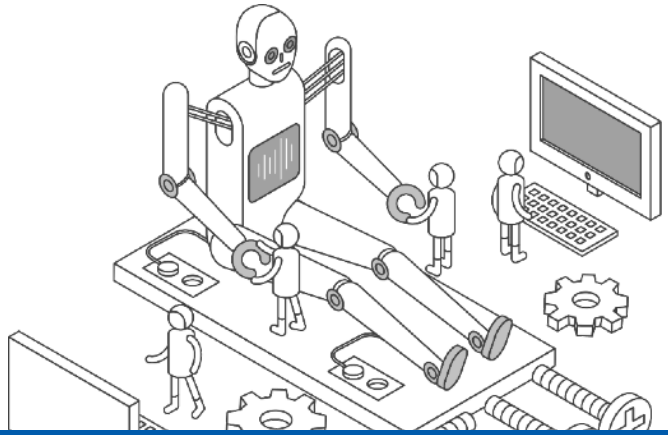
2022 디지털 전환을 위한 AI 전문가 과정

ARTIFICIAL INTELLIGENCE
BIG DATA
SMART FACTORY

AI·빅데이터 심화과정

"트랜스포머로 만드는 대화형 챗봇"



http://jalammar.github.io/illustrated-transformer/

1. Attention is all you need논문에서 Transformer 모델이 제안되었습니다.
2. Transformer는 RNN, LSTM과 같은 sequential model을 사용하지않고 attention 메카니즘만 이용하여 sequential data를 처리합니다.
3. RNN과 다르게 병렬적으로 작동하므로 트레이닝 시간이 매우 줄어들며, inference 성능 또한 매우 우수하여 딥러닝 발전에 매우 공헌하였습니다.

Attention Is All You Need

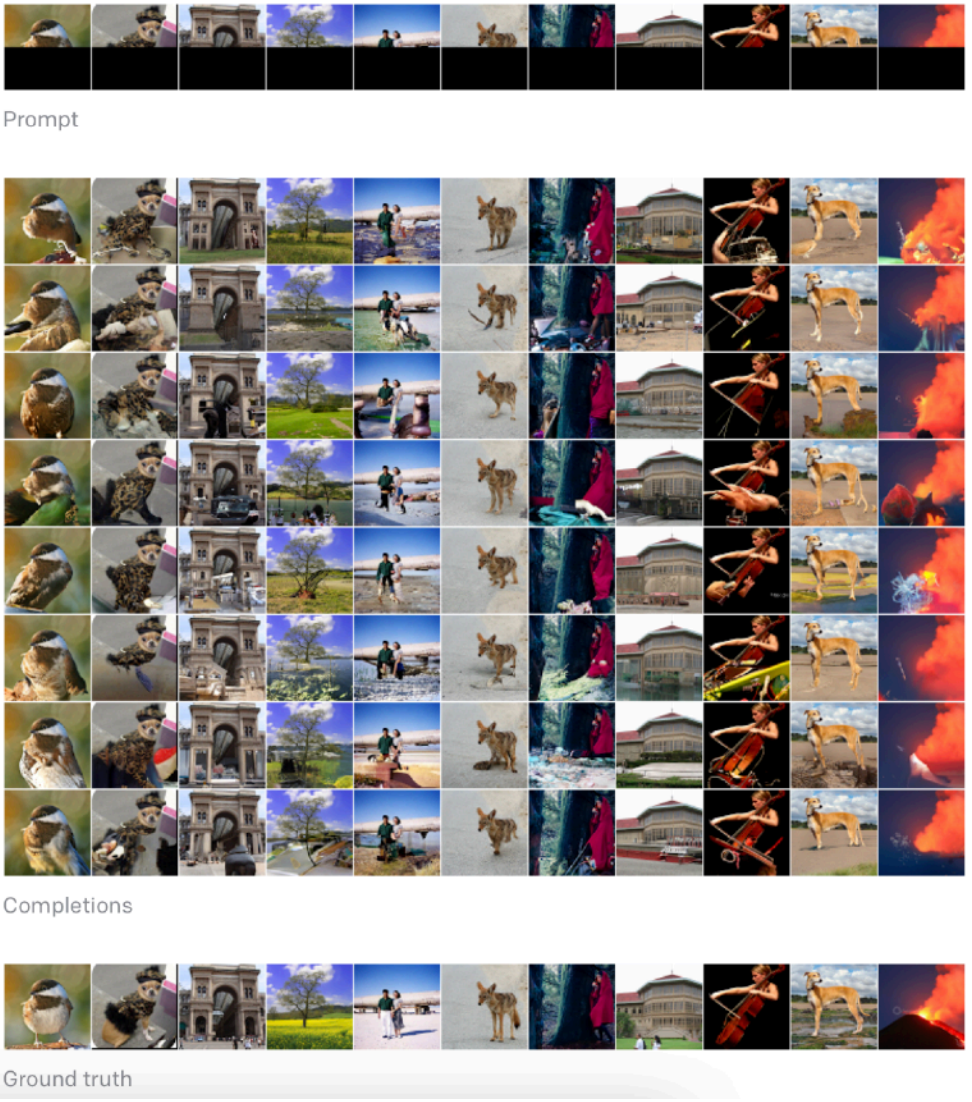
Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Sparse Transformer



