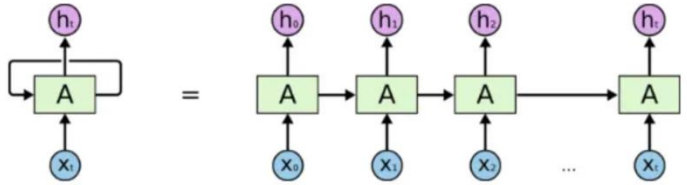
Transformer

Attention Is All You Need

1.0 Transformer의 등장 배경: 순차적 처리의 한계



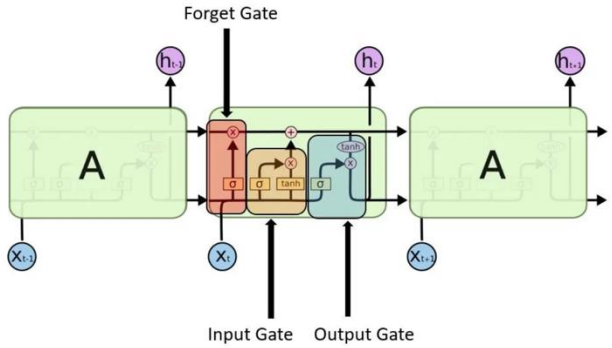
An unrolled recurrent neural network.

1.1 도입
- 인공지능(AI) 분야에서 텍스트나 시계열 데이터와 같은 순차적 데이터를 모델링하는 것은 가장 근본적인 도전 과제 중 하나
- 기계 번역, 음성 인식, 감성 분석 등 수많은 응용 분야의 성공에 직결

1.2 순차적 데이터와 순환 신경망 (RNN)
- 순차적 데이터란 "나는 학교에 간다"와 같은 문장처럼 순서가 중요한 의미를 가지는 데이터
- 대표적인 초기 해결책은 **순환 신경망(Recurrent Neural Network, RNN)**
- RNN의 핵심 메커니즘은 데이터를 한 번에 하나씩(예: 단어 하나씩) 순차적으로 처리하면서, '은닉 상태(hidden state)'라는 내부 메모리를 유지하는 것. 이 은닉 상태 덕분에 모델은 이전에 처리했던 정보를 기억하고 현재의 입력을 이해하는 데 활용
- RNN은 두 가지 치명적인 한계를 가짐.
- 1. **장기 의존성 문제 (The Long-Term Dependency Problem)** