# [논문 리뷰] Attention is all you need

2021. 9. 9. 23:05

#attention #self-attention #transformer #RNN #seq2seq #CNN #multi-head attention #encoder #decoder #positional encoding #scaled dot product attention

[제목] Attention is all you need
[학회] NIPS
[년도] 2018년
[주저자] Google Researcher

오늘은 Attention is all you need 논문에 나오는 attention에 대한 개념과 논문의 핵심인 transformer 모델에 대해 알아보려고 합니다. Attention이란 무엇이고 trasnformer라는 모델은 왜 나오게 되었을까요? 설명에 앞서 Harper님의 블로그를 많이 참조하였음을 밝힙니다.

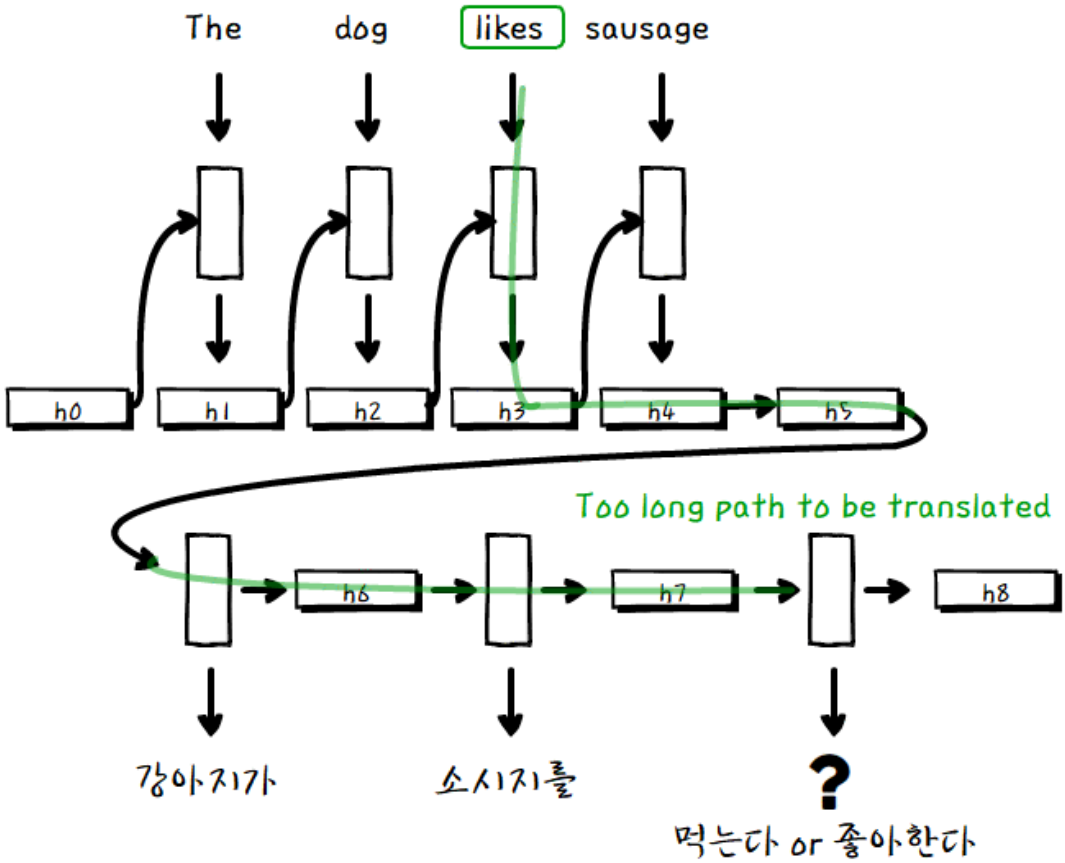
## Attention 메커니즘의 도입 배경

먼저 Attention이라고 하는 개념은 쉽게 말해 조금 더 집중해서 보겠다라는 의미입니다. 집중해서 본다는 것은 당연 그만큼 중요하단 의미입니다. 그렇다면 attention은 무엇에 더 집중한다는 것일까요? attention은 주로 자연어 처리에 사용되는데, attention은 해당 시점에서 예측해야할 단어와 관련있는 입력 단어(input word)를 중점적으로 본다고 할 수 있습니다.

그렇다면 이 attention이라고 하는 개념이 왜 나오게 되었을까요? 바로 기존의 seq2seq 모델의 단점을 극복하고자 나오게 되었습니다. seq2seq는 RNN 계열의 모델이며 2개의 RNN을 연결하여 하나의 시계열 데이터를 다른 시계열 데이터로 변환하는 것입니다. seq2seq 모델에 대한 구체적인 내용은 2014년 12월에 발표된 Sequence to Sequence Learning with Neural Networks 논문을 참조하면 됩니다. 이러한 seq2seq 모델에는 크게 4가지 단점이 있고 아래와 같습니다.

1. 상당한 계산 복잡도를 가진다.
2. RNN 모델에서 병렬처리가 불가능하다.
3. long-range dependency를 가지는 단어를 참조할 때 거리가 멀어 참조가 잘되지 않는다.
4. 구조적으로 고정 크기의 벡터에 모든 정보를 압축하다보니 발생하는 정보손실 문제를 가진다.

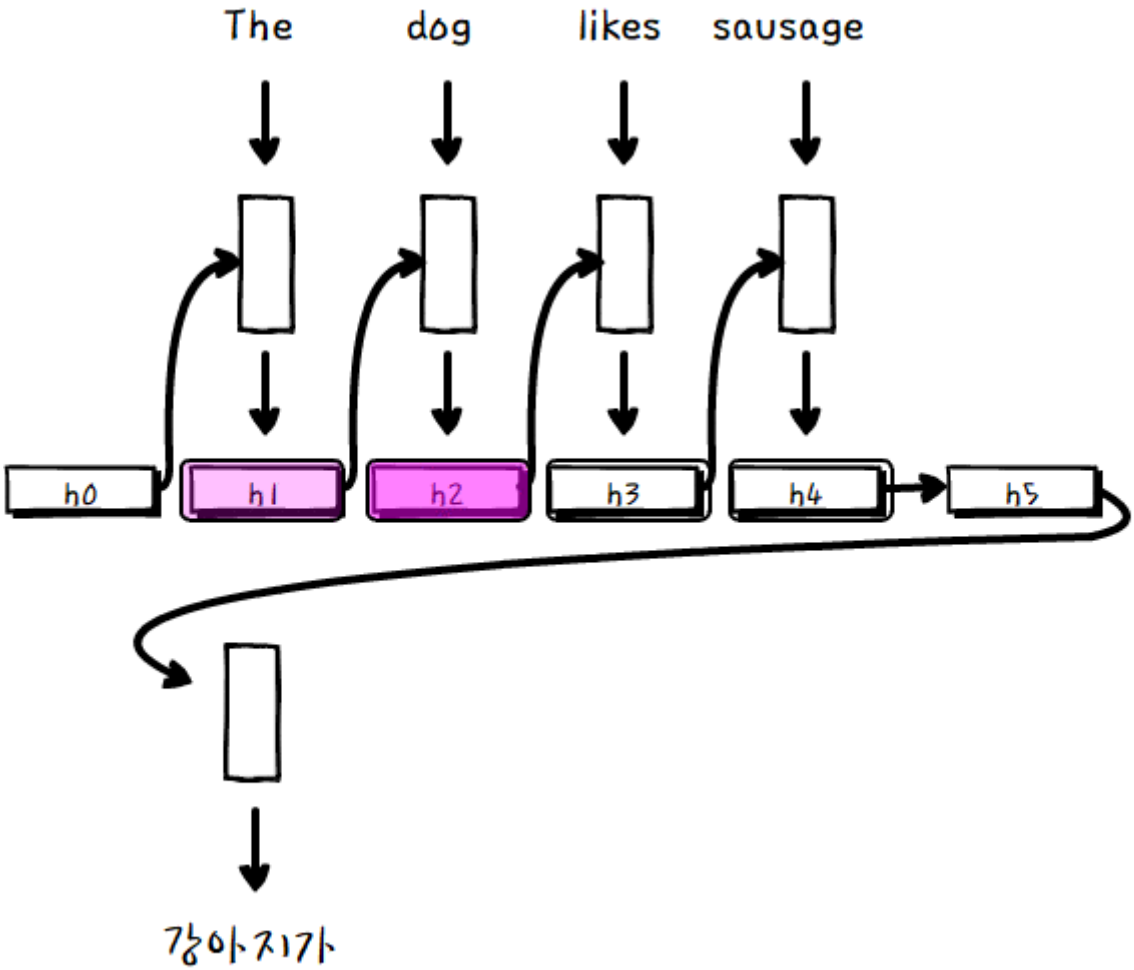
이러한 RNN 모델의 단점을 잘 설명하는 그림은 아래와 같습니다.




[Figure 1] RNN을 이용한 번역 (영어 --> 한글)

참고로 RNN의 경우 NMT(Neural Machine Translation)에서 사용되며 위 그림은 영어를 한글로 번역하는 것을 나타내었습니다. 강아지가 소시지를 먹는다 또는 좋아한다를 예측하기 위해 likes를 참조합니다. 하지만 likes라는 단어는 여러번의 hidden layer를 거치면서 그 의미가 희석되었을 확률이 높습니다.

따라서 이러한 RNN의 단점을 극복하기 위해 나온 개념이 아래와 같이 attention이 추가된 RNN 입니다.

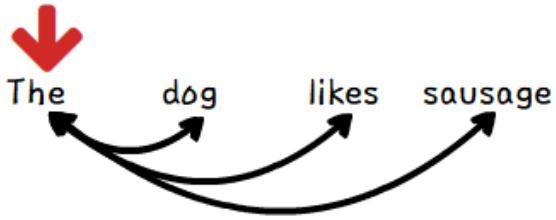



[Figure 2] RNN with attention model

attention은 Input word에 해당하는 hidden layer와 출력해야 할 단어와의 연관성이 있는 단어에 가중치를 주어 output word를 출력하는 것입니다. 하지만 이 또한 병렬 처리가 불가능하다는 근본적인 RNN의 단점과 attention 계산까지 더해져 계산복잡도가 높아진다는 단점이 존재합니다.

## Transformer 모델 도입 배경

Transformer 모델은 기존의 RNN 모델의 단점인 long-range dependency 문제를 극복하고자 제시 되었던 RNN with attention 모델의 단점을 극복하고자 나오게 되었습니다. 단점은 위 설명과 동일하게 병렬 처리 불가능과 attention 계산으로 인한 계산 복잡도가 높다는 것입니다. Transformer 모델의 아이디어는 attention을 통해 참조해야할 word의 위치(position) 정보를 얻음으로서 RNN with attention 모델의 두 단점을 극복하고자 하였습니다.




[Figure 3] self-attention machnism

기존의 output word를 예측할 때만 attention을 사용했던 기존 모델과 달리 Transformer는 sequence 내의 단어 관계 정보를 self-attention을 사용하여 미리 계산해둡니다. 이를 통해 Transformer는 다음 단어를 효율적으로 예측하게 됩니다. 여기서 효율적이라는 의미는 RNN with attention 모델의 단점이었던 계산 복잡도를 줄였다고 볼 수 있습니다.

## Trasnformer 모델 아키텍처

self-attention 메커니즘이 들어간 transformer의 핵심 아키텍처는 아래와 같습니다.

