적절한 출전을 명시하는 경우, 구글은 이 논문 내의 표와 그림을 저널리즘 또는 학술적 목적에 한해 복제할 수 있는 권한을ここに 부여합니다.

주의(Attention)만 있으면 충분하다

아시什 바스와니*
구글 브레인
avaswani@google.com

노암 샤지어*
구글 브레인
noam@google.com

니키 파마*
구글 리서치
nikip@google.com

야코부스 우스크로이트*
구글 리서치usz@google.com

Llion Jones*
Google Research
llion@google.com

에이던 N. 고메즈*
토론토대학교
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin*
illia.polosukhin@gmail.com

arXiv:1706.03762v7 [cs.CL] 2 Aug 2023

초록

우세한 순서 변환 모델은 인코더와 디코더를 포함하는 복잡한 재귀적 또는 컨볼루션 신경망에 기반한다. 가장 성능이 높은 모델들은 또한 인코더와 디코더를 주의 메커니즘을 통해 연결한다. 우리는 재귀와 컨볼루션을 완전히 배제하고 오직 주의 메커니즘에만 기반한 새로운 간단한 네트워크 구조인 Transformer를 제안한다. 두 기계 번역 과제에 대한 실험 결과, 이 모델들은 품질 면에서 우위에 있으며 병렬화가 더 용이하며 훈련에 필요한 시간이 현저히 적게 소요됨을 보여준다. 우리 모델은 WMT 2014 영어-독일어 번역 과제로 28.4의 BLEU 점수를 달성하여, 엔블블(집합)을 포함한 기존 최상 결과보다 2 BLEU 이상 향상시켰다. WMT 2014 영어-프랑스어 번역 과제에서는, 문헌에 reported된 최고 수준의 모델들의 훈련 비용에 비해 매우 작은 부분인 8개의 GPU에서 3.5일간 훈련한 후, 우리 모델은 41.8의 새로운 단일 모델 최첨단 BLEU 점수를 기록하였다. 우리는 Transformer가 큰 데이터와 제한된 훈련 데이터를 모두 사용하여 영어 구문 구성 분석에 성공적으로 적용됨으로써 다른 과제에도 잘 generalizes(적응/확장)함을 보여준다.

*동등 기여. 나열 순서는 무작위임. 야코브(R Jakob)는 RNN을 self-attention으로 대체할 것을 제안하고 이 아이디어를 평가하기 위한 노력을 시작함. 아시시(Ashish)는 일리아(Illia)와 함께 첫 번째 Transformer 모델을 설계 및 구현했으며, 이 작업의 모든 측면에 핵심적으로 관여함. 노암(Noam)은 스케일된 점곱 주의机制(scaled dot-product attention), 멀티 헤드 주의机制(multi-head attention), 매개변수 없는 위치 표현을 제안하며 거의 모든 세부 사항에 관여한 또 다른 주요 인물이 됨. 니키(Niki)는 우리 원본 코드베이스와 tensor2tensor에서 수없이 많은 모델 변형을 설계, 구현, 튜닝 및 평가함. 리온(Llion) 또한 novel model variants로 실험했으며, 초기 코드베이스와 효율적인 추론 및 시각화를 담당함. 루카시(Lukasz)와 에 (Aidan)은 수많은 긴 날들을 보내 여러 부분을 설계하고 our earlier codebase를 대체하는 tensor2tensor를 구현하여 연구 결과를 크게 향상시키고 연구 속도를 획기적으로 가속화함.

†구글 브레인 재직 시 수행된 연구.
구글 리서치 재직 중 수행된 연구.

제31회 신경 정보 처리 시스템Conference (NIPS 2017), 미국 캘리포니아주 롱비치

1 서론

재귀 신경망, 특히 장단기 메모리 [13] 및 게이트드 재귀 [7] 신경망은 언어 모델링 및 기계 번역 [35, 2, 5]과 같은 시퀀스 모델링 및 전도 문제에서 최첨단 기법으로 확고히 자리 잡았습니다. 이후 수많은 연구가 재귀 언어 모델 및 인코더-디코더 아키텍처의 한계를 넓혀 나가는 데 기여해 왔습니다 [38, 24, 15].

재귀 모델은 일반적으로 입력 및 출력 시퀀스의 기호 위치를 따라 연산을 분해한다. 이러한 위치를 계산 시간의 단계에 정렬하여, 그들은 이전 숨겨진 상태 h_{t-1}과 위치 t에 대한 입력을 함수로 하여 숨겨진 상태 h_t의 시퀀스를 생성한다. 이러한 본질적인 순차적 성질은 훈련 예제 내 병렬화를 불가능하게 하며, 이는 메모리 제약이 예제 간 배치(batching)를 제한하기 때문에 더 긴 시퀀스 길이에서 핵심적인 문제가 된다. 최근 연구는 [21]의 분해 기법과 조건부 연산 [32]을 통해 계산 효율성에서 상당한 개선을 이루었으며, 후자의 경우에는 모델 성능 또한 향상시켰다. 그러나 순차적 연산에 대한 근본적인 제약은 여전히 남아 있다.

주의 메커니즘은 다양한 작업에서 설득력 있는 시퀀스 모델링 및 변환 모델의 필수적인 부분이 되었으며, 입력 또는 출력 시퀀스 내 거리와 무관하게 종속성을 모델링할 수 있게 합니다 [2, 19]. 그러나 몇몇 예외 [27]를 제외하고는, 이러한 주의 메커니즘은 대부분 순환 네트워크와 결합되어 사용됩니다.

이 연구에서 우리는 순환을 배제하고 대신 입력과 출력 간의 전역 종속성을 포착하기 위해注意力 메커니즘에만 의존하는 모델 아키텍처인 Transformer를 제안한다. Transformer는 병렬化处理을 크게 증대시킬 수 있으며, 8대의 P100 GPU에서 최소 12시간만 훈련하여 번역 품질 분야에서 새로운 최상수준(new state of the art)에 도달할 수 있다.

2 배경

순차 연산의 감축 목표는 Extended Neural GPU [16], ByteNet [18], ConvS2S [9]의 기초를 이루며, 이 모든 모델은 기본 구성 요소로卷积神经网络를 사용하며, 모든 입력 및 출력 위치의 숨겨진 표현을 병렬로 계산한다. 이러한 모델에서 두 임의의 입력 또는 출력 위치 간의 신호를 연결하는 데 필요한 연산 횟수는 위치 간 거리에 따라 증가하는데, ConvS2S인 경우 선형적으로, ByteNet인 경우 로그함수적으로 증가한다. 이는远距离 위치 사이의 종속성을 학습하는 것을 더 어렵게 만든다 [12]. Transformer에서는 이를 일정한 연산 횟수로 줄였으나, 이는attention-weighted positions를 평균화하여 유효한 해상도가 감소한다는 대가를 치르는 것이다. 우리는 3.2절에서 설명하듯 Multi-Head Attention를 통해 이러한 효과를 상쇄한다.

셀프-attention(때로는 인트라-attention이라 불림)은 시퀀스의 다른 위치들 간의 관계를 설정하여 해당 시퀀스의 표현을 계산하기 위한 attention 메커니즘이다. 셀프-attention은 독해 이해, 추상적 요약, 텍스트 함의 및 태스크 독립적 문장 표현 학습 등 다양한 작업에서 성공적으로 활용되어 왔다 [4, 27, 28, 22].

