Attention is all you need

Attention mechanism을 활용한 Transformation Model

25.04.06

정보컴퓨터공학부
202055528 김태환

Contents

초록

- **기존의 Sequence 변환 Model**
  - Encoder와 Decoder 를 포함하는 Complex RNN 또는 CNN에 기반한다.

- **최고의 성능 Model**
  - Attention Mechanism을 통해 Encoder와 Decoder를 연결한다.
  - Recurrence와 Convolution을 없애고 오직 Attention Mechanism에반 기반한 Model이다.

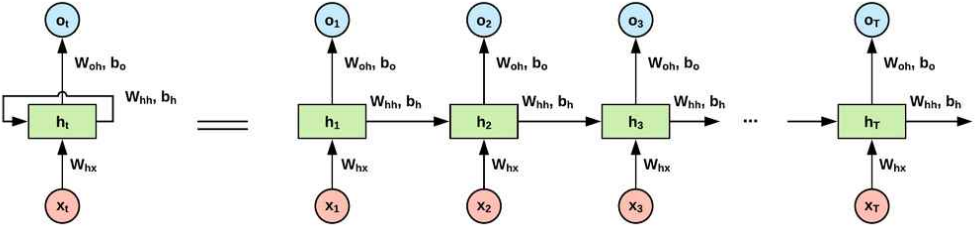
- **훈련 퍼포먼스**
  - 두번의 기계번역 실험은 더 적은 훈련시간을 요구한다.
  - 병렬화 측면에서 용이하다.
  - 영어-독어, 영어-불어 번역 대회에서 더 적은 훈련시간을 사용하여 최고의 성능을 보였다.

소개

- **기존에 확립되어 있던 Models**
  - RNN, LSTM과 같은 모델들은 변환 문제와 sequence modelling 분야에서 최고의 접근방식으로써 확립되어졌다.
- **한계를 뛰어넘기 위한 노력**
  - 수많은 연구들이 RNN 언어모델과 인코더-디코더 구조의 한계를 뛰어넘기 위해 노력했다.
- **최근 동향**
  - 연산향상성과 모델 성능을 높였지만, 여전히 순차연산의 제약은 존재했다.
- **이전 모델에서의 Attention Mechanism 활용**
  - Attention 메커니즘은 단어간의 거리에 대한 의존성을 제거한다.
  - 설득력있는 모델을 제시하지만, 기본적으로 RNN과 함께 사용되어진다.
- **논문이 제시하는 모델**
  - 논문은 Recurrence를 배제하며 인풋과 아웃풋간의 전역적인 의존 관계를 도출시킨다.
  - 이를 통해 병렬처리가 가능해진다.

소개

- **RNN 모델의 단점**
  - 기존의 RNN 모델은 sequence 처리를 위해 h_{t-1} 와 x_t 를 필요로 한다.
  - 즉, symbol 처리를 위해서는 이전 단계의 은닉층 출력값을 기다려야 한다.



ex) "I love you" 라는 sequence가 주어질 때, 각 단어의 임베딩 벡터가 x_t이고 이전 출력값인 h_{t-1} 와 함께 연산되어 세 단어의 의미를 함축한 최종 벡터를 생성한다.

출처:https://velog.io/@rsj9987/딥러닝-용어정리

배경

- **순차연산을 줄이려는 목표**
  - 순차연산을 줄이려는 목표는 External Neural GPU, Bytenet, ConvS2S 모델들에서의 CNN을 활용한 병렬연산의 적용으로 이어졌다.
  - 하지만 두 단어간의 관계를 계산하는 것은 선형, log적으로 연산량이 늘어났다.
    - 두 단어가 멀리 떨어져있다면 연관성을 계산하기가 복잡해진다.

- **Self-Attention의 동작방식**
  - Sequence의 단어들을 연결하여 연관성을 표현하는 벡터를 만든다.
  - 해당 기술은 독해, 요약, 문장추론 등과 같은 분야에서 좋은 성과를 도출했다.

- **논문이 제시하는 모델**
  - CNN과 RNN없이 Self-attention만으로 입력과 출력을 표현하는 최초의 모델이다.

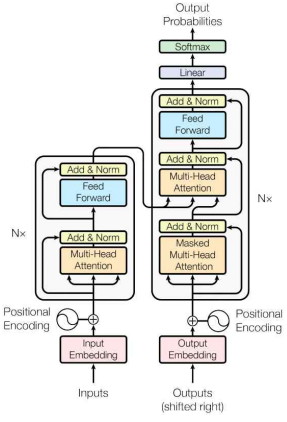
모델 구조

- **Encoder-Decoder 구조**
  - 대부분의 강력한 모델들은 Encoder-Decoder 구조를 취한다.
  - 즉, 입력해석-출력생성 구조를 가진다.

