Transformer 모델의 기본 개념, 구조, 작동 원리 설명 리포트

딥러닝 프레임워크[00]
최인엽 교수님

201904236 산업데이터사이언스학부 전병준
202184042 데이터 사이언스 학과 이재성

1. 서론

Transformer 모델의 소개

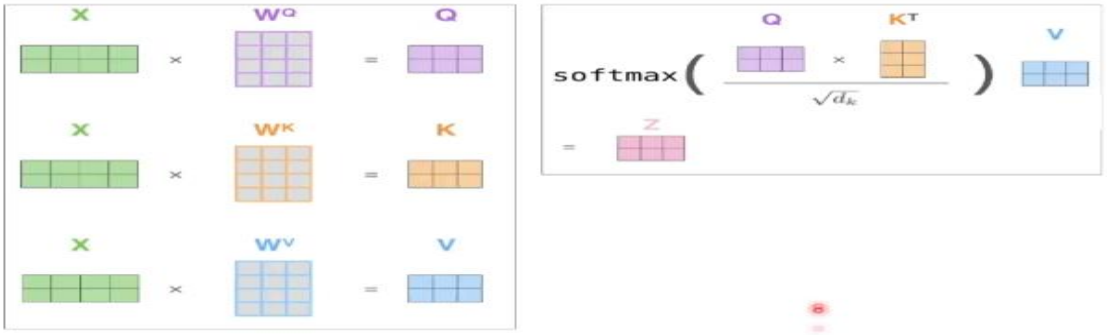
Attention은 2015년에 나왔지만 이 기법을 적용하고 좀더 높은 성능으로 업그레이드하여 transformer라는 이름으로 2017년에 나왔고,
현재 자연어처리 분야를 넘어서 인공지능의 많은 분야에 인용되고 있다.
기존의 RNN이나 LSTM과 같은 seq2seq 방식은 하나의 벡터에 모두 압축해야 했고,
이로 인해 병목현상이 발생한다는 단점이 있었다.
따라서 입력 시퀀스 전체에서 정보를 추출하여 병렬연산을 하는 방식으로 발전하게 되었고,
인코더-디코더 구조를 통해 시퀀스 데이터를 효과적으로 처리하는 transformer가 나오게 되었다.

Transformer 모델이 왜 중요한지, NLP 및 기타 분야에서의 역할

RNN을 사용하지 않는 기계변역이기 때문에 학습이 빠르고 성능이 빠르기 때문이다.
또한 Transformer모델은 NLP모델 분야에서 중요한 역활을 담당하고 있다.
Transformer모델은 다양한 NLP모델들에서 사용된다.
BERT - 트랜스포머가 갖는 인코더만 사용하여 진행하는 모델 이는 단어 인베딩과 문서인베딩과 분류작업, Q&N 에서 사용된다.
GPT - 생성 모형이며 트랜스포머가 갖는 디코더만 사용하며, 새로운 텍스트를 생성하는 목적으로 사용된다.
BART - 인코더와 디코더를 사용하여 텍스트 서머라이즈의 목적으로 사용된다.
추가적으로 컴퓨터 비전, 음성 처리, 게임 AI등 다양한 분야에서 응용이 되고 있다.

2. 이론적 배경

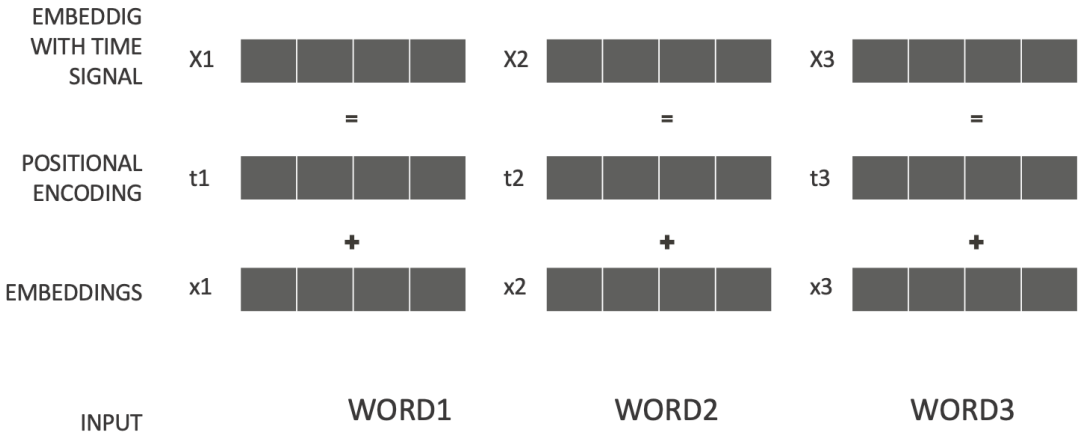
self Attention



각각의 임베딩된 벡터가 self attention에 input으로 들어오면 output은 input과 같은 dimension의 벡터를 내보냅니다.
그리고 output으로 나온 벡터들은 Neural Network에 각각 들어가게 됩니다.
임베딩된 벡터가 attention에 들어오면 각각의 query weight, key weight, value weight와 연산을 해서 각각의 query, key, value vector를 구할 수 있습니다.
그러면, 찾고자 하는 단어의 query vector를 가지고 해당 문장의 모든 key와 전부 내적을 합니다.

같은 벡터일수록 내적하면 큰 값이 나오기 때문에 내적했을 때의 가장 큰 값이 쿼리와 유사한 벡터가 됩니다.
따라서 query와 key vector를 내적해서 score를 구합니다.
쿼리가 자기자신의 키 벡터와는 매우 유사하므로 스코어가 크고, 다른 단어의 키와 연산한 스코어는 비교적 낮을 것입니다.
이후, 해당하는 dimension의 루트값으로 나눠줍니다. 이 과정은 정규화를 위한 과정인데, 벡터값이 너무 큰경우 정규화를 통해 기울기를 안정적으로 해주기 위한 과정입니다.
그리고 query, key 말고도 value가 있었는데 이 모든 임베딩에서 나온 softmax값과 value를 전부 곱해줍니다. 그 값을모두 더해주면 output으로 벡터가 나오게 됩니다.
이것을 여러번 수행하면 Multi Head Attention입니다.

Positional Encoding

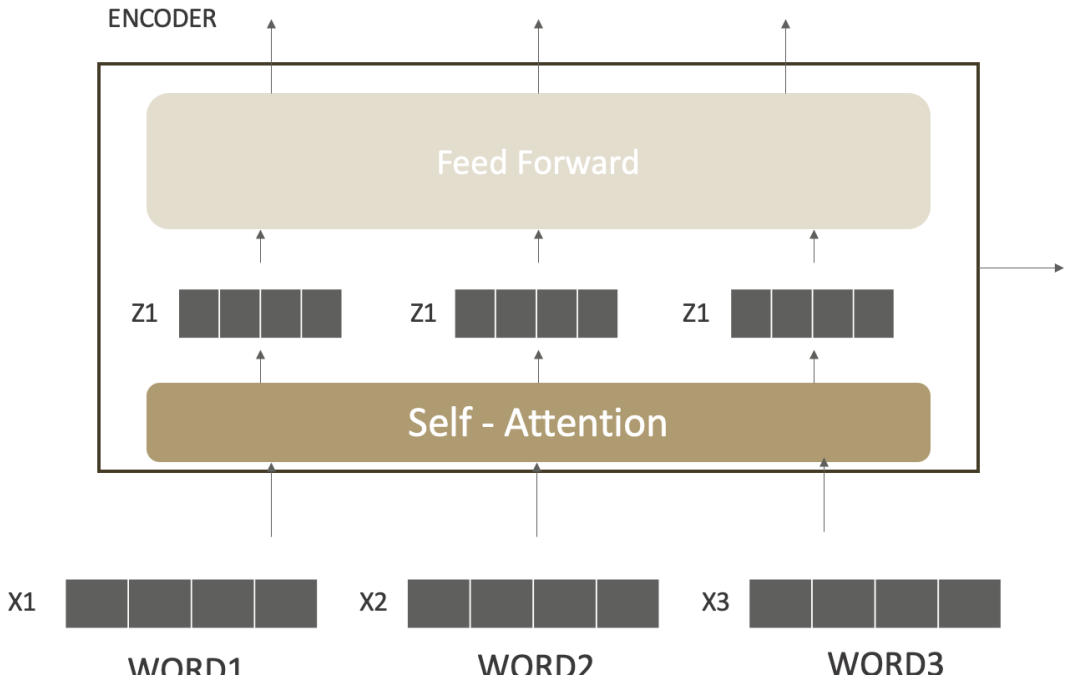


먼저 단어들은 워드 임베딩을 통해 각각의 벡터로 변환됩니다.
그리고 각각의 벡터들이 한꺼번에 들어가므로 위치정보가 없는 상태입니다.

따라서, 위치값을 저장해주기 위해 positional encoding을 함께 수행해줍니다.
다시말해, 임베딩된 벡터에 positional encoding vector를 더해주어 최종적으로 input으로 들어갈 벡터를 생성하게 됩니다.

Transformer 구조 분석

Encoder 과 Decoder

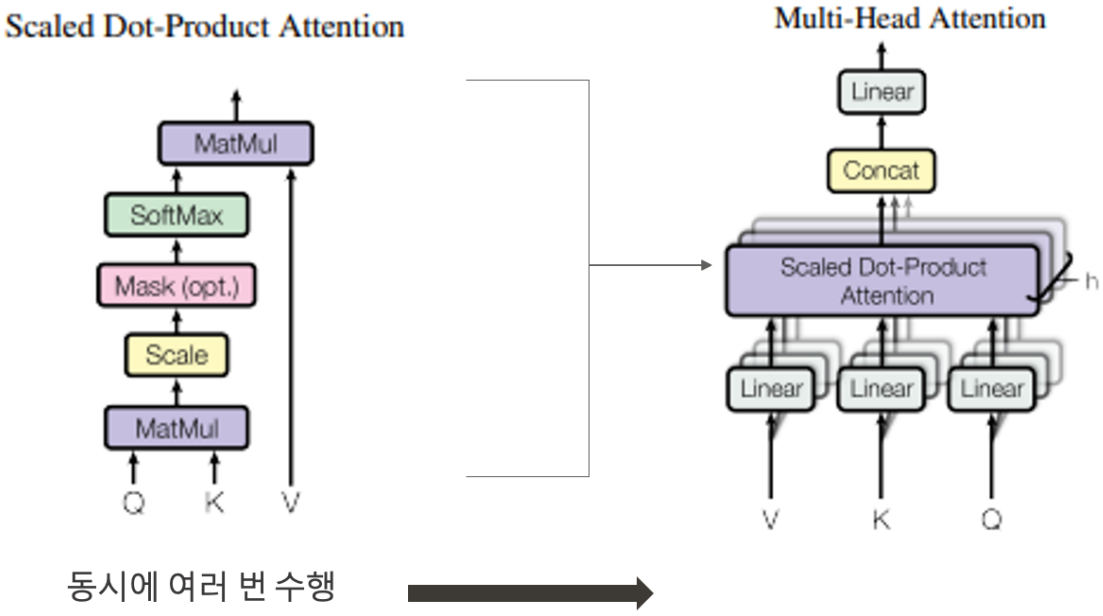


Encoder : 입력 시퀀스를 입력 받아 내부 표현으로 변환하는 역할을 한다.
주어진 문장을 고정된 길이의 벡터 형태인 표현으로 인코딩을 하고
이 표현은 Decoder로 전달되어 번역이나 문장 생성과 같은 작업을 수행한다.

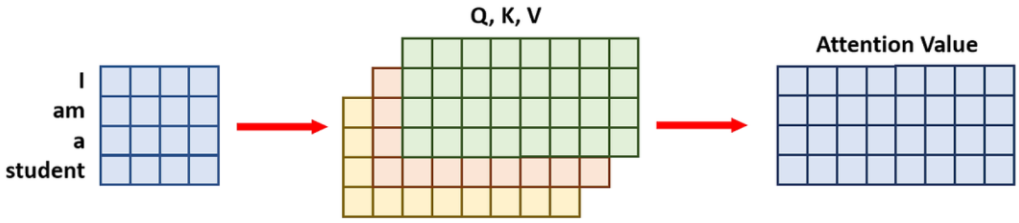
Decoder : 인코더의 출력을 입력으로 받아 목표 시퀀스를 생성하는 역할을 한다.
인코더의 출력과 이전 시간 단계의 출력을 사용하여 번역이나 문장 생성과 같은 작업을 수행한다.

Multi-Head Attention

지금까지의 과정들이 attention의 과정이고 이 과정을 동시에 여러번 수행하는 것이 multi head attention입니다.



multi head attention에서 나온 벡터들이 feed forward Neural Network에 들어갑니다.
feed forward Neural Network는 그냥 hidden layer가 하나인 fully connected layer입니다.
이 과정에서 같은 인코더 내에서 feed forward의 weight는 공유하지만, 다른 인코더 와는 공유하지 않습니다.이 모든 과정이 인코더 하나의 과정인데 이 과정을 총 6번 반복합니다.(논문에서 제시한 숫자)



즉 이는 Multi-Head Attention은 Quary,Key,Value값을 한번에 계산하지 않고 Head 수만큼 나눠 계산 후 나중에 Attention Value들을 합치는 메커니즘입니다.
위의 내용을 요약하자면

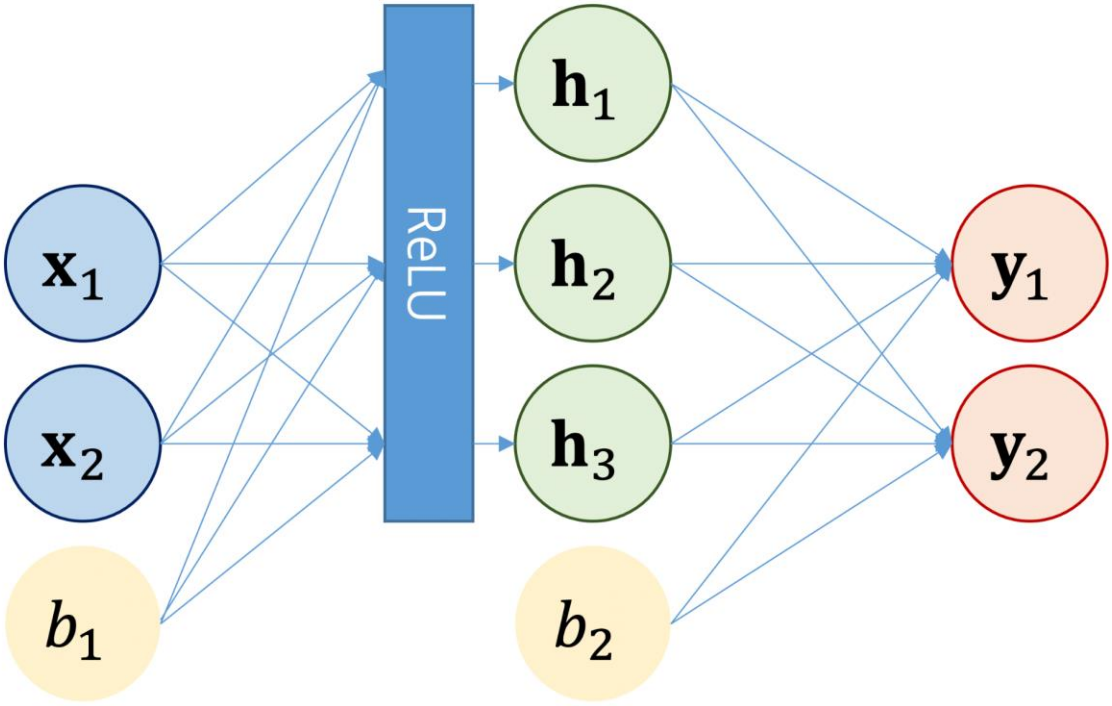
1. Quary, Key, Value 행렬 값을 Head의 수만큼 분할하고
2. 분할된 행렬 값을 통해, 각 Attention Value값들을 도출합니다.
3. 도출된 Attentiom Value값들을 Concate하여 최종 Attention Value를 도출합니다.

Feed-Forward Networks

$$FFN(z) = max(0, zW_1 + b_1)W_2 + b_2$$

의 공식을 제공하고 있으며
이는 Feed Forward Neural Network는 인코더와 디코더에서 공통으로 가지고 있는 서브층으로 Fully

connected Layer로 이루어진 신경망을 말합니다. 시각적인 자료로 나타내면 아래의 사진과 같습니다.