엔드 투 엔드 메모리 네트워크는 시퀀스 정렬 반복 대신 순환 주의 메커니즘에 기반하며, 간단한 언어 질문 응답 및 언어 모델링 작업에서 우수한 성능을 보이는 것으로 나타났다 [34].

저자의 지식에 의하면, Transformer는 서열 정렬 RNN이나 합성곱을 사용하지 않고 입력 및 출력의 표현을 계산하기 위해 자기 주의 Mechanism(self-attention)에만 의존하는 최초의 전도 모델입니다. 다음 섹션에서 우리는 Transformer를 설명하고, 자기 주의 Mechanism(self-attention)의 필요성을 논하며, 이를 [17, 18] 및 [9]와 같은 모델에 비해優位性(advantages)을 논의할 것입니다.

3 모델 아키텍처

대부분의 경쟁력 있는 신경 시퀀스 전도 모델은 인코더-디코더 구조를 가진다 [5, 2, 35]. 여기서 인코더는 기호 표현의 입력 시퀀스 $(x_1, \ldots, x_n)$을 연속적 표현 $\mathbf{z} = (z_1, \ldots, z_n)$의 시퀀스로 매핑한다. $\mathbf{z}$가 주어지면, 디코더는 기호의 출력 시퀀스 $(y_1, \ldots, y_m)$을 한 요소씩 생성한다. 각 단계에서 모델은 자동 회귀적(auto-regressive)이며 [10], 다음을 생성할 때 이전에 생성된 기호를 추가 입력으로 사용한다.

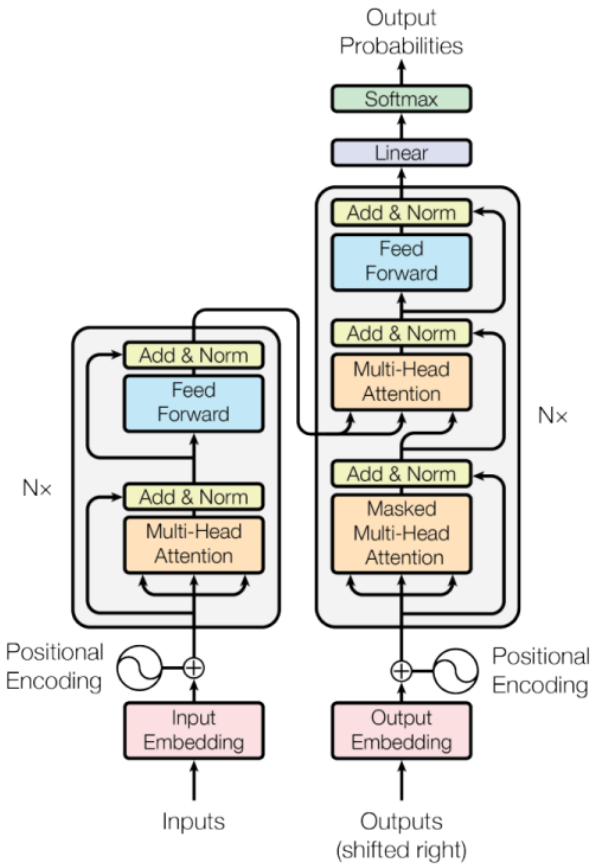


그림 1: 트랜스포머 모델 아키텍처.

Transformer는 인코더와 디코더 모두에 대해 쌓인 자기 주의력(self-attention)과 점별 전 연결층(point-wise, fully connected layers)을 사용하는 이 전체 아키텍처를 따르며, 이는 각각 1번 도서의 좌측 및 우측 부분에 나타납니다.

3.1 인코더 및 디코더 스택

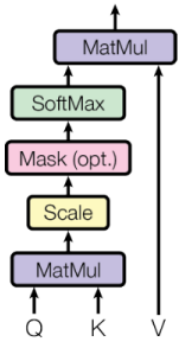
인코더: 인코더는 $N=6$개의 동일한 레이어로 구성된 스택으로 이루어져 있다. 각 레이어는 두 개의 서브레이어를 갖는다. 첫 번째 서브레이어는 멀티 헤드 셀프 어텐션 메커니즘이며, 두 번째는 위치별 전 연결 피드포워드 네트워크이다. 우리는 두 서브레이어 주변에 잔차 연결(residual connection) [11]을 적용하고, 그 다음 계층 정규화(layer normalization) [1]을 수행한다. 즉, 각 서브레이어의 출력은 $\mathrm{LayerNorm}(x+\mathrm{Sublayer}(x))$이며, 여기서 $\mathrm{Sublayer}(x)$는 서브레이어 자체가 구현한 함수를 의미한다. 이러한 잔차 연결을 위해 모델 내의 모든 서브레이어 및 임베딩 레이어는 $d_{\mathrm{model}}=512$ 차원의 출력을 산출한다.

디코더: 디코더도 $N=6$개의 동일한 레이어로 구성된 스택으로 이루어져 있습니다. 인코더 레이어의 두 서브레이어 외에도, 디코더는 인코더 스택의 출력에 대해 멀티 헤드 어텐션을 수행하는 세 번째 서브레이어를 삽입합니다. 인코더와 마찬가지로, 우리는 각 서브레이어 주변에 잔차 연결(residual connections)을 적용하고, 그 다음 레이어 정규화(layer normalization)를 수행합니다. 또한 디코더 스택의 셀프 어텐트(sub-layer)를 수정하여 특정 위치가 이후 위치들(subsequent positions)에 어텐트(attention)를 할 수 없도록 합니다. 이 마스킹(masking)과 출력 임베딩(output embeddings)이 한 위치만큼 오프셋(offset)되어 있다는 사실은, 위치 i에 대한 예측이 오직 i보다 작은 위치들의已知된 outputs에만 의존하도록 보장합니다.

3.2 주의력

주의 함수는 쿼리와 키-값 쌍의 집합을 출력으로 매핑하는 것으로 설명할 수 있으며, 여기서 쿼리, 키, 값, 그리고 출력은 모두 벡터이다. 출력은 가중 합으로 계산된다.

스케일드 도트-프로덕트 어텐션



Multi-Head Attention

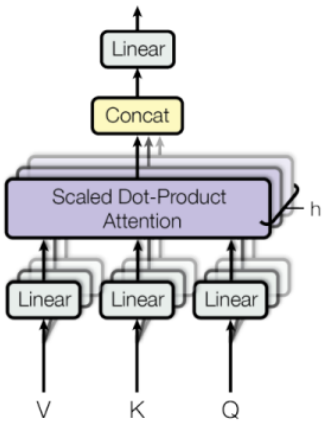


그림 2: (왼쪽) 스케일된 점곱attention. (오른쪽) 멀티헤드attention은 병렬로 작동하는 여러 attention 레이어로 구성됩니다.

값들 중 일부이며, 각 값에 할당된 가중치는 해당 키와 쿼리 간의 호환성 함수에 의해 계산됩니다.

3.2.1 스케일드 도트-프로덕트 어텐션

우리의 특정.attention을 "스케일된 점积.attention" (Figure 2)라고 부릅니다. 입력은 차원 d_k의 쿼리 및 키와 차원 d_v의 값으로 구성됩니다. 우리는 쿼리와 모든 키의 점积을 계산한 뒤, 각값을 $\sqrt{d_k}$로 나누고 softmax 함수를 적용하여 값에 대한 가중치를 얻습니다.

실제로는, 일련의 쿼리를 동시에 처리하기 위해 행렬 Q에-Pack하여注意力 함수를 계산합니다. 키와 값 또한 행렬 K 및 V에-Pack하여 함께 묶습니다. 우리는 출력의 행렬을 다음과 같이 계산합니다:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

가장 널리 사용되는 두 가지attention 함수는 가산 attention [2]과 점곱(multiplicative) attention입니다. 점곱 attention은 $\frac{1}{\sqrt{d_k}}$의 스케일링 인자를 제외하면 우리 알고리즘과 동일합니다. 가산 attention은 단일 숨겨진 계층을 가진 전방 연결 네트워크를 사용하여 호환성 함수를 계산합니다. 두 방법은 이론적 복잡성이 유사하지만, 점곱 attention은 고도로 최적화된 행렬 곱셈 코드를 통해 구현할 수 있어 실제로는 훨씬 빠르고 공간 효율적입니다.

d_k의 값이 작을 때 두 메커니즘은 유사하게 작동하지만, 가산 주의Mechanism(additive attention)은 d_k가 클 때 스케일링 없는 내적 곱 주의Mechanism(dot product attention)보다 우월하다 [3]. 우리는 d_k의 값이 클 때 내적 곱의 크기가 커져 softmax 함수를 그 기울기가 극도로 작은 영역으로 밀어 넣는다고 추측한다 4. 이 효과를 상쇄하기 위해, 우리는 내적 곱을 $\frac{1}{\sqrt{d_k}}$으로 스케일링한다.

3.2.2 멀티 헤드 어텐션

d_{model} 차원의 키, 값, 쿼리를 사용하는 단일 주의 함수를 수행하는 대신, 쿼리, 키 및 값을 h回, 서로 다른 학습된 선형 투영을 통해 각각 d_k, d_k 및 d_v 차원으로 선형 투영하는 것이 유리함을 발견했습니다. 쿼리, 키 및 값의 이러한 투영된 버전에 대해 병렬로 주의 함수를 수행하여 d_v 차원의 결과를 얻습니다.

[4]점积의 크기가 커지는 이유를 설명하기 위해, q와 k의 성분이 평균 0, 분산 1인 독립적인 확률변수라고 가정하자. 그러면 그들의 점积 $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$는 평균 0과 분산 d_k를 가진다.

출력 값들. 이들은 연결된 후 다시 투사되어, Figure 2에 묘사된 바와 같이 최종 값들을 산출한다.

멀티 헤드 어텐션은 모델이 서로 다른 위치에서 다양한 표현 서브스페이스의 정보를 동시에 주시할 수 있게 합니다. 단일 어텐션 헤드를 사용할 경우, 평균화 과정이 이를 방해합니다.

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

여기서 투사 행렬 $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ 및 $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ 는 매개변수 행렬이다.

이 연구에서 우리는 $h = 8$개의 병렬attention 레이어, 즉 헤드(heads)를 사용한다. 각 헤드에 대해 $d_k = d_v = d_{\text{model}}/h = 64$ 를 적용한다. 각 헤드의 차원이 축소되므로, 총 계산 비용은 완전한 차원을 가진 단일 헤드attention과 유사하다.

3.2.3 우리 모델에서의 어텐션 적용

Transformer는 세 가지 다른 방식으로 멀티 헤드 어텐션을 사용합니다:

"엔코더-디코더 어텐션" 계층에서 쿼리는 이전 디코더 계층에서 오고, 메모리 키와 값은 엔코더의 출력에서 온다. 이를 통해 디코더의 모든 위치가 입력 시퀀스의 모든位置上에 어텐션을 할 수 있다. 이는 [38, 2, 9]와 같은 시퀀스-투-시퀀스 모델의 전형적인 엔코더-디코더 어텐션 메커니즘을 모방한다.

인코더는 셀프 어텐션 레이어를 포함한다. 셀프 어텐션 레이어에서 키, 값, 쿼리는 모두 같은 위치에서 오는데, 이 경우 인코더의 이전 레이어 출력에서 온다. 인코더의 각 위치는 인코더의 이전 레이어의 모든 위치에 어텐트할 수 있다.

similarly, decoder의 self-attention 레이어는 각 위치가 그 위치와 그 이전의 모든 위치까지 주의를 기울일 수 있게 합니다. 자동 회귀 성질을 보존하기 위해 Decoder 내의 좌측 정보 흐름을 차단해야 합니다. 우리는 이를scaled dot-product attention 내부에서 구현하는데, softmax의 입력에 해당하는 불법 연결에 대응하는 모든 값($-\infty$)을 마스킹(설정)함으로써 수행합니다. Figure 2를 참고하십시오.

3.3 위치별 전방 피드-forward 네트워크

주의 하위 계층 외에, 엔코더와 디코더의 각 레이어는 Fully Connected 피드포워드 네트워크를 포함하며, 이는 각 위치별로 개별적이고 동일하게 적용됩니다. 이 네트워크는 그 사이에 ReLU 활성화 함수가 있는 두 개의 선형 변환으로 구성됩니다.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

선형 변환은 서로 다른 위치에서 동일하게 적용되지만, 레이어마다 매개변수는異합니다. 이를 두 가지 커널 크기가 1인 컨볼루션으로 설명할 수도 있습니다. 입력 및 출력의 차원은 $d_{\text{model}} = 512$이며, 내부 레이어의 차원은 $d_{ff} = 2048$입니다.

3.4 임베딩과 소프트맥스

다른 시퀀스 전도 모델과 마찬가지로, 우리는 입력 토큰과 출력 토큰을 d_{model} 차원의 벡터로 변환하기 위해 학습된 임베딩을 사용한다. 또한 디코더 출력을 예측된 다음 토큰 확률로 변환하기 위해 일반적인 학습된 선형 변환 및 softmax 함수를 사용한다. 본 모델에서는 두 임베딩 계층과 pre-softmax 선형 변환 사이에서 동일한 가중치 행렬을 공유하며, 이는 [30]과 유사하다. 임베딩 계층에서 우리는 해당 가중치에 $\sqrt{d_{\text{model}}}$ 를 곱한다.

표 1: 서로 다른 레이어 유형에 대한 최대 경로 길이, 레이어별 복잡도 및 순차 연산의 최소 개수. n은 시퀀스 길이, d는 표현 차원, k는 컨볼루션의 커널 크기, r은 제한된 자기 주의机制(attention)에서의 근접 영역 크기를 나타낸다.

레이어 유형	레이어당 복잡도	순차적 연산	최대 경로 길이
자기 주의 (Self-Attention)	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
순환적	$O(n \cdot d^2)$	$O()$	$O(n)$
합성곱 (Convolutional)	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log$□$(n))$
자기 주의 (제한적)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 위치 인코딩

우리의 모델에는 재귀와卷积이 포함되어 있지 않으므로, 모델이 서열의 순서를 활용할 수 있도록 서열 내 토큰의 상대적 또는 절대적 위치에 대한 정보를 주입해야 합니다. 이를 위해 엔코더 및 디코더 스택 하단의 입력 임베딩에 "위치 인코딩"을 추가합니다. 위치 인코딩은 임베딩과 동일한 차원 d_{model}을 가지므로 두 가지를 더할 수 있습니다. 위치 인코딩의 선택 방법은 다양하며, 학습되는 방식과 고정되는 방식으로 나뉩니다 [9].

이 연구에서는 서로 다른 주파수의 사인 및 코사인 함수를 사용한다:

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

pos는 위치를, i는 차원을 나타낸다. 즉, 위치 인코딩의 각 차원은 사인파에 대응한다. 파장은 2π에서 $10000 \cdot 2\pi$까지 기하급수적 배열을 이룬다. 우리는 이 함수를 선택한 이유는 모델이 상대적 위치에 대한 주의를 쉽게 학습할 수 있을 것으로 가설했기 때문이며, 임의의 고정된 오프셋 k에 대해 PE_{pos+k}는 PE_{pos}의 선형 함수로 표현될 수 있기 때문이다.

또한 대안으로 학습된 위치 인코딩[9]을 적용해 보았으나, 두 버전이 거의 동일한 결과를 내는 것을 확인했습니다(표 3 행 (E) 참조). 우리는 모델이 훈련 중 접한 시퀀스 길이보다 더 긴 길이에 대해 외삽(extrapolate)할 수 있을 것으로 기대되어 사인파(sinusoidal) 버전을 선택하였습니다.

4 자주 Attention(주의)이 필요한 이유

이 섹션에서 우리는 기호 표현의 변수 길이 서열 $(x_1, \ldots, x_n)$을 동일한 길이의 다른 서열 $(z_1, \ldots, z_n)$로 매핑하는 데 일반적으로 사용되는 순환 및 합성층과 자기 주의(self-attention) 계층의 다양한 측면을 비교한다. 여기서 $x_i, z_i \in \mathbb{R}^d$이며, 이는 일반적인 서열 전도 인코더 또는 디코더의 숨겨진 계층에 해당한다.我们自己注意를 사용하는 동기를 부여하기 위해, 우리는 세 가지 요구 사항을 고려한다.

하나는 계층별 총 계산 복잡도이다. 다른 하나는 필요한 최소 순차 연산 수로 측정되는 병렬화 가능한 계산량이다.

세 번째는 네트워크 내 장거리 의존성 사이의 경로 길이이다. 장거리 의존성을 학습하는 것은 많은 시퀀스 전도 작업에서 핵심적인 과제로 남아 있다. 이러한 의존성을 학습하는 능력에 영향을 미치는 주요 요인 중 하나는 프로그래밍 및 백워드 신호가 네트워크를 통과해야 하는 경로의 길이이다. 입력 및 출력 시퀀스의 위치 조합 간 이러한 경로가 짧을수록 장거리 의존성을 학습하기가 쉬워진다 [12]. 따라서 우리는 서로 다른 레이어 유형으로 구성된 네트워크의 임의의 두 입력 및 출력 위치 사이의 최대 경로 길이를 비교한다.

Table 1에서 언급한 바와 같이, 셀프-어텐션 레이어는 순차적으로 실행되는 연산의 수를 일정하게 유지하며 모든 위치를 연결하지만, 재귀 레이어는 $O(n)$개의 순차적 연산을 필요로 한다. 계산 복잡도 측면에서 시퀀스 길이가 일정 수준에 도달하면 셀프-어텐션 레이어가 재귀 레이어보다 빠르다.

길이 n이 표현 차원 d보다 작은 경우가 가장 흔한데, 이는 기계 번역의 최첨단 모델에서 사용되는 문장 표현, 예를 들어 word-piece [38] 및 byte-pair [31] 표현에서 그렇다. 매우 긴 서열을 다루는 작업에 대한 계산 성능을 향상시키기 위해, self-attention은 respective 출력 위치를 중심으로 하는 크기 r의 인접 영역만 고려하도록 제한할 수 있다. 这样的话 maximum path length는 $O(n/r)$로 증가한다. 우리는 향후 연구에서 이 접근법을 더 깊이 조사할 계획이다.

커널幅 $k < n$ 인 단일 합성층은 모든 입력 및 출력 위치 쌍을 연결하지 못한다. 연속 커널의 경우 이를 위해 $O(n/k)$ 개의 합성층 스택이 필요하며, 확장 합성의 경우 $O(\log_k(n))$ 개가 필요하다 [18]. 이는 네트워크 내 임의의 두 위치 사이의 최장 경로 길이를 증가시킨다. 합성층은 일반적으로 재귀층보다 k 배 정도 비용이 많이 든다. 그러나 가분 합성(separable convolutions) [6]은 복잡도를 크게 줄여 $O(k \cdot n \cdot d + n \cdot d^2)$ 로 만든다.即便如此, $k = n$ 인 경우에도 가분 합성의 복잡도는 자기 주의(self-attention) 층과 지점별 피드포워드(layer)의 결합과 동일하며, 이것이 우리 모델에서 채택하는 방법이다.

부수적인 이점으로, 셀프 어텐션은 더 해석 가능한 모델을 생성할 수 있다. 우리는 모델의 어텐션 분포를 조사하며 부록에서 예시를 제시하고 논의한다. 개별 어텐션 헤드가 서로 다른 작업을 수행하도록 명확하게 학습하는 것뿐만 아니라, 많은 헤드가 문장의 구문 및 의미 구조와 관련된 행동을 보이는 것으로 보인다.

5 학습

이 섹션은 우리 모델의 훈련 체계를 설명한다.

5.1 학습 데이터 및 배치

우리는 약 450만 문장 쌍으로 구성된 표준 WMT 2014 영어-독일어 데이터셋을 사용하여 학습하였다. 문장은 공유된 소스-타겟 어휘가 약 37,000개의 토큰인 바이트-페어 인코딩(byte-pair encoding) [3]을 사용하여 인코딩되었다. 영어-프랑스어의 경우, 약 3,600만 문장으로 구성된 훨씬 더 큰 WMT 2014 영어-프랑스어 데이터셋을 사용하였으며, 토큰을 32,000개 단어 조각(word-piece) 어휘로 분할하였다 [38]. 문장 쌍은 근사한 시퀀스 길이에 따라 배치(batch)되었다. 각 학습 배치는 약 25,000개의 소스 토큰과 25,000개의 타겟 토큰을 포함하는 문장 쌍 세트를 포함하였다.

5.2 하드웨어 및 스케줄

우리는 8개의 NVIDIA P100 GPU가 장착된 단일 기계에서 모델들을 훈련시켰다. 본 논문 전반에 서술된 하이퍼파라미터를 사용한 기본 모델의 경우, 각 훈련 단계당 약 0.4초가 소요되었다. 기본 모델들은 총 100,000단계 또는 12시간 동안 훈련되었다. 대형 모델들(표 3의 하단 줄에 서술됨)의 경우, 단계당 소요 시간은 1.0초였다. 대형 모델들은 300,000단계(3.5일) 동안 훈련되었다.

