BERT

Bidirectional Encoder Representations from Transformer

이은수

BERT
목차

BERT
서론

BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformer

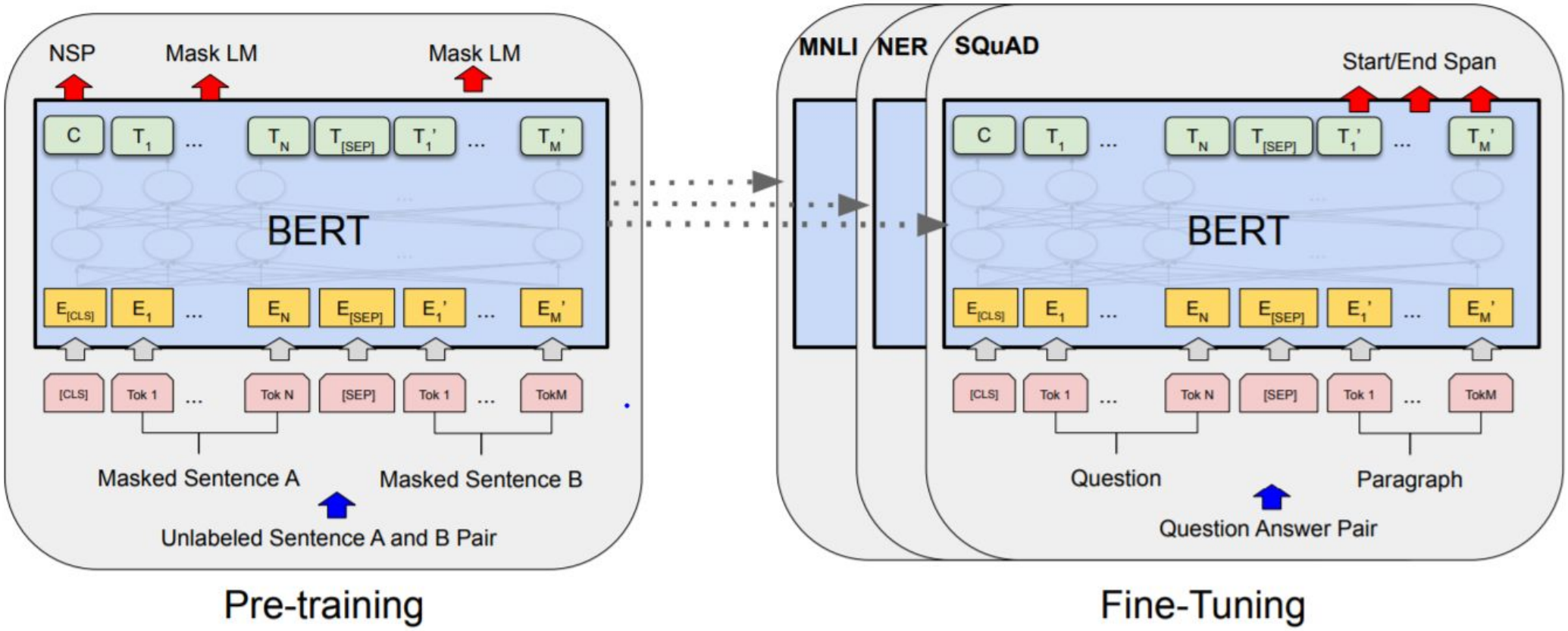
- Google AI Language 팀이 2018년 10월에 공개한 논문
- BERT는 Transformer 모델의 Encoder만을 양방향으로 연결한 모델
- 11개의 NLP Task에서 SOTA를 기록함
- 대형 코퍼스를 통해 pre-training하고, NLP Task마다 fine-tuning함

BERT
서론

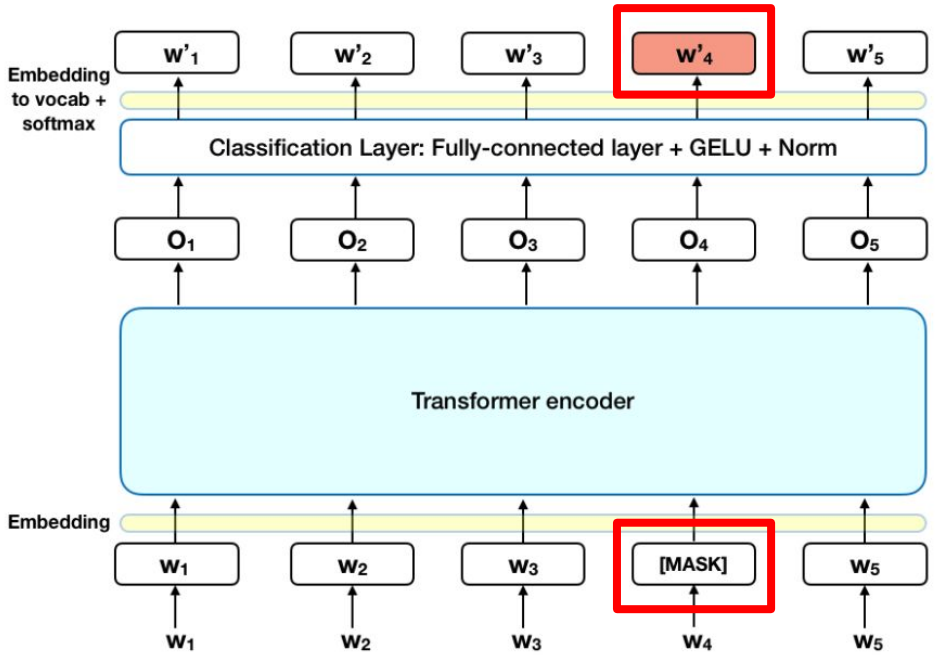
Transfer Learning

- 기존에 잘 학습된 모델을 사용해 새로운 모델을 만드는 데 사용함으로써 학습 속도 및 예측 성능을 높이는 방법
- 잘 학습된 모델을 만드는 과정을 **pre-training**이라고 하고, 이를 특정 NLP Task에 적용하고자 할 때 initial weight로 적용해 다시 학습하는 과정을 **fine-tuning**이라고 함
- 이미지 처리에서 미리 잘 학습된 ImageNet을 사용하는 것과 같음

BERT
서론


NSP
Mask LM
Mask LM
BERT
Masked Sentence A
Masked Sentence B
Unlabeled Sentence A and B Pair
Pre-training
MNLI
NER
SQuAD
Start/End Span
BERT
Question
Paragraph
Question Answer Pair
Fine-Tuning

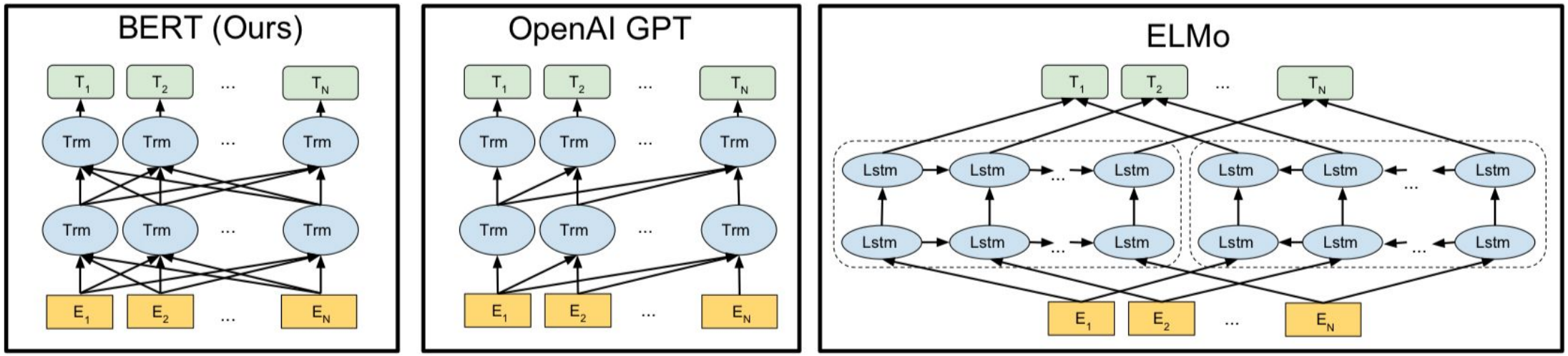
BERT
서론



- Input에서 무작위하게 몇 개의 token을 [MASK] 처리
- 주변 단어의 context만을 보고 mask된 단어를 예측하는 모델

BERT
서론

Pre-Training
Masked Language Model



- OpenAI GPT에서는 앞 단어를 보고 뒷 단어를 예측하는 transformer decoder 채용
- BERT에서는 input 전체와 mask된 token을 한 번에 transformer encoder에 넣고 원래 token 값을 예측하므로 deep bidirectional함

Pre-Training

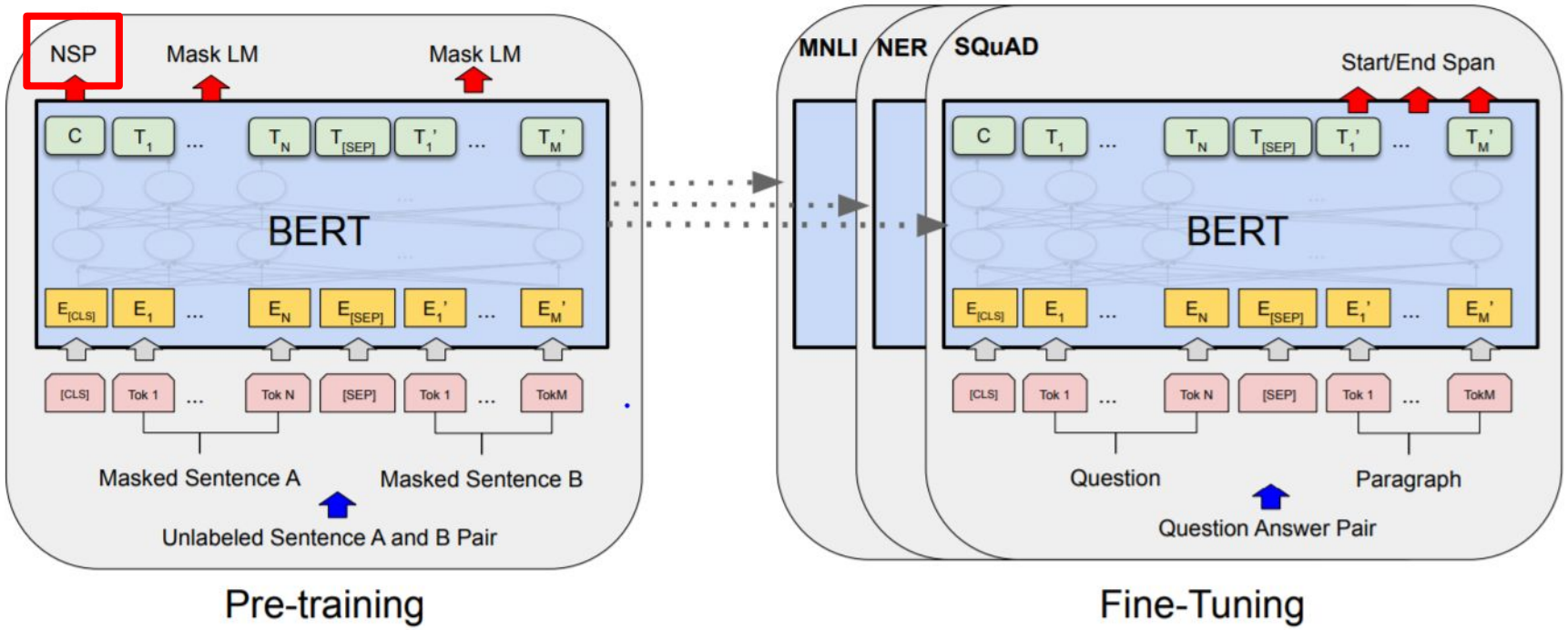
Next Sentence Prediction

- 두 문장을 pre-training할 때 같이 넣어 두 문장이 이어지는 문장인지 아닌지 맞추는 것(*IsNext, NotNext*)
- pre-training할 때 50:50 비율로 실제로 이어지는 두 문장과, 랜덤하게 추출된 두 문장을 넣어 BERT가 예측하도록 함
- NLI와 같은 task를 수행할 때 도움이 됨

BERT
서론

Pre-Training

Next Sentence Prediction



Transformer
서론

Transformer

Attention is all you need

- Google Brain & Research, Toronto Univ. 팀이 2017년에 공개한 논문
- seq2seq 모델에서 RNN을 제거하고 Attention만으로 Encoder, Decoder 모델을 구현함
- 병렬 처리가 가능해짐
- Neural Machine Translation을 위해 제안됨

Transformer
서론

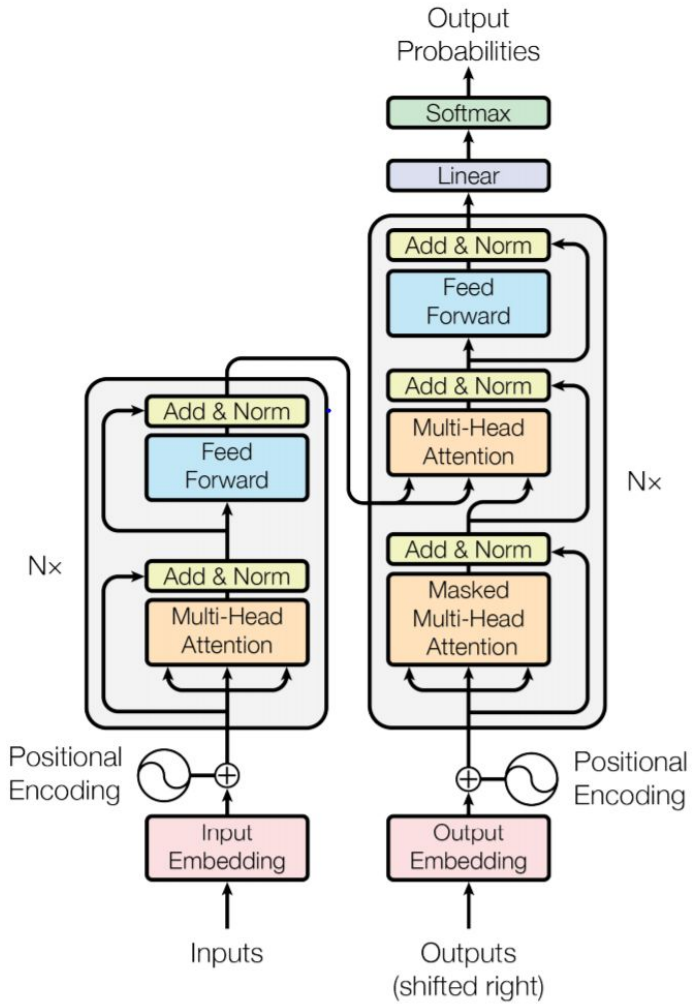


Figure 1: The Transformer - model architecture.

Transformer
서론

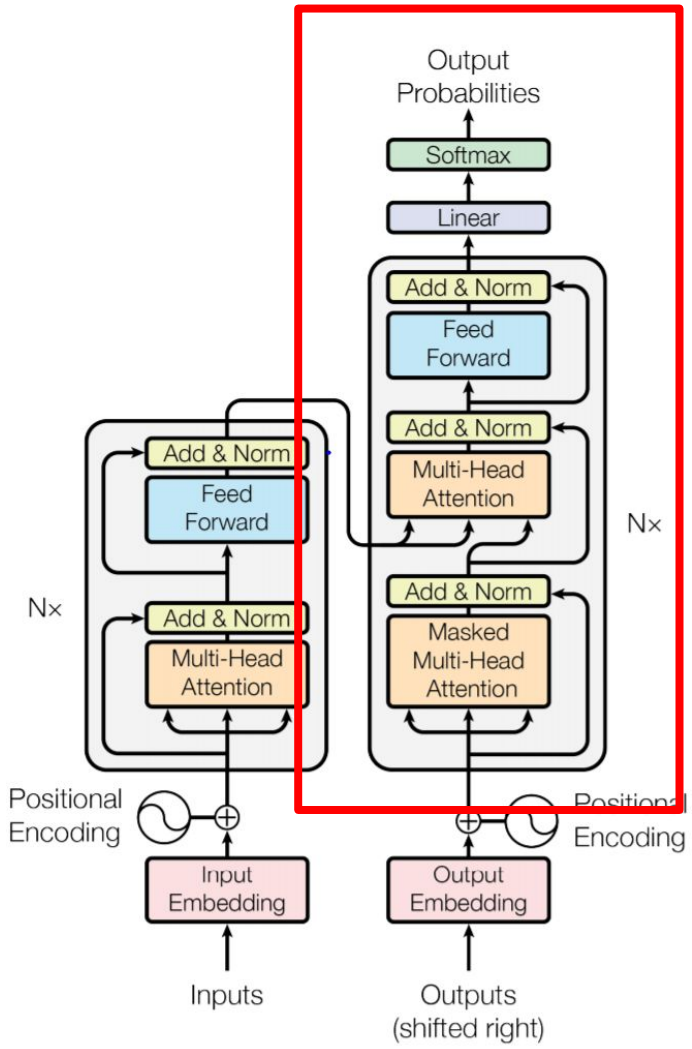


Figure 1: The Transformer - model architecture.

Decoder

Transformer
서론

Encoder

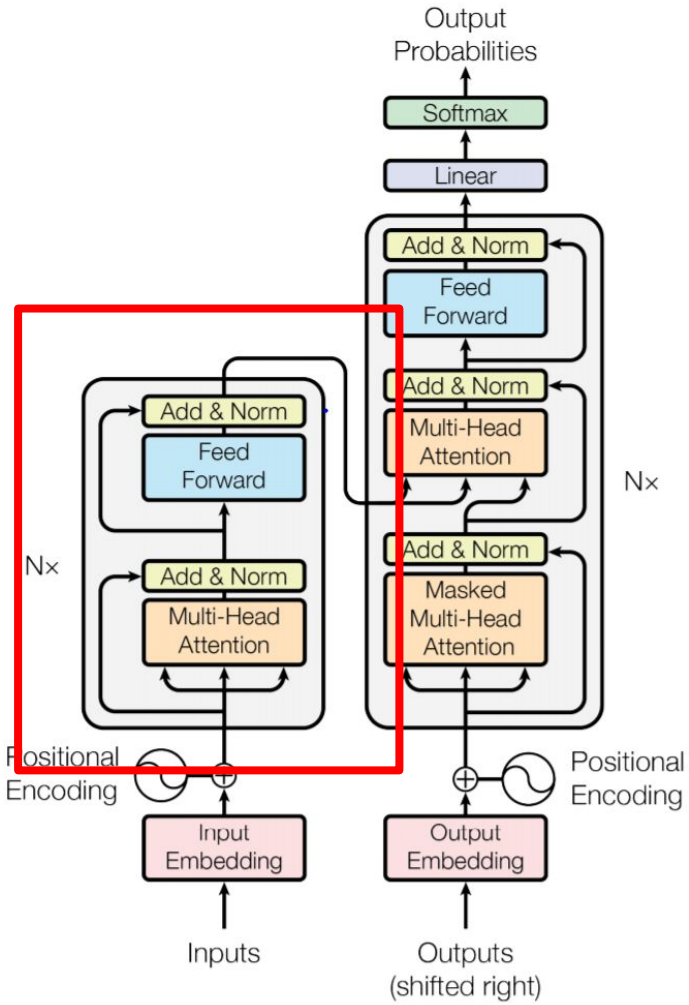
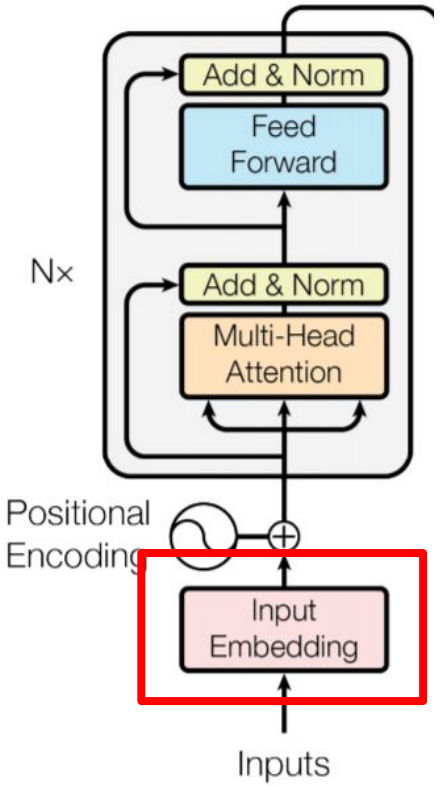


Figure 1: The Transformer - model architecture.

Input Embedding
모델

널리 알려져 있는 Word Embedding 층을 말함

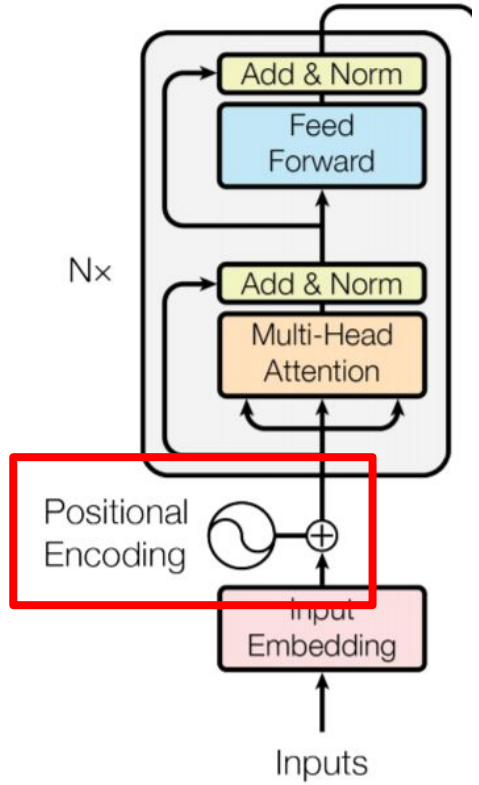
Word2Vec, GloVe… 어떤 모델을 사용하는지는 구체적으로 정해지지 않음



