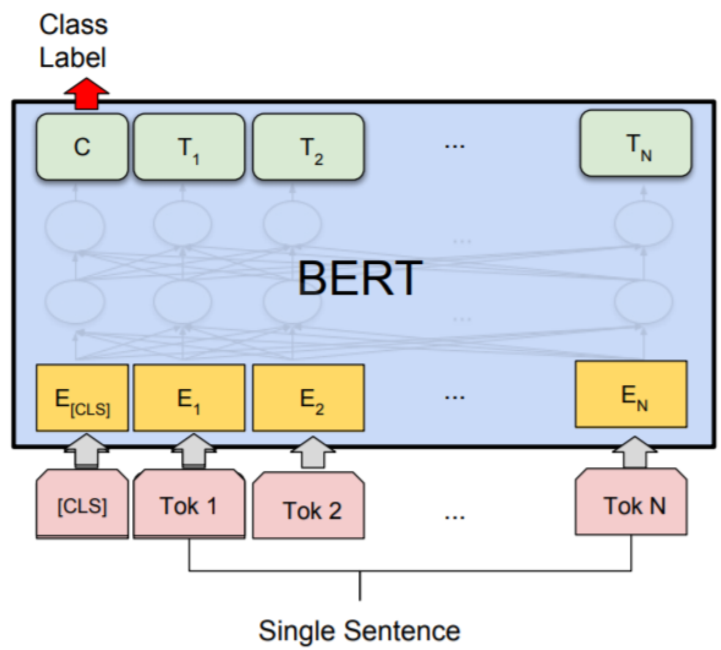
- Baseline GPT
  - 이 방법론도 [CLS], [SEP] 사용, sentence A/B
- GLUE datasets (2018)
  - MNLI: entailment classification task(text and hypothesis에 대해서 참, 중립, 거짓)
  - QQP: 두 질문 문장이 의미론 적으로 동등한지 이진 분류
  - QNLI: Q & A에 대한 두 문장들이 참/거짓을 분류하는 이진 분류
  - SST-2: binary sentiment classification
  - CoLA: 영어 문장이 문법적인 오류 있는지/없는지 분류
  - STS-B: 뉴스기사 제목에 대한 두 문장이 얼마나 유사한지 1~5점 척도로 매긴 데이터
  - MRPC: 두 문장이 의미론적으로 동등한지 이진 분류
  - RTE: entailment classification task

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$BERT_{BASE}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

성능비교



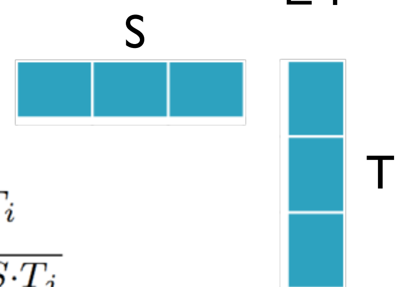
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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

성능비교

- SQuAD
  - Input(Question, paragraph), output(Answer)
  - Question, paragraph를 하나의 pair of sentence로 간주
  - Start vector(S)
  - End vector(E)



$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$$

- Input Question:

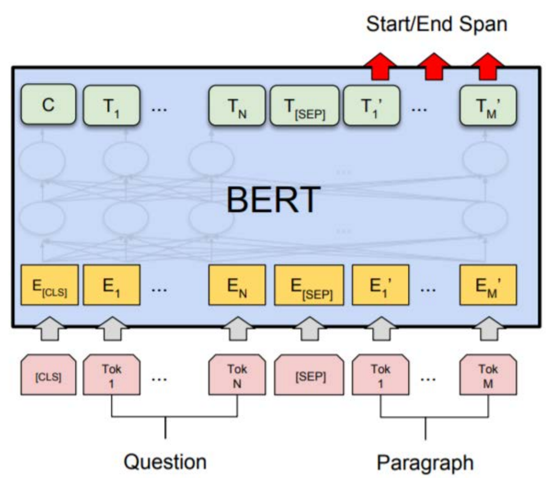
  Where do water droplets collide with ice crystals to form precipitation?

- Input Paragraph:

  ... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. ...

- Output Answer:

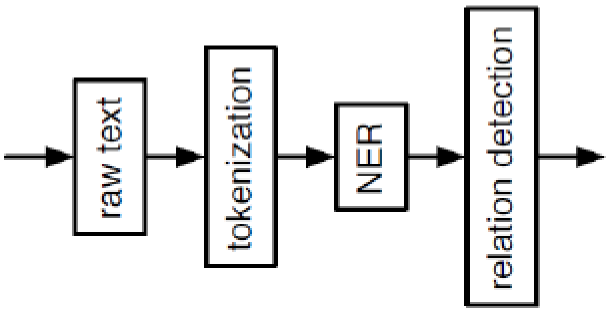
  within a cloud



(c) Question Answering Tasks:
SQuAD v1.1

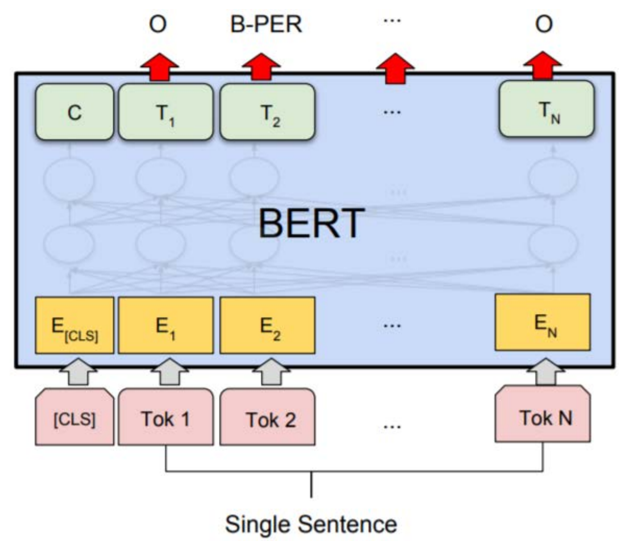
성능비교

- CoNLL 2003 NER
  - Person, Organization, Location, Miscellaneous, others
  - WordPiece tokenizer




```
Jim   Hen   ##son was a puppet ##eer
I-PER I-PER X     O   O O      X
```

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
BERT$_{\text{BASE}}$	96.4	92.4
BERT$_{\text{LARGE}}$	**96.6**	**92.8**



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER