Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* illia.polosukhin@gmail.com

개요

대다수의 시퀀스 변환 모델은 인코더와 디코더를 포함하는 복잡한 반복 또는 컨볼루션 신경망을 기반으로 합니다. 가장 성능이 좋은 모델도 어탠션 메커니즘을 통해 인코더와 디코더를 연결합니다. 우리는 RNN과 컨볼루션을 완전히 배제하고 어탠션 메커니즘만을 기반으로 하는 새로운 간단한 네트워크 아키텍처인 Transformer(트랜스포머)를 제안합니다. 두 가지 기계 번역 작업에 대한 실험은 우리의 모델이 품질이 우수하면서도 병렬처리가 가능하고 교육 시간이 훨씬 더 적게 소요된다는 것을 보여줍니다. 우리 모델은 WMT 2014 영어-독일 번역 작업에서 28.4 BLEU를 달성하여 앙상블을 포함한 기존 최상의 결과를 2BLEU 이상 개선합니다. WMT 2014 영어-프랑스어 번역 작업에서 우리 모델은 8개의 GPU에 대한 3.5일 교육 후 41.8BLEU의 새로운 단일 모델로서는 최고 점수를 받았습니다. 최고의 모델들의 교육 비용에 대해 문학에서 알 수 있습니다. 우리는 트렌스포머가 크고 제한된 교육 데이터로 영어 문법 분석에 성공적으로 적용함으로써 다른 작업에도 잘 일반화된다는 것을 보여줍니다.

1 소개

특히 RNN, LSTM 및 게이트 반복 신경망은 언어 모델링 및 기계 번역과 같은 시퀀스 모델링 및 변환 문제에서 최첨단 접근 방식으로 확고하게 확립되었습니다. 그 이후로 반복되는 언어 모델과 인코더-디코더 아키텍처의 경계를 확장하기 위한 수많은 노력이 계속되고 있습니다. 반복 모델은 일반적으로 입력 및 출력 시퀀스의 기호 위치를 따라 계산을 고려합니다. 계산 시간의 단계에 위치를 정렬하면 이전 숨겨진 상태 ht 1과 위치 t에 대한 입력의 함수로 숨겨진 상태 ht 시퀀스를 생성합니다. 이러한 본질적으로 순차적 특성은 훈련 예제 내의 병렬화를 배제합니다. 이는 메모리 제약으로 인해 예제 간 일괄 처리가 제한되므로 시퀀스 길이가 길어질수록 중요해집니다. 최근의 연구는 인수 분해 속임수와 조건부 계산을 통해 계산 효율성이 크게 향상되는 한편 후자의 경우 모델 성능도 개선되었습니다. 그러나 순차적으로 해야 하는 계산의 근본적인 제약은 남아 있습니다.

어탠션(Attention) 메커니즘은 다양한 작업에서 강력한 시퀀스 모델링 및 변환 모델의 필수적인 부분이 되어 입력 또는 출력 시퀀스의 거리에 관계없이 종속성을 모델링할 수 있습니다. 그러나 몇 가지 경우를 제외하고는 모두 이러한 주의 메커니즘이 반복 네트워크와 함께 사용됩니다.

이 작업에서 우리는 RNN 대신 입력과 출력 사이의 글로벌 의존성을 이끌어내기 위해 어텐션 메커니즘에 전적으로 의존하는 모델 아키텍처인 트랜스포머를 제안합니다. 트렌스포머는 훨씬 더 많은 병렬 연산을 허용하고 8개의 P100 GPU에서 12시간 동안만 교육을 받은 후 번역 품질에서 새로운 첨단 기술에 도달할 수 있습니다.

2 배경

순차 계산을 극복하기 위한 목표는 또한 확장 신경 GPU, ByteNet 및 ConvS2S의 기초를 형성하며, 모든 입력 및 출력 위치에 대해 숨겨진 표현을 병렬로 계산하는 컨볼루션 신경망을 기본 구성 요소로 사용합니다. 이러한 모델에서, 두 개의 임의 입력 또는 출력 위치의 신호를 연관시키는 데 필요한 작업 수는 ConvS2S의 경우 선형으로, ByteNet의 경우 로그로 위치 사이의 거리에 따라 증가합니다. 이것은 먼 위치 사이의 의존성을 배우는 것을 더 어렵게 만듭니다. 트랜스포머에서는 평균 주의 가중 위치로 인해 유효 분해능이 감소하더라도 일정한 수의 작업으로 감소합니다. 이 효과는 3.2장에서 설명한 바와 같이 다중 헤드 어텐션으로 대응합니다.

셀프 어탠션, 혹은 인트라 어텐션은 시퀀스의 표현을 계산하기 위해 단일 시퀀스의 다른 위치와 관련된 어탠션 메커니즘입니다. 셀프 어탠션은 독해, 추상 요약, 텍스트 수반 및 작업 독립적인 문장 표현을 학습하는 것을 포함한 다양한 작업에 성공적으로 사용되었습니다.

엔드 투 엔드 메모리 네트워크(End-to-end memory networks)는 시퀀스 정렬 반복 대신 반복 어탠션 메커니즘을 기반으로 하며 간단한 언어 질문 답변 및 언어 모델링 작업에서 우수한 성능을 발휘하는 것으로 나타났습니다.

우리가 아는 트랜스포머는 시퀀스 정렬 RNN 또는 컨볼루션 없이 입력 및 출력 표현을 계산하는 데 전적으로 셀프 어탠션에 의존하는 첫 번째 변환 모델입니다. 다음 장에서는 트랜스포머에 대해 설명하고, 셀프 어탠션이 무엇인지 환기하며 다른 모델에 비해 가지는 이점에 대해 설명합니다.