- **Encoder**
  - 인코더는 입력 Sequence의 symbol representation **x**를 **z**로 변환시킨다.

- **Decoder**
  - 디코더는 **z**가 주어지면 output sequence를 만들어낸다.

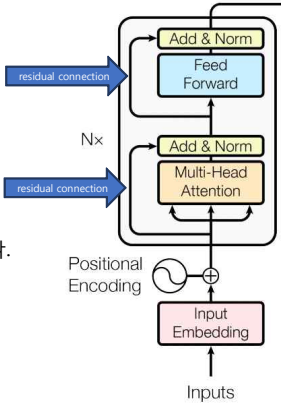
- **Transformer 구조**
  - Transformer 또한 위의 구조를 따르며 layer를 겹겹이 쌓은 구조이다.



출처:논문-Attention is all you need

모델 구조 - Encoder

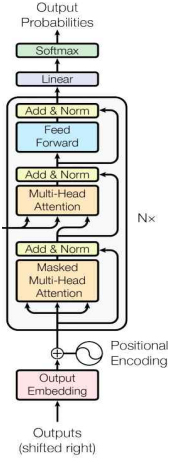
- **Encoder의 구조**
  - input을 embedding하여 위치정보를 더해준다.
  - 같은 구조의 layer 6개를 겹겹히 쌓은 형태이다.
  - 각 layer는 두 개의 sub-layer를 가진다.
    - Multi-head attention
    - Feed forward
  - 각 sub-layer에 대해 residual connection과 Layer normalization을 실시한다.
    - residual connection : 모델의 단순화를 위해 input을 다시한번 입력한다.
    - layer normalization : 정규화를 진행한다.



출처:논문-Attention is all you need

모델 구조 - Decoder

- **Decoder의 구조**
  - encoder와 같이 6개의 layer를 가진다.
  - 각 layer는 세 개의 sub-layer를 가진다.
    - Multi-head attention
    - Encoder-Decoder attention
    - Feed forward
  - 각 sub-layer에 대해 residual connection과 Layer normalization을 실시한다.
    - residual connection : 모델의 단순화를 위해 input을 다시한번 입력한다.
    - layer normalization : 정규화를 진행한다.

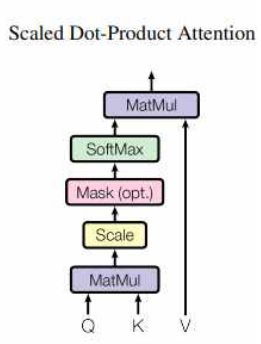


출처:논문-Attention is all you need

모델 구조 - Attention

- **Attention은 무엇인가?**
  - 특정단어가 문장에서 다른 단어들과 가지는 연관성을 반영한 수치이다.
  - **Q**(Query) : 중심이 될 단어 / **K**(Key) : 다른 단어들 / **V**(Value) : **Q**의 실제 정보
- **Attention이 이루어지는 방법**
  - **Q**(Query)와 **K**(Key) 벡터를 점곱하여 유사도를 구한다.
    - QK^T 는 특정단어와 다른 단어들간의 유사도를 의미한다.
  - **QK^T** 에 $\sqrt{d_k}$를 나누어 수를 간소화한다.
  - 간소화된 벡터에 **V**를 곱하여 최종 벡터를 구한다.
    - 실제 값인 V를 곱함으로써 단어들의 가중치가 반영된 벡터가 만들어진다.
    - 도출된 벡터는 **"sequence내에서"** 특정 단어가 갖는 의미를 나타낸다.

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



출처:논문-Attention is all you need

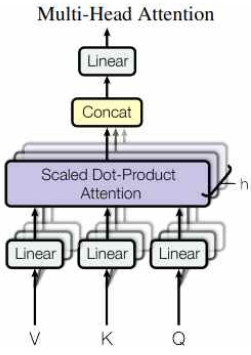
모델 구조 - Multi-head Attention

- **Multi-head Attention**
  - Attention을 병렬로 계산한다.
    - Q, K, V를 달리하여 h번 attention을 진행한다.
    - 선형변환을 통해 다양한 시점에서의 출력 값을 얻는다.
    - 풍부한 단어들과의 관계 값을 구할 수 있다.
  - 앞에서 구한 여러 attention값을 합쳐 최종 값을 구한다.
  - Multi-head Attention을 표현하는 공식은 아래와 같다.
    - head의 개수에 따라 연산량을 조절하면서 총 연산량을 유지할 수 있다.
      - 즉, Q, K, V의 차원을 줄이면 head의 개수가 늘어나도 총 연산량은 동일하다.

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1, ..., \mathrm{head}_\mathrm{h})W^O$$

$$\text{where } \mathrm{head}_\mathrm{i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

      - 각각의 head에 따른 관점으로 분석하기 위해 W_i를 곱해준다.

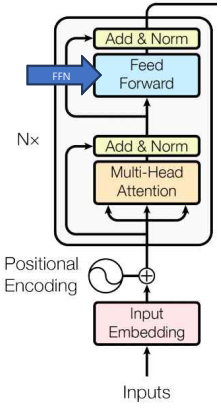


출처:논문-Attention is all you need

모델 구조 - Position-wise Feed-Forward Networks

- **Position-wise Feed-Forward Networks**
  - 인코더와 디코더에는 동일하게 Feed-Forward network가 존재한다.
    - 각 단어에 대해 동일하게 적용된다.
    - 두 번의 선형변환을 진행한다.
    - 두 번의 bias를 더한다.
  - 6개의 레이어는 서로 다른 파라미터(W_1, W_2)를 가진다.
    - 각 레이어마다 다른 해석이 가능해진다.
  - FFN을 표현하는 공식은 아래와 같다.
    - W_1의 역할 : 단어의 벡터공간을 확장하여 더 많은 특징을 추출한다.
    - max의 역할 : ReLU함수를 적용함으로써 중요한 값을 강조한다.
    - W_2의 역할 : 기존의 벡터공간으로 축소하여 새 특징들을 적용한다.

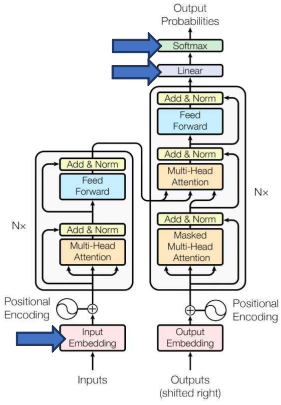
$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



출처:논문-Attention is all you need

모델 구조 - Embeddings and Softmax

- **Embeddings and Softmax**
  - Embeddings
    - Input tokens를 output tokens로 변환한다.
    - 다른 변환 모델과 마찬가지로 tokens의 d_{model}차원으로의 벡터 변환이 필요하다.
  - Softmax
    - 디코더의 output vector를 선형변환하여 단어들의 확률 분포 벡터로 변환한다.
    - 위의 확률 분포 벡터를 Softmax를 활용하여 확률값으로 변환한다.



출처:논문-Attention is all you need

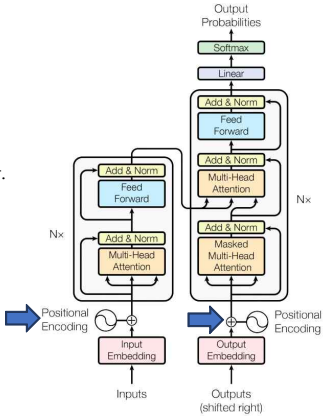
모델 구조 - Positional Encoding

- **Positional Encoding**
  - Sequence내에서 token들 간의 위치 정보를 반영해야 한다.
    - 이 모델에서는 RNN도 CNN도 활용하지 않기 때문이다.
  - 임베딩된 vector에 positional encoding vector를 더해준다.
    - 단어의 의미를 내포하는 vector에 위치정보를 삽입하는 과정이다.
    - 임베딩 vector와 같은 차원(d_{model})을 가지게 함으로써 단순히 더할 수 있다.
  - 다양한 positional encoding 방식이 존재한다.
    - 이를 학습시키는 방법도 존재하나, 고정된 수학 함수로 사용한다.
    - 이를 통해 구한 위치정보를 기반으로 상대적 위치를 구할 수 있다.

      ex) PE vector의 거리가 가까우면 $Q*K^T$에서 높은 수치를 보인다.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



출처:논문-Attention is all you need

결론

- **전적으로 Attention에 기반한 sequence 변환 모델**
  - RNN기반 구조에서 탈피함으로써 병렬연산이 가능하도록 한다.
  - Attention에 전반적으로 기반한 최초의 모델이다.
  - 2014 WMT 영어-독어, 영어-불어 번역 작업에서 최고 성능을 달성했다.
  - 하지만 decoder에서 병렬연산이 불가능하다는 단점이 존재한다.

감사합니다