https://openai.com/blog/sparse-transformer/

DALL·E

TEXT PROMPT

an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES

Edit prompt or view more images↓

TEXT PROMPT

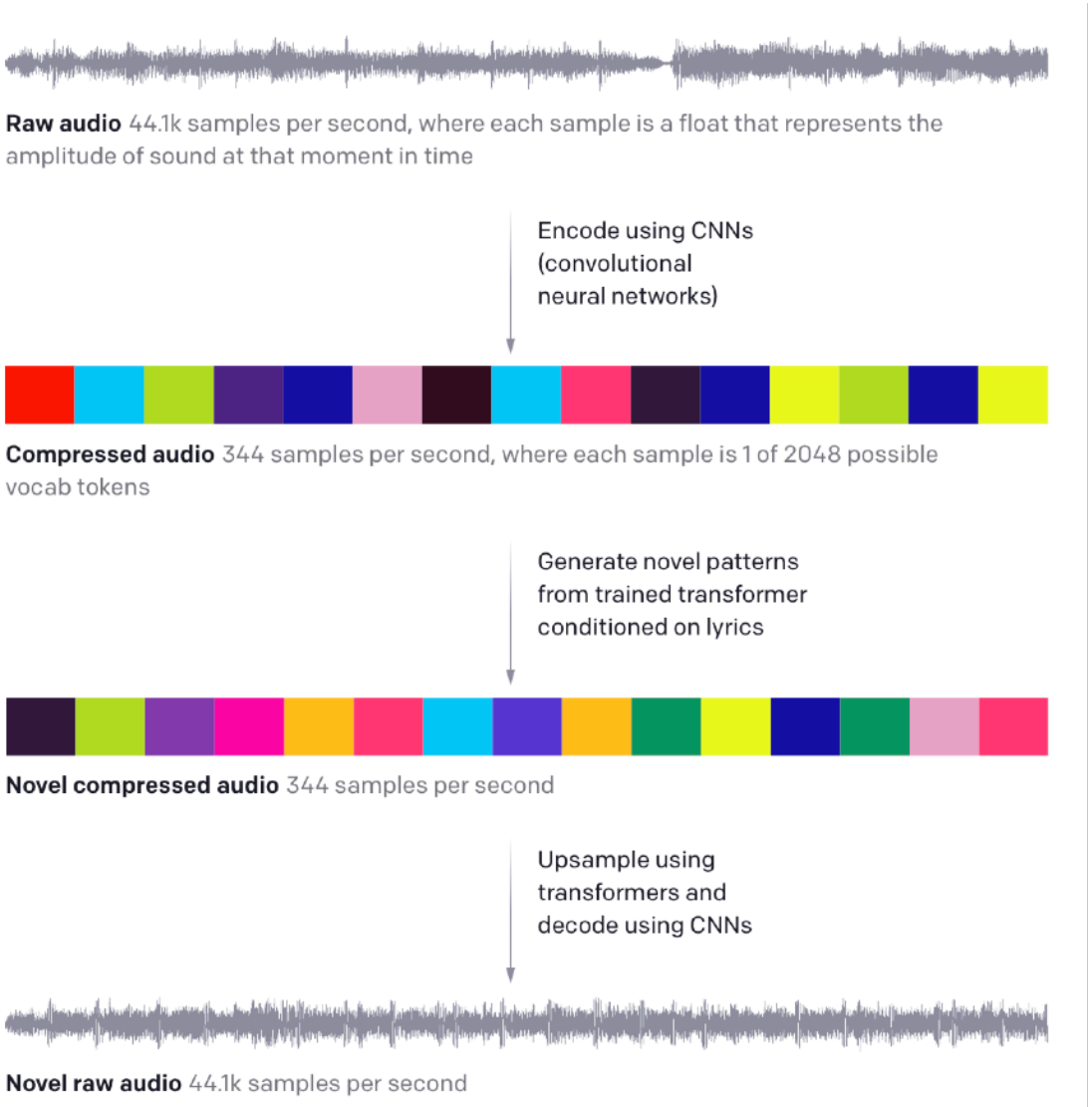
a store front that has the word 'openai' written on it. . . .