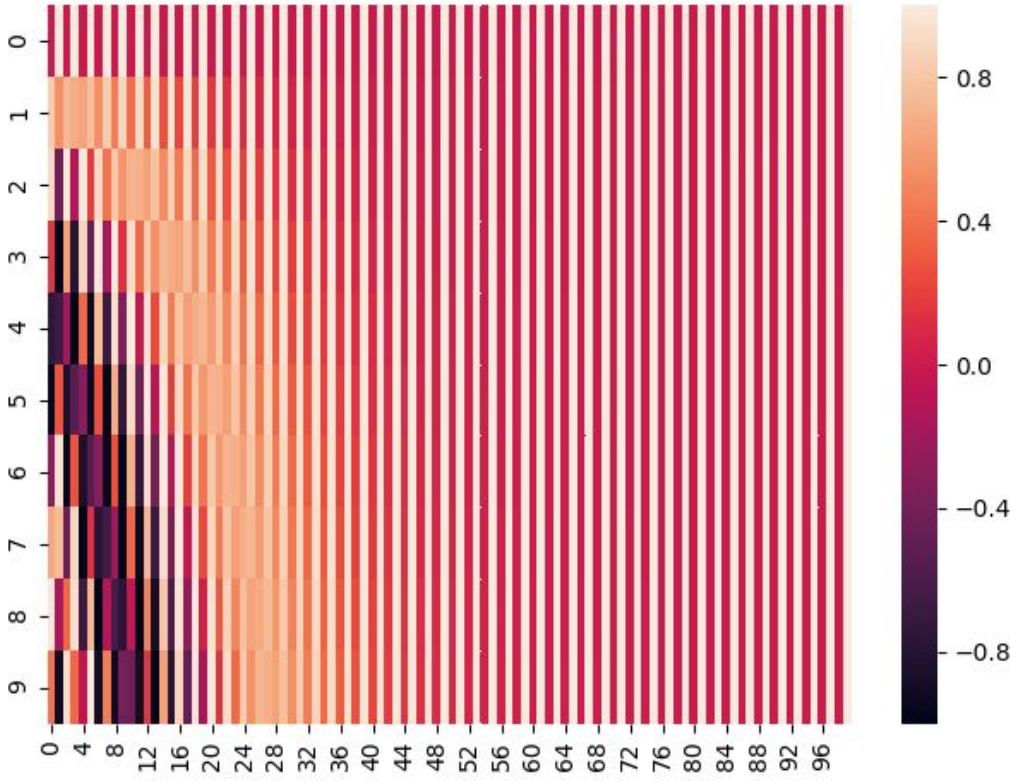
Positional Encoding
모델

Transformer 모델은 RNN을 사용하지 않기 때문에 문장의 순서 정보를 반영해주는 작업이 필요함

주기 함수의 출력값을 더해 순서 정보를 반영해주도록 함



Positional Encoding
모델

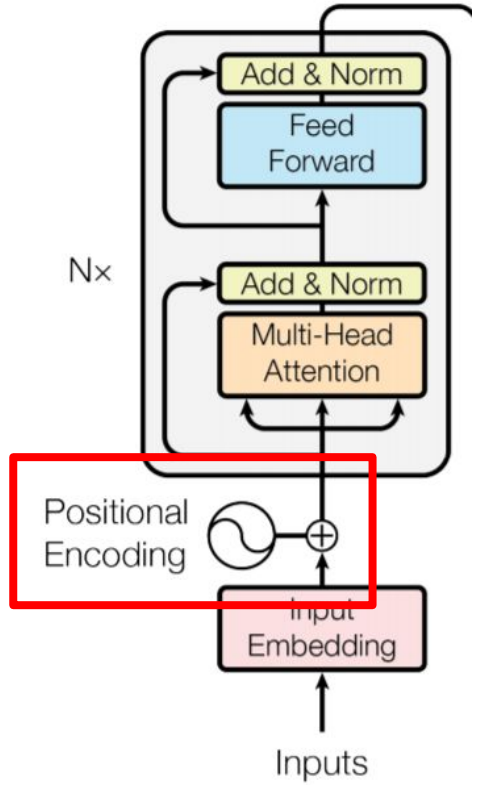


$$\mathrm{PE}_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

*i*번째 토큰에 대해 위와 같은 값을 더함

주기 함수는 무한대의 sequence 길이를 커버할 수 있음



Position Embedding
모델

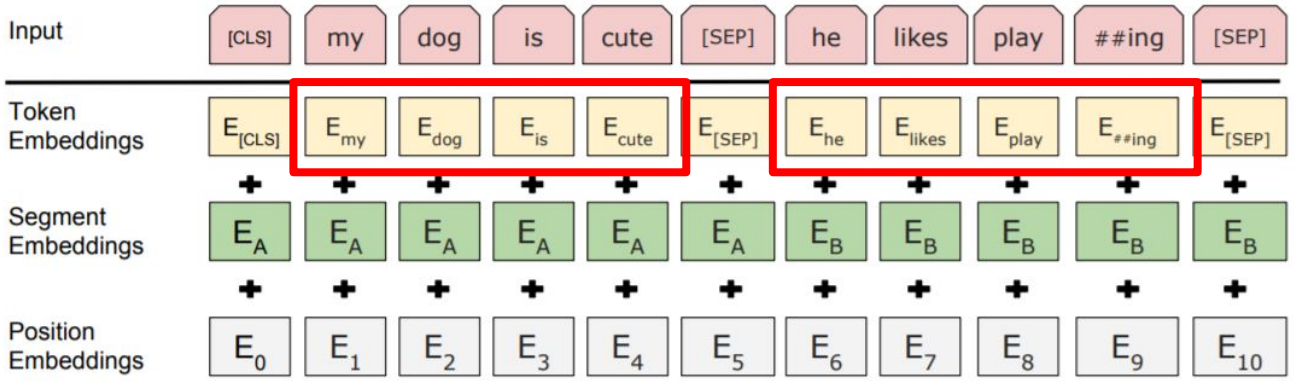
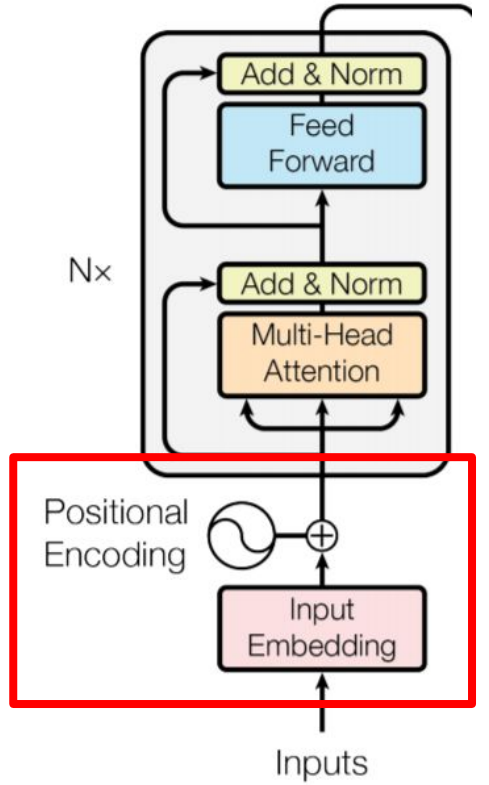


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.



- BERT는 3가지 Embedding을 더해 Input으로 사용함
- 문장 분석에 WordPiece Embedding 사용 (BERT-english의 경우 30,000개의 token 사용)
- 두 개 이상의 Input Sequence를 합쳐 사용하기 때문에 문장 구분을 위해 Segment Embedding (고정값) 사용
- RNN을 사용하지 않으므로 Positional Encoding 사용

Position Embedding
모델

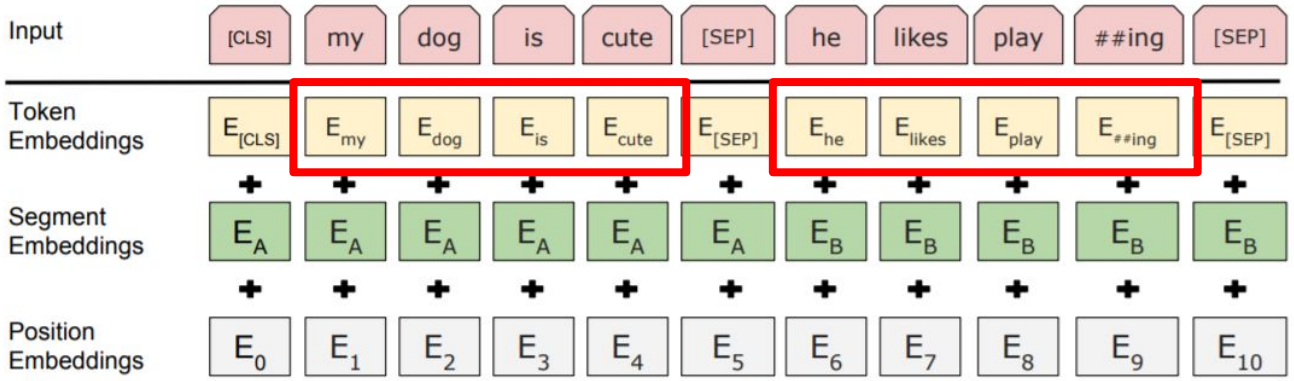
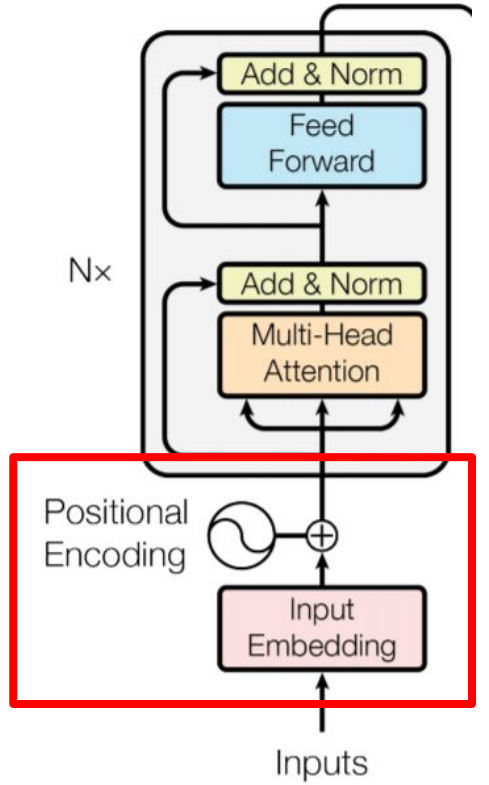


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

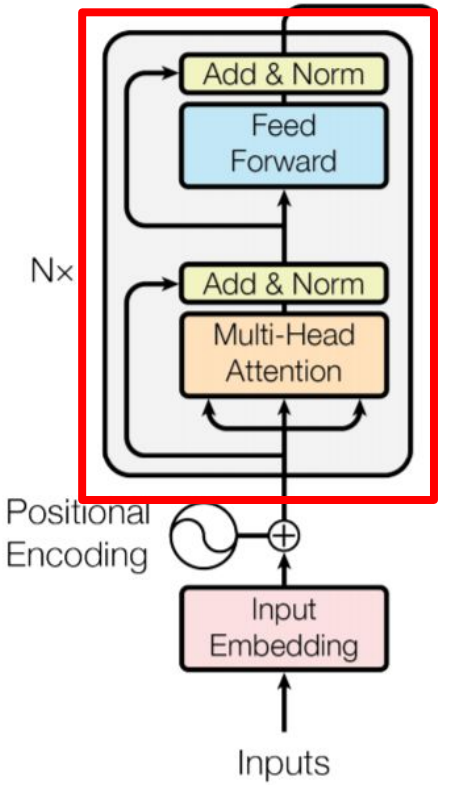


- **[CLS] token**: Classification token으로, 학습을 거치고 나면 token sequence를 결합한 것의 의미가 학습되고, 이를 이용해 classification을 수행한다
- **[SEP] token**: Separation token으로, 2개 이상의 문장을 입력으로 넣을 때 구분자 역할을 한다

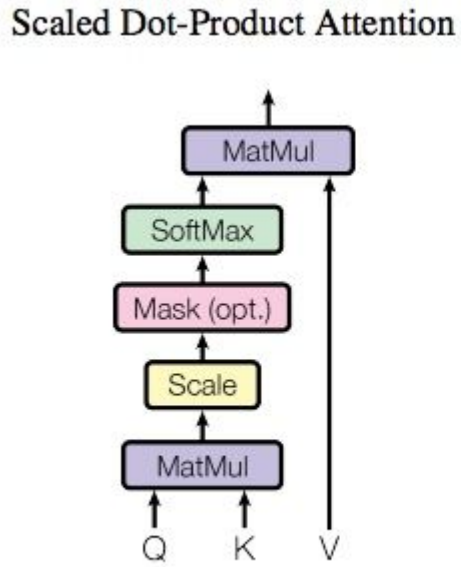
Encoder Block
모델

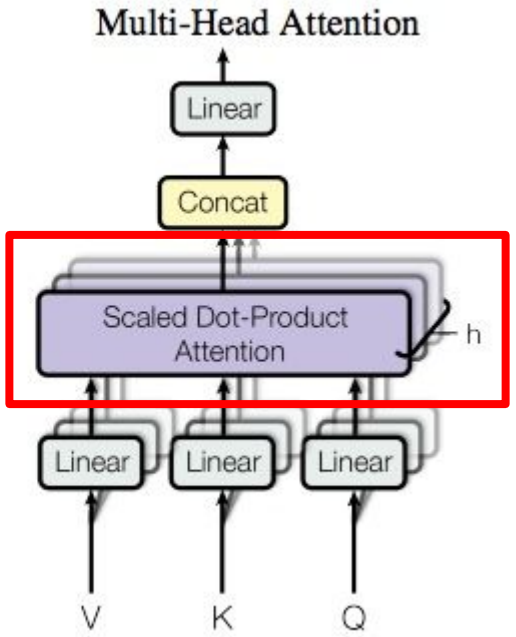
Encoder Block은 아래와 같은 Sub Block을 가지고, *N*개 쌓는 형태로 구현됨

