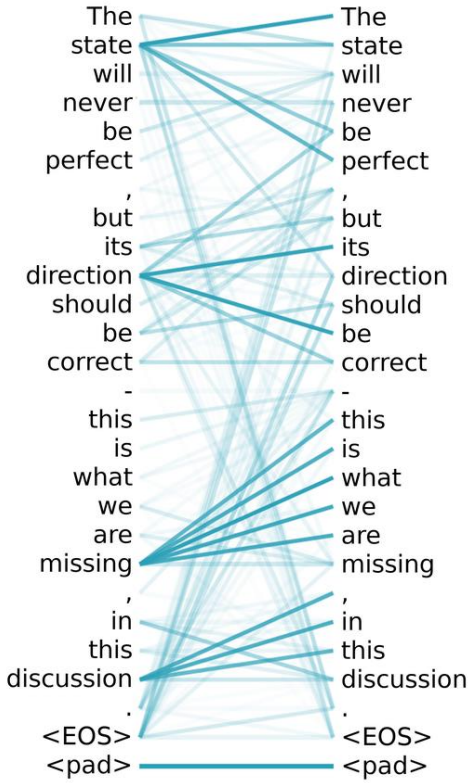
Transformer

김균엽

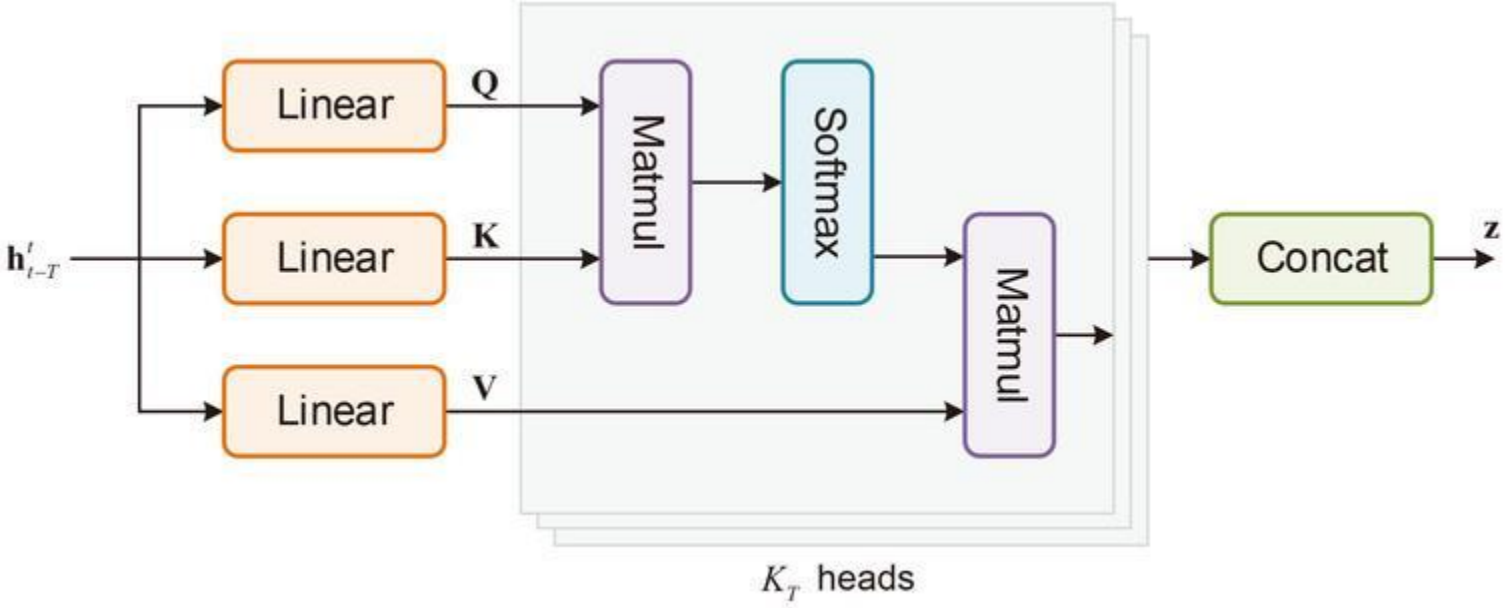
Self-attention

- **Self-attention**
  - Q와 K, V가 같은 단어인 파생된 Attention.
  - 문장과 문장과의 attention을 통해 단어간의 관계를 학습하는 network



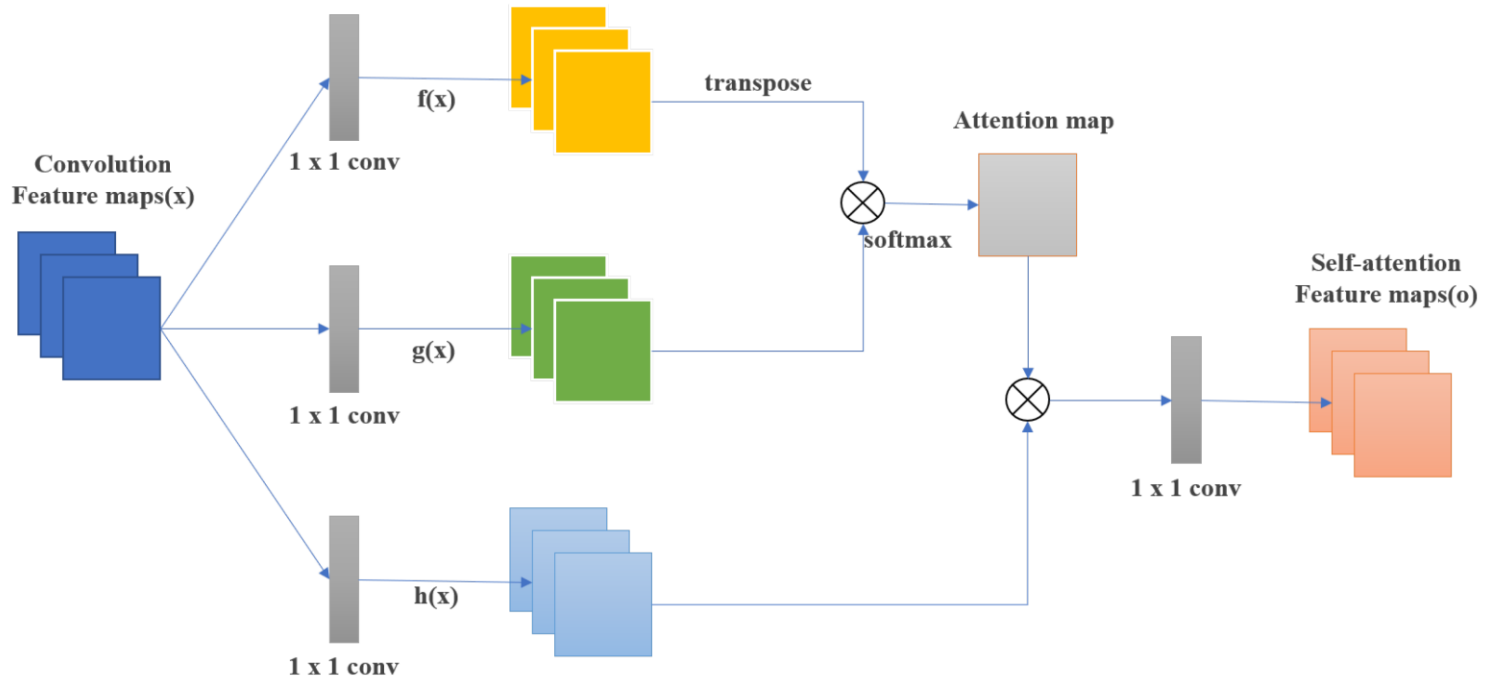
Self-attention

- **Self-attention**
  - 한 문장의 단어들을 MLP를 이용하여 Q,K,V로 분리한다
  - QKV를 통해 attention을 진행한다.