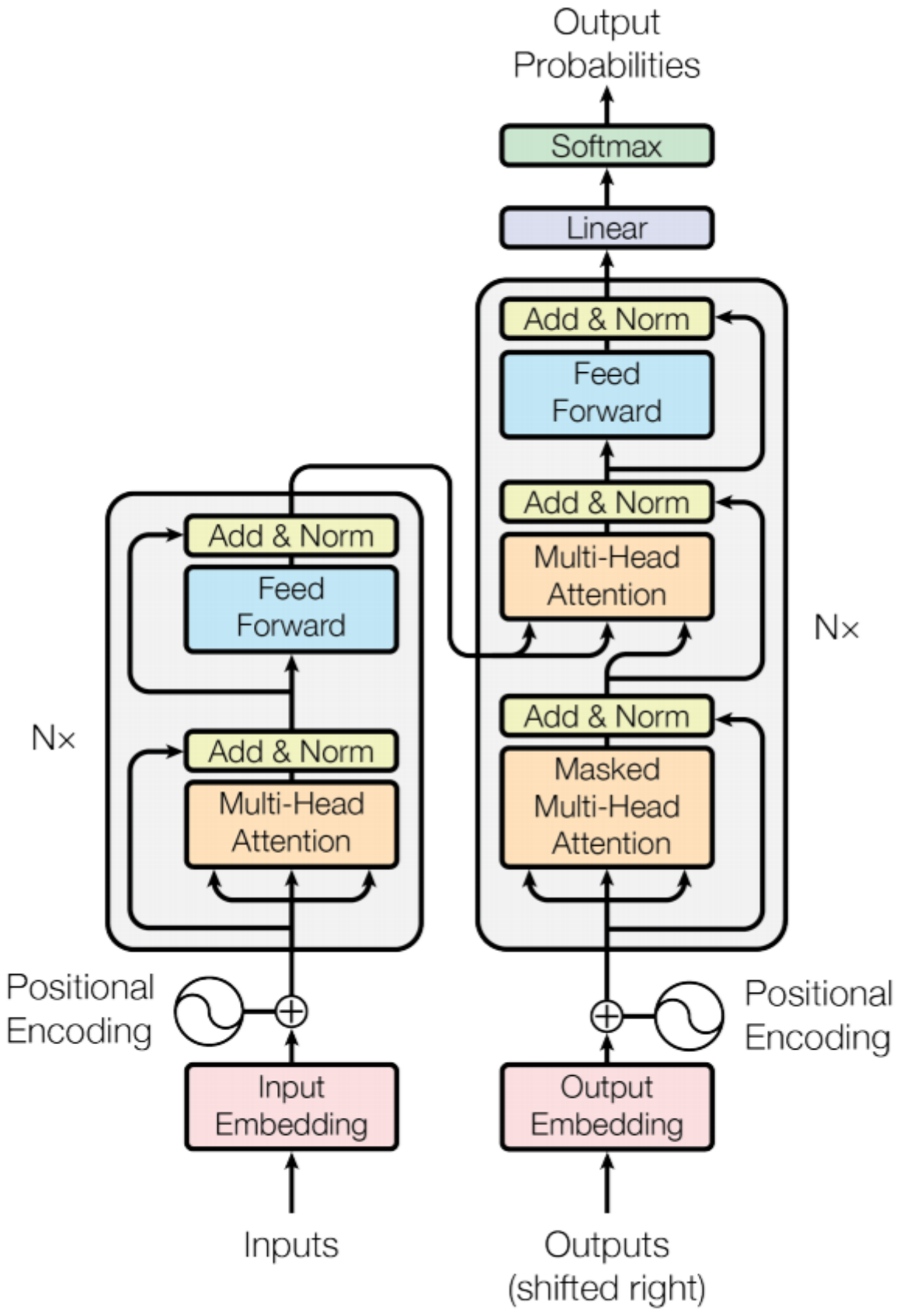