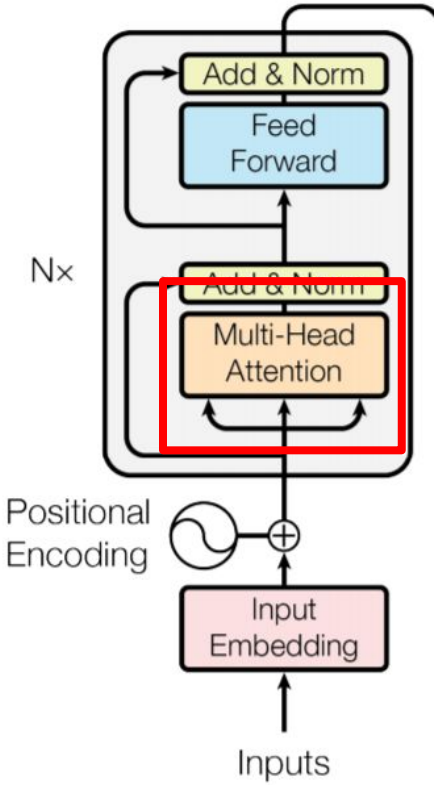
- **Multi-Head Attention**
- Residual Connection
- Normalization
- **Position-wise Feed Forward Network**
- Residual Connection
- Normalization



Scaled Dot-Product Attn.
모델

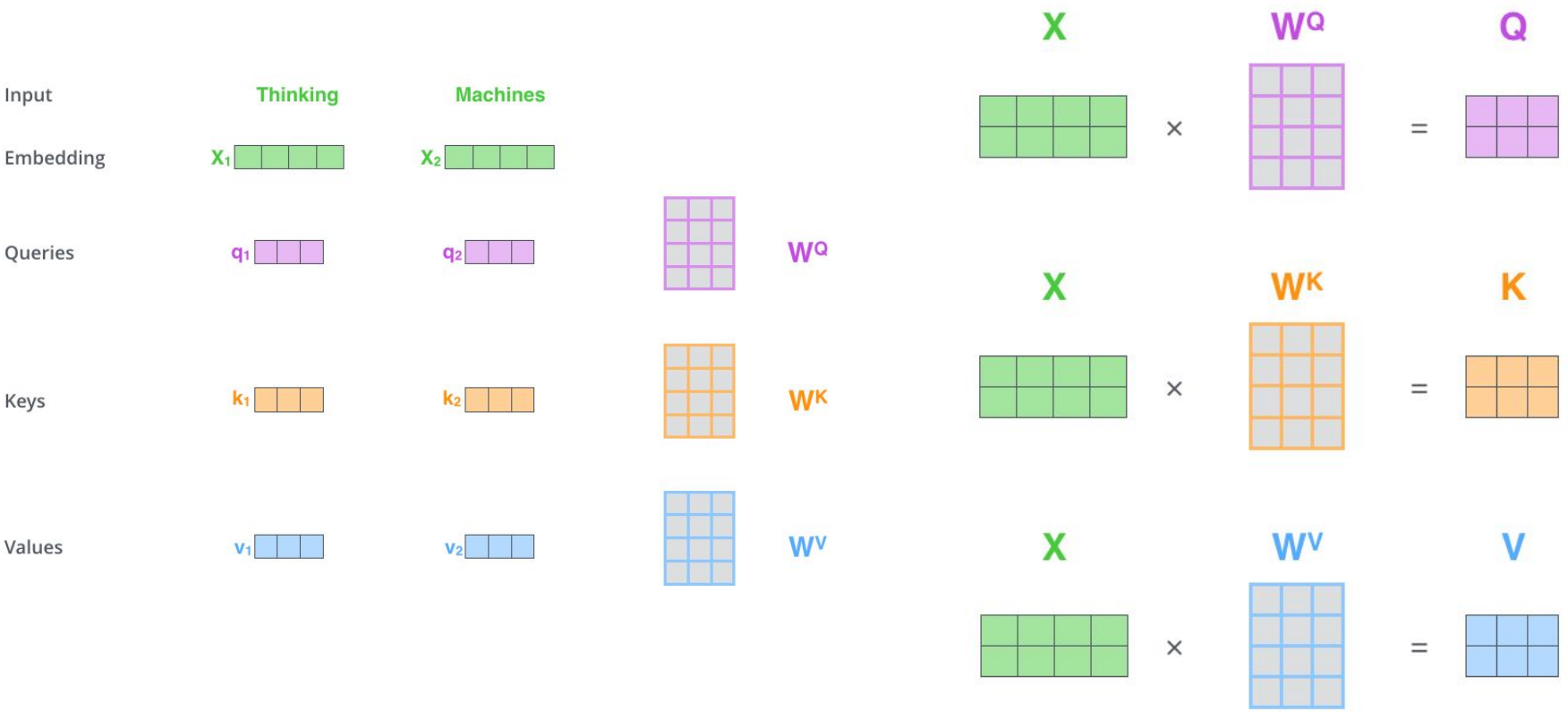


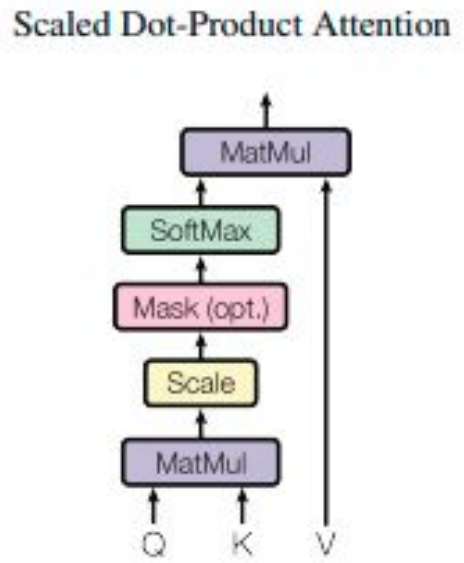




Multi-Head Attention의 Head 하나에는 Scaled Dot-Product Attention으로 구현된다

Scaled Dot-Product Attn. 모델





Q, K, V는 Input X와 Q, K, V Weight Matrix를 곱하여 구한다

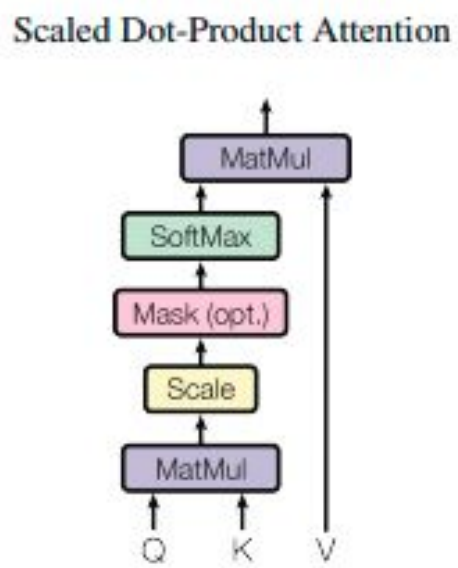
Scaled Dot-Product Attn.
모델

Query? Key? Value?