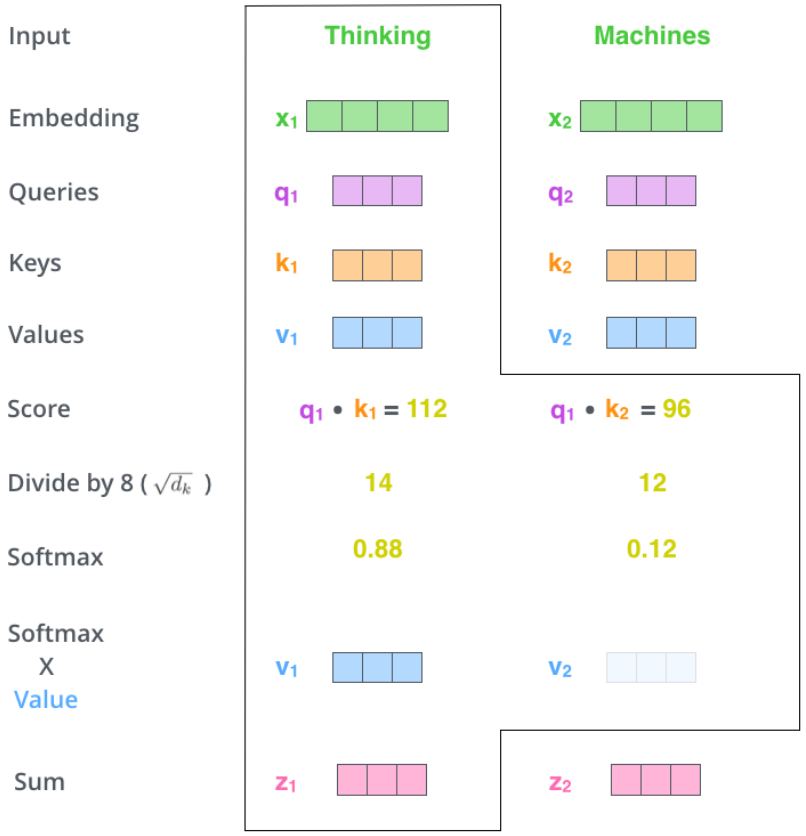
Self-attention

- **Self-attention**
  - 한 문장의 단어들을 MLP를 이용하여 Q,K,V로 분리한다
  - QKV를 통해 attention을 진행한다.



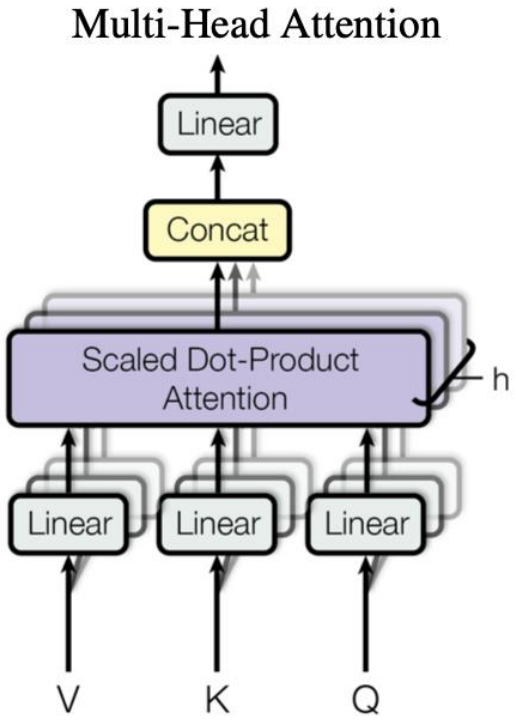
Multi-head attention

- **Self-attention**
  - 한 문장의 단어들을 MLP를 이용하여 Q,K,V로 분리한다
  - QKV를 통해 attention을 진행한다.



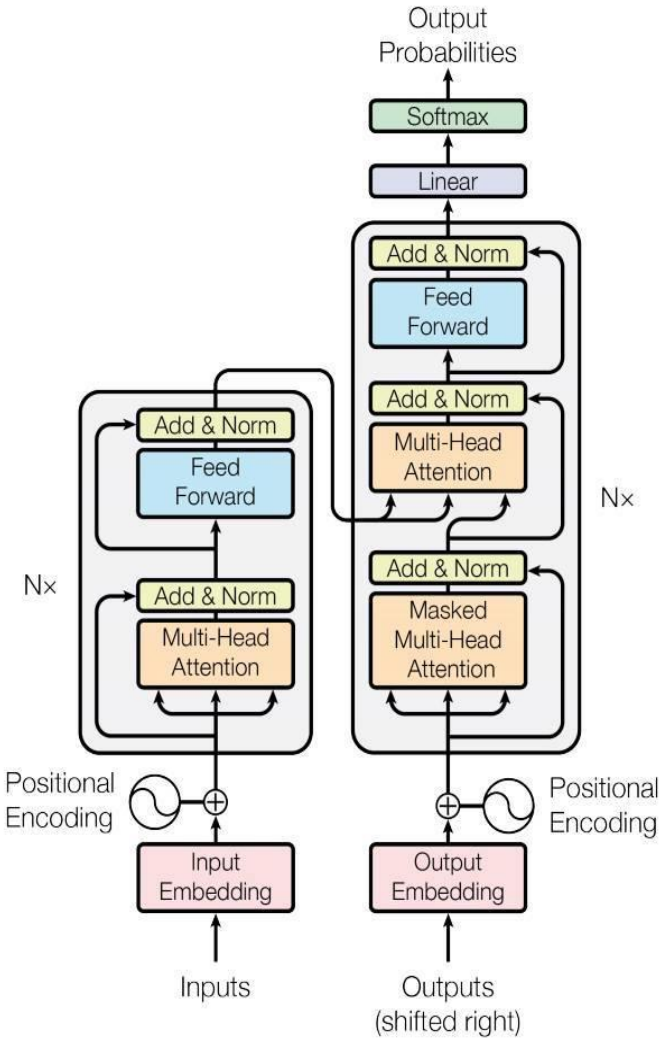
Multi-head attention

- **Multi-head attention**
  - Attention을 여러 번 진행하여 학습하는 방법
  - 문법, 의미 등 여러가지의 관계를 학습할 수 있을것이라 기대하고 학습



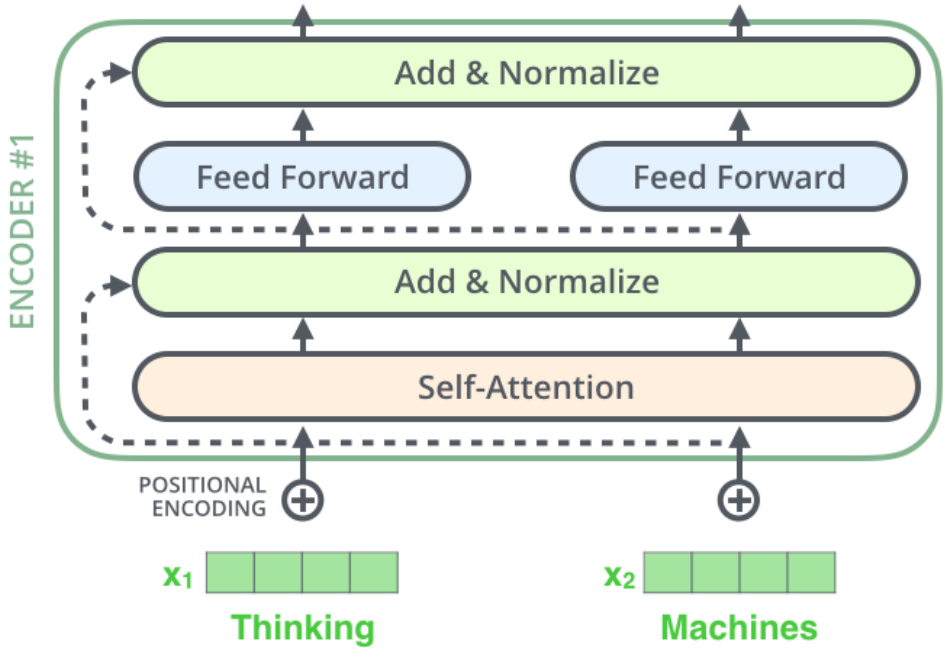
Transformer

- **Transformer**
  - Attention을 기반으로 문장을 학습하는 model
  - 기존의 RNN/LSTM의 순서정보에 중점을 맞춘 것이 아닌 단어간의 관계를 통해 문장을 학습하는 방법론
  - 크게 encoder와 decoder로 이루어져 있는 seq2seq 형태를 띄며 RNN대신 self-attention을 통해 학습한다.



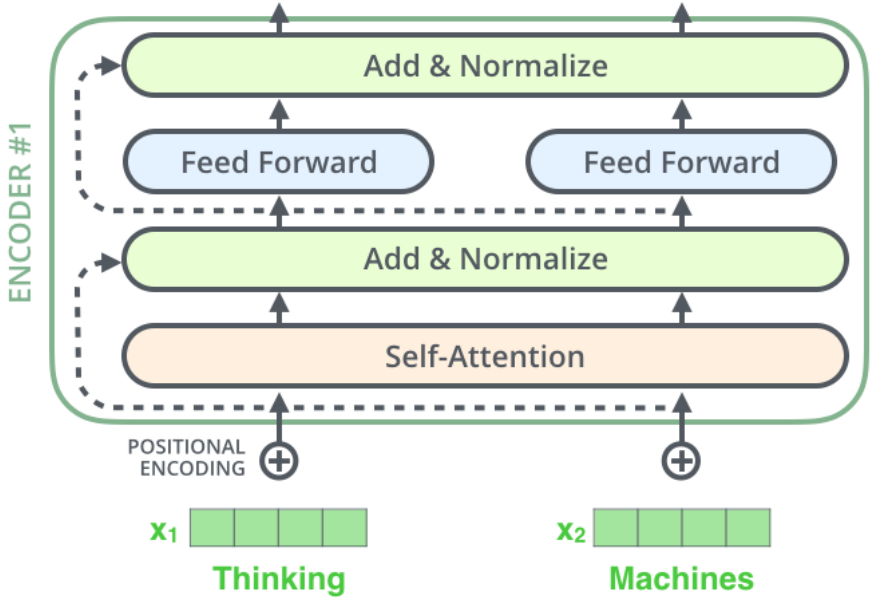
Transformer

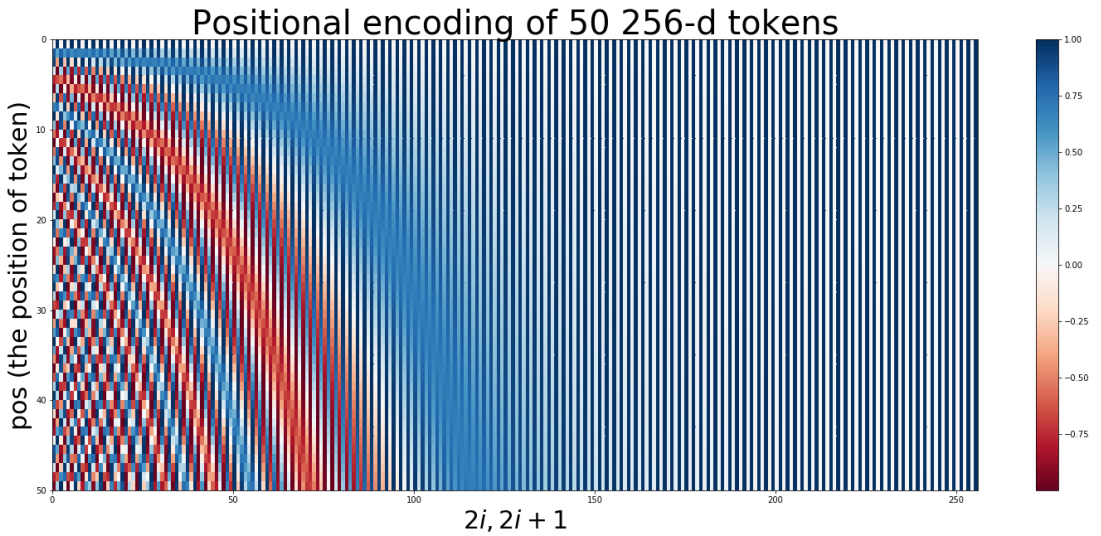
- **Transformer Encoder**
  - Self attention 과 2-layer Linear 로 구성된 network
  - 입력 문장의 모든 token을 self-attention을 통해 연산한다.
  - 이후 residual block과 MLP를 통해 최종 결과를 연산한다.