3 모델 구조

대부분의 경쟁 신경 시퀀스 변환 모델은 인코더-디코더 구조를 가지고 있습니다. 여기서 인코더는 기호 표현$(x_1, ..., x_n)$의 입력 시퀀스를 연속 표현 $\mathbf{z} = (z_1, ..., z_n)$의 시퀀스에 매핑합니다. $\mathbf{z}$가 주어지면 디코더는 한 번에 한 요소씩 심볼의 출력 시퀀스$(y_1, ..., y_m)$를 생성합니다. 각 단계에서 모델은 자동 회귀 분석을 수행하며, 이전에 생성된 기호를 다음 단계를 생성할 때 추가 입력으로 사용합니다.

트랜스포머는 그림 1의 왼쪽과 오른쪽에 각각 나와 있는 것처럼 인코더와 디코더 모두에 대해 스택형 셀프 어탠션 및 포인트별 완전 연결(FC) 레이어를 사용하여 이 전체 아키텍처를 따릅니다.

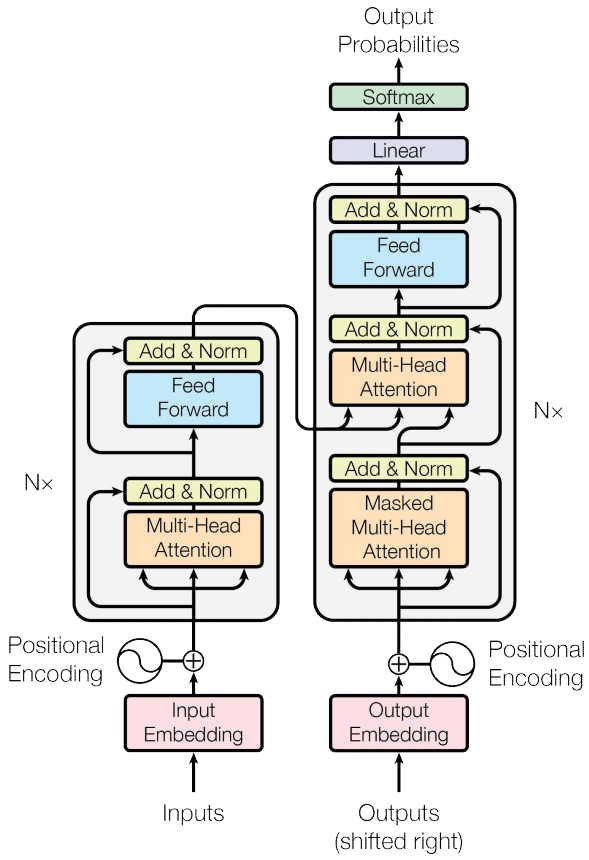

그림 1: 트랜스포머 모델 구조

3.1 인코더-디코더 스택

인코더: 인코더는 N = 6개의 동일한 레이어로 구성된 스택으로 구성됩니다. 각 층에는 두 개의 하위 층이 있습니다. 첫 번째는 다중 헤드 셀프 어탠션 메커니즘이고, 두 번째는 단순하고 위치에 따라 완전히 연결된 피드-포워드 네트워크입니다. 우리는 두 하위 레이어 주위에 잔류 연결을 사용하고 이어서 레이어 정규화를 사용합니다. 즉, 각 하위 계층의 출력은 $\mathbf{LayerNorm}(x + \mathbf{Sublayer}(x))$ 이며, 여기서 하위 계층(x)은 하위 계층 자체에 의해 구현되는 함수입니다. 이러한 잔차 연결을 용이하게 하기 위해 임베딩 계층뿐만 아니라 모델의 모든 하위 계층은 차원 $d_{\text{model}} = 512$의 출력을 생성합니다.

디코더: 디코더는 또한 N = 6개의 동일한 레이어의 스택으로 구성됩니다. 디코더는 각 인코더 레이어에 있는 두 개의 하위 레이어 외에도 세 번째 하위 레이어를 삽입하여 인코더 스택의 출력에 대해 다중 헤드 어탠션을 수행합니다. 인코더와 유사하게, 우리는 각 하위 계층 주위에 잔여 연결을 사용하고 이어서 계층 정규화를 사용합니다. 또한 디코더 스택의 자기 주의 하위 레이어를 수정하여 위치가 후속 위치로 가는 것을 방지합니다. 이러한 마스킹은 출력 임베딩이 한 위치만큼 상쇄된다는 사실과 결합되어 i 위치에 대한 예측이 i보다 작은 위치의 알려진 출력에만 의존할 수 있도록 합니다.

3.2 어탠션

주의 기능은 쿼리, 키, 값 및 출력이 모두 벡터인 출력에 쿼리 및 키-값 쌍을 매핑하는 것으로 설명될 수 있습니다. 출력은 값의 가중치 합으로 계산되며, 여기서 각 값에 할당된 가중치는 해당 키와의 쿼리의 호환성 함수에 의해 계산됩니다.

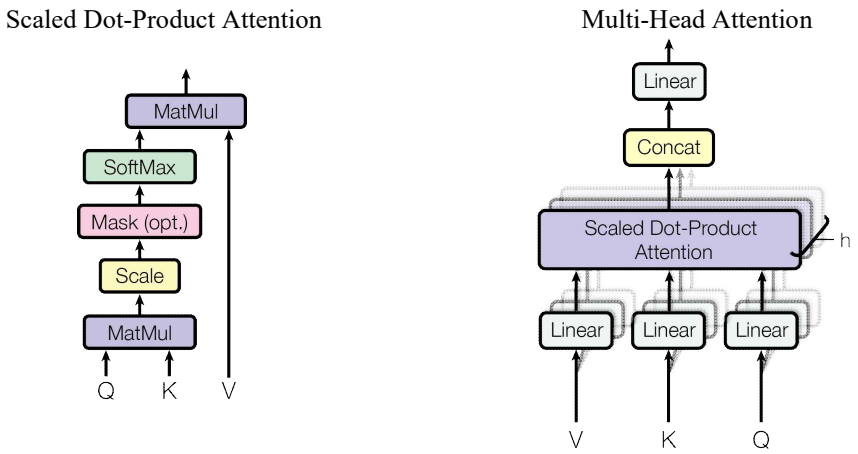

그림 2: (왼쪽) Scaled Dot-Product Attention. (오른쪽) Multi-Head Attention은 병렬로 실행되는 여러 주의 계층으로 구성.

3.2.1 스케일드 닷-프로덕트 어텐션(Scaled Dot-Product Attention)

우리는 특히 주의를 "스케일드 닷-프로덕트 어텐션"(그림 2)라고 부릅니다. 입력은 차원 d_k의 쿼리 및 키, 차원 d_v의 값으로 구성됩니다. 우리는 모든 키로 쿼리의 점 곱을 계산하고 각 값을 나눕니다.

실제로, 우리는 매트릭스 Q로 함께 채워진 일련의 쿼리에 대한 어탠션 함수를 동시에 계산합니다. 키와 값은 행렬 K와 V로 함께 채워집니다. 출력 행렬은 다음과 같이 계산합니다.

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

가장 일반적으로 사용되는 두 가지 어탠션 기능은 애디티브 어탠션과 닷-프로덕트 어텐션입니다. 닷-프로덕트 어텐션은 $\frac{1}{\sqrt{d_k}}$의 스케일링 팩터를 제외하고 우리의 알고리즘과 동일합니다. 애디티브 어탠션은 숨겨진 단일 레이어가 있는 피드-포워드 네트워크를 사용하여 호환성 함수를 계산합니다. 두 개의 이론적인 복잡성은 유사하지만, 고도로 최적화된 행렬 곱셈 코드를 사용하여 구현할 수 있기 때문에 닷-프로덕트 어텐션은 실제로 훨씬 빠르고 공간 효율적입니다.

d_k의 값이 작은 경우 두 메커니즘이 유사하게 수행되지만, 애디티브 어탠션은 d_k의 큰 값에 대한 스케일링 없이 닷-프로덕트 어텐션을 능가합니다. 우리는 d_k의 값이 큰 경우, 닷-프로덕트 어텐션이 크기가 크게 성장하여 소프트맥스 함수를 극도로 작은 기울기 4를 갖는 영역으로 밀어넣을 것으로 예상합니다. 이러한 효과를 상쇄하기 위해, 우리는 닷-프로덕트 어텐션을 $\frac{1}{\sqrt{d_k}}$로 스케일링합니다.

3.2.2 다중 헤드 어탠션

d_{model}-dimensional keys, 값 및 쿼리를 사용하여 단일 주의 기능을 수행하는 대신, 우리는 쿼리, 키 및 값을 각각 d_k, d_k 및 d_v 차원에 서로 다른 학습된 선형 투영으로 선형적으로 투영하는 것이 유익하다는 것을 발견했습니다. 이렇게 투영된 각 버전의 쿼리, 키 및 값에서 주의 기능을 병렬로 수행하여 d_v-dimensional 출력 값을 산출합니다. 그림 2와 같이 연결되고 다시 투영되어 최종 값이 생성됩니다.

다중 헤드 어텐션을 통해 모델은 서로 다른 위치에 있는 서로 다른 표현 하위 공간의 정보를 공동으로 처리할 수 있습니다. 단일 어탠션 헤드에서는 평균화가 이를 억제합니다.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$.

이 작업에서 우리는 h = 8개의 병렬 어탠션 계층 또는 헤드를 사용합니다. 이 각각에 대해 $d_k = d_v = d_\text{model} / h = 64$를 사용합니다. 각 헤드의 치수가 줄어들기 때문에 총 계산 비용은 전체 치수의 단일 헤드 어탠션과 비슷합니다.

3.2.3 우리의 모델에 어탠션을 적용하기

트랜스포머는 다음 세 가지 방법으로 다중 헤드 어텐션을 사용합니다.

- "인코더-디코더 어탠션" 계층에서 쿼리는 이전 디코더 계층에서 나오고 메모리 키와 값은 인코더의 출력에서 나옵니다. 이렇게 하면 디코더의 모든 위치가 입력 시퀀스의 모든 위치에 걸쳐 참석할 수 있습니다. 이것은 시퀀스 간 모델에서 일반적인 인코더-디코더 주의 메커니즘을 모방합니다.
- 인코더에는 셀프 어탠션 계층이 있습니다. 셀프 어탠션 계층에서 모든 키, 값 및 쿼리는 동일한 위치에서 나옵니다. 이 경우 인코더의 이전 계층의 출력입니다. 인코더의 각 위치는 인코더의 이전 계층에 있는 모든 위치에 적용될 수 있습니다.
- 마찬가지로, 디코더의 셀프 어탠션 계층은 디코더의 각 위치가 해당 위치까지 디코더의 모든 위치에 어탠션을 할 수 있도록 합니다. 우리는 자동 회귀 속성을 보존하기 위해 디코더의 왼쪽 정보 흐름을 방지해야 합니다. 우리는 잘못 된 연결에 해당하는 소프트맥스의 입력에 있는 모든 값을 마스킹하여 스케일드 닷-프로덕트 어탠션 내부에서 구현합니다. (그림 2. 참조)

3.3 위치별 피드-포워드 네트워크

어탠션 하위 계층 외에도, 인코더와 디코더의 각 계층에는 완전히 연결된 피드-포워드 네트워크가 포함되어 있으며, 이는 각 위치에 개별적으로 동일하게 적용됩니다. 이것은 ReLU 활성화를 사이에 두고 두 개의 선형 변환으로 구성됩니다.

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

선형 변환은 서로 다른 위치에 걸쳐 동일하지만 계층마다 다른 매개변수를 사용합니다. 이를 설명하는 또 다른 방법은 커널 크기가 1인 두 개의 컨볼루션입니다. 입력과 출력의 차원은 d_model = 512이고 내부 레이어의 치수 d_{ff} = 2048입니다.

3.4 임베딩과 소프트 맥스

다른 시퀀스 변환 모델과 유사하게, 우리는 학습된 임베딩을 사용하여 입력 토큰과 출력 토큰을 차원 d 모델의 벡터로 변환합니다. 우리는 또한 일반적으로 학습된 선형 변환과 소프트맥스 함수를 사용하여 디코더 출력을 예측된 다음 토큰 확률로 변환합니다. 우리 모델에서, 우리는 두 임베딩 계층과 사전 소프트맥스 선형 변환 사이에 동일한 가중치 매트릭스를 공유합니다. 임베딩 레이어에서, 우리는 그 가중치에 $\sqrt{d_{\text{mod}el}}$을 곱합니다.

3.5 포지션 인코딩

우리의 모델은 반복이나 컨볼루션(convolution)을 포함하지 않기 때문에, 모델이 문장의 순서를 기억하기 위해, 우리는 상대적인 위치 또는 절대적인 위치에 대한 정보를 주입해야 합니다.

표 1: 여러 계층 유형에 대한 최대 경로 길이, 계층별 복잡성 및 최소 순차 작업 수. n은 시퀀스 길이, d는 표현 차원, k는 컨볼루션의 커널 크기, r은 제한된 셀프 어탠션에서 이웃의 크기.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

토큰이 시퀀스에 있습니다. 이를 위해 인코더 및 디코더 스택 하단에 있는 입력 임베딩에 "위치 인코딩"을 추가합니다. 위치 인코딩은 임베딩과 동일한 치수 d_{model}를 가지므로 두 개가 합해질 수 있습니다. 학습된 포지션 인코딩과 고정된 포지션 인코딩에는 많은 선택사항이 있습니다.

이 작업에서는 서로 다른 사인 및 코사인 함수를 사용합니다.

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

여기서 pos는 위치이고 I 는 치수입니다. 즉, 포지션 인코딩의 각 차원은 정현상에 해당합니다. 파장은 2π에서 $10000 * 2\pi$까지의 기하학적 진행을 형성합니다. 이 함수를 선택한 이유는 고정 오프셋 k 에 대해 PE_{pos+k}이 PE_{pos}의 선형 함수로 표현될 수 있기 때문에 모델이 상대 위치에 의해 참석하도록 쉽게 배울 수 있기 때문입니다.

또한 학습된 포지션 임베딩을 대신 사용하여 실험한 결과 두 버전이 거의 동일한 결과를 나타냈습니다(표 3 행(E) 참조). 우리는 정현파 버전을 선택했는데, 이는 모델이 훈련 중에 접한 것보다 더 긴 시퀀스 길이를 추론할 수 있기 때문입니다.

4 왜 셀프 어탠션 인가

이 장에서는 일반적인 시퀀스 변환 인코더 또는 디코더에서 숨겨진 레이어와 같이 $x_i, z_i \in R^d$를 사용하여 하나의 가변 길이 기호 표현 시퀀스($x_1, ..., x_n$) 를 동일한 길이의 다른 시퀀스($z_1, ..., z_n$)로 매핑하는 데 일반적으로 사용되는 반복 및 컨볼루션 레이어와 셀프 어탠션 레이어의 다양한 측면을 비교합니다. 우리의 셀프 어탠션 사용에 동기를 부여하기 위해 우리는 세 가지 이상적인 자료를 고려합니다.

하나는 계층당 총 계산 복잡도입니다. 또 다른 방법은 필요한 최소 순차 작업 수로 측정되는 병렬로 처리할 수 있는 계산의 양입니다.

세 번째는 네트워크에서 장거리 종속성 사이의 경로 길이입니다. 장거리 의존성을 배우는 것은 많은 시퀀스 변환 작업에서 핵심 과제입니다. 이러한 종속성을 학습하는 능력에 영향을 미치는 핵심 요소 중 하나는 네트워크에서 전방 및 후방 신호가 통과해야 하는 경로의 길이입니다. 입력과 출력 시퀀스의 위치 조합 사이의 이러한 경로가 짧을수록 장거리 종속성을 학습하는 것이 더 쉽습니다. 따라서 우리는 또한 다른 계층 유형으로 구성된 네트워크에서 두 입력 및 출력 위치 사이의 최대 경로 길이를 비교합니다.

표 1에서 언급한 바와 같이, 셀프 어탠션 계층은 모든 위치를 일정한 수의 순차적으로 실행되는 연산으로 연결하는 반면, RNN 계층은 $O(n)$ 순차 연산을 필요로 합니다. 계산 복잡성의 측면에서, 셀프 어탠션 계층은 시퀀스 길이 n이 표현 차원 d보다 작을 때 RNN 계층보다 빠르며, 이는 워드피스 및 바이트 쌍 표현과 같은 기계 번역에서 최첨단 모델에 의해 사용되는 문장 표현과 가장 자주 발생합니다. 매우 긴 시퀀스를 포함하는 작업에 대한 계산 성능을 향상시키기 위해, 셀프 어탠션은 각 출력 위치를 중심으로 하는 입력 시퀀스에서 크기

r의 주변만 고려하는 것으로 제한될 수 있습니다. 이렇게 하면 최대 경로 길이가 $O(n/r)$로 늘어납니다. 우리는 향후 작업에서 이 접근 방식을 더 조사할 계획입니다.

커널 너비가 $k < n$ 인 단일 컨볼루션 레이어는 모든 입력 및 출력 위치 쌍을 연결하는 것은 아니다. 그렇게 하려면 연속 커널의 경우 $O(n/k)$ 컨볼루션 레이어의 스택이 필요하고, 확장 컨볼루션의 경우 $O(log_k(n))$가 필요하므로 네트워크의 모든 두 위치 사이의 가장 긴 경로의 길이가 증가합니다. 컨볼루션 레이어는 일반적으로 k배만큼 반복 레이어보다 더 비쌉니다. 그러나 분리 가능한 나선형은 복잡도를 $O(k * n * d + n * d^2)$.로 상당히 감소시킵니다. 그러나 $k = n$ 을 사용하더라도 분리 가능한 컨볼루션의 복잡성은 우리가 모델에서 취하는 접근 방식인 셀프 어탠션 레이어와 포인트별 피드-포워드 레이어의 조합과 동일합니다.

부수적인 이점으로서, 셀프 어탠션은 더 해석 가능한 모델을 산출할 수 있습니다. 우리는 모델에서 주의 분포를 검사하고 부록의 예를 제시하고 논의합니다. 개별 어탠션 헤드는 다른 작업을 수행하는 방법을 분명히 배울 뿐만 아니라, 많은 사람들이 문장의 구문 및 의미 구조와 관련된 행동을 보이는 것으로 보입니다.

5 학습

이 장에서는 우리 모델의 학습 방법에 대해 설명합니다.

5.1 학습 데이터와 배칭

우리는 약 450만 개의 문장 쌍으로 구성된 표준 WMT 2014 영어-독일 데이터 세트에 대해 훈련했습니다. 문장은 약 37,000개의 토큰의 공유 소스 대상 어휘를 가진 바이트 쌍 인코딩을 사용하여 인코딩되었습니다. 영어-프랑스어의 경우, 3,600만개의 문장으로 구성된 상당히 큰 WMT 2014 영어-프랑스어 데이터 세트를 사용하고 토큰을 32,000개의 단어 단위 어휘로 분할했습니다. 문장 쌍은 대략적인 시퀀스 길이로 함께 배치되었습니다. 각 교육 배치에는 약 25,000개의 소스 토큰과 25,000개의 대상 토큰이 포함된 문장 쌍 세트가 포함되어 있습니다.

5.2 하드웨어와 학습주기

우리는 8개의 NVIDIA P100 GPU가 있는 하나의 기계에서 모델을 교육했습니다. 논문 전반에 걸쳐 설명된 하이퍼 파라미터를 사용하는 기본 모델의 경우, 각 교육 단계는 약 0.4초가 걸렸습니다. 우리는 기본 모델을 총 10만 단계 또는 12시간 동안 교육했습니다. 대형 모델(표 3의 맨 아래 줄에 설명)의 경우 단계 시간은 1.0초였습니다. 이 대형 모델들은 30만 단계(3.5일) 동안 훈련을 받았습니다.

5.3 옵티마이저

우리는 $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$.의 Adam 옵티마이저를 사용했습니다. 우리는 훈련 과정에 따라 학습률을 다음과 같은 공식에 따라 변화시켰습니다.

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$

이는 첫 번째 *warmup_steps* 훈련 단계에 대해 학습률을 선형적으로 증가시킨 후 단계 번호의 역제곱근에 비례하여 감소시키는 것에 해당합니다. *warmup_steps* = 4000을 사용했습니다.

5.4 정규화

우리는 교육 중에 세 가지 유형의 정규화를 사용합니다.

잔차 드롭아웃 우리는 드롭아웃을 각 하위 계층의 출력에 적용하고, 그것이 하위 계층 입력에 추가되고 정규화되기 전에 적용합니다. 또한, 우리는 인코더와 디코더 스택의 임베딩과 위치 인코딩의 합계에 드롭아웃을 적용합니다. 기본 모형의 경우 P_{drop} = 0.1의 비율을 사용합니다.

표 2: Transformer는 영어-독일어 및 영어-프랑스어 newstest2014 테스트에서 이전 최신 모델보다 적은 비용으로 더 나은 BLEU 점수를 획득함.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

레이블 스무딩 교육 중에 값 ϵ_{ls} = 0.1의 레이블 스무딩을 사용했습니다. 이것은 모델이 더 확신하지 못하는 것을 배우는 단점은 있지만, 정확도와 BLEU 점수를 향상시킵니다.

6 결과

6.1 기계 번역

WMT 2014 영어-독일 번역 대회에서 큰 트랜스포머 모델(표 2의 트랜스포머(대형))은 2.0 BLEU 이상 이전에 보고된 최고의 모델(앙상블 포함)을 능가하여 28.4의 새로운 최첨단 BLEU 점수를 확립했습니다. 이 모델의 구성은 표 3의 맨 아래 줄에 나와 있습니다. 8개의 P100 GPU에 대한 교육에는 3.5일이 걸렸습니다. 심지어 우리의 기본 모델도 경쟁 모델의 교육 비용의 일부만으로 이전에 발표된 모든 모델과 앙상블을 능가합니다.

WMT 2014 영어-프랑스어 번역 작업에서, 우리의 큰 모델은 41의 BLEU 점수를 달성하여 이전 최첨단 모델의 1/4 미만의 교육 비용으로 이전에 발표된 모든 단일 모델을 능가합니다. 영어-프랑스어 간을 위해 훈련된 큰 트랜스포머 모델은 0.3 대신 드롭아웃 비율 P_{drop} = 0.1을 사용했습니다.

기본 모델의 경우 10분 간격으로 작성된 마지막 5개의 체크포인트를 평균하여 얻은 단일 모델을 사용했습니다. 대형 모델의 경우 마지막 20개의 체크포인트를 평균했습니다. 빔 크기가 4이고 길이 패널티 α = 0.6인 빔 검색을 사용했습니다. 이러한 초 매개 변수는 개발 세트에 대한 실험 후에 선택되었습니다. 추론 중 최대 출력 길이를 입력 길이 + 50으로 설정하지만 가능하면 일찍 종료합니다.

표 2는 결과를 요약하고 번역 품질 및 교육 비용을 문헌의 다른 모델 아키텍처와 비교합니다. 우리는 훈련 시간, 사용된 GPU의 수 및 각 GPU의 지속적인 단일 정밀 부동소수점 용량의 추정치를 곱하여 모델을 훈련하는 데 사용되는 부동소수점 연산 수를 추정합니다.

6.2 다양한 모델들

트랜스포머의 다양한 구성 요소의 중요성을 평가하기 위해, 우리는 개발 과정, newsest 2013에서 영어-독일 번역의 성능 변화를 측정하면서 다양한 방식으로 기본 모델을 다양화했습니다. 이전 장에서 설명한 대로 빔 검색을 사용했지만 체크포인트 평균은 사용하지 않았습니다. 우리는 이러한 결과를 표 3에 적어 놓았습니다.

표 3 행(A)에서는 3.2.2 장에 설명된 대로 계산량을 일정하게 유지하면서 어탠션 헤드의 수와 어탠션 키 및 값 치수를 변경합니다. 단일 헤드 어탠션은 최상의 설정보다 0.9BLEU 더 나쁘지만, 헤드가 너무 많으면 품질도 떨어집니다.

표 3: 트렌스포머 구조에 대한 변화. 목록에 없는 값은 기본 모델의 값과 동일. 모든 지표는 영어-독일어 번역 개발 세트, 뉴스 테스트 2013에 기반한 것. 나열된 복잡성은 바이트 쌍 인코딩에 따라 워드 단위이며 워드 단위 복잡성과 비교해서는 안 됨.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

표 4: 트랜스포머는 영어 문법 구문 분석에 잘 일반화 됨.

Parser	**Training**	**WSJ 23 F1**
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

표 3 행(B)에서 어탠션 키 크기 d_k를 줄이면 모델 품질이 저하된다는 것을 관찰합니다. 이는 호환성을 결정하는 것이 쉽지 않으며 닷-프로덕트보다 더 정교한 호환성 기능이 유용할 수 있음을 시사합니다. 우리는 또한 (C) 행과 (D) 행에서 예상대로 큰 모델이 더 낫고, 드랍은 과적합을 피하는 데 매우 도움이 된다는 것을 관찰할 수 있습니다. 행 (E)에서 우리는 사인파 위치 인코딩을 학습된 위치 임베딩으로 교체하고 기본 모델과 거의 동일한 결과를 관찰할 수 있습니다.

6.3 영어 문법 분석

트랜스포머가 다른 작업으로 일반화할 수 있는지 평가하기 위해 영어 문법 구문 분석에 대한 실험을 수행했습니다. 이 작업은 특정 과제를 제시합니다. 출력은 강력한 구조적 제약을 받으며 입력보다 상당히 깁니다. 또한, RNN sequence-to-sequence 모델은 소규모 데이터 체제에서 최첨단 결과를 얻을 수 없었습니다.

Penn Treebank의 WSJ(Wall Street Journal) 부분에서 d_{model} = 1024로 4계층 트랜스포머를 학습시켰는데, 이는 약 40,000개의 학습 문장입니다. 우리는 또한 약 1,700만 문장의 더 큰 고신뢰와 Berkley Parsercorpa를 사용하여 준지도 환경에서 훈련했습니다. WSJ 전용 설정에는 16,000 토큰의 어휘를, semi-supervised 설정에는 32,000 토큰의 어휘를 사용했습니다.

우리는 어탠션과 잔차 드롭아웃(5.4장), 섹션 22 개발 세트의 학습 속도 및 빔 크기를 모두 선택하기 위해 소수의 실험만 수행했으며, 다른 모든 매개 변수는 영어-독일어 기반 번역 모델에서 변경되지 않았습니다. 추론하는 동안 최대 출력 길이를 입력 길이 + 300으로 늘렸습니다. 우리는 WSJ와 semi-supervised 설정 모두에 대해 21과 α = 0.3의 빔 크기를 사용했습니다.

표 4의 결과는 튜닝이 없음에도 불구하고 우리 모델이 놀라울 정도로 잘 수행되어 RNN을 제외하고 이전에 보고된 모든 모델보다 더 나은 결과를 산출한다는 것을 보여줍니다.

RNN sequence-to-sequence 모델과 달리 트랜스포머는 WSJ 40K 문장 훈련 세트에 대해서만 훈련하는 경우에도 Berkeley-Parser를 능가합니다.

7 결론

본 연구에서는 인코더-디코더 구조에서 가장 일반적으로 사용되는 반복 레이어를 다중 헤드 셀프 어텐션으로 대체하여 전적으로 어탠션을 기반으로 하는 첫 번째 시퀀스 변환 모델인 트랜스포머를 제시했습니다.

번역 작업의 경우 트랜스포머는 반복 또는 컨볼루션 레이어를 기반으로 하는 아키텍처보다 훨씬 빠르게 훈련될 수 있습니다. WMT 2014 영어-독일어 및 WMT 2014 영어-프랑스어 번역 작업에서 우리는 새로운 최첨단 기술을 달성합니다. 이전 작업에서 우리의 최고의 모델은 이전에 보고된 모든 앙상블을 능가합니다.

우리는 어탠션 기반 모델의 미래에 대해 흥분하고 있으며 다른 작업에 적용할 계획입니다. 우리는 트랜스포머를 텍스트 이외의 입력 및 출력 양식과 관련된 문제로 확장하고 이미지, 오디오 및 비디오와 같은 대규모 입력 및 출력을 효율적으로 처리하기 위한 로컬 제한 어탠션 메커니즘을 연구할 계획입니다. 세대를 덜 순차적으로 만드는 것은 우리의 또 다른 연구 목표입니다.