Query는 **유사도를 계산**할 hidden state vector

Key는 **비교 대상**인 hidden state vector

Value는 **유사도만큼 가중합**할 hidden state vector



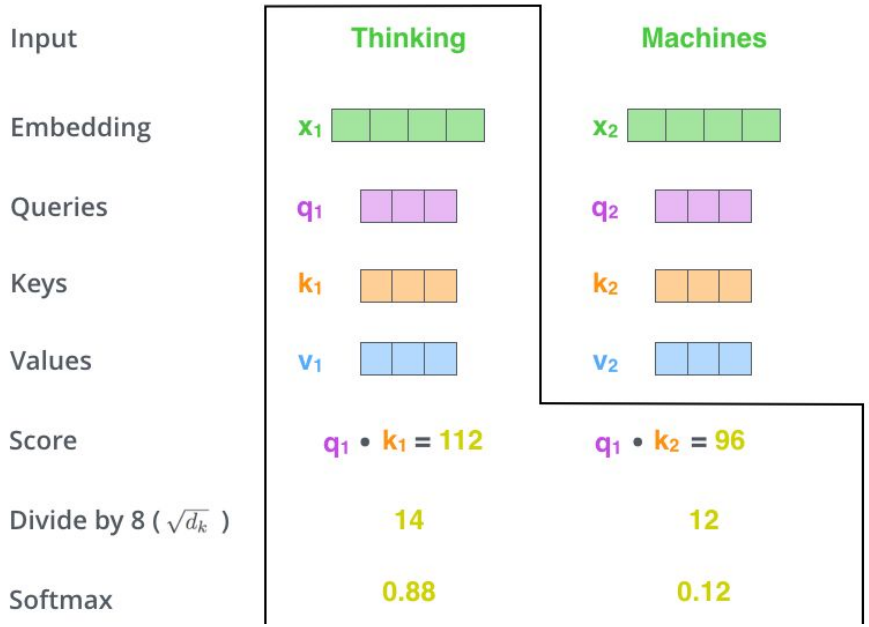
seq2seq 모델과 비교하면…

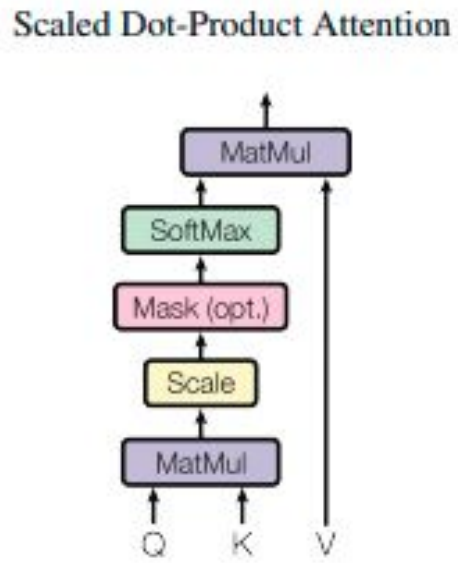
Query=**Decoder**

Key=**Encoder**

Value=**Encoder**

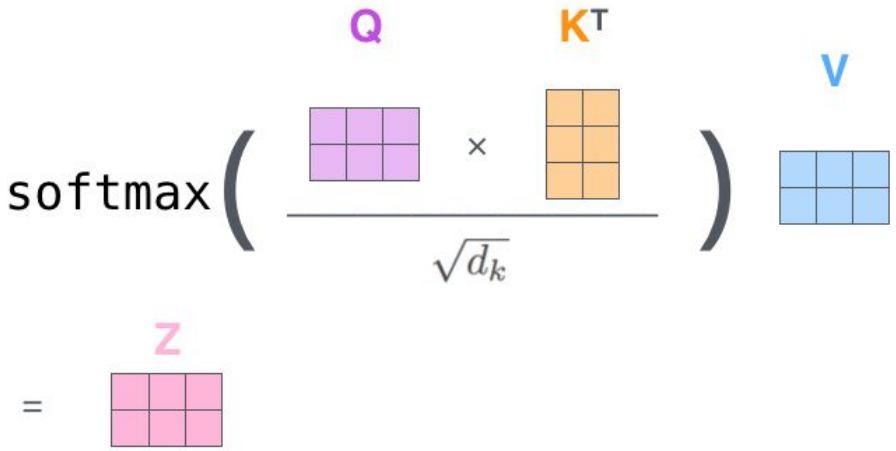
Scaled Dot-Product Attn.
모델

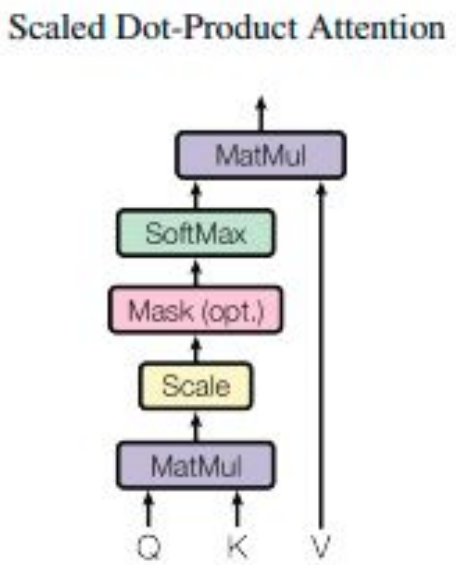




Q와 K의 유사도를 Dot-Product로 구한 후, 값이 커지는 것을 보정하기 위해 Dimension of K로 Scaling한다

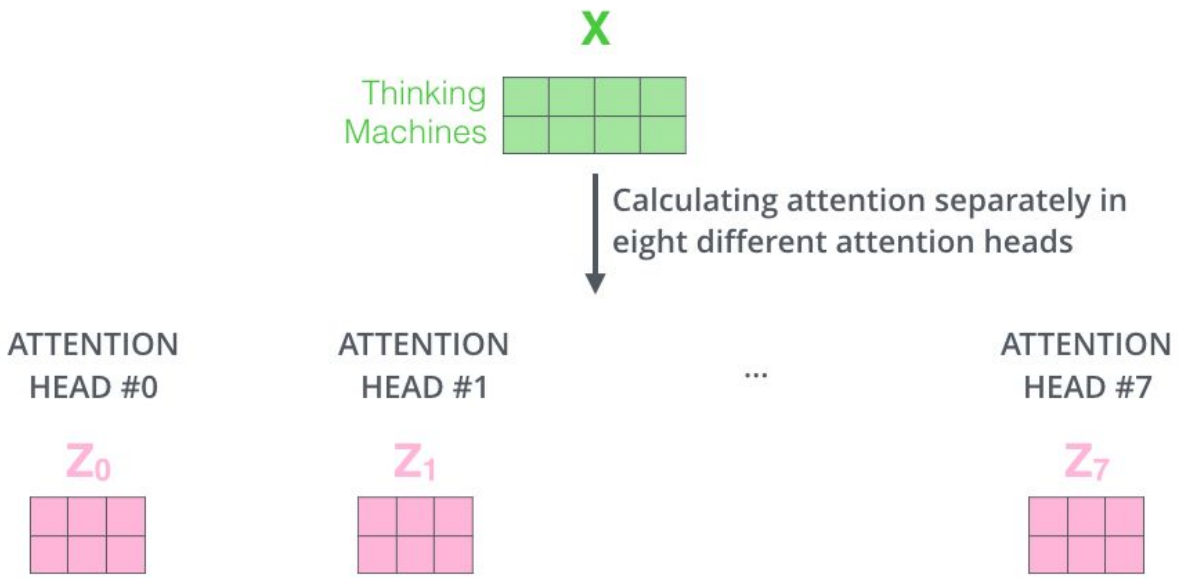
Scaled Dot-Product Attn.
모델

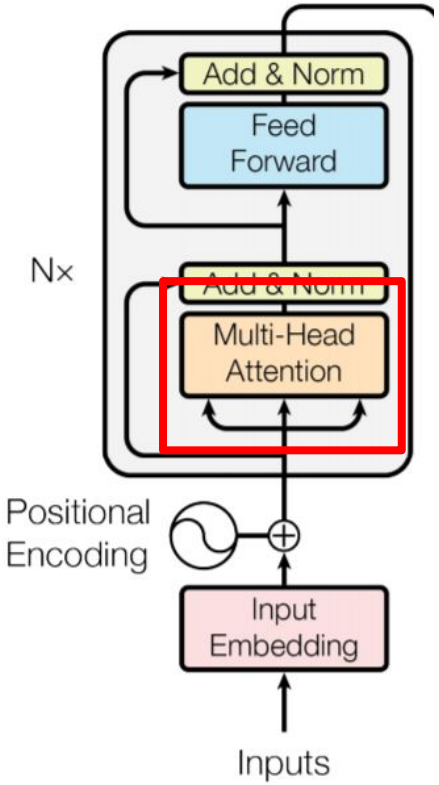




Q와 K의 유사도를 V와 가중합하면 Attention 계산이 끝난다

Multi-Head Attn.
모델

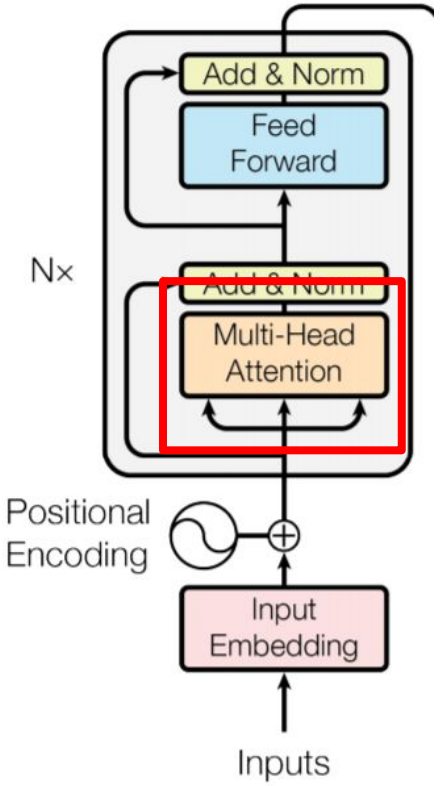




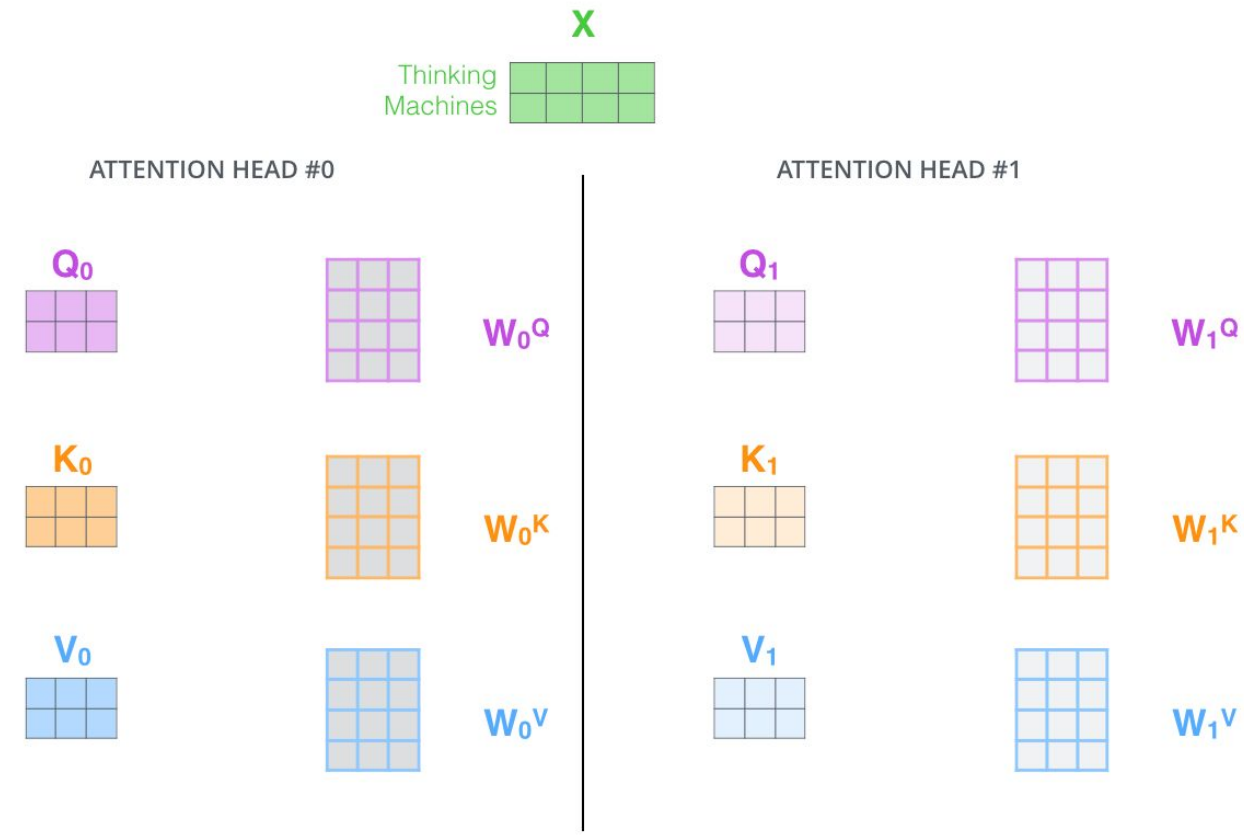
Attention을 수행하는 데 있어서 여러 관점(Multi-Head)에서 수행함

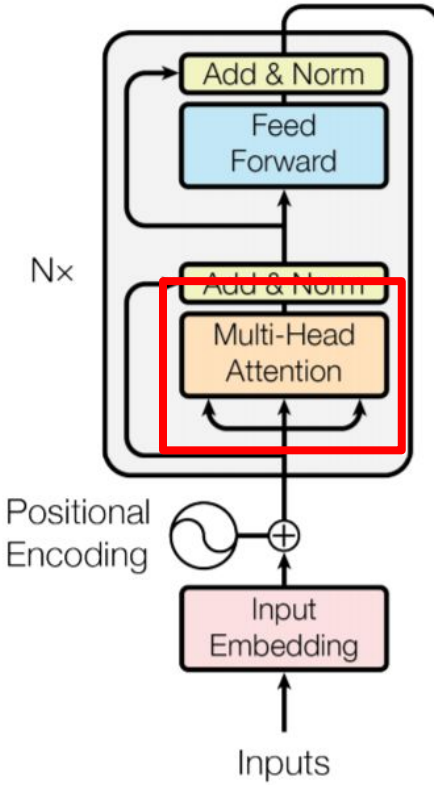
Multi-Head Attn.
모델

- 저자는 Q, K, V에 대한 Single Attention을 수행하는 것보다 Q, K, V를 각각 병렬적으로 Attention을 수행하는게 더 효과적인 것을 발견함
- Multi-Head Attention은 모델이 서로 다른 위치의 정보를 함께 집중하게 함
- Single Attention으로는 averaging*(weighted sum > scaling)*이 이를 억제함



Multi-Head Attn.
모델





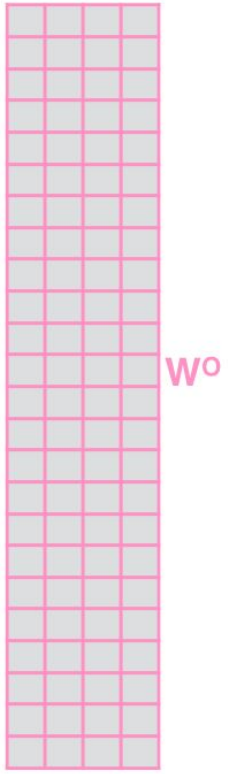
- 문장간 Attention을 수행하는 데 있어서 여러 관점(Multi-Head)에서 수행함
- 각각의 Attention Head는 서로 다른 Q, K, V Weight Matrix를 가지고 있음

Pos-wise FFN
모델

1) Concatenate all the attention heads

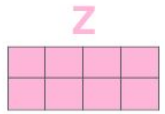
2) Multiply with a weight matrix W^O that was trained jointly with the model