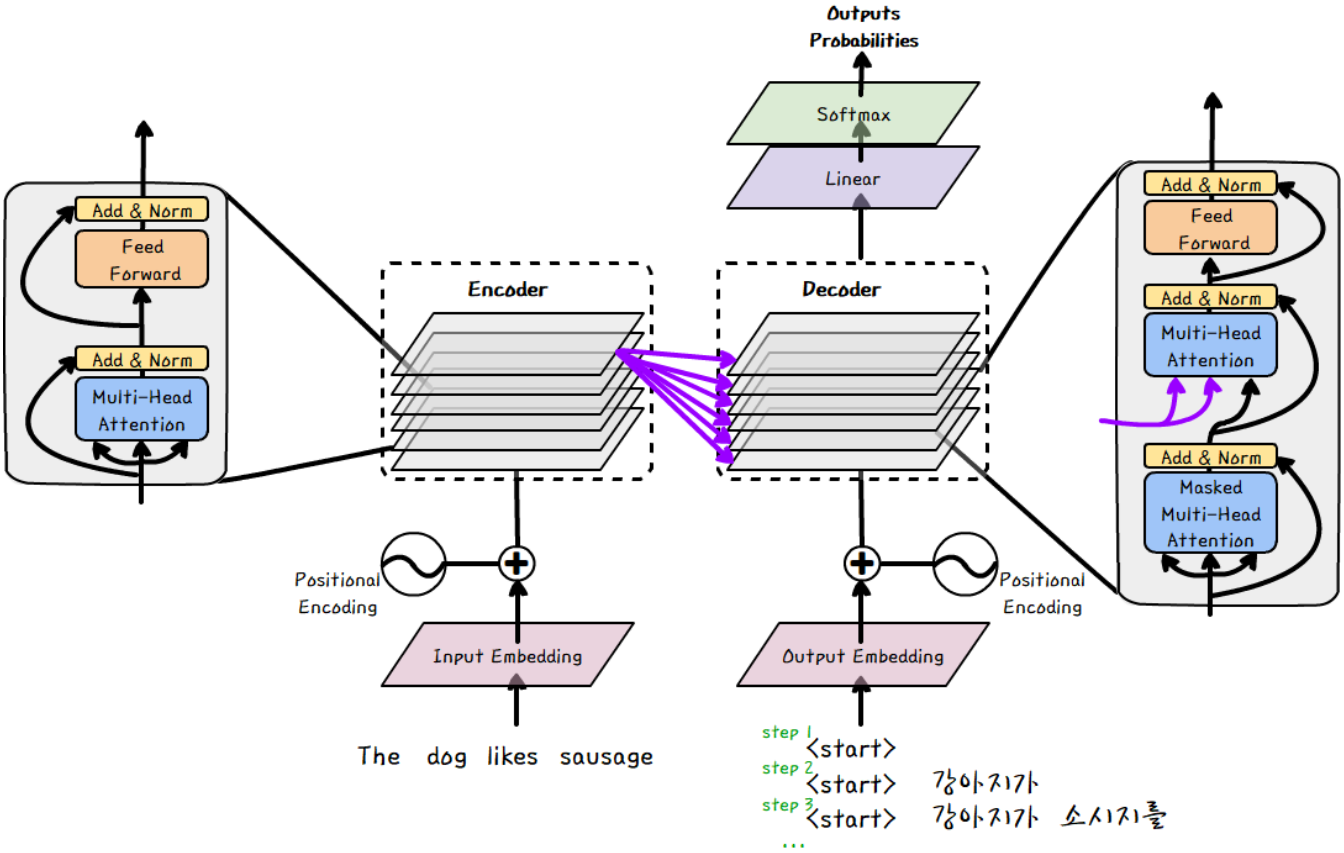