Feed forward Neural Network는 그냥 hidden layer가 하나인 fully connected layer입니다. attention을 통해 나온 벡터를 weight와 연산하고 bias(편향)를 넣어준 뒤에 ReLU Activation을 사용합니다. 그리고 한번 더 weight를 곱한 뒤 bias를 더해주는데, 이 과정을 확대해보면, input dimension이 512일 때의 기준으로 weight를 곱하고 bias를 취한 뒤 ReLU Activation function으로 2048 dimension까지 늘린 뒤에 다시 weight를 곱하고 bias를 더하는 방식으로 원래의 512dimension으로 되돌려 놓습니다. 이런 방식은 mlp방식과 유사한데, 보통 mlp에서 차원을 확장할 때 input dimension의 4배로 설정을 한다고 합니다. 셀프 어텐션을 통해 특정 단어와 다른 단어 간의 관계를 학습하여 관계성이 높은 단어를 특정하고 feed forward network를 통해 중요한 단어에 대해 더 깊은 의미를 부여하고 표현을 강화하는 역할을 합니다.

Residual Connections & Layer Normalization

Residual Connections

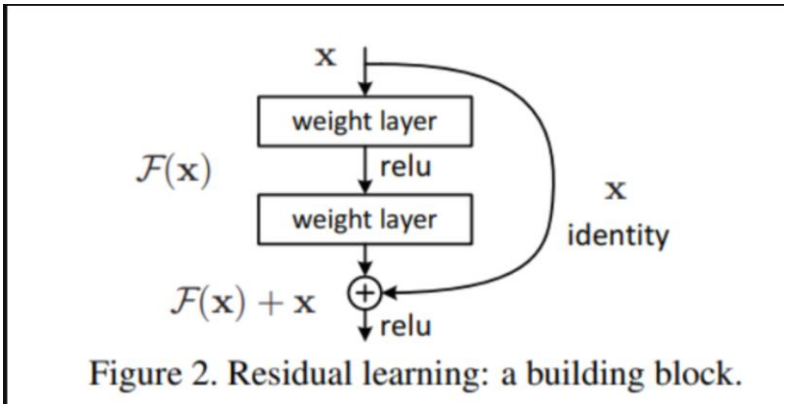
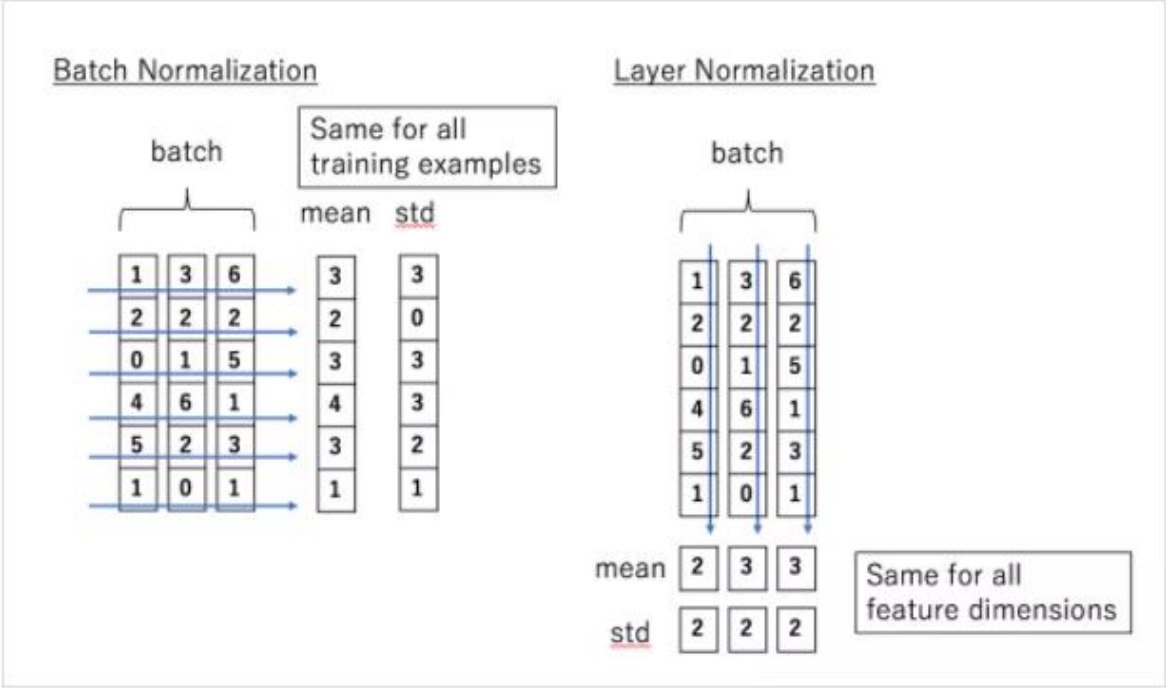