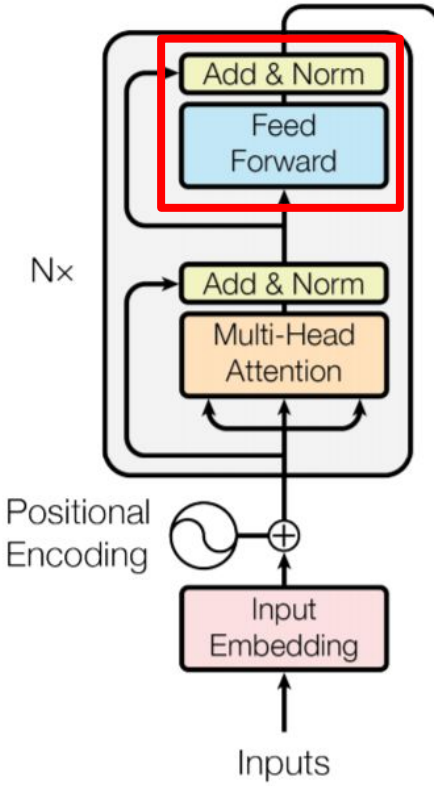
X



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

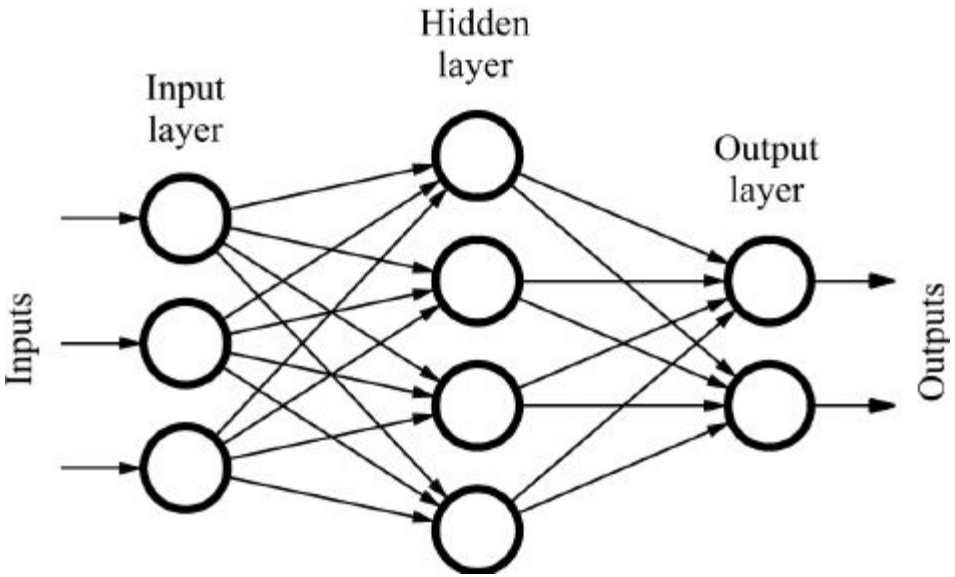
=

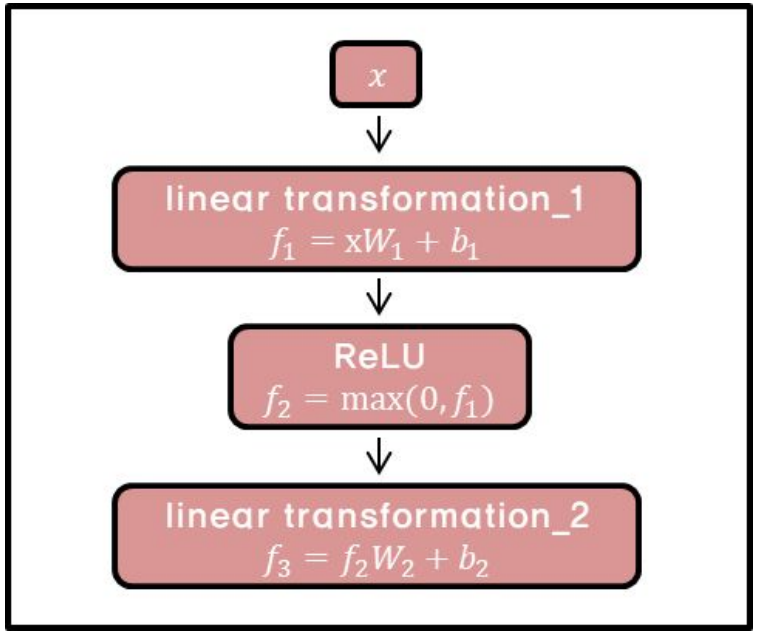


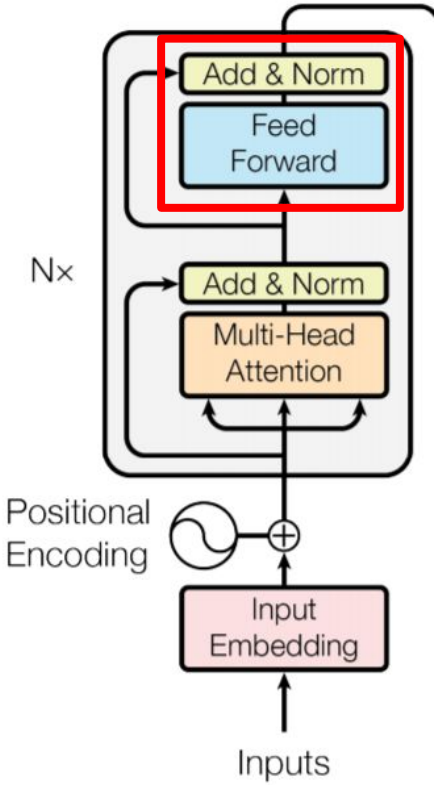


Attention 결과를 단순히 이어붙여 Output Weight Matrix에 곱하면 Multi-Head 끝!

Pos-wise FFN
모델







- 이제 여러 관점으로 바라본 Attention을 하나로 합치기 위해 Pos-wise FFN을 통과한다
- Position마다, 각 단어마다 적용되기 때문에 Pos-wise하다고 이름 붙여졌다

Pos-wise FFN
모델

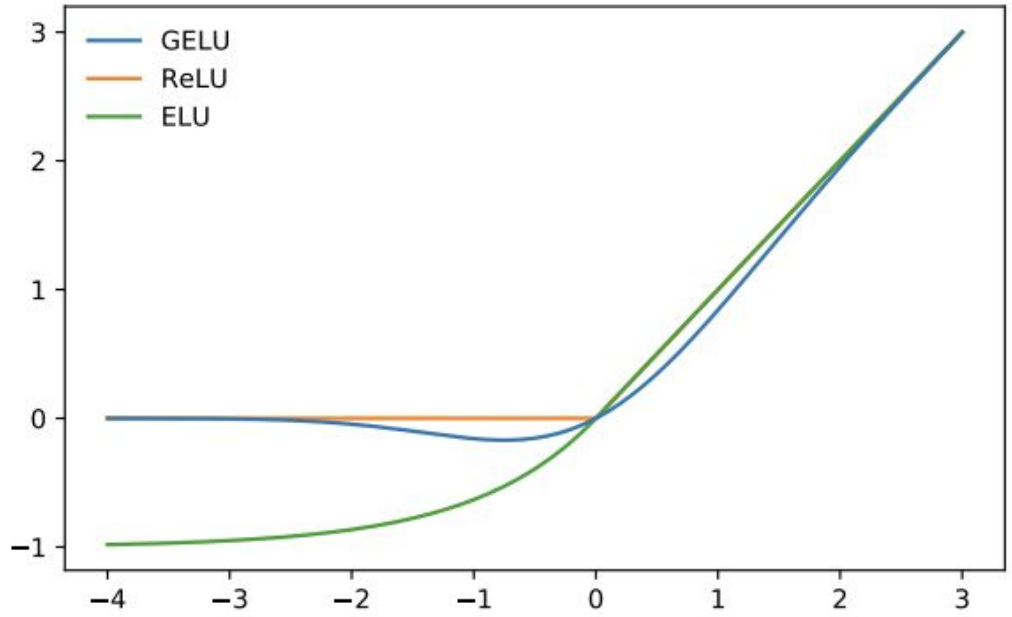
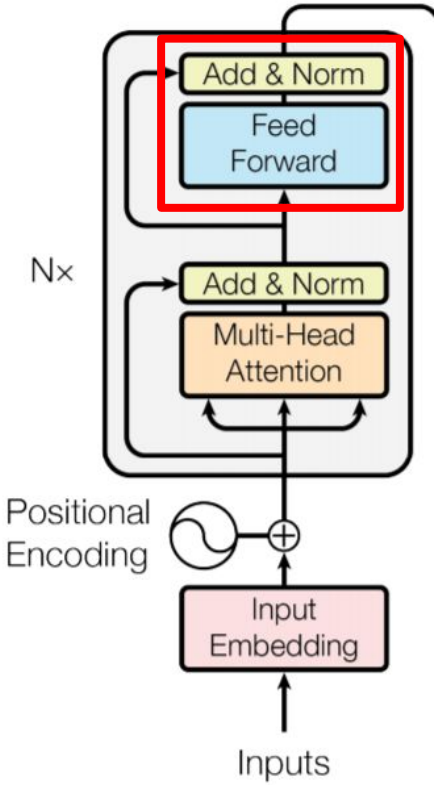


Figure 1: The GELU ($\mu = 0, \sigma = 1$), ReLU, and ELU ($\alpha = 1$).



- Original Transformer는 이 네트워크에 ReLU를 적용함
- BERT는 GELU를 적용해 음수에서도 미분할 수 있도록 하여 약간의 gradient를 전달함

Multi-Head Attn.
모델

1) This is our input sentence*

2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

Thinking
Machines

X

W_0^Q W_0^K W_0^V

Q_0 K_0 V_0

Z_0

W_1^Q W_1^K W_1^V

Q_1 K_1 V_1

Z_1

...