모델을 교육하고 평가하는 데 사용한 코드는 https://github.com/tensorflow/tensor2tensor에서 확인할 수 있습니다.

도움을 주신 분들 Nal Kalchbrenner와 Stephan Gouws의 유익한 논평, 수정 및 영감에 감사합니다.

참조

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. of NAACL*, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolu- tional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.

[10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841. ACL, August 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[16] Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In *Advances in Neural Information Processing Systems, (NIPS)*, 2016.

[17] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In *International Conference on Learning Representations (ICLR)*, 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099v2*, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[21] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. *arXiv preprint arXiv:1703.10722*, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[26] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159. ACL, June 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In *Empirical Methods in Natural Language Processing*, 2016.

[28] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 433–440. ACL, July 2006.

[30] Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdi- nov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 434–443. ACL, August 2013.

어탠션 시각화

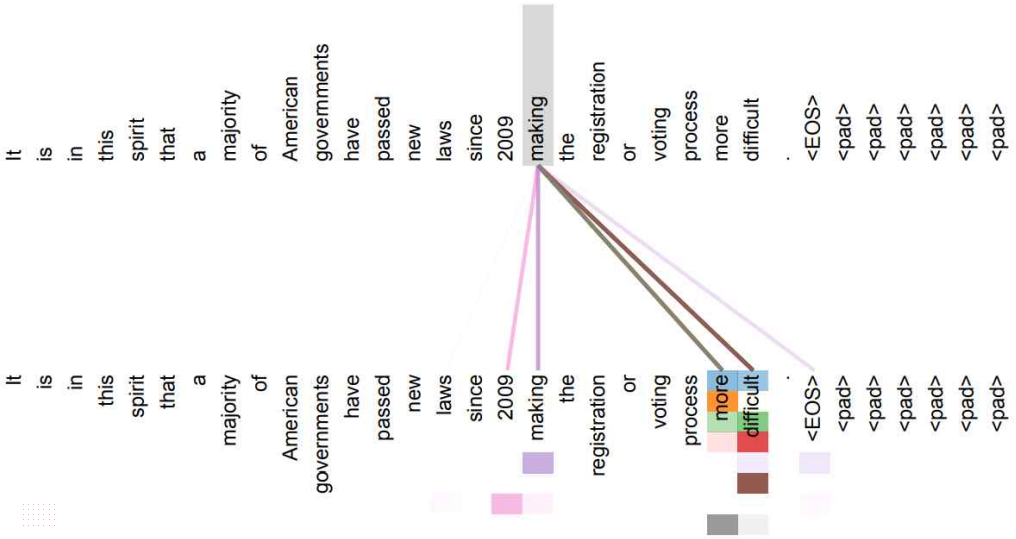

그림 3: 5 ~ 6개의 레이어에서 인코더 자체 주의에서 장거리 의존성을 따르는 주의 메커니즘의 예입니다. 많은 어탠션 헤더들은 'making'이라는 동사의 먼 의존성에 주목하여 'making... more difficult'라는 구절을 완성합니다. 여기서는 'making'이라는 단어에 대해서만 주의를 기울입니다. 다른 색은 다른 머리를 나타냅니다. 가장 잘 보이는 색상입니다.