Transformer

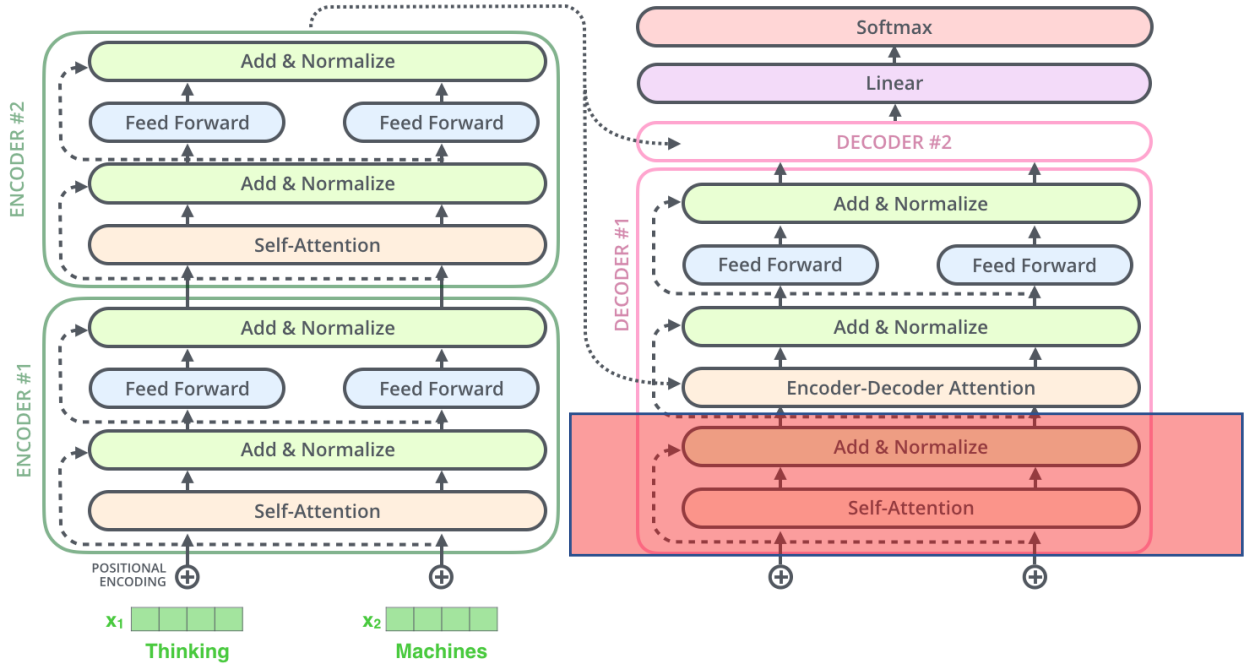
- **Transformer Encoder**
  - Positional Encoding을 통해 위치정보를 추가한다.
  - Positional Encoding이란 Sin과 Cos기반의 값을 통해 벡터를 구성하여 각 위치마다 다른 벡터를 구성
  - Positional Encoding을 word embedding에 더해서 위치정보를 입력함





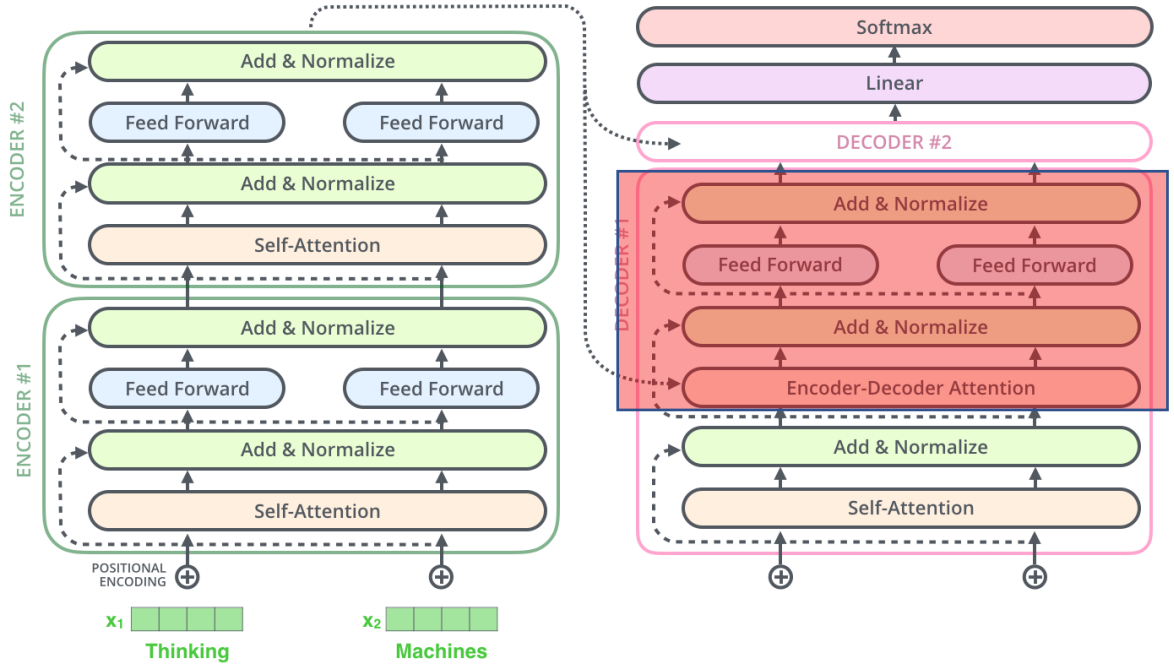
Transformer

- **Transformer Decoder**
  - Self-Attention을 통해 각 단어를 encoding한다.
  - Decoder는 auto-regressive하게 한단어씩 생성하기 때문에
  - 현재 time의 이전 단어까지만 self-attention을 한다.



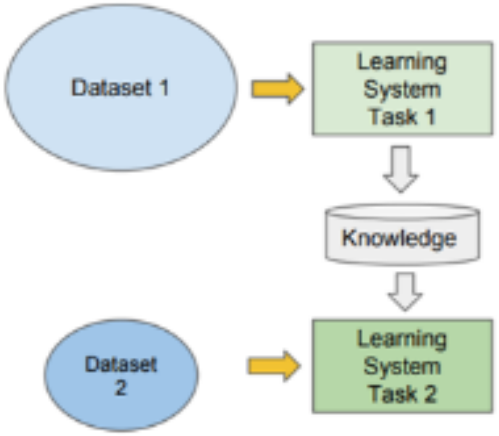
Transformer

- **Transformer Decoder**
  - Self-attention 이후 encoder-decoder attention을 통해 문장을 encoding한다.
  - Q는 decoder의 input이고 K,V는 encoder의 output으로 attention한다.
  - 이후 MLP를 통해 학습한다.



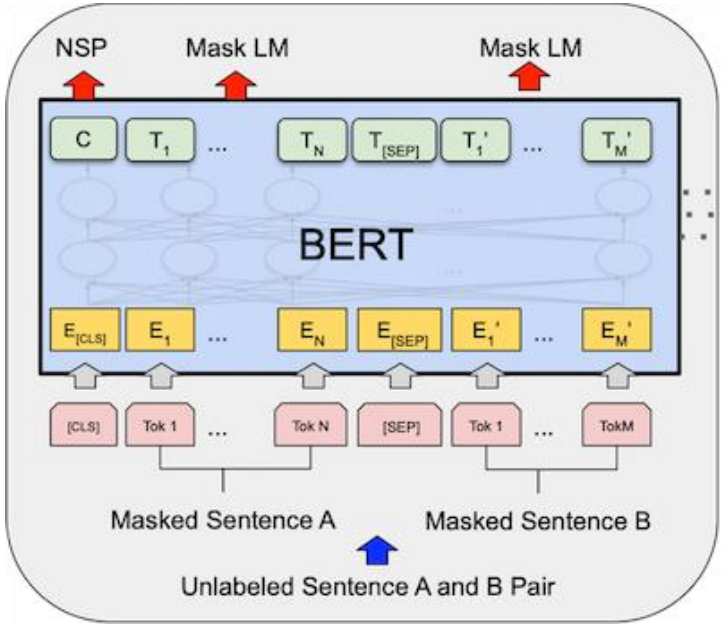
Transfer learning

- **Transfer learning**
  - 전이학습이라는 개념으로 매우 대량의 데이터를 통해 학습하면 다른 task에서도 성능향상을 보일 수 있다는 내용이다.
  - 예를들어 text classificatio에 대한 아주 많은 domain(병원, 공항, 호텔 등등)을 통해 학습하였을때 전혀 관계없는 QA나 chat-bot의 성능을 올릴 수 있다.
  - transfer learning은 대용량 데이터를 통해 학습하는 pre-train과 실제 목적 task를 학습하는 fine-tune으로 나뉜다.
  - 자연어처리에서는 이것을 self-supervised learning을 통해 학습함으로써 좋은 성능을 보였다.



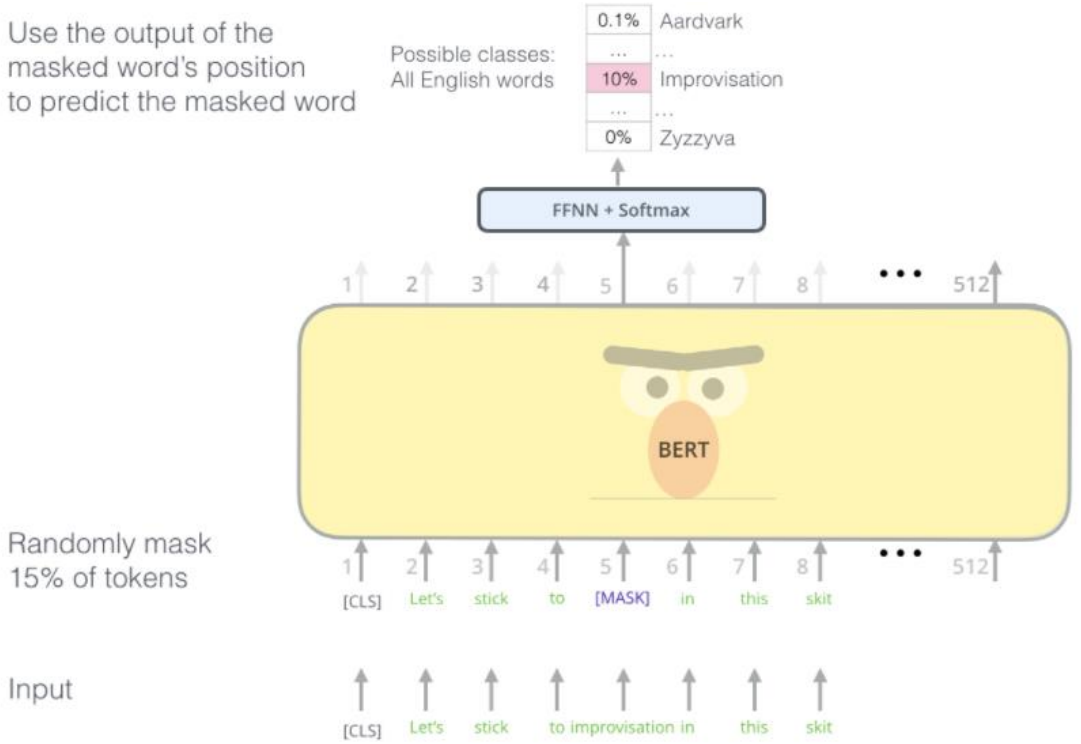
BERT

- **BERT**
  - Transformer encoder기반의 model이다.
  - 문장의 모든 token을 입력으로 사용하여 각 단어에 대해 학습된 vector로 변환한다.



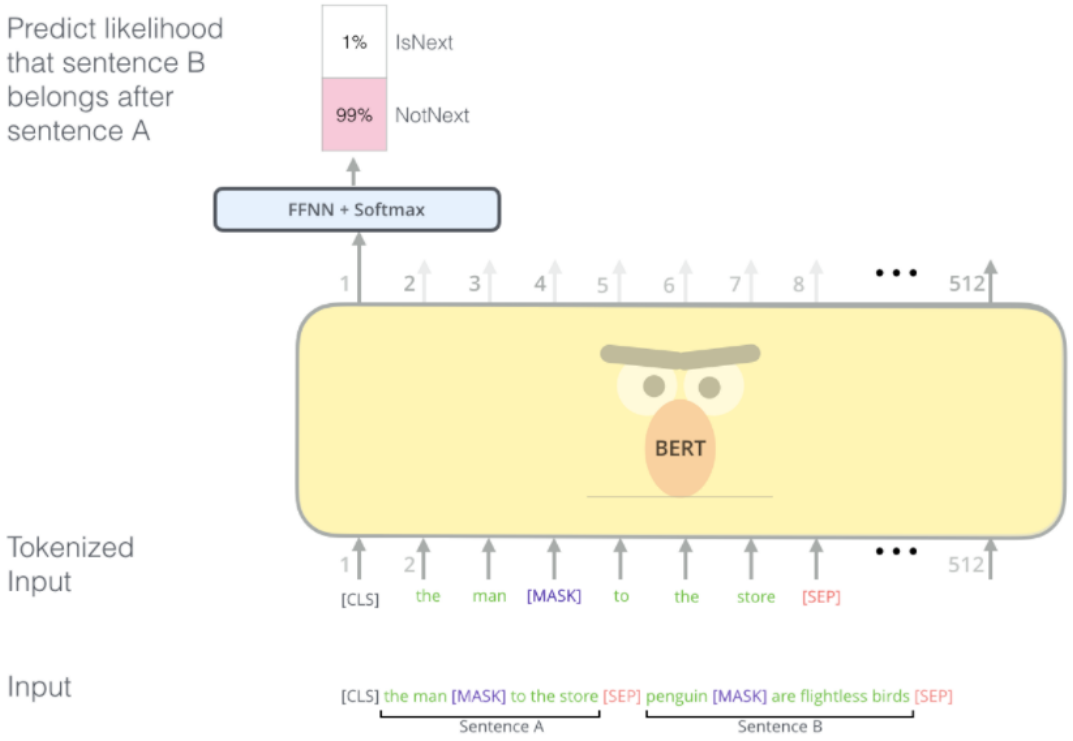
BERT

- **BERT pre training**
  - BERT는 masked language model로 masking된 token을 예측하며 각 단어의 의미를 학습한다.
  - 15%의 token을 랜덤하게 마스킹하고 BERT의 결과에서 해당 단어가 뭔지 예측한다.



BERT

- **BERT pre training**
  - 또한 Next Sentence Prediction를 통해 두문장이 이어지는 문장인지 판별하는 task를 통해 학습한다.
  - 이때 CLS token을 통해 학습하며 CLS는 전체문장에 대한 정보를 학습하기를 기대한다.

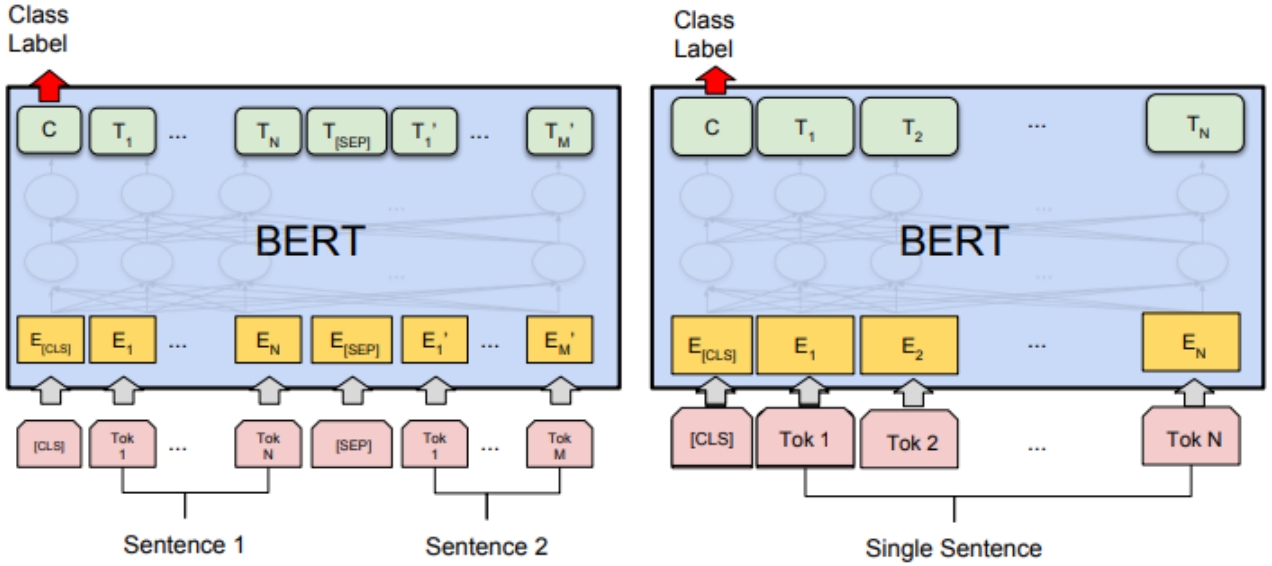


BERT

- **BERT pre training**
  - BERT의 pre-training방법은 대체로 정답을 따로 tagging하지 않고 문장내에서 정답을 구성하는 self-supervised learning을 사용한다.
  - 이를 통해 wiki와 같은 데이터를 대용량으로 crawling해와서 학습한다.
  - 일반적으로 300G정도의 데이터에 대해서 학습하여 pre-trainin한다.

BERT

- **BERT fine-tuning**
  - pre-train된 parameter들을 load해서 실제 목적 task를 진행한다.
  - 그리고 각 task를 위한 linear layer들을 추가해서 학습한다.
  - 예를 들어 문장 분류에서는 cls token에 MLP를 이용하여 각 label에 대한 확률을 구한다.

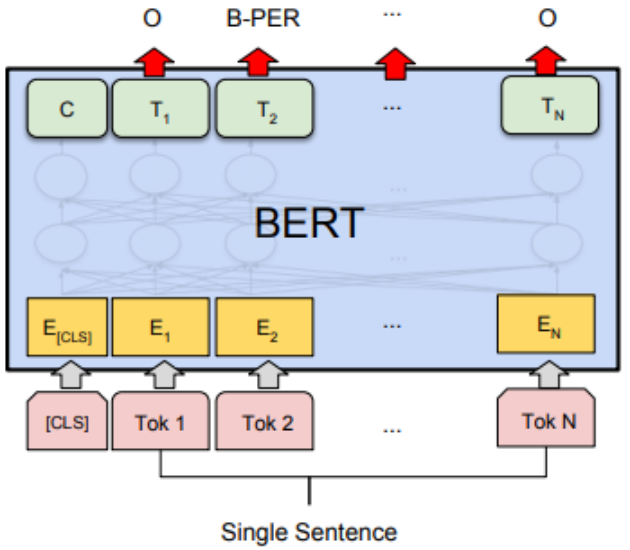


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

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

BERT

- **BERT fine-tuning**
  - pre-train된 parameter들을 load해서 실제 목적 task를 진행한다.
  - 그리고 각 task를 위한 linear layer들을 추가해서 학습한다.
  - NER같은 경우에는 각 token의 결과에 MLP를 이용하여 BIO tag class에 대한 확률을 구한다.



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

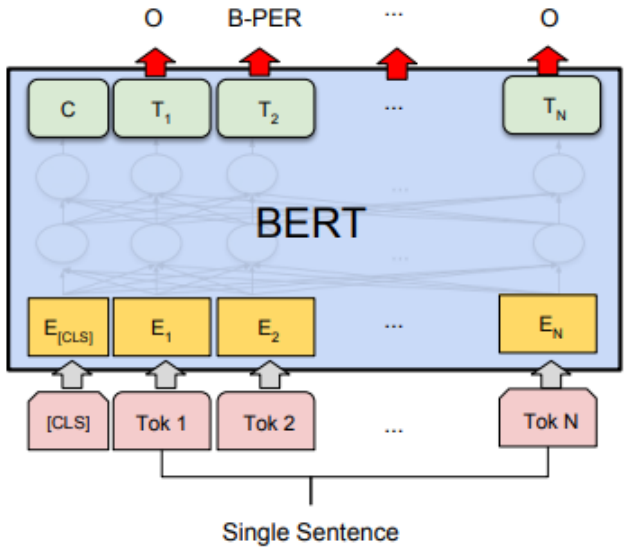
BERT

- **BERT fine-tuning**
  - pre-train된 parameter들을 load해서 실제 목적 task를 진행한다.
  - 그리고 각 task를 위한 linear layer들을 추가해서 학습한다.
  - QA는 질문이 들어왔을때 주어진 지문에서 해당 답을 추출하는 task이다.
  - QA 같은 경우에는 각 token에 대해 MLP를 이용하여 해당 토큰이 시작 token일 확률 end token일 확률을 구한다.