W_7^Q W_7^K W_7^V

Q_7 K_7 V_7

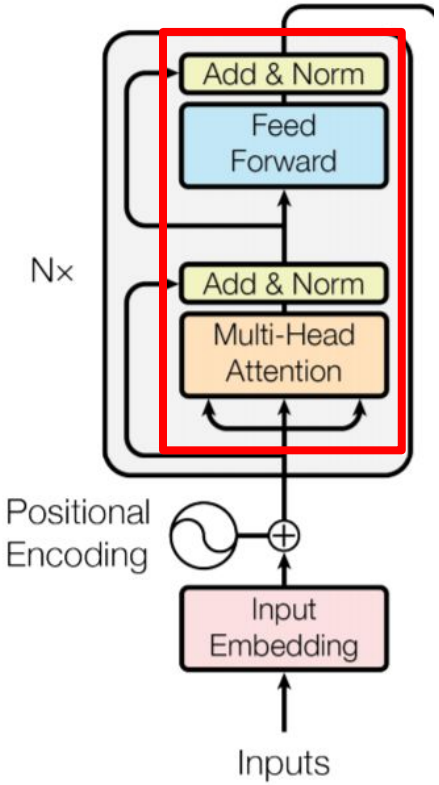
Z_7

W^O

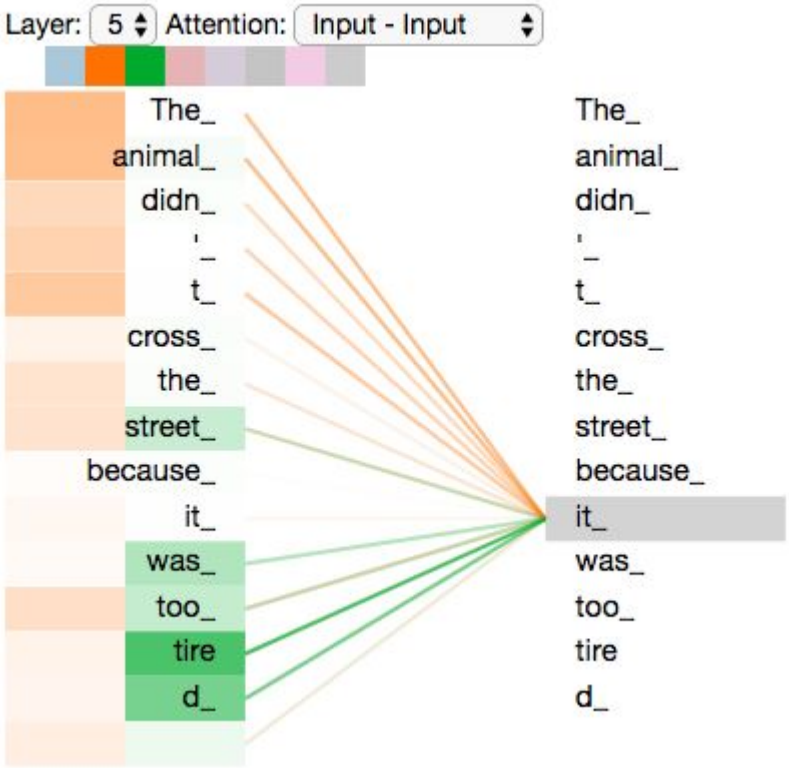
Z

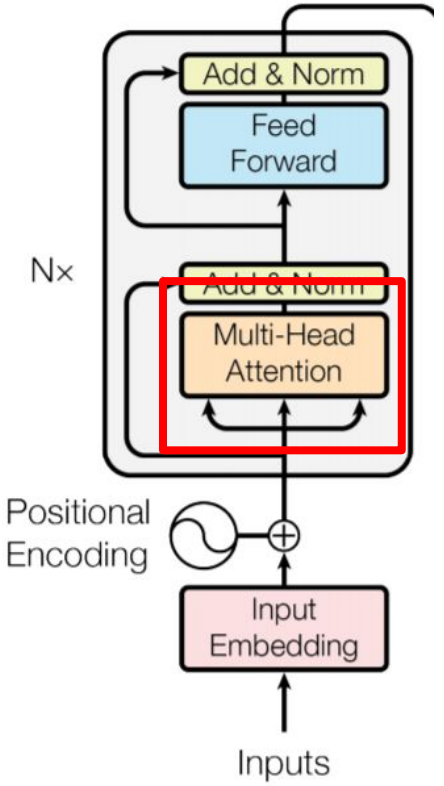
R

* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



Multi-Head Attn.
모델





Multi-Head Attention 시각화

Multi-Head Attn.
모델

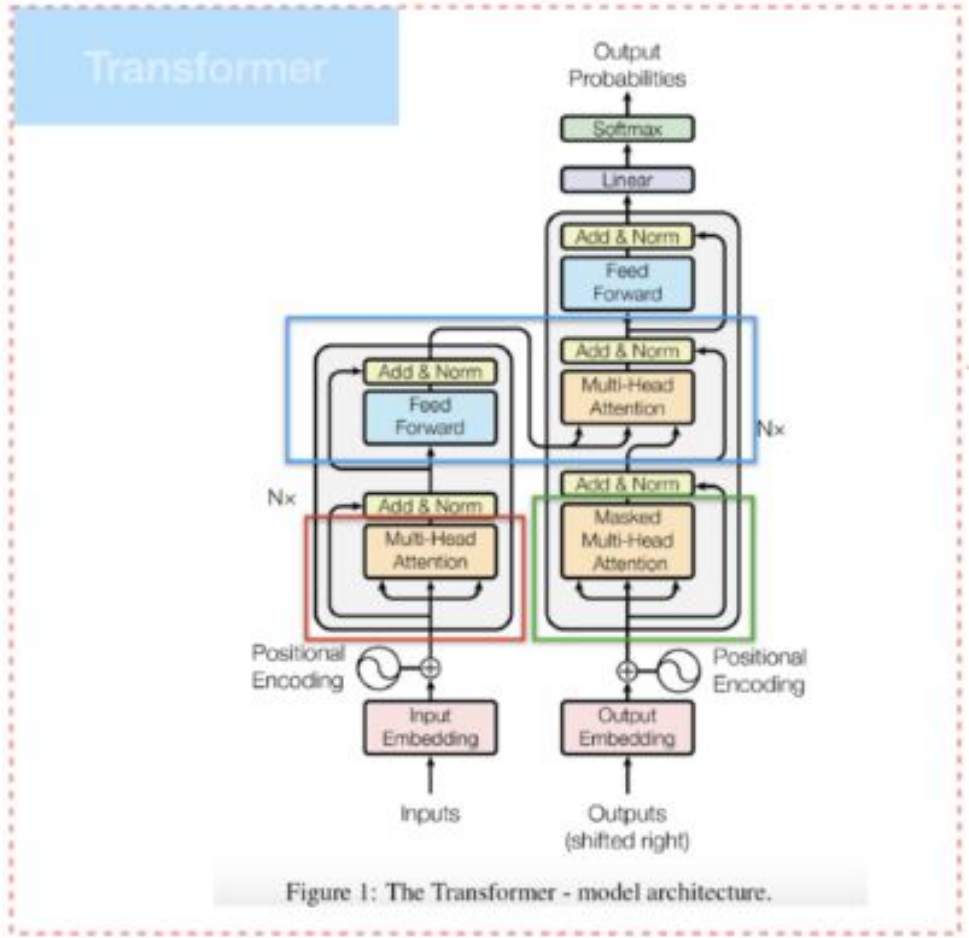


Figure 1: The Transformer - model architecture.

encoder self attention

1. Multi-head Attention
2. Query=Key=Value

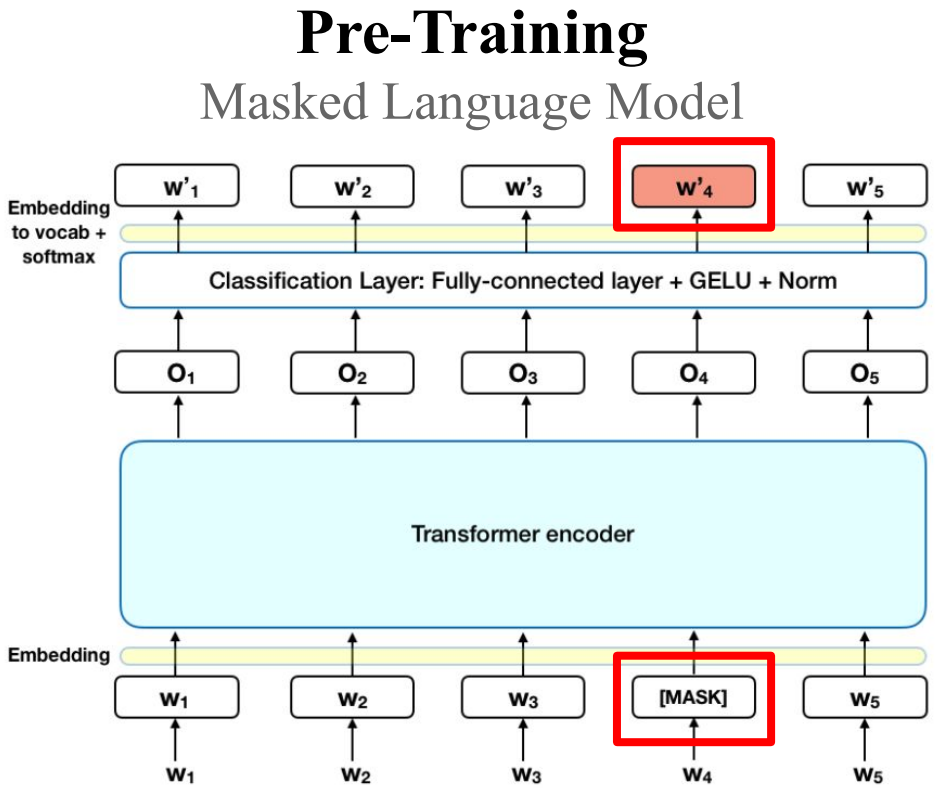
decoder self attention

1. Masked Multi-head Attention
2. Query=Key=Value

encoder-decoder attention

1. Multi-head Attention
2. Encoder Self attention=Key=Value
3. Decoder Self attention=Query

Masked LM
학습



- Input에서 무작위하게 몇 개의 token을 [MASK] 처리
- 주변 단어의 context만을 보고 mask된 단어를 예측하는 모델

Masked LM
학습

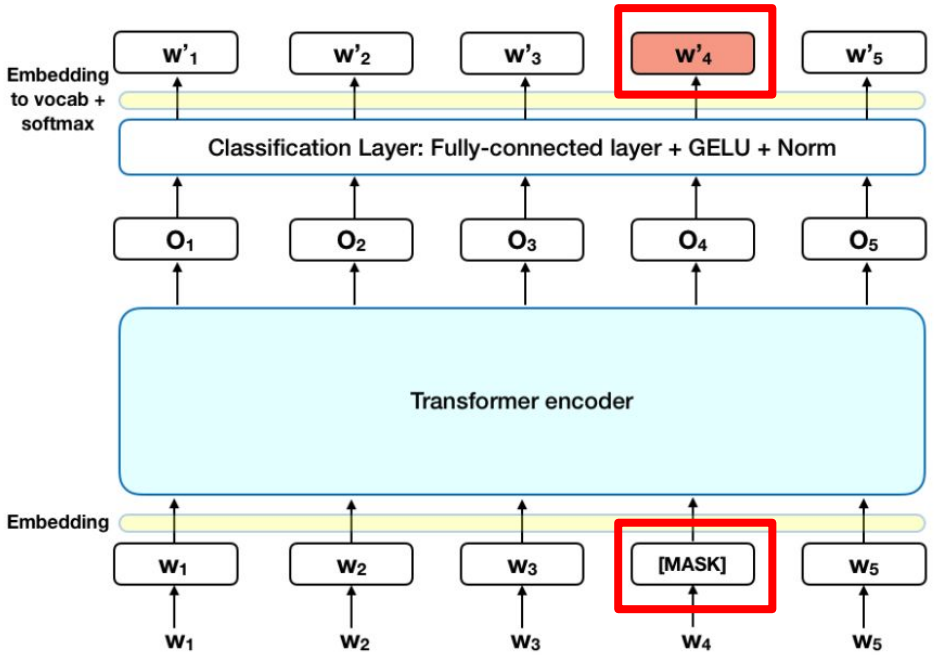
MLM Output Layer의 크기는 Vocab Size이고, Softmax Activation한다

MLM의 단점

1. pre-training과 fine-tuning 사이의 간극 발생 ([MASK] token만 학습하면 fine-tuning 시 이 token을 못 볼 것이고, 아무것도 예측할 필요가 없다고 판단해 성능에 해를 끼침)
2. 각 batch의 15%([MASK] token 비율)만 학습
3. 높은 Softmax 연산 비용

