- 본인 아이디 및 닉네임

jinsuc28(블로그 아이디), 6팀_최진수

- 게시글 URL

https://jinsuc.tistory.com/9

- 게시글 캡처

1. Transformer는 왜 나왔는가?

transformer 이전 모델들은 **recurrent**한 모델들이 많이 사용되었다.

아래와 같은 recurrent 모델의 단점을 보완한 모델이라고 할 수 있다.

1. 순차적으로 학습하기 때문에 마지막 출력단에서 첫번째 입력 단어에 대한 정보를 잃기 쉽다는 점이다. (긴 문장 학습 불리)

2. 병렬학습이 아니기 때문에 학습하는데 시간이 오래걸린다.

2. Transformer의 구조

2. Transformer의 구조

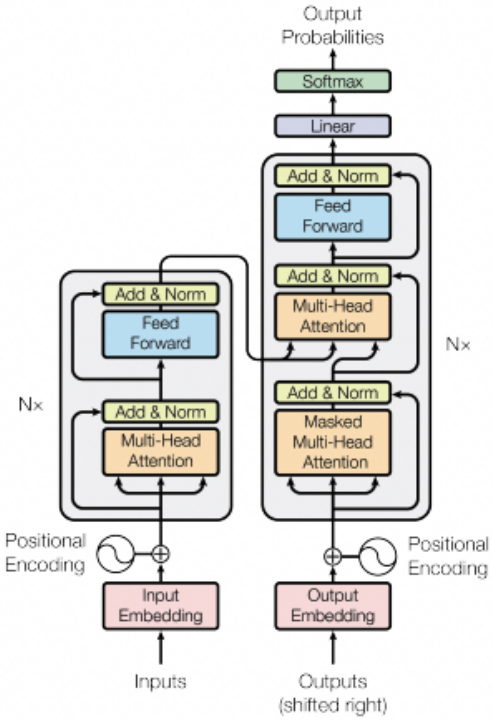


Figure 1: The Transformer - model architecture.

Transformer model Architecture

transformer는 주로 기계번역에서 많이 이용되고 이때 encoder와 decoder라는 개념이 사용된다.

transformer의 모델 구조는 이후 BERT와 GPT에서 사용되는데

encoder만 이용한 것이 BERT decoder부분만 이용한 것이 GPT 모델 구조이다.

2-1. encoder

positional encoder라는 개념이 있다. 줄여서 pos로 불리기도 한다.
positional encoder는 단어의 위치 정보를 주기 위해 도입된 것이다.

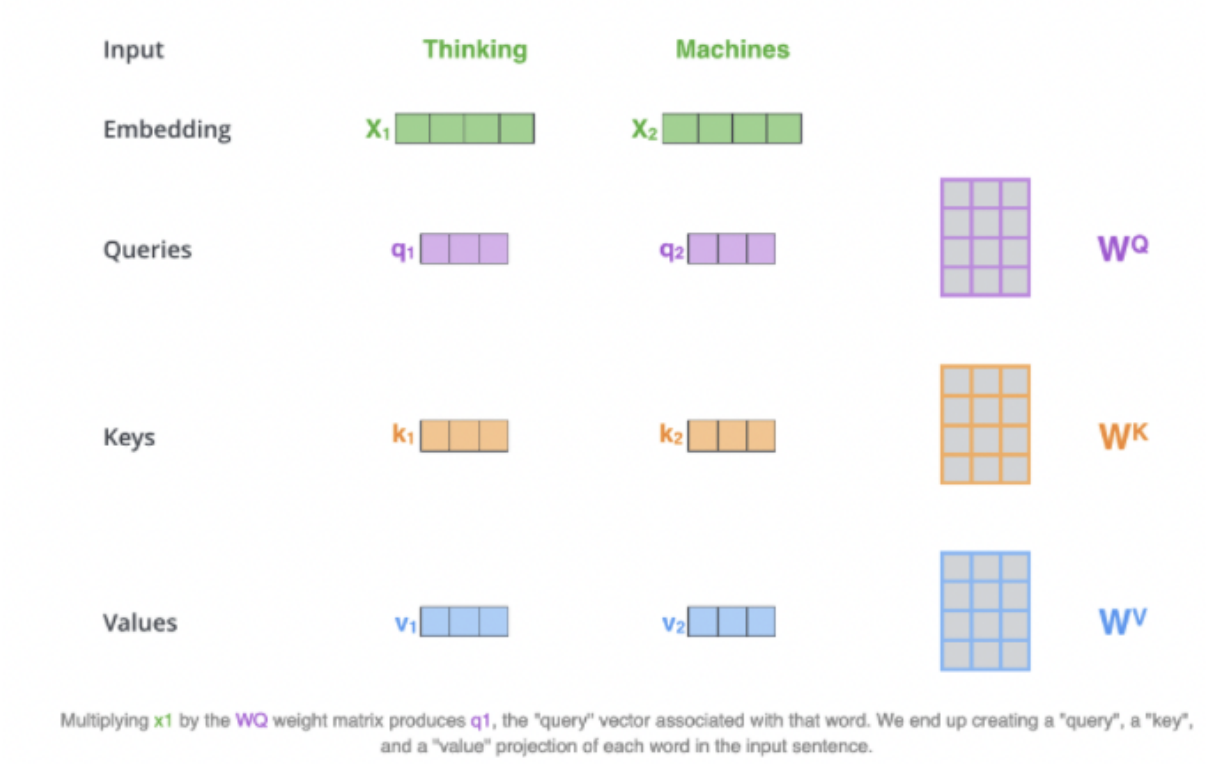
기존 ELMO같은 recurrent모델에 경우 순차적으로 데이터가 입력되어 학습되기 때문에 위치정보가 그대로 반영이 된다.