AI-GENERATED IMAGES

Edit prompt or view more images↓

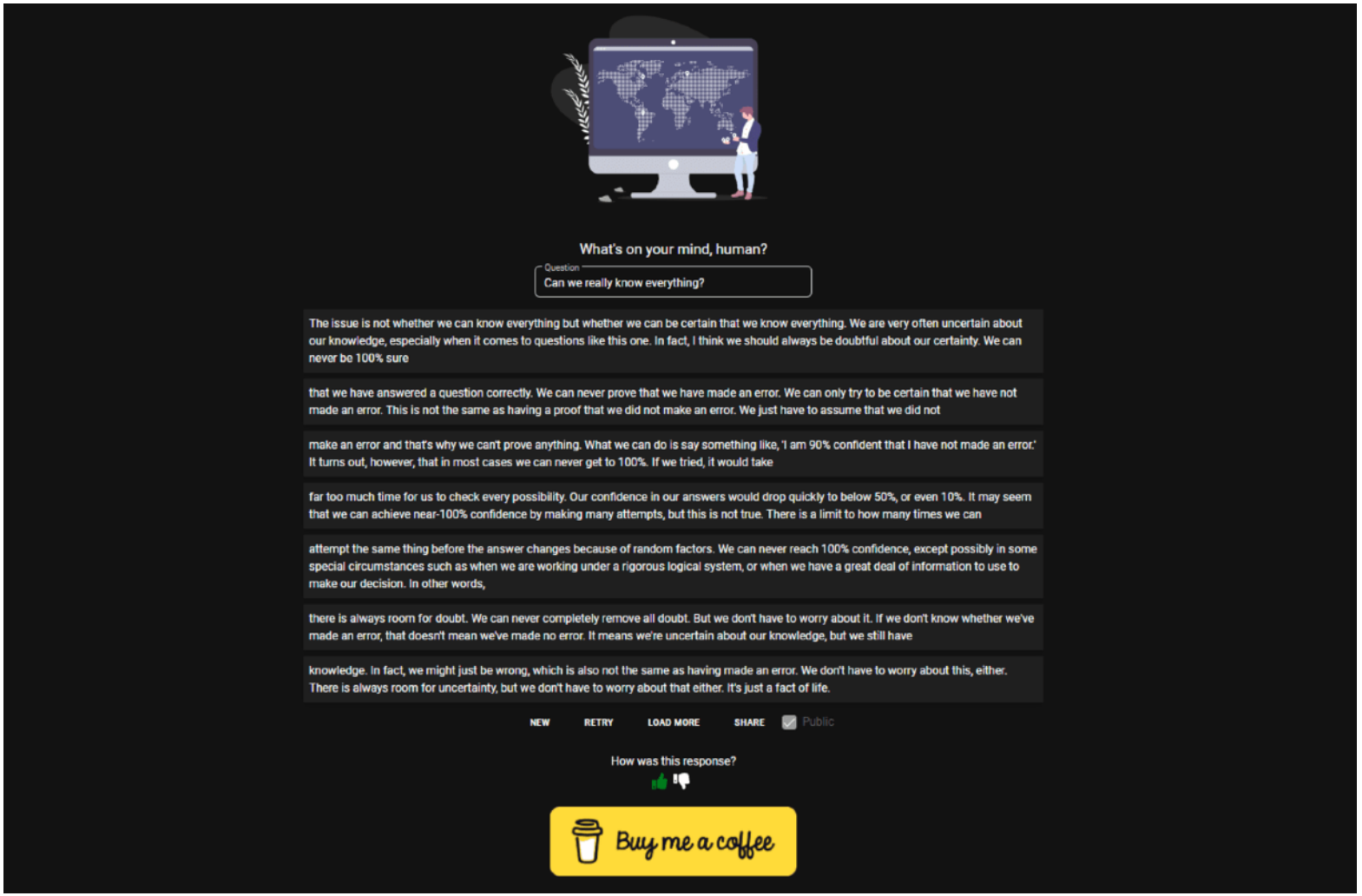
https://openai.com/blog/dall-e/

Jukebox



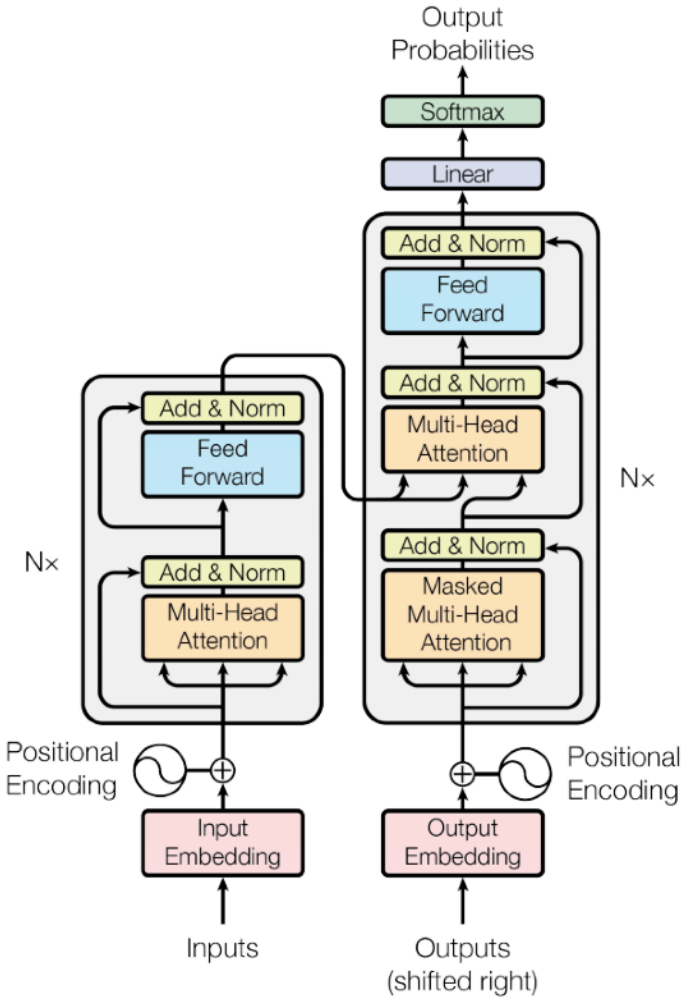
https://openai.com/blog/jukebox/

Philosopher AI



https://philosopherai.com

Overall Architecture



Transformer는 크게 Inputs 데이터를 인코딩하는 Encoder와, 인코딩된 데이터를 기반으로 Outputs을 디코딩하여 Outputs의 확률분포 파라메터를 출력하는 Decoder로 구성된다.