Figure 2. Residual learning: a building block.

Residual connection은 ResNet에서 등장한 개념이고 서브층의 입력과 함수를 거친 출력을 더해주는 것입니다. transformer에서는 행렬의 연산으로 multi head attention, feed forward network를 구하고 마지막에 가중치값을 구해서 입력값과 출력값이 동일한 차원을 갖게 만들고 있습니다. 이로인해 Residual connection에서는 덧셈 연산이 가능하게 됩니다.

Layer Normalization



Data sample 단위로 평균(mean)과 표준편차(std)를 계산해서 정규화를 실행한다.

특성의 개수와 상관없이 batch 내부의 데이터 개수가 3 개이기 때문에, 3 개의 데이터 샘플에 3 개의 평균, 3 개의 표준편차 값을 계산하여 Layer Normalization 을 실행한다.

Layer Normalization 은 이처럼 batch 단위가 아니라 input 을 기준으로 평균과 분산을 계산하게 된다.

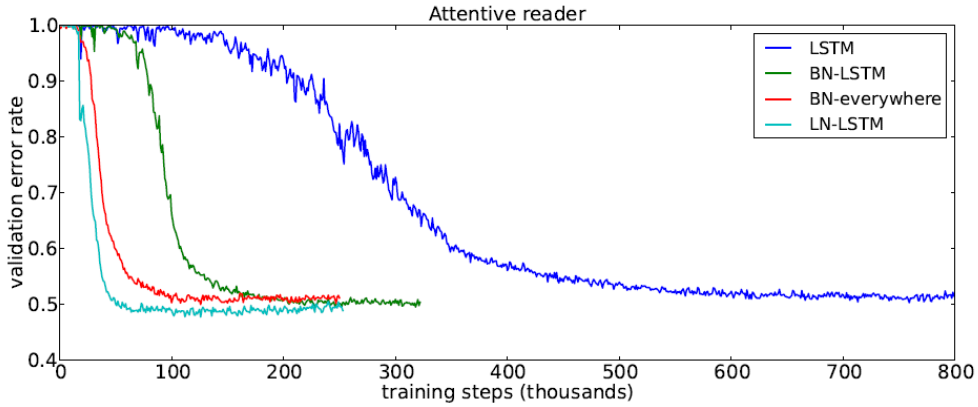


Figure 2: Validation curves for the attentive reader model. BN results are taken from [Cooijmans et al., 2016].

(위의 사진은 질문에 포함된 빈칸에 대한 알맞은 단어를 예측하는 실험을 수행했을 때 그래프이다. 위의 결과와 같이 Layer Normalization을 이용한 LSTM 모델이 가장 좋은 성능을 나타낸다는 것을 확인 할 수 있다.)

특징으로는

1. 데이터마다 각각 다른 normalization tern을 갖는다.
2. Mini - batch 크기에 영향을 받지 않는다.(size=1 이어도 작동)
3. 서로 다른 길이를 갖는 sequence가 batch 단위의 입력으로 들어오는 경우에도 적용할 수 있다.

일반적인 RNN 모델에서는 많은 time-step을 거칠 때 gradient가 explode 또는 vanish 하는 현상이 발생한다. 그러나 LM이 적용될 경우 Layer의 scale에 영향을 받지 않기 때문에, 안정적인 학습이 진행된다.

이고 장점은

1. Batch Normalization은 작은 배치 크기에 극단적인 결과를 내는데 반해, 작은 batch size에도 효과적인 이용이 가능하다.