5.3 최적화기

우리는 $\beta_1 = 0.9$, $\beta_2 = 0.98$ 및 $\varepsilon = 10^{-9}$인 Adam 최적화기 [20]를 사용하였다. 학습률은 다음 공식에 따라 훈련 과정 전체에 걸쳐 변경하였다:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5}) \tag{3}$$

이는 첫 $warmup_{steps}$ 학습 단계 동안 학습률을 선형적으로 증가시키고, 이후에는 단계 수의 제곱근 역수에 비례하여 감소시키는 것에 해당합니다. 우리는 $warmup_{steps} = 4000$을 사용했습니다.

5.4 정칙화

우리는 학습 과정에서 세 가지 유형의 정칙화를 적용합니다:

표 2: Transformer은 훈련 비용의 일부로만 영어-독어 및 영어-프랑스어 newstest2014 테스트에서 이전 최첨단 모델보다 높은 BLEU 점수를 달성합니다.

모델	BLEU		학습 비용 (FLOPs)	
	영어-독일어	영어-프랑스어	영어-독일어	영어-프랑스어
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 앙상블 [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL 앙상블 [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S 앙상블 [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (기본 모델)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (대형)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

잔차 드롭아웃 각 서브레이어의 출력에 드롭아웃 [33]을 적용하여, 이를 서브레이어 입력에 더하고 정규화하기 전에 수행합니다. 또한 인코더 및 디코더 스택 모두에서 임베딩과 위치 인코딩의 합에 대해 드롭아웃을 적용합니다. 기본 모델의 경우, $P_{drop} = 0.1$의 비율을 사용합니다.

라벨 스무딩 훈련 중, 우리는 값 $\varepsilon_{ls} = 0.1$인 라벨 스무딩을 적용하였다 [36]. 이는 모델이 더 불확실하게 학습되도록 하여 퍼플렉스(perplexity)를 높이지만, 정확도와 BLEU 점수는 향상시킨다.

6. 결과

6.1 기계 번역

WMT 2014 영어-독어 번역 과제에서, Table 2의 대형 트랜스포머 모델(Transformer(big))은 엔블블을 포함한 Previously 보고된 모든 모델보다 2.0 BLEU를 초과하여 우위이며, 새로운 최첨단 BLEU 점수인 28.4를 달성하였다. 이 모델의 구성은 Table 3의 하단 행에 나열되어 있다. 훈련은 8대의 P100 GPU에서 3.5일이 소요되었다.是我们的 기본 모델조차도 경쟁 모델들의的训练 비용의 일부분으로, previously published된 모든 모델과 엔블블을 초월한다.

WMT 2014 영어-프랑스어 번역 과제를 대상으로, 본 대형 모델은 BLEU 점수 41.0을 달성하여 기존에 발표된 모든 단일 모델보다 우수하며, 이전 최첨단 모델의 학습 비용의 1/4 미만으로 achievement하였다. 영어-프랑스어 번역을 위해 훈련된 Transformer(대형) 모델은 0.3 대신 드롭아웃율 $P_{drop} = 0.1$을 사용하였다.

베이스 모델의 경우, 10분 간격으로 작성된 마지막 5개의 체크포인트를 평균화하여 얻은 단일 모델을 사용했습니다. 대형 모델의 경우, 마지막 20개의 체크포인트를 평균화했습니다. 비암搜索(beam search)를 사용했으며, 비암 크기(beam size)는 4, 길이 페널티 α는 0.6 [38]로 설정했습니다. 이러한 하이퍼파라미터들은 개발셋(development set)에 대한 실험을 통해 선정되었습니다. 추론 시 최대 출력 길이는 입력 길이 + 50으로 설정하되, 가능하면 조종 종료(early termination)를 적용했습니다 [38].

표 2는 우리 결과를 요약하고, 문헌에 제시된 다른 모델 아키텍처와 비교하여 우리 번역의 품질과 훈련 비용을 보여줍니다. 우리는 각 GPU의 지속 가능한 단일 정밀도 부동소수점 성능 추정을 사용된 GPU의 수 및 훈련 시간에 곱하여 모델을 훈련하는 데 사용된 부동소수점 연산의 개수를 추정합니다 [5].

6.2 모델 변형

Transformer의 구성 요소별 중요도를 평가하기 위해, 우리는 기본 모델을 다양한 방식으로 변형하여 영어-독일어 번역에서의 성능 변화를 측정하였다.

[5] K80, K40, M40, P100에 대해서는 각각 2.8, 3.7, 6.0, 9.5 TFLOPS의 값을 사용하였다.

표 3: Transformer 아키텍처의 변형. 나열되지 않은 값은 기본 모델과 동일하다. 모든 지표는 영어-독어 번역 개발セット(newstest2013) 상에서 측정되었다. 나열된 퍼플렉시티(perplexities)는 바이트-페어 인코딩(byte-pair encoding)에 따른 단단어piece당 값이며, 단단어(per-word) 퍼플렉시티와 비교해서는 안 된다.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_s	학습 단계	PPL (개발)	BLEU (개발)	파라미터 수 $\times 10^6$
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	09
(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	06
(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)	사인 함수 대신 위치 임베딩 사용									4.92	25.7	
big	6	1024	4096	16			0.3		300K	4.33	26.4	213

개발 세트, newstest2013. 이전 섹션에서 설명한 바와 같이 빔 서치를 사용했으나 체크포인트 평균화는 적용하지 않았다. 이러한 결과는 Table 3에 제시한다.

표 3의 행 (A)에서, 3.2.2절에 설명된 바와 같이 연산량을 일정하게 유지한 채 주의력 헤드 수 및 주의력 키 및 값 차원을 변경하였다. 단일 헤드 주의력은 최상 설정 대비 BLEU 점수가 0.9 낮지만, 헤드가 너무 많을 경우 품질도 저하된다.

표 3의 행 (B)에서 우리는 attention 키 크기 d_k를 줄이면 모델 품질이 저하됨을 관찰합니다. 이는 호환성을 결정하기 어렵다는 것을 시사하며, 내적 곱(dot product)보다 더 정교한 호환성 함수가 유익할 수 있음을 의미합니다.我们又行 (C)와 (D)에서 예상대로 더 큰 모델일수록 성능이 우수하며, dropout은 과적합(over-fitting)을 방지하는 데 매우 도움이 된다는 것을 추가로 관찰합니다. 행 (E)에서는 기사(sinusoidal) 위치 인코딩을 학습된 위치 임베딩[9]으로 교체하여, 기본 모델(base model)과 거의 동일한 결과를 얻었습니다.

6.3 영어 구성소 구문 분석

Transformer가 다른 과업에도 적용 가능한지 평가하기 위해 영어 구성소 구문 분석에 대한 실험을 수행하였습니다. 이 과업은 출력이 강력한 구조적 제약을 받으며 입력보다 훨씬 길다는 점이라는 특유의 난이점을 가지고 있습니다. 또한, RNN 순서-순서 모델은 소량 데이터 환경에서 최첨단 성능을 달성하지 못해 왔습니다 [37].