The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

Answer: through contact with Persian traders

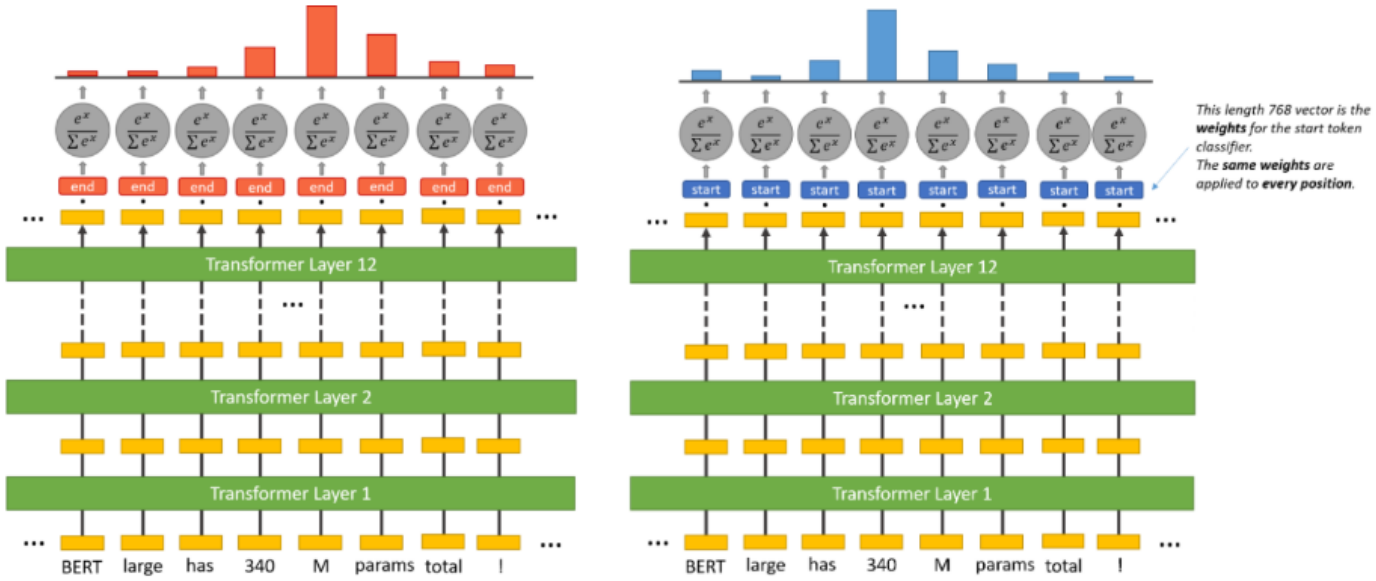


(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

BERT

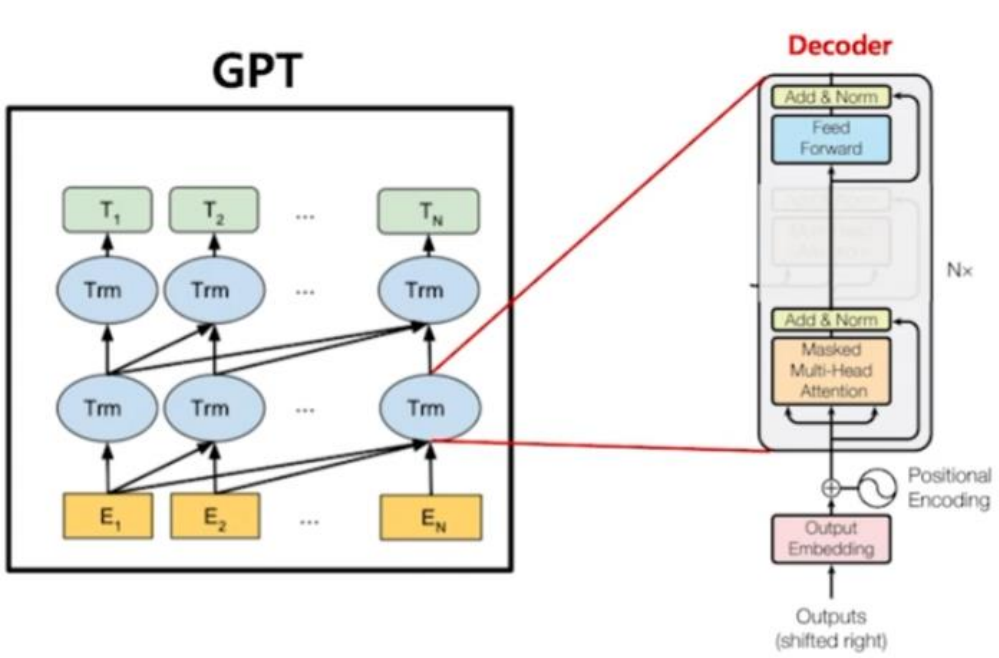
- **BERT fine-tuning**
  - pre-train된 parameter들을 load해서 실제 목적 task를 진행한다.
  - 그리고 각 task를 위한 linear layer들을 추가해서 학습한다.
  - QA는 질문이 들어왔을때 주어진 지문에서 해당 답을 추출하는 task이다.
  - QA 같은 경우에는 각 token에 대해 MLP를 이용하여 해당 토큰이 시작 token일 확률 end token일 확률을 구한다.

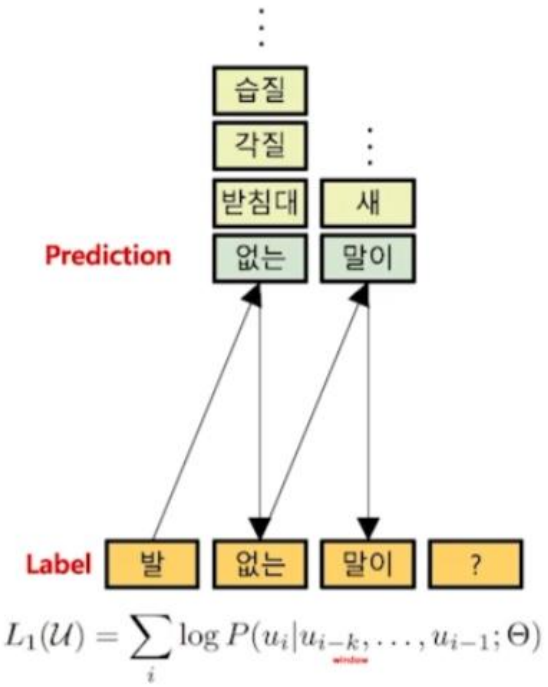
Paragraph: BERT-large is really big ... it has 24 layers and an embedding size of 1024 for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple of minutes to download.
Question: How many parameters does BERT-large have?
Answer: 340M parameters
Start token: 340
End token: parameters



GPT

- **GPT2**
  - Transformer Decoder로 구성된 model
  - Decoder로 구성되었기에 문장 생성에 뛰어나다.





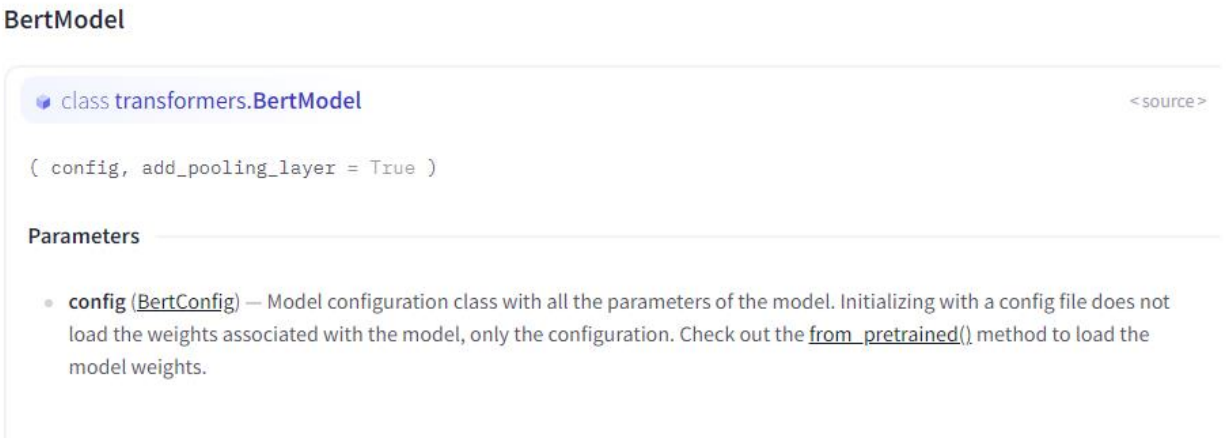
$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