간극을 줄이기 위해 [MASK] token을 다음 규칙에 따라 치환한다

1. Input의 15%는 다음 방법으로 치환된다
2. 80%는 [MASK] token으로 치환한다
3. 10%는 random word로 치환한다
4. 10%는 기존 단어 그대로 두고, 이 단어를 예측한다

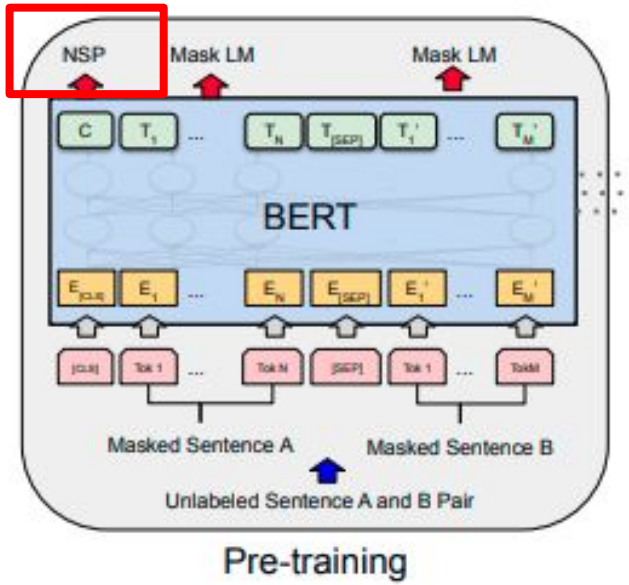


NSP
학습

Pair Sentence에 대해 문장 관계가 *IsNext*인지 *NotNext*인지 판별

QA, NLI에서 필요한 sentence간의 relation 학습

- QA: 질문이 주어지면 답변 문장 출력
- NLI: 전제가 주어졌을 때 가설이 참인지, 거짓인지, 미결정인지 판별



Fine-Tuning
학습

BERT는 상당히 큰 모델이다

$Transformer Block = L, Embedding Size = H, Attention Head = A$ 일 때,

$$BERT_{BASE}(L = 12, H = 768, A = 12, Total Parameters = 110M)$$
$$BERT_{LARGE}(L = 24, H = 1024, A = 16, Total Parameters = 340M)$$

Fine-Tuning
학습

BERT는 상당히 큰 모델이다

$Transformer Block = L, Embedding Size = H, Attention Head = A$ 일 때,

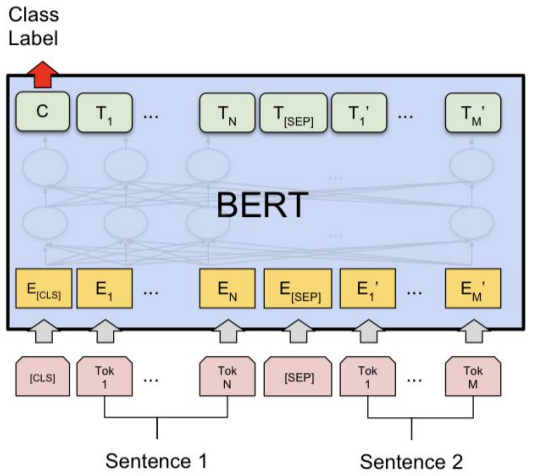
$$BERT_{BASE}(L = 12, H = 768, A = 12, Total Parameters = 110M)$$

$$BERT_{LARGE}(L = 24, H = 1024, A = 16, Total Parameters = 340M)$$

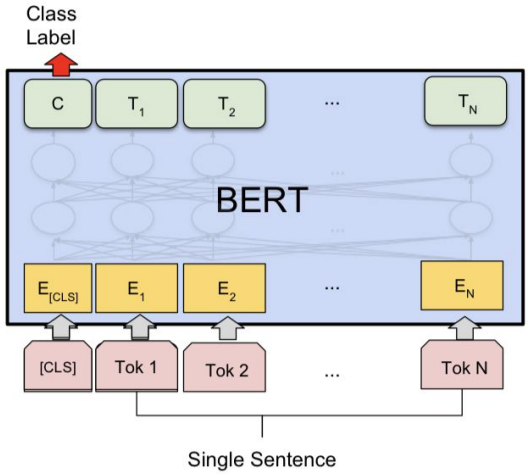
SOTA(state-of-the-art) 모델

Fine-Tuning
학습

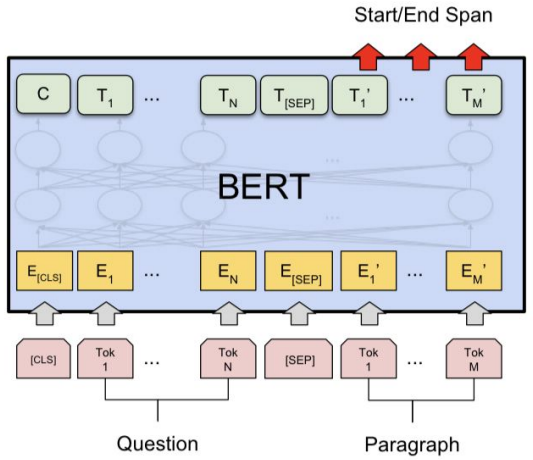
11개 NLP Task마다 fine-tuning 방법이 조금씩 다름



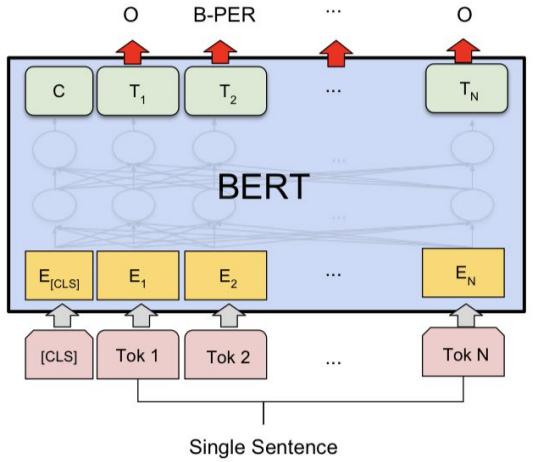
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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

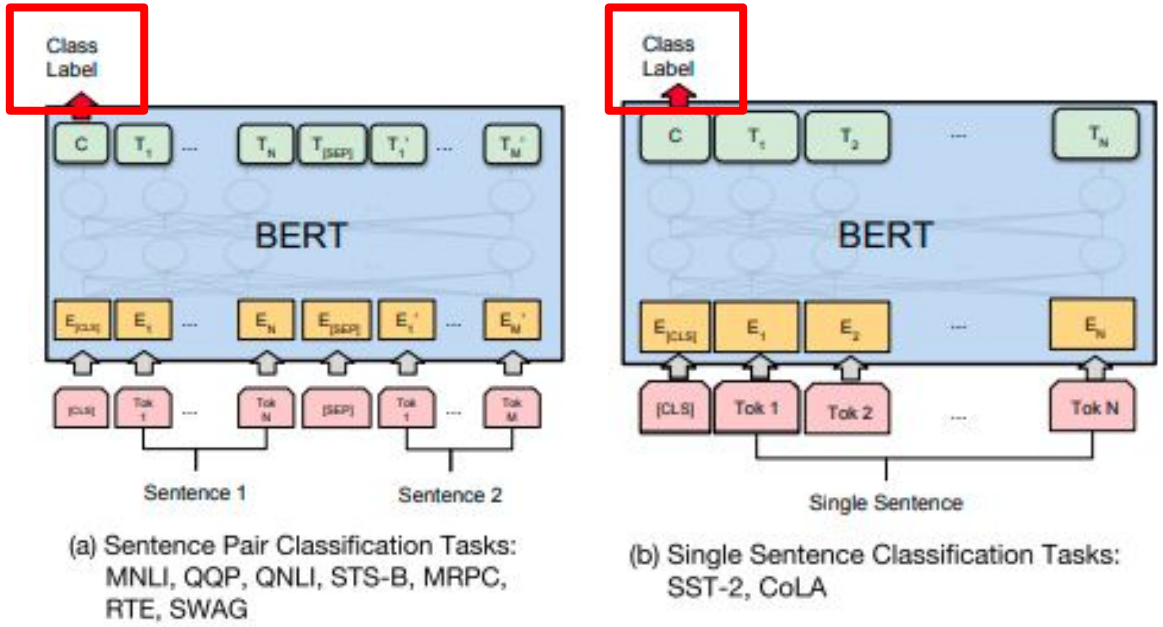


(c) Question Answering Tasks: SQuAD v1.1



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

Fine-Tuning
학습

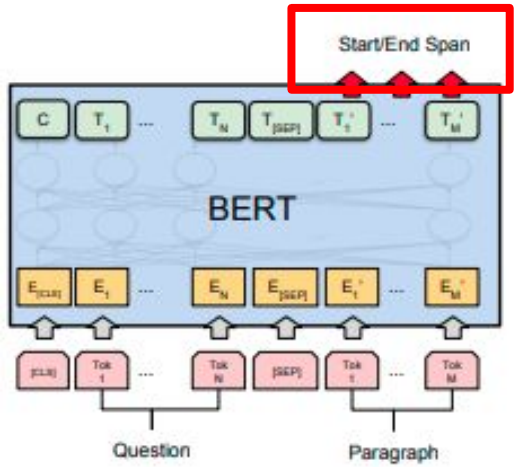


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

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

[CLS] token output에 classification layer(softmax)를 붙여 fine-tuning함

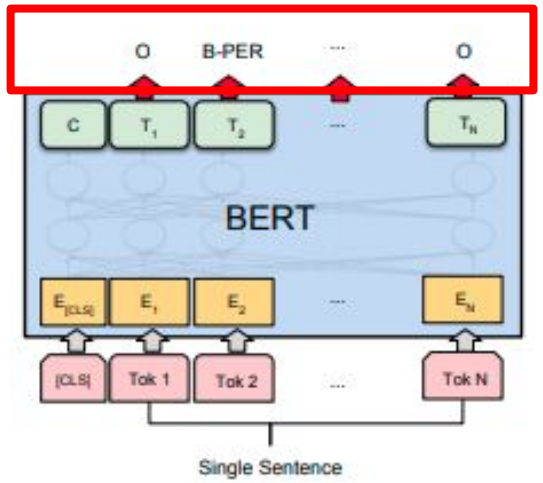
Fine-Tuning
학습



(c) Question Answering Tasks:
SQuAD v1.1

QA는 Question에 정답이 되는 Paragraph의 substring을 찾아내는 문제이므로,
[CLS] 이후의 token 중에서 시작 지점과 끝 지점을 찾아내는 task로 fine-tuning함

Fine-Tuning
학습



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

형태소 분석 또는 Named Entity Recognition(NER)은 token마다 classification layer를 붙여 fine-tuning함

참고문헌

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). **Attention is all you need**. Advances in Neural Information Processing Systems, 2017-Decem(Nips), 5999–6009.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. Mlm.
- **"Season 07 - 05. Attention Is All You Need."** HYU-AILAB/ai-seminar. 2020년 4월 13일 접속, https://github.com/HYU-AILAB/ai-seminar/tree/master/season_07/05.%20Attention%20is%20All%20you%20Need.
- **"Season 04 - 09. [NLP] BERT (Pre-training of Deep Bidirectional Transformers for Language Understanding)."** HYU-AILAB/ai-seminar. 2020년 4월 13일 접속, https://github.com/HYU-AILAB/ai-seminar/tree/master/season_04/09.%20%5BNLP%5D%20BERT%20(Pre-training%20of%20Deep%20Bidirectional%20Transformers%20for%20Language%20Understanding).
- **"Transformer: All you need is Attention (설명/요약/정리)."** 만렙개발자. 2019년 12월 31일 수정, 2020년 4월 13일 접속, https://lv99.tistory.com/26.
- **"BERT 논문정리."** Mino-Park7 NLP Blog. 2018년 12월 12일 수정, 2020년 4월 13일 접속, https://mino-park7.github.io/nlp/2018/12/12/bert-논문정리/.
- **"인공지능(AI) 언어모델 'BERT(버트)'는 무엇인가."** 인공지능신문. 2019년 1월 3일 수정, 2020년 4월 13일 접속, http://www.aitimes.kr/news/articleView.html?idxno=13117.
- **"BERT 톺아보기."** The Missing Papers. 2019년 1월 24일 수정, 2020년 4월 13일 접속, http://docs.likejazz.com/bert/.
- **"The Illustrated Transformer."** NLP in Korean. 2018년 12월 20일 수정, 2020년 4월 13일 접속, https://nlpinkorean.github.io/illustrated-transformer/.
- **"Transformer: Attention Is All You Need 리뷰."** Huiwon. 2018년 12월 6일 수정, 2020년 4월 13일 접속, https://vanche.github.io/NLP_Transformer/.
- **"Attention is all you need paper 뽀개기."** 포자랩스의 기술 블로그. 2018년 9월 15일 수정, 2020년 4월 13일 접속, https://pozalabs.github.io/transformer/.