우리는 Penn Treebank [25]의 Wall Street Journal (WSJ) 부분(약 40K 학습 문장)에서 모델 차원 $d_{model} = 1024$인 4층 트랜스포머를 훈련시켰다. 또한 약 17M 문장을 포함하는 더 큰 고품질 신뢰도(high-confidence) 및 BerkleyParser 코퍼스를 사용하여 반-supervised 설정에서도 훈련하였다. WSJ 전용 설정에는 16K 토큰 어휘를, 반-supervised 설정에는 32K 토큰 어휘를 사용하였다.

드롭아웃, 어텐션 및 잔차(5.4절), 학습률 및 빔 크기를 선택하기 위해 소수의 실험만 수행하였으며, 모든 다른 매개변수는 영어-독어 기본 번역 모델에서 그대로 유지되었습니다. 추론 중에는

표 4: Transformer는 영어 구성소 분석에 잘 일반화된다 (결과WSJ의 섹션 23 기준)

파서	학습	WSJ 23 F1
Vinyals 및 Kaiser 외 (2014) [37]	WSJ 전용, 판별적	88.3
Petrov 외 (2006) [29]	WSJ 전용, 판별적	90.4
Zhu 외 (2013) [40]	WSJ 전용, 판별적	90.4
Dyer 외 (2016) [8]	WSJ 전용, 판별적	91.7
Transformer (4층)	WSJ 전용, 판별적	91.3
Zhu 외 (2013) [40]	반지도 학습	91.3
Huang & Harper (2009) [14]	반지도 학습	91.3
McClosky 외 (2006) [26]	반지도 학습	92.1
Vinyals 및 Kaiser 외 (2014) [37]	반지도 학습	92.1
Transformer (4층)	반지도 학습	92.7
Luong 외 (2015) [23]	다중 작업	93.0
Dyer 외 (2016) [8]	생성적	93.3

최대 출력 길이를 입력 길이 + 300로 늘렸습니다. WSJ 전용 및 반 Supervised 설정 모두에서 빔 크기를 21, α를 0.3으로 사용했습니다.

표 4의结果表明, 비록 작업 특화 튜닝이 이루어지지 않았음에도 불구하고 본 모델은 놀라울 정도로 우수한 성능을 보이며, 재귀 신경망 문법[8]을 제외하고는 이전 보고된 모든 모델보다 더 나은 결과를 산출한다.

RNN 순서-순서 모델 [37]과 달리, Transformer는 40K 문장의 WSJ 학습 세트에서만 훈련된 경우에도 Berkeley-Parser [29]를 초월한다.

7 결론

본 연구에서 우리는 전 внимание에 기반하여 완전히 구축된 최초의 시퀀스 전도 모델인 트랜스포머를 제시하였으며, 엔코더-디코더 아키텍처에서 가장 일반적으로 사용되는 순환 계층을 멀티 헤드 셀프 어텐션으로 대체하였다.

번역 작업에서 트랜스포머는 순환 또는 컨볼루션 레이어 기반 아키텍처에 비해 훨씬 빠르게 훈련할 수 있다. WMT 2014 영어-독일어 번역 및 WMT 2014 영어-프랑스어 번역 작업 모두에서 최첨단 기술의 새로운 기록을 달성했다. 전자의 경우, 우리 모델이 Previously 보고된 모든 엔semble(집합 학습)보다 더 높은 성능을 보인다.

우리는attention 기반 모델의 미래에 대해 기대를 걸고 있으며, 이를 다른 작업에 적용할 계획이다. 우리는 Transformer를 텍스트가 아닌 입력 및 출력 모달리티를 포함하는 문제로 확장하고, 이미지, 오디오, 비디오와 같은 대규모 입력과 출력을 효율적으로 처리하기 위해 국소적이고 제한된 attention 메커니즘을 조사할 계획이다. 생성 과정을 더 순차적이지 않게 만드는 것은 우리의 또 다른 연구 목표이다.

우리가 모델의 학습 및 평가를 위해 사용된 코드는 https://github.com/tensorflow/tensor2tensor에서 확인할 수 있습니다.

감사 인사 Nal Kalchbrenner와 Stephan Gouws분께 유익한 의견, 수정 및 영감을 주신 점에 감사드립니다.

참고문헌

[1] 지미 레이 바, 제이미 라이언 키로스, 조피리 E 힌튼. 레이어 정규화. arXiv 사전 인인 arXiv:1607.06450, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, 및 Yoshua Bengio. 정렬과 번역을 공동으로 학습하는 신경 기계 번역. CoRR, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. 신경 기계 번역 아키텍처의 대규모 탐색. CoRR, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, 및 Mirella Lapata. 기계 독해를 위한 긴 단기 메모리 네트워크. arXiv 사전 인쇄물 arXiv:1601.06733, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, 및 Yoshua Bengio. 통계 기계 번역을 위한 RNN 인코더-디코더를 이용한 구절 표현 학습. CoRR, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: 심층 학습을 위한 depthwise separable convolutions를 이용한deeplearning. arXiv 사전 인쇄본 arXiv:1610.02357, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, 및 Yoshua Bengio. 시퀀스 모델링에서 게이트드 순환 신경망에 대한 경험적 평가. CoRR, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, 및 Noah A. Smith. 재귀 신경망 문법. NAACLProceedings에 수록, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin.卷积형 순서에서 순서로 학습. arXiv 사전 인출 arXiv:1705.03122v2, 2017.

[10] Alex Graves. 순환 신경망을 이용한 시퀀스 생성. arXiv 사전 인쇄본, arXiv:1308.0850, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, 및 Jian Sun. 이미지 인식을 위한 깊은 잔차 학습. IEEE计算机视觉 및 패턴 인식 회의록(Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition), 770 – 778페이지, 2016년.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, 및 Jürgen Schmidhuber. 순환 네트워크에서의 기울기 흐름: 장기 의존성 학습의 어려움, 2001.

[13] 세프 호크라이터와 융어른 슈미트후버. "롱 숏 트임 메모리." 니얼 컴퓨테이션, 9(8):1735 – 1780, 1997.

[14] Zhongqiang Huang 및 Mary Harper. 언어 간 잠재 주석과 함께 self-training PCFG 문법. In Natural Language Processing의 경험적 방법에 관한 2009ConferenceProceedings, 832 – 841쪽. ACL, 2009년 8월.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 언어 모델의 한계 탐구. arXiv 사전 인쇄본 arXiv:1602.02410, 2016.

[16] Łukasz Kaiser and Samy Bengio. 액티브 메모리가 어텐션을 대체할 수 있는가? , (NIPS), 2016.

[17] Łukasz Kaiser 및 Ilya Sutskever. 신경 GPU는 알고리즘을 학습한다. In International Conference on Learning Representations (ICLR), 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 선형 시간 내 신경 기계 번역. arXiv 사전 인쇄물 arXiv:1610.10099v2, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, 및 Alexander M. Rush. 구조화된 주의 네트워크. In 국제 학습 표현Conference, 2017.

[20] Diederik Kingma 및 Jimmy Ba. Adam: 확률적 최적화를 위한 한 방법. ICLR에, 2015.

