Attention is all you need

1. Introduction

- 기존의 시퀀스 변환 모델은 대부분 RNN, LSTM, GRU 기반의 인코더-디코더 구조.
- RNN 계열 모델은 순차적 계산으로 인해 병렬화가 어렵고 긴 시퀀스에서 학습 효율이 떨어짐.
- Attention 메커니즘은 입력과 출력 시퀀스 간의 장거리 의존성을 효과적으로 학습하게 해줌.
- 하지만 기존 모델에서는 attention이 RNN에 보조적으로만 사용됨.
- Transformer는 순수하게 attention만으로 구성된 모델로, recurrence와 convolution을 모두 제거함.
- 병렬 처리와 학습 속도가 향상되면서 번역 품질도 기존 최고 모델보다 BLEU 점수가 더 높음.

---

2. Background

- 기존의 순차 계산 문제를 해결하기 위해 Extended Neural GPU, ByteNet, ConvS2S 같은 CNN 기반 모델 등장.
- CNN은 병렬 계산이 가능하지만 멀리 떨어진 단어 간 관계를 학습하기 위해서는 레이어를 깊게 쌓아야 함 → 경로 길이가 증가.
- Self-Attention은 한 레이어에서 모든 단어 간 직접적인 관계를 계산할 수 있음.
- Memory Networks, Structured Attention Networks 등도 존재하지만 Transformer는 RNN과 CNN 없이 완전한 self-attention 기반의 최초 모델.

---

3. Model Architecture

- Transformer는 일반적인 시퀀스-투-시퀀스 모델처럼 인코더와 디코더 구조를 가짐.
- 각 부분은 self-attention과 position-wise feed-forward network를 쌓아 올린 형태.

3.1 Encoder and Decoder Stacks

Encoder

- 인코더는 N=6개의 동일한 레이어로 구성.
- 각 레이어는
  1. Multi-Head Self-Attention: 입력 시퀀스의 각 단어가 다른 모든 단어를 동시에 주목할 수 있게 함.
  2. Feed-Forward Network: 각 위치별로 독립적인 fully connected layer 적용.
- 각 서브레이어에는 Residual Connection + Layer Normalization이 적용됨:
  - 출력 = `LayerNorm(x + Sublayer(x))`
- 모든 서브레이어와 임베딩 차원은 동일하게 **d_model = 512.**

Decoder

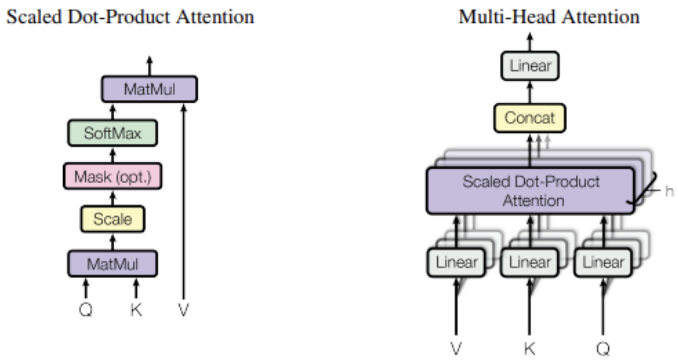
- 디코더도 N=6개의 동일한 레이어로 구성.
- 각 레이어는 3개의 서브레이어 포함

1. Masked Multi-Head Self-Attention: 미래 단어를 참조하지 못하도록 마스킹.
2. Encoder-Decoder Attention: 인코더 출력의 정보를 사용.
3. Feed-Forward Network.

- Residual Connection과 Layer Normalization 적용.

3.2 Attention

transformer는 **attention만으로** 구성된 모델이며, 이 중에서도 가장 핵심이 되는 것이 Scaled Dot-Product Attention과 Multi-Head Attention임



3.2.1 Scaled Dot-Product Attention

- 입력: Query Q, Key K, Value V
- 계산식:

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

- Query와 Key의 내적을 통해 유사도를 계산
- $\sqrt{d_k}$로 나누는 이유: 차원이 커질수록 softmax gradient가 매우 작아지는 현상 방지
- **dot-product attention**은 연산 효율이 뛰어나고 병렬화에 유리함

3.2.2 Multi-Head Attention

- attention을 한 번만 수행하는 대신, 여러 개의 서로 다른 head를 두어 **다양한 표현 공간에서 병렬로 attention** 수행
- 각 head마다 서로 다른 가중치 행렬로 Q, K, V를 투영한 뒤 attention 수행 → 결과를 concat → 최종 출력 생성
- 계산식:

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

- 실험에서는 $h = 8, d_k = d_v = 64$
- 다양한 표현 공간(subspace)에서 정보를 병렬적으로 학습
- 한 head에서는 포착하지 못한 관계를 다른 head가 포착 가능

3.2.3 Applications of Attention in our Model

Transformer는
총 3가지 방식으로 attention을 사용

위치	종류	설명
인코더	Self-Attention	인코더 내에서 입력 간 관계를 모델링
디코더	Masked Self-Attention	디코더 내에서 이전 위치까지만 보도록 마스킹 적용
디코더	Encoder-Decoder Attention	디코더가 인코더 출력에 attend (기존 seq2seq 방식과 유사)

디코더의 self-attention에서는 미래 위치로의 정보 흐름을 방지하기 위해 softmax 입력값을 -∞로 마스킹함.