하지만, transformer는 입력이 병렬로 한번에 되기 때문에 위치 정보를 주어야한다.

이를 주기 위해서 논문은 sin함수를 이용하여서 input embedding vector의 ids 정보와 vector 값으로 주어 position encoder를 구하게 된다.

이때 사용되는 함수는 sin함수가 아니여도되며 cos함수 라던지 여러 방법론이 존재한다.

다음으로 **Multi-head attention** 개념에 대해 알아보겠다.
positional encoder와 embedding vector를 더한 값을 input으로 받으며
이때 Key, Query, Value를 생성하게 된다.



Multiplying x1 by the WQ weight matrix produces q1, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

각각의 key,query,value는 embedding 각각의 단어에 w(k,q,v)값이 행렬곱 생성된다.

따라서 단어마다 key,query,value 3개의 생성됩니다.

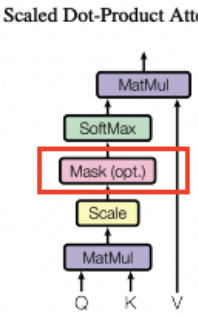
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

dk는 Embedding vector의 차원수를 말하며 query와 key가 서로 행렬곱 되어지고
이를 softmax취하여 상대적으로 중요한 단어는 0의 가깝게 취하고 value값을 행렬곱 해줍니다.

이러한 작업을 통하여 나온 것 1 attention입니다. 이를 여러번 취하게 된 attention을 합하면
mulit head attention이 됩니다. 이때 필요한 작업이 있습니다.

각 attention의 head 값들을 embedding vector와 같은 사이즈로 각각 꺼내고(이는 서로 다른 정보를 내포하게됨)
이를 concat하며 또 다른 w값을 행렬곱하여 embedding vector와 같은 사이즈의 multi head attention을 뽑게 됩니다.

이 작업은 각 단어의 관계 정보와 중요도를 수치화 하기 위함입니다.



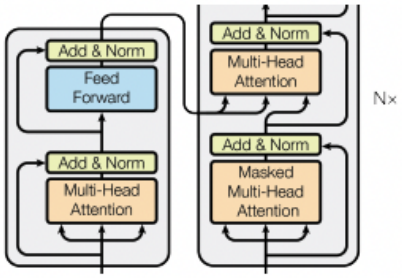
논문 중 설명

이때 Mask는 옵션이며 이 attention 작업 중에 중요한 단어의 정보를 0의 가까운 수를 주어
상대적으로 중요하지 않는 단어로 만들 수 가 있습니다.

2-2. decoder

마지막 attention layer에서 뽑힌 output은 다시 decoder마다 attention으로 들어가게 됩니다.

이때 주의할 점은 decoder는 1 layer의 attention이 두번 시행된다는 점입니다.
그리고 encoder에서 output은 각각의 decoder의 두번째 attention에 들어가게 됩니다.



이때 encoder의 attention은 decoder의 두번째 attention의 query형태로 들어가게 됩니다.
이러한 작업을 통해 encoder의 vector 정보를 고려한 attention작업을 수행하게 됩니다.

주의할 점은 decoder layer는 encoder layer의 숫자와 같다는 점입니다.