[21] Oleksii Kuchaiev와 Boris Ginsburg. LSTM 네트워크를 위한 인수화 기법. arXiv预印本, arXiv:1703.10722, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, 및 Yoshua Bengio. 구조화된 자기 주의 문장 임베딩. arXiv 사전 인쇄본, arXiv:1703.03130, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, 및 Lukasz Kaiser. 멀티태스크 시퀀스 투 시퀀스 학습. arXiv 프리프린트 arXiv:1511.06114, 2015.

[24] Minh-Thang Luong, Hieu Pham, 및 Christopher D Manning. 어텐션 기반 신경 기계 번역을 위한 효과적인 접근법. arXiv 프리프린트 arXiv:1508.04025, 2015.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, 및 Beatrice Santorini. 영어의 대규모 주석 부여 코퍼러스 구축: 펜 트리뱅크. 계산 언어학, 19(2):313 – 330, 1993.

[26] David McClosky, Eugene Charniak, 및 Mark Johnson. 구문 분석을 위한 효과적인 자생 학습. In NAACL 인간 언어 기술conferenceProceedings, 메인 컨퍼런스, 152 – 159페이지. ACL, 2006년 6월.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, Jakob Uszkoreit. 분해 가능한 주의력 모델. Empirical Methods in Natural Language Processing, 2016.

[28] Romain Paulus, Caiming Xiong, 및 Richard Socher. 추상적 요약에 대한 딥 강화 모델. arXiv 사전 인쇄물 arXiv:1705.04304, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 정밀하고 압축 가능하며 해석 가능한 트리 주석 학습. 제21회 국제 계산 언어학 회의 및 ACL 제44회 연례 회의록, 433 – 440페이지. ACL, 2006년 7월.

[30] Ofir Press and Lior Wolf. 언어 모델을 개선하기 위한 출력 임베딩 활용. arXiv 전간 arXiv:1608.05859, 2016.

[31] 리코 센리치, 바리 해도우, 알렉산드라 버치. 서브워드 유닛을 이용한 드문 단어의 신경 기계 번역. arXiv 사전 인쇄본 arXiv:1508.07909, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, 및 Jeff Dean. 극도로 큰 신경 네트워크: 희소 게이트형 전문가의 혼합 계층. arXiv 프리프린트 arXiv:1701.06538, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, 및 Ruslan Salakhutdinov. Dropout: 신경망의 과적합을 방지하는 간단한 방법. Journal of Machine Learning Research, 15(1):1929 – 1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus. 엔드투엔드 메모리 네트워크. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett, 편집자, Neural Information Processing Systems 28에 대한 연구 진전, 2440 – 2448쪽. Curran Associates, Inc., 2015.

[35]伊利亞·蘇茨克維爾, 오리오르 비날스, 그리고 쿽 VV 레. 신경망을 이용한 순서에서 순서로 학습. Neural Information Processing Systems 진전에서, 3104 – 3112페이지, 2014년.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, 및 Zbigniew Wojna. 컴퓨터 비전을 위한 인셉션 아키텍처에 대한 재고찰. CoRR, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, 및 Hinton. 'Grammat 외국어로서(Grammar as a foreign language)'. 《Neural Information Processing Systems 진전(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, 외. 구글의 인공 신경 기계 번역 시스템: 인간과 기계 번역 사이의 격차 해소. arXiv 사전 인쇄본, arXiv:1609.08144, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, 및 Wei Xu. 신경 기계 번역을 위한 고속 전진 연결이 있는 깊은 순환 모델. CoRR, abs/1606.04199, 2016.

[40] 무화 주, 요장 장, 원량 첸, 민 장, 징보 주. 빠르고 정확한 시프트-감소 구성 요소 구문 분석. ACL 제51차 연례 회의 논문집 (제1권: 롱 페이퍼), 434~443페이지. ACL, 2013년 8월.

주의력 시각화

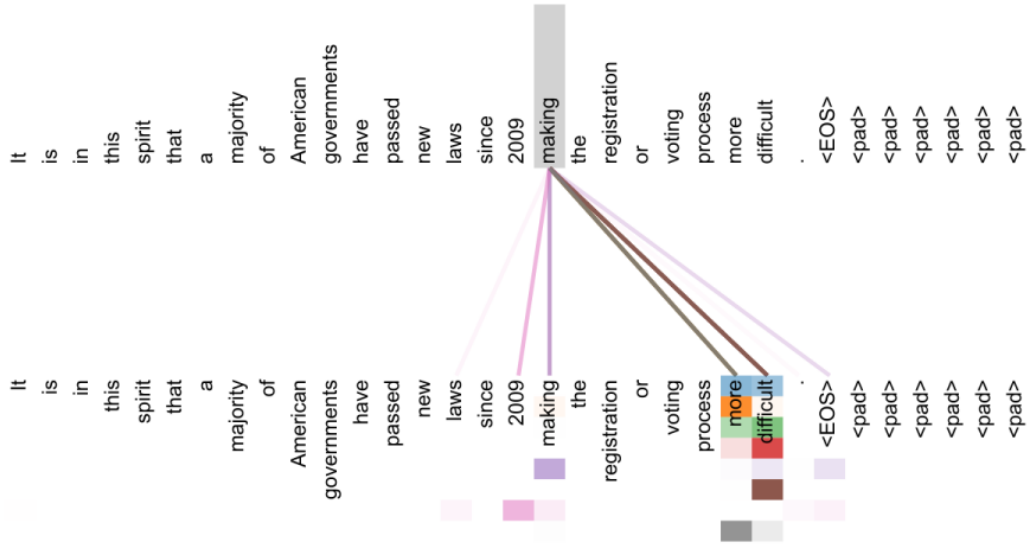


그림 3: 6층 중 5층의 인코더 자기attention(attention mechanism)에서 장거리 의존성(long-distance dependencies)을 따르는 자기attention의 예시. 많은 attention 헤드는 동사 'making'의 먼 의존성에 주의를 기울이며, 구 'making...more difficult'를 완성한다. 여기서는 단어 'making'에 대한 attention만 표시하였다. 다른 색상은 서로 다른 헤드를 나타낸다. 색상으로 볼 때 가장 명확하다.