3.3 Position-wise Feed-Forward Networks

- Attention만으로는 비선형 표현 능력이 부족하기 때문에 각 위치별로 독립적으로 **FFN을 추가**함.
- 각 Transformer block에는 attention 뒤에 다음 연산이 포함됨: $\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$
- 이는 ReLU가 들어간 두 개의 선형 계층으로 구성된 MLP
- 모든 위치에 **같은 파라미터**를 적용하지만, **layer 간에는 서로 다른 파라미터**를 사용
- 구조적으로는 1×1 convolution과 유사한 연산
- 차원: 입력/출력 $d_{\text{model}} = 512$, 내부 차원 $d_{\text{ff}} = 2048$

3.4 Embeddings and Softmax

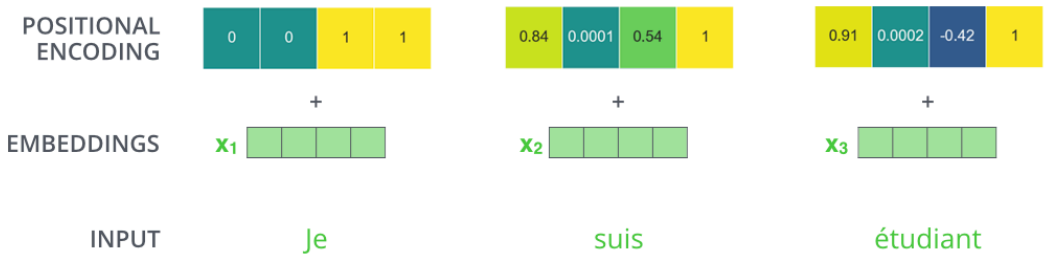
- 입력/출력 토큰을 **512차원 벡터로 임베딩**함
- 임베딩 행렬과 **출력 softmax 직전의 선형 계층은 파라미터를 공유**함 (Press & Wolf, 2016 아이디어 활용)
- 특징:
  - 임베딩 행렬의 가중치는 $\sqrt{d_{\text{model}}}$만큼 곱해서 크기 조정
  - **단어 분포가 너무 편향되지 않도록 안정성 확보**

3.5. Positional Encoding

Transformer 모델이 사용하는 Self-Attention 메커니즘은 시퀀스 내의 모든 위치를 동시에 처리하므로 **토큰 간의 순서 정보를 반영하지 못함**

→ 모델이 시퀀스의 순서 정보를 활용하도록 하기 위해 **Positional Encoding**이 필요

- 입력 임베딩(input embeddings)과 동일한 차원(d_{model})을 갖는 positional encoding 벡터를 생성
- 이 벡터를 Encoder와 Decoder 스택의 가장 아래층에 있는 입력 임베딩 벡터에 더해줌으로써 모델은 각 토큰의 의미적 정보(임베딩)와 위치 정보를 함께 학습



A real example of positional encoding with a toy embedding size of 4
[
https://jalammar.github.io/illustrated-transformer/]

논문에서는 Positional Encoding 함수로 사인과 코사인 함수를 사용

- 훈련 때 보지 못했던 아주 긴 문장이 들어와도 Transformer는 당황하지 않고 위치 정보를 부여받아 처리할 수 있음 → 외삽(extrapolation)에 강함

❓ 다른 함수를 사용했을 때 나타날 수 있는 문제점

1. 학습된(learned) 위치 임베딩: 훈련 데이터에 나온 가장 긴 문장의 길이까지만 위치 정보를 학습할 수 있으므로, 그보다 더 긴 문장이 들어오면 성능이 나빠짐
2. 간단한 선형 함수: 언어에는 반복되는 패턴, 구조(주어-동사-목적어)가 많은데, 단순한 숫자는 이런 주기성을 표현하지 못함, 긴 거리에서는 정보가 희미해짐

4. Why Self-Attention

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Computational Complexity per Layer

- 시퀀스 길이 n과 표현 차원 d에 대해 $O(n^2 \cdot d)$의 복잡도를 가지므로 RNN과 비교했을 때 $n < d$ 인 경우 전체 계산량이 더 작음

Sequential Operations

- RNN은 n개의 데이터를 순차적으로 입력 받기에 $O(n)$의 복잡도를 갖지만, Transformer는 입력 데이터를 한번에 처리하기에 $O(1)$의 복잡도를 가짐

Maximum Path Length

- 네트워크 내에서 입력 및 출력 시퀀스의 임의의 두 위치 사이를 신호가 이동해야 하는 경로의 길이로 경로가 짧을수록 모델이 장거리 의존성을 학습하기 더 쉬움
- Self-Attention는 시퀀스 내의 모든 위치를 직접 연결하므로, 모든 위치 간의 경로 길이가 상수 $O(1)$임
- Transformer가 기존 RNN, LSTM 등의 방법들과 비교했을때 더 긴 정보를 잘 처리함

5. Training & 6. Results

Adam 옵티마이저와 학습률 스케줄링, 그리고 잔차 드롭아웃 및 레이블 스무딩과 같은 정규화 기법을 적용하여 훈련함

WMT 번역 task에서 기존 SOTA 대비 더 적은 학습 비용을 가지고 더 좋은 성능을 달성

	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

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

- (A): 적절한 head의 개수를 찾는 것이 성능 향상에 도움이 됨
- (B): key 크기를 줄이면 모델 성능이 크게 하락할 수 있음
- (C), (D): 사이즈가 큰 모델일 수록 성능이 향상되는 경향이 있으며, drop-out이 overfitting을 완화하여 성능을 개선함

- (E): Learned Positional Encoding을 사용해도 결과가 유사함

7. Conclusion

기존의 encoder-decoder 아키텍처에서 주로 사용되던 recurrent 레이어를 Multi-Head Self-Attention으로 완전히 대체하여, **오직 Attention 메커니즘만을 기반으로 하는 최초의 Sequence Transduction 모델인 Transformer를 제시**

WMT 2014 영어-독일어 및 영어-프랑스어 번역 태스크 모두에서 새로운 SOTA(state-of-the-art)를 달성

task-specific tuning이 부족했음에도 불구하고 다른 task(영어 구문 분석)에서도 우수한 결과를 보임