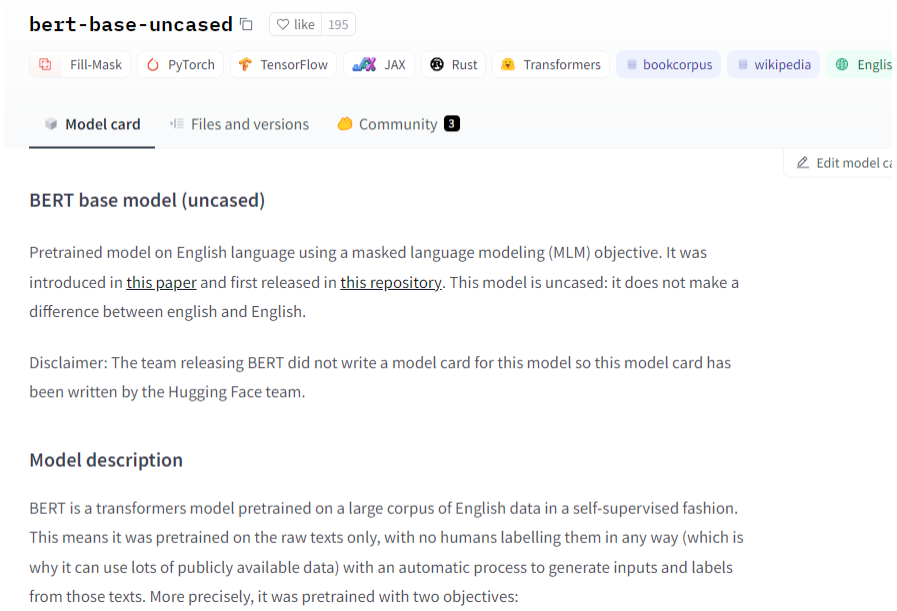
Transformer in pytorch

김균엽

huggingface

- **huggingface**
  - transformers기반의 모델들을 제공하는 python package
  - 또한 사전학습된 paramete들을 load할수 있는 hub역할도 한다.
  - www.huggingface.co
  - pip install transformers로 설치하고 import transformers로 import한다





huggingface BERT

- **transformers.BertTokenizer**
  - BERT의 형식에 맞는 값을 반환하는 tokenizer
  - BertTokenizer.from_pretrained(model url)을 통해 다른사람이 사전에 학습시켜놓은 tokenizer를 불러올 수 있다.
    - ex) BertTokenizer.from_pretrained("bert-base-uncased")
  - input으로는 한 개의 문장 또는 두개의 문장이 입력으로 들어간다.
  - output으로는 input_ids(tokenization과 indexing한 결과), token_type_ids(각 token이 몇번째 문장인지 입력하는 id), attention_mask(attention을 어디까지 진행할 것인지)가 반환된다.
    - ex) tokenizer("I am student") → input_ids = [101,123,1,32,102,103,103,103](101:cls 102:eos 103:pad)
      token_type_ids = [1,1,1,1,1,1,1,1]
      atttention_mask = [1,1,1,1,1,0,0,0]

huggingface BERT

- **transformers.BertModel**
  - BERT의 형식에 맞는 값을 반환하는 tokenizer
  - BertModel.from_pretrained(model url)을 통해 사전학습된 model을 불러올 수 있다.
    - ex) BertModel.from_pretrained("bert-base-uncased")
  - input으로는 input_ids, token_type_ids, attention_mask 등이 입력된다.
  - output으로는 last_hidden_state(모든 token에 대한 bert 결과값), pooler_output(cls token에 대한 결과값) 등이 있다.
  - https://huggingface.co/docs/transformers/v4.20.1/en/model_doc/bert#transformers.BertModel 참조

BERT with QA

- **QA**
  - SQuaD를 통해 QA학습
  - input_data = question과 paragraph에 대한 input_ids, token_type_ids, attention_mask
  - target = answer의 input_data에서의 start positio과 end position
  - output = 각 token이 start token일 확률, 각 token이 end token일 확률

S_2 : Pharmacists may also be **small-business proprietors**, owning the pharmacy in which they practice.

Question: What other role do many pharmacists play?

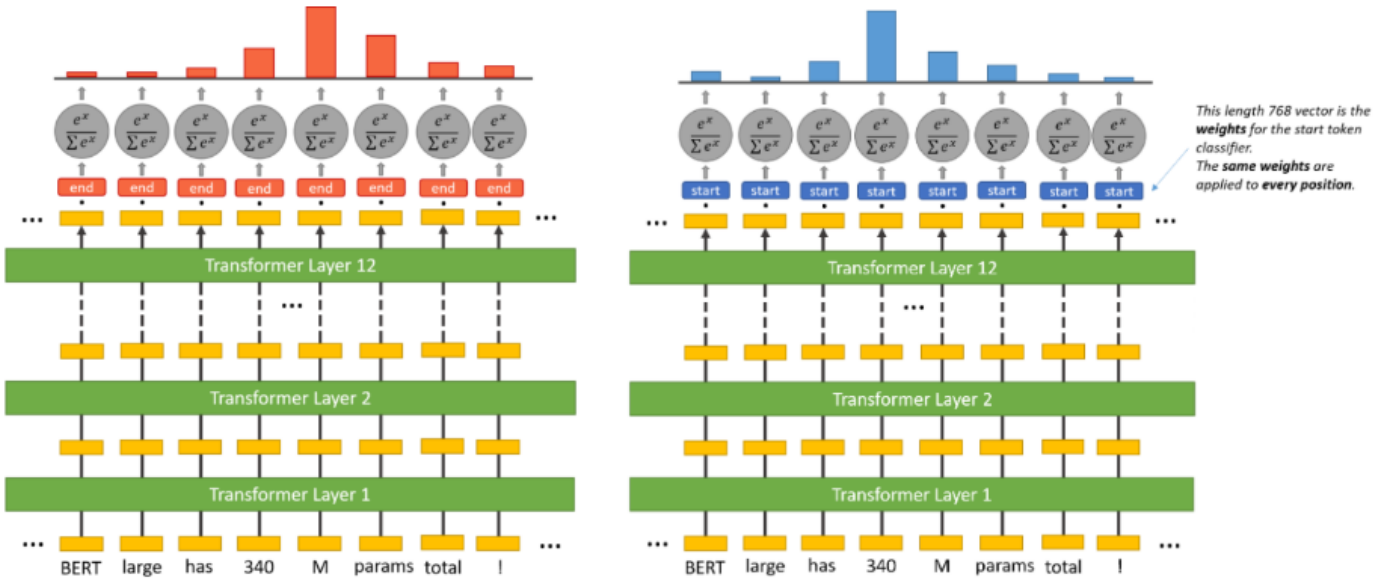
Answer: **small-business proprietors**

---

target: start: 5, end: 7

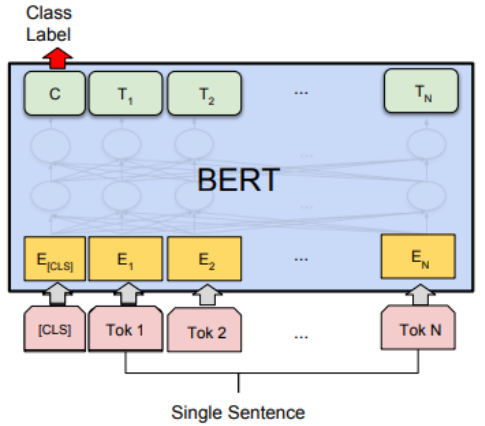
BERT with QA

- **QA**
  - SQuaD를 통해 QA학습
  - model 구성: BERT의 모든 token의 output인 last hidden state에 linear를 통해 1차원으로 변형한다. 결과로 나온 1차원 벡터의 값은 이 단어가 시작/끝 토큰일 확률을 나타낸다.
  - output = 각 token이 start token일 확률, 각 token이 end token일 확률



classification with BERT

- **classification**
  - spam dataset을 통해 QA학습
  - input_data = data 문장에 대한 input_ids, token_type_ids, attention_mask(tokenizer의 out)
  - target = 데이터가 spam일때는 1 ham일때는 0
  - model = bert의 output중 cls token에 대한 output인 pooler output에 linear를 적용하여 2차원 벡터로 변환, 변환된 벡터는 각각 spam일 확률 ham일 확률
  - output = spam일 확률 ham일 확률



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