Figure 1: The Transformer - model architecture.

[Figure 4] Transformer model architecture

크게 Encoder와 Decoder로 나뉘어져 있습니다. Encoder와 Decoder의 경우 둘 모두 크게 3가지로 Positional Encodin
Multi-Head Attention과 Feed Forward NN이 사용되는 것을 확인할 수 있습니다. 논문에서는 Encoder와 Decoder의

N 값을 6으로 주어 Transformer 모델을 구성했다고 합니다. 이를 입체적으로 나타내면 아래와 같습니다.




[Figure 5] Transformer model architecture

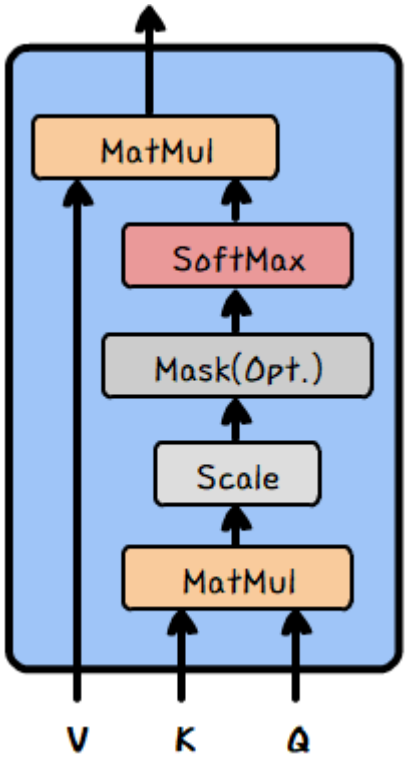
전반적으로 Abstract한 과정은 먼저 Encoder와 Decoder에 sequence가 입력이 됩니다. 이후 positional encoding을 통해 Encoder/Decoder로 들어갑니다. Encoder의 출력을 보면 모든 Decoder의 Layer에 영향을 미치는 것을 확인할 수 있습니다. 이후 Decoder는 Linear layer와 Softmax layer를 거쳐서 단어의 확률 값을 나타낸다고 볼 수 있습니다.

## Position Encoding

먼저 Positional Encoding은 단어의 위치 정보를 나타내기 위한 방법입니다. position을 나타내주는 벡터를 각각의 단어에 더해주는 방식으로 사용됩니다. 결론적으로 이러한 posotion encoding으로 각 단어의 상대적인 위치 정보를 포함하도록 합니다. 이러한 position encoding이 가지는 의미는, 기존의 임베딩과 같은 차원의 position encoding을 만들어 더해줌으로써 time signal을 가진 embedding을 인풋으로 받을 수 있게 된다는 것입니다.

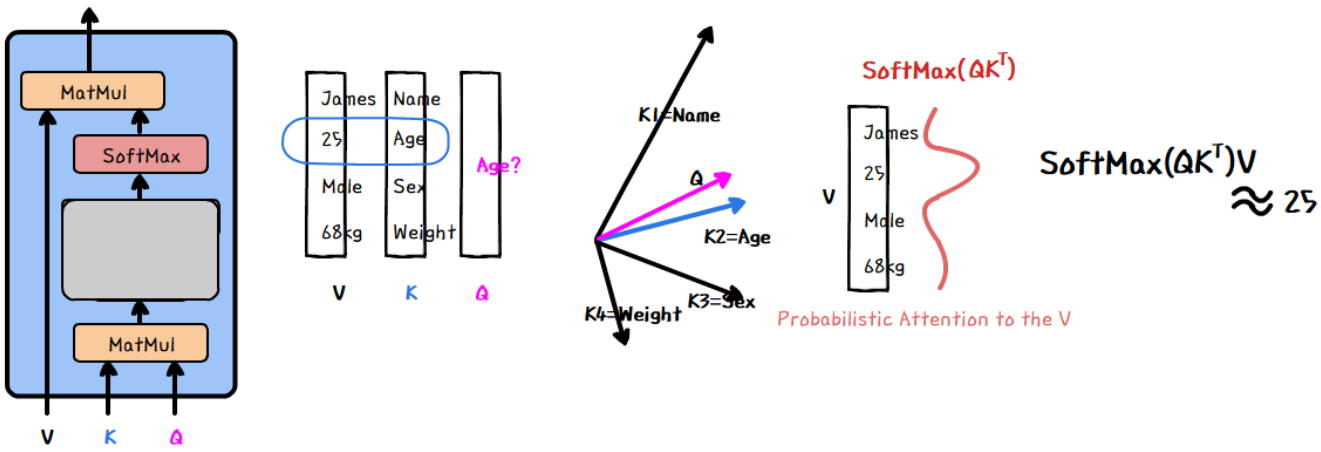
## Scaled Dot-Product Attention

Transformer에서 attention을 구하기 위한 가장 작은 단위입니다.




[Figure 6] Scaled Dot Product Attention

Scaled Dot-Product Attention은 전반적으로 두 개의 행렬 K와 행렬 Q의 multiplication을 통해 나머지 행렬 V 내의 정보를 추출하는 과정이라 할 수 있습니다. 구체적인 내용은 아래와 같습니다.




[Figure 7] Scaled Dot-Product Attention

행렬 K와 행렬 Q를 Dot-product하게 되면 행렬 K를 이루는 벡터와 행렬 Q를 이루는 벡터 간의 내적이라 볼 수 있습니다. 이를 통해 K와 Q의 유사한 row vector 값의 연산값이 높게 나타나게 됩니다. 이후 이러한 결과를 Softmax 함수를 통과하게 되면 Q와 유사한 K의 row들이 확률의 형태로 나타나게 됩니다. 유사하다면 높은 확률로 나타날 것이고 이 확률 값과 V를 내적하게 되면 얻고자 하는 정보를 추출할 수 있습니다.

여기서 Query와 Key와 Value는 아래와 같은 의미를 가집니다.

Query: 영향을 받는 단어 A를 나타내는 변수

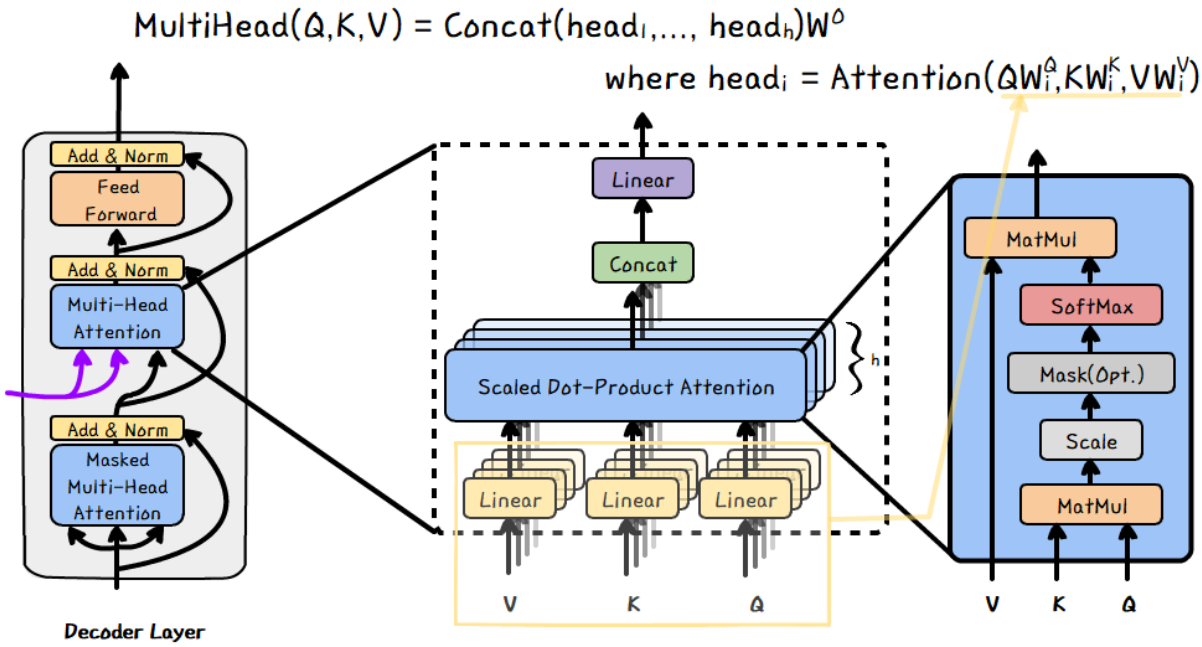
Key: 영향을 주는 단어 B를 나타내는 변수

Value: 영향에 대한 가중치를 나타내는 변수

이를 기반으로 다시 말해, Query를 예측하고자 하는 단어를 도출하기 위해 Key와 내적하여 가장 유사하여 확률 값이 높은 Key의 row를 추출하고, 이를 다시 Value와 내적하게 되면 가장 높은 확률 값을 구할 수 있게 된다고 볼 수 있습니다.

## Multi-Head Attention

위에서 Q, K, V를 여러 다른 차원으로 projection 시킨 후 각각에서 Scaled Dot-Product Attention에 적용하여 종합한 결과를 도출하는 것이 Multi-head attention이라 할 수 있습니다. 다른 말로 multi-head attention은 self-attention을 병렬적으로 사용한 것이라 볼 수 있습니다.




[Figure 8] Multi-Head Attention

V, K, Q 각각을 h번 다른 linear projection을 통해 변환시키고 병렬적으로 각각의 attnetion을 계산합니다. 이후 이를 종합하여 선형 변환을 통해 최종 값을 계산하게 됩니다.

## Feed Forward NN

일반적인 feed Forward Neural Network와 동일하며, 문장내의 정보를 추출하는 역할을 한다고 볼 수 있습니다.

## Transformer 모델 특징

위와 같은 메커니즘을 통해 transformer는 크게 학습에 있어 CNN과 RNN보다 significant하게 빠르다는 것이 장점입니다. 또한 input/output sequence의 거리에 관계 없이 동작가능하다는 것도 장점인데 이는 병렬처리를 하여 attention을 미리 계산해두었기 때문에 가능합니다.

## 복잡도 비교

아래의 표는 Self-Attention과 Recurrent, Convolution 레이어에 대한 계산 복잡도를 비교한 표입니다.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. $n$ is the sequence length, $d$ is the representation dimension, $k$ is the kernel size of convolutions and $r$ the size of the neighborhood in restricted self-attention.

| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|---|---|---|---|
| Self-Attention | $O(n^2 \cdot d)$ | $O(1)$ | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(log_k(n))$ |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

이 중 Transformer 모델에 도입한 Self-Attention 레이어가 가장 효율적임을 알 수 있습니다.

## 실험 결과

Transforemr 모델의 성능을 보면 아래와 같습니다. 아래는 BLEU 데이터셋에 대한 벤치마크 스코어와 부동소수점(FLOPs) 연산에 대한 결과 입니다.

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [18] | 23.75 | | | |
| Deep-Att + PosUnk [39] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [38] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [32] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [39] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | **$3.3 \cdot 10^{18}$** | |
| Transformer (big) | **28.4** | **41.8** | $2.3 \cdot 10^{19}$ | |

base model로 영어 → 독일어 번역 성능을 앞질렀고, 이후 big model로 영어 → 프랑스 번역으로 SOTA를 달성한 것을 확인할 수 있습니다. 또한 부동 소수점 연산 또한 base model을 사용하여 대략 100배 가량 계산 효율성이 좋을 것을 확인할 수 있습니다.

## Reference

https://machinereads.wordpress.com/2018/09/26/attention-is-all-you-need/
https://velog.io/@dscwinterstudy/%EB%B0%91%EB%B0%94%EB%8B%A5%EB%B6%80%ED%84%B0-%EC%8B%9C%EC%9E%91%ED%95%98%EB%8A%94-%EB%94%A5%EB%9F%AC%EB%8B%9D2-8%EC%9E%A5
https://machinereads.wordpress.com/2018/09/26/attention-is-all-you-need/
https://melon-buffer-f27.notion.site/Attention-is-All-You-Need-d484b19f68a54d589cfdb2d76495c73c
https://omicro03.medium.com/attention-is-all-you-need-transformer-paper-정리-83066192d9ab
http://ko.gravatar.com/harperpark
https://tech.kakaoenterprise.com/45



'AI Reasearch > 논문 리뷰' 카테고리의 다른 글

[논문 리뷰] All About GAN (Generative Adversarial Nets) (0) 2021.09.14
[논문 리뷰] Attention is all you need (0) 2021.09.09

NAME

PASSWORD

HOMEPAGE

http://

SECRET WRITE

PREV 1 2 NEXT

+ Recent posts


[독서노트#17] 타이탄의 도구…

[독서노트#16] 울트라 러닝 (…

[독서노트#15] 7막 7장 (홍정욱)

[독서노트#14] 라플라스의 악…