Tobigs 15기 Week9 Transformer – 15기 이성범

Attention is All You Need 리뷰

리뷰

일단 논문을 리뷰하기에 앞서 Attention Mechanism에 대하여 설명하고자 한다. 기본적으로 Attention Mechanism이 등장하게 된 배경은 기존의 모델인 RNN에 기반한 seq2seq 모델이 크게 2가지의 문제점을 가지고 있었기 때문이다. 첫번째 하나의 고정된 크기의 벡터에 모든 정보를 압축하고자 하니 정보의 손실이 발생하고 두번째 RNN의 고질적인 문제인 기울기 소실 문제가 발생하는 것이다. 위의 두가지 문제점에 위해서 입력문장이 길면 길수록 번역의 품질이 감소하는 현상이 나타났다. 따라서 이를 해결하고자 Attention Mechanism이 등장하게 되었다.

Attention Mechanism의 기본 아이디어는 Decoder에서 출력 단어를 예측하는 매 시점 마다 Encoder에서 전체 입력 문장을 다시 참고함으로써 기존의 문제점을 해결하고자 하는 것이다. 그런데 여기서 중요한 점은 전체 입력 문장을 전부 참고하는 것이 아닌 해당 시점에서 예측해야할 단어와 가장 연관이 있는 단어를 좀 더 집중해서 참고하자는 것이다. 좀 더 집중해서 참고를 하기 위해서 softmax함수를 활용하여 중요한 단어의 확률을 높이는 output을 만들 수 있다.

본 논문은 이러한 기존의 문제점을 해결할 수 있는 Attention Mechanism을 RNN과 CNN에 적용하는 것이 아닌 완전히 Attention Mechanism만을 활용하여 Encoder와 Decoder을 만든 모델인 Transformer 모델을 제안했으며 본 논문이 제안한 Transformer의 성능은 기존의 기계번역 모델들의 성능을 뛰어넘었으며 이 논문 이후에 대부분의 기계번역 논문이 Transformer을 가지고 발전되어 왔다. 최근에도 기계번역에서 가장 성능이 좋은 모델들은 대부분 Transformer를 기본 베이스로 활용하고 있다.

본 논문을 Positional encoding, Multi head attention, Self attention, Masked attention 의 키워드를 중심으로 리뷰를 진행하도록 하겠다. 우선 Positional encoding이다. Positional encoding은 단어의 임베딩 벡터에 위치정보를 더하는 방식을 의미한다. 기존의 RNN의 경우 hidden state를 통해서 과거 단어의 정보를 입력받기 때문에 입력 값 자체적으로 위치 정보를 가질 수 있었다. 하지만 Transformer의 경우 RNN처럼 단어를 순차적으로 입력 받지 않기 때문에 임베딩 벡터에 위치정보를 알려줄 필요가 존재했다. 따라서 본 논문에서는 아래의 두 함수를 통해서 단어의 위치정보를 생성하여 입력 값인 임베딩 벡터에 더해 줌으로써 위치 정보를 저장한다. 또한 본 논문에서는 함수를 통하여 위치 정보를 생성하지 않고 학습을 통하여서도 위치정보를 얻을 수도 있다고 언급하고 있다.

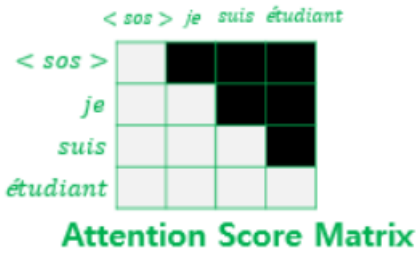
$$PE_{(pos,\,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

그 다음의 키워드는 Multi head attention이다. 기존의 모델들과 Transformer의 가장 큰 차이점이며 Transformer의 최고의 장점으로 attention은 병렬로 처리한다는 것을 의미한다. 병렬로 처리함으로써 모델 학습의 시간 상의 위를 얻을 수 있었고 성능 또한 향상되었다고 한다.

그 다음은 Self attention 이다. 이는 Encoder에서 이루지는 것으로 어텐션을 자기 자신에게 수행한다는 것을 의미한다. Q, K, V를 모두 자기 자신 단어 벡터의 값을 넣음으로써 자기 자신의 문장에서 가장 중요한 단어를 output으로 출력하는 방식이다. 한마디로 '나는 동물을 좋아한다. 왜냐하면 그것이 귀엽기 때문이다.' 라는 문장이 존재할 때 Qury에 '그것'이라는 단어가 들어갔을 때 Key에는 나는, 동물을, 좋아한다,.....때문이다. 등이 존재한다고 가정했을 때 그것이 어떤 단어들과 가장 유사한지를 계산하여 '그것'이 '동물'과 유사한다는 것을 모델이 알 수 있도록 Value로 동물과 매우 유사하다는 벡터 값을 출력해주는 방식이다.

마지막으로 Masked attention이다. 이는 self attention과 동일한 연산을 수행하지만 미래의 단어 즉 나보다 앞에 있는 단어는 참고하지 못하도록 설계한 attention이다. 아래의 그림과 같이 나보다 미래에 존재하는 단어의 경우에는 mask 처리를 하여 참고하지 못하도록 하여 나와 가장 유사한 단어를 찾아가는 방식이다. 이러한 처리를 하는 이유는 이 층을 지난 후에 Encdoer에서 출력된 값과 합쳐지는 인코더-디코더 어텐션 층을 거치게 되기 때문이다. Masked attention을 Qury 값으로 활용하여 질문을 만들기 위해서 이며 Encoder에서 나온 값들은 key와 value 값으로 활용되어 이제 앞으로 나타날 단어를 예측할 수 있게 된다.



Attention Score Matrix

참고자료

- https://www.youtube.com/watch?v=AA621UofTUA&t=937s

- https://reniew.github.io/43/

- https://wikidocs.net/31379

- https://wikidocs.net/22893