Self-attention -> 독해, 추상적 요약, 텍스트 수반 및 학습 과제, 독립적인 문장 표현을 포함한 다양한 과제에서 성공적으로 사용됨

End-to-end memory 시퀀스 정렬반복 대신, recurrent attention 메커니즘을 기반으로해서

간단한 언어 질문 응답 및 언어 모델링 작업에서 잘 수행됨

인코더 기호 : 입력 기호 x1~xn, 연속 기호 : z1~zn

z에의해 생성된 출력 기호 y1~ym

**인코더**

는 6개의 동일한 layer stack으로 구성됨

각 layer에는 두개의 하위 layer가 있음

첫번째는 multi-head self-attention이고

두번째는 fully connected feed forward network

두개의 하위 layer 각각 주위에 residual connection 사용 + layer normalization 사용

즉 output of each sub-layer isLayerNorm(x Sublayer(x)),

이러한 residual connections을 용이하게 하기 위해 모든 하위 layer 와 embedding layer 의 dimension = 512 출력을 생성함.

디코더

는 6개의 동일한 layer stack으로 구성됨

인코더 layer의 2 하위 layer + encoder의 출력에 대해 multi-head attention을 수행하는 세번째 하위 layer 삽입

인코더와 유사하게 각 하위 layer 주위에 residual connection 사용하고 계층 normalization 사용

또한 디코더 스택의 self-attentionsub-layer를 수정하여 위치가 후속 위치에 주의를 기울이는 것을 방지합니다.

Attention

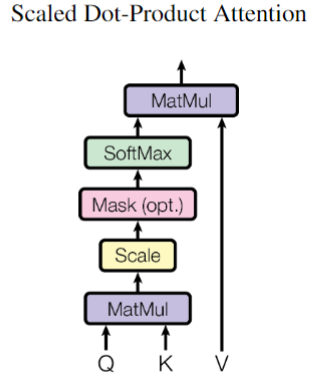
Query, key-value 쌍 집합을 출력에 mapping 하는것

Query, key, value, output은 모두 vector임

출력은 값의 가중치 합계로 계산

각 값에 대한 가중치는 key와query의 호환성 함수에 의해 계산

Scalec dot product attention



Query 와 모든 key에 대해 dot product 진행 후

루트 d\_k로 나누고, softmax 통해서 가중치 값을 얻음

텍스트이(가) 표시된 사진

자동 생성된 설명

정리한 수식

내적이 행렬곱으로 표현 가능해서 훨씬 빠르고 공간 효율적.

D\_k의 큰 값에 대해 내적 진행시, softmax함수를 Small gradients 밀어 넣는걸 방지 하기 위해 1/루트d\_k사용해서 스케일링 해줌

Multi-head attention

Single attention으로 진행하는 것 보다 각각 선형으로 프로젝션 하는게 더 유익하단걸 발견

Multi-head attention을 통해 모델은 서로 다른 위치에 있는 서로 다른 표현 하위 공간의 정보에 공동으로 주의를 기울일 수 있다.

수식

텍스트이(가) 표시된 사진

자동 생성된 설명

8깊이의 병렬 형식 attention layer 또는 head를 사용

각각은 64깊이 사용

* 각 헤드의 축소된 차원으로 인해 총 계산 비용은 전체 차원의 single head attention과 유사

**Applications of attention in our model**

Transformer 는 3가지 방법으로 multi head attention을 사용한다.

1. "encoder-decoder Attention" 계층에서 쿼리는 이전 디코더 계층에서 가져오고 메모리 키와 값은 인코더의 출력에서 ​​가져옵니다. 이를 통해 디코더의 모든 위치가 입력 시퀀스의 모든 위치를 처리할 수 있습니다. 이것은 다음과 같은 sequence-to-sequence 모델에서 전형적인 인코더-디코더 주의 메커니즘을 모방합니다.