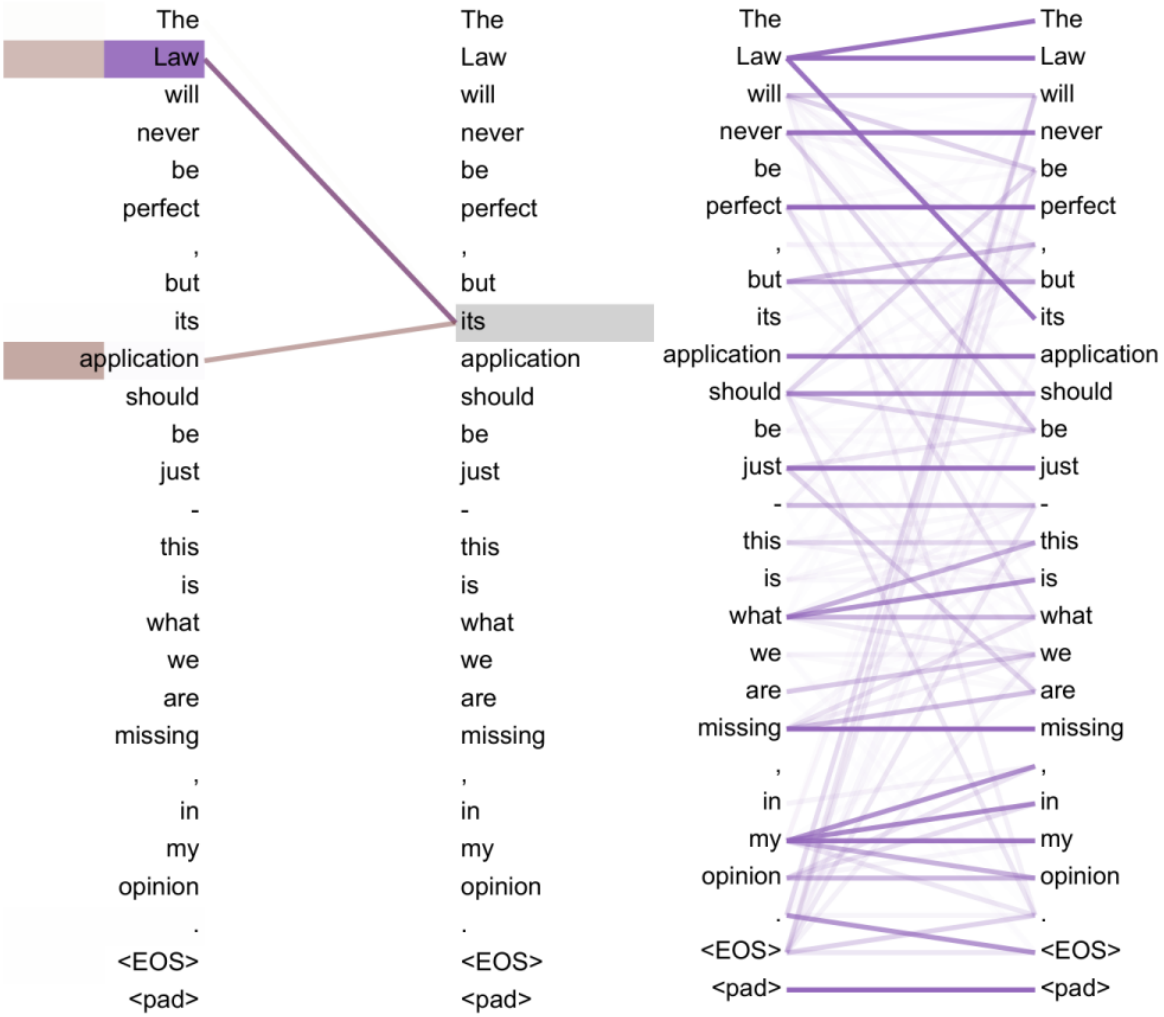


그림 4: 두 개의attention 헤드, 5/6 레이어에서도 안aphora 해결에 관여하는 것으로 보임. 위: 5번 헤드의 모든 attention. 아래: 단어 'its'에 대한 5번 및 6번 헤드의 isolated attention. 이 단어에 대한 attention이 매우 날카로움을 유의하십시오.

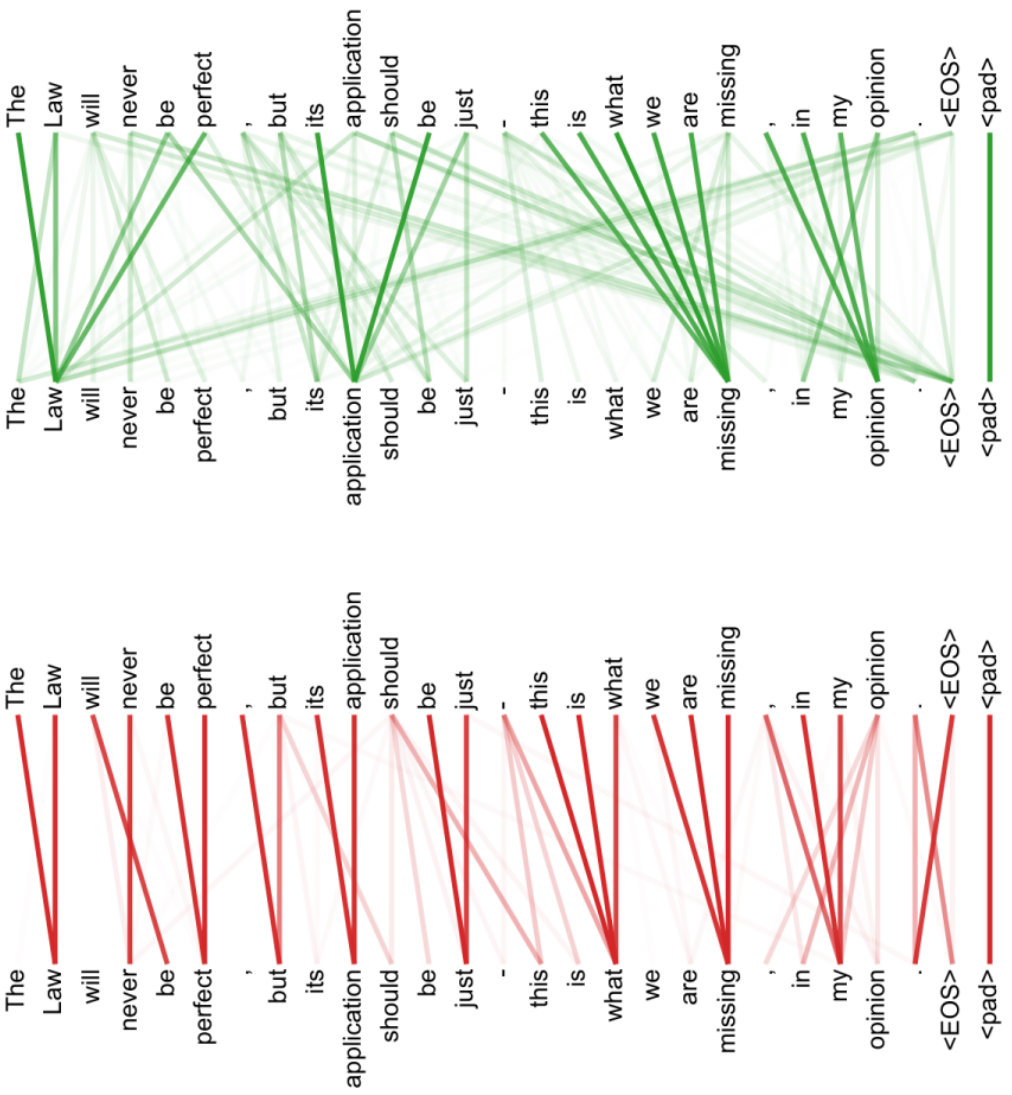


그림 5: 많은 어텐션 헤드가 문장의 구조와 관련된 행동을 보입니다. 위에는 인코더 셀프-어텐션의 6층 중 5층에서 나온 두 다른 헤드의 예시를 제시하였습니다. 이 헤드들은 명확하게 서로 다른 작업을 수행하도록 학습되었습니다.