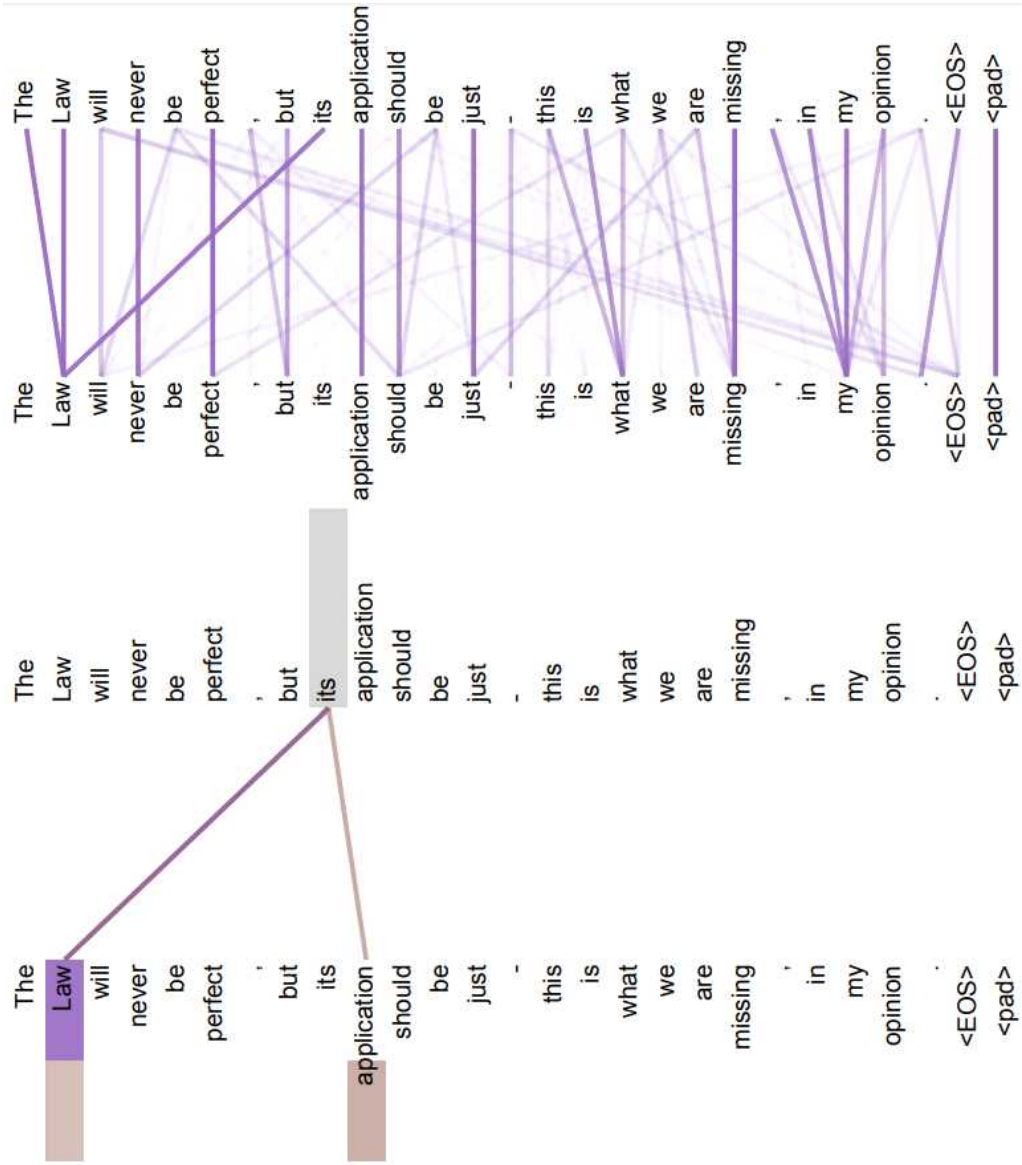

그림 4: 또한 5 ~ 6개의 레이어에 있는 두 개의 어탠션 헤드는 분명히 '대용' 해결법과 관련이 있습니다. 상단 부분은 헤더 5에 대한 완전한 어탠션입니다. 하단은 어탠션 헤드 5와 6에 대한 'its'라는 단어에서 어탠션을 분리합니다. 이 단어에 대한 어탠션은 매우 날카롭습니다.

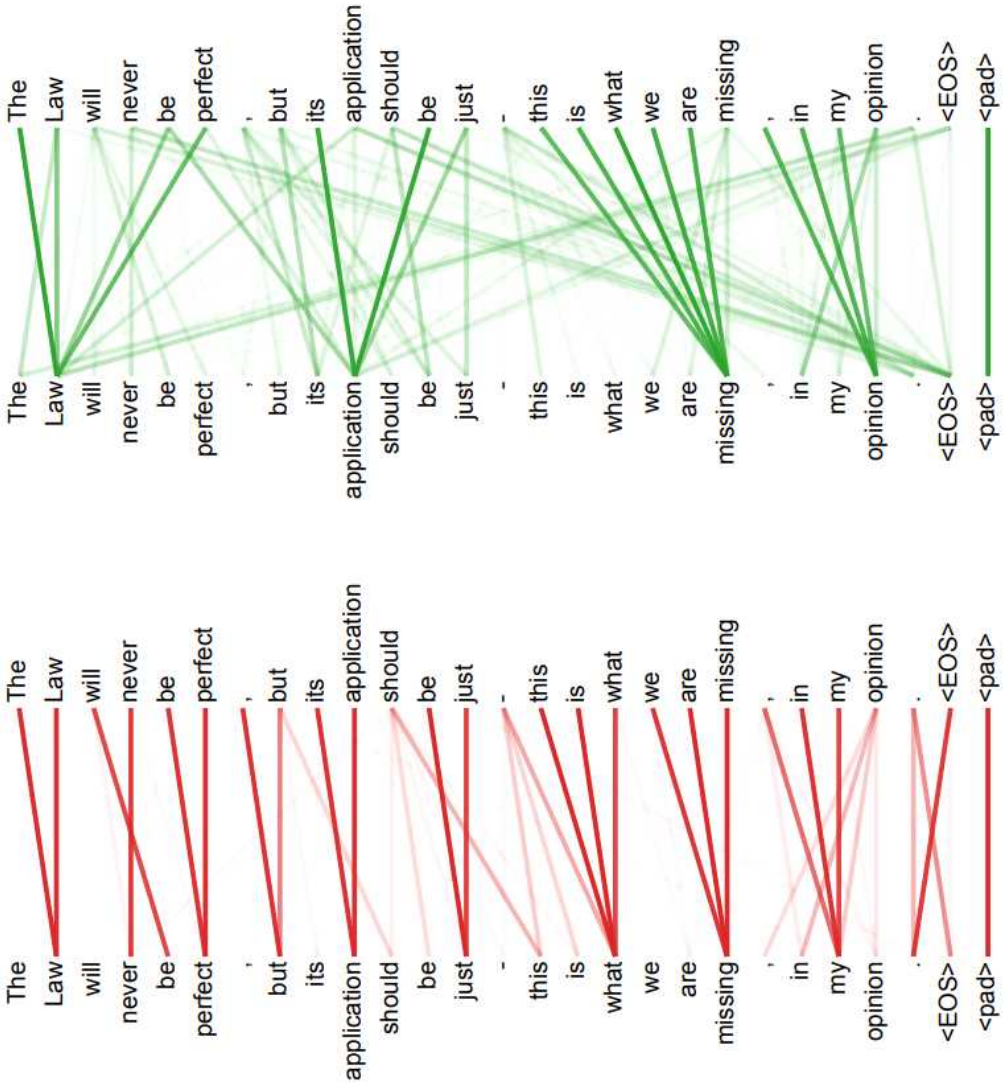

그림 4: 많은 어탠션 헤더들은 문장의 구조와 관련이 있는 것처럼 보이는 행동을 보입니다. 우리는 위의 두 가지 예를 제시합니다. 인코더 셀프 어탠션 레이어 5, 6의 두 개의 다른 헤드에서. 헤더들은 분명히 다른 임무를 수행하는 법을 배웠습니다.