2. 인코더에는 자체 주의 레이어가 포함되어 있습니다. self-attention layer에서 모든 키, 값 및 쿼리는 같은 위치에서 옵니다. 이 경우에는 인코더에서 이전 레이어의 출력입니다. 인코더의 각 위치는 인코더의 이전 계층에 있는 모든 위치에 주의할 수 있습니다.

3. 유사하게, 디코더의 셀프 어텐션 레이어는 디코더의 각 위치가 그 위치를 포함하는 디코더의 모든 위치에 주의를 기울일 수 있도록 합니다. 자동 회귀 속성을 유지하려면 디코더에서 왼쪽 정보 흐름을 방지해야 합니다. 우리는 불법 연결에 해당하는 softmax 입력의 모든 값을 마스킹(1으로 설정)하여 scaled dot-product Attention 내부에서 구현합니다.

**Position wise feed forward networks**

주의 하위 계층 외에도 인코더 및 디코더의 각 계층에는 완전히 연결된 피드포워드 네트워크가 포함되어 있으며 각 위치에 개별적으로 동일하게 적용됩니다. 이것은 사이에 ReLU 활성화가 있는 두 개의 선형 변환으로 구성됩니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

선형 변환은 다른 위치에서 동일하지만 레이어마다 다른 매개변수를 사용합니다. 이것을 설명하는 또 다른 방법은 커널 크기가 1인 두 개의 컨볼루션입니다. 입력 및 출력의 차원은 dmodel= 512이고 내부 레이어의 차원은 dff= 2048입니다.

**Embeddings and softmax**

다른 시퀀스 변환 모델과 유사하게 학습된 임베딩을 사용하여 입력 토큰과 출력 토큰을 차원 모델의 벡터로 변환합니다. 또한 일반적인 학습된 선형 변환 및 softmax 함수를 사용하여 디코더 출력을 예측된 다음 토큰 확률로 변환합니다. 모델에서는 [24]와 유사하게 두 개의 임베딩 레이어와 pre-soft max linear 변환 간에 동일한 가중치 행렬을 공유합니다. 임베딩 레이어에서 이러한 가중치를 루트 dmodel로 곱합니다.

**Positional Encoding**

우리 모델은 반복과 컨볼루션을 포함하지 않기 때문에 모델이 시퀀스의 순서를 사용하려면 시퀀스에서 토큰의 상대 또는 절대 위치에 대한 정보를 주입해야 합니다. 이를 위해 인코더 및 디코더 스택의 맨 아래에 있는 입력 임베딩에 "위치 인코딩"을 추가합니다. 위치 인코딩은 임베딩과 동일한 차원의 모델을 가지므로 둘을 합산할 수 있습니다. 학습되고 고정된 위치 인코딩에 대한 많은 선택이 있습니다

이 작업에서는 서로 다른 주파수의 사인 및 코사인 함수를 사용합니다

여기서 위치와 차원입니다. 즉, 위치 부호화의 각 차원은 정현파에 해당한다. 파장은 2에서 10000 2까지 기하학적 진행을 형성합니다. 이 함수를 선택한 이유는 모델이 상대 위치에 참석하는 방법을 쉽게 학습할 수 있다고 가정했기 때문입니다. 고정 오프셋 k에 대해 PEpos k는 PEpos의 선형 함수로 표시될 수 있기 때문입니다. 또한 학습된 위치 임베딩[8]을 대신 사용하여 실험한 결과 두 버전이 거의 동일한 결과를 생성했다는 것을 알 수 있습니다(표 3 행(E) 참조). **우리는 사인파 버전을 선택했습니다. 왜냐하면 모델이 훈련 중에 발생하는 것보다 긴 시퀀스 길이로 외삽할 수 있기 때문입니다.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* 위에 있는 사인파 버전 채택

Why self attention

Self attention을 사용하도록 동기부여 하기 위해 세가지 요구사항 고려한다.

1. 계층당 총 계산 복잡성

2. 필요한 순차 작업의 최소 수로 측정한 병렬 처리