이러한 구조는 seq2seq 모델이라고 불리며 원래 언어의 문장을 대상 언어의 문장으로 변환시키는 번역 작업이나, 질의 문장을 토대로 응답 문장을 생성해내는 질의 응답 작업 등에 사용되었다.

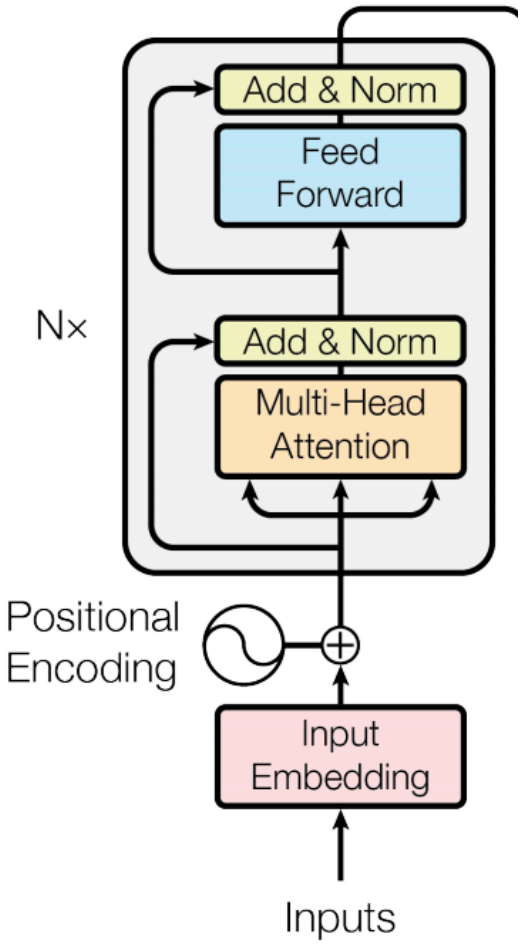
Encoder는 원래 언어의 문장의 토큰을 입력으로 받아 임베딩 테이블을 이용해 벡터로 변환하고 여러 인코딩 블록을 거쳐 최종적으로 Decoder에 들어갈 형태로 출력한다.

Decoder도 마찬가지로 대상 언어의 문장의 토큰을 입력으로 받아 임베딩 테이블을 이용해 벡터로 변환하고 여러 디코딩 블록을 거쳐 다음 토큰에 해당하는 확률 분포의 파라메터를 출력한다.

이 때, 디코딩 블록 내에서 어텐션 메카니즘을 통해 Encoder에서 인코딩된 정보를 가져온다.

어텐션 시 여러 헤드로 분리하여 각각 다양한 부분의 정보를 가져올 수 있도록 하며, 가져온 정보들은 concatenation하여 다음 레이어로 전해진다.

Encoder



Encoder는 Inputs, Input Embedding, Positional Encoding, Attention Blocks으로 구성되며, Attention Blocks내부는 Multi-Head Attention, Feed forward와 Layer Normalization 으로 구성된다.

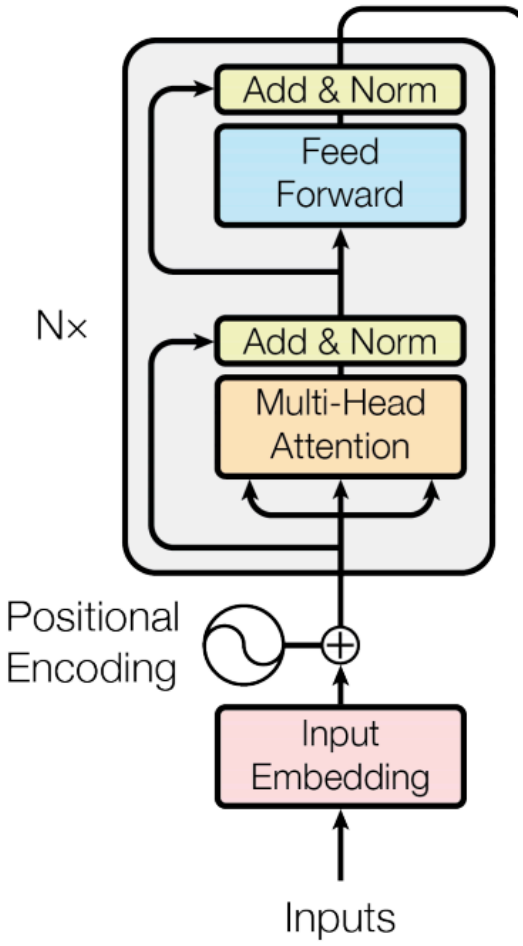
Inputs : 번역 작업의 경우 원본 문장, 질의 응답 작업의 경우 질의에 해당하는 입력 문자열에 해당한다. 문자열에 대응하는 토큰이 입력으로 주어진다.

Input Embedding : 임베딩 테이블을 통해 각 토큰을 임베딩 벡터로 변환한다. 임베딩 테이블은 그래디언트를 받아 트레이닝 가능한 상태로 둔다.

Positional Encoding : Attention 레이어는 CNN, RNN과 다르게 위치에 대한 정보가 출력에 반영되지 않으므로 별도로 위치 정보를 임베딩하는 것이 필요하다. 위치 값을 아래와 같은 sin과 cos함수에 적용한 출력 값을 Input Embedding 결과에 더한다.

$$PE_{(pos,2i)} = \sin\left(\text{pos}/10000^{2i/d_{\text{novel}}}\right)$$
$$PE_{(\text{pos},2i+1)} = \cos\left(\text{pos}/10000^{2i/d_{\text{nodel}}}\right)$$

Encoder



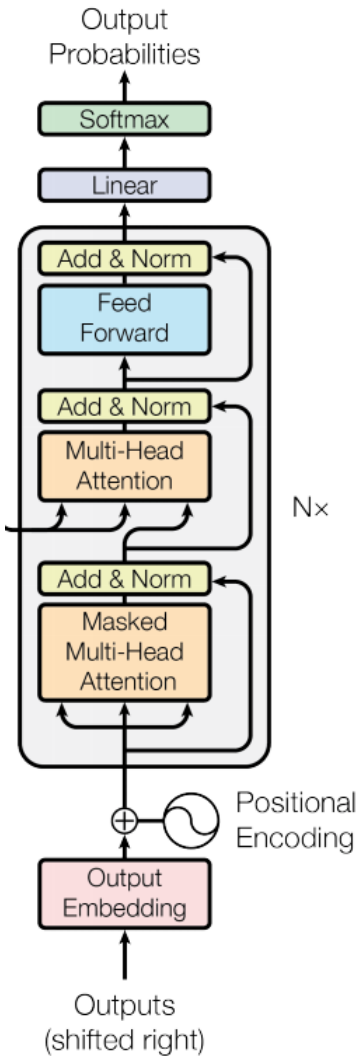
Multi-Head Attention : 입력으로 들어온 벡터를 여러 Head로 나누어 각각 다른 어텐션 정보를 계산한다. 이러한 작업으로 어텐션 영역 내에서 다양한 정보를 가져다 사용할 수 있는 능력을 부여한다.

Add & Norm : Multi-Head Attention의 결과와 입력을 더한다. 이러한 Residual Connection을 통해 깊은 네트워크를 구축하면서도 Backpropagation시 Gradient가 잘 전파되도록 돕는다. 또한 Layer Normalization을 사용하여 Add 연산이 반복됨에도 안정된 값을 출력하도록 한다.

Feed Forward : 두 개의 Fully Connected 레이어와 ReLU Activation으로 구성된다. 내부 레이어의 차원은 입력보다 4배로 크게 설정된다.

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

Decoder



Decoder는 Outputs, Outputs Embedding, Positional Encoding, Attention Blocks으로 구성되며, Attention Blocks내부는 Masked Multi-Head Attention, Feed Forward와 Layer Normalization 으로 구성된다.

Outputs : 번역 작업의 경우 대상 문장, 질의 응답 작업의 경우 응답문장에 해당한다. 문자열에 대응하는 토큰이 입력으로 주어진다.

Outputs Embedding : Encoder의 Inputs Embedding과 같은 방식으로 작동한다.

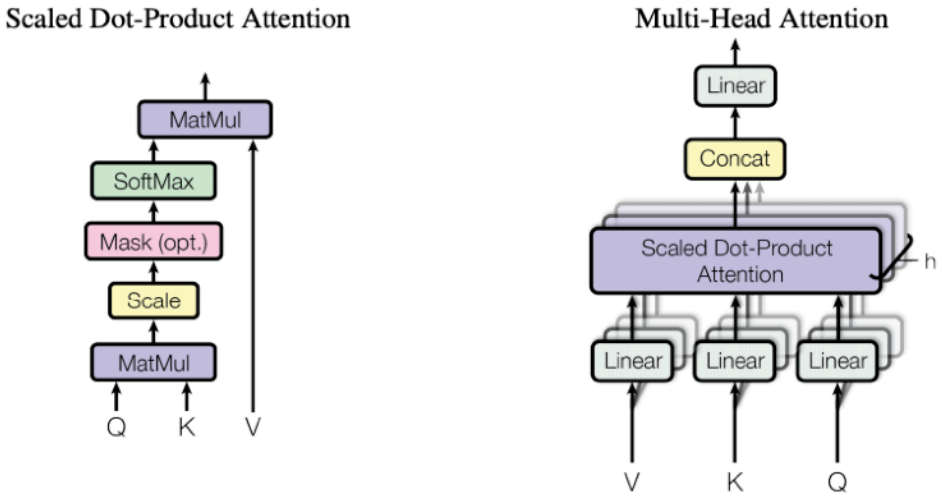
Positional Encoding : Encoder의 Positional Encoding과 같은 방식으로 작동한다.

Masked Multi-Head Attention : Encoder의 Multi-Head Attention과 같지만 현재보다 과거의 정보들만을 참조하도록 Mask를 사용한다.

Add & Norm : Encoder의 Add & Norm과 같은 방식으로 작동한다.

Feed Forward : Encoder의 Feed Forward과 같은 방식으로 작동한다.

Multi-Head Attention



Multi-Head Attention은 입력 벡터들을 선형 변환하여 Value, Key, Query 값을 얻고, 각 Query와 Key 값들 간에 dot-product를 취하여 얼마나 정보를 취할지 결정하는 Attention Score를 얻은 뒤, 이 값을 토대로 Value 값들을 선형 결합하여 최종적인 출력 벡터를 얻어낸다. 이 때 입력 벡터를 각 Head로 분리하고 출력 벡터를 concatenation을 통해 다시 합쳐 다양한 어텐션 가능성을 얻을 수 있도록 한다.

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

Query와 Key의 내적을 각 Head의 차원 d_k의 루트 값으로 나누어줌으로써 Head의 차원에 관계없이 내적값이 안정되도록 도와준다.


똑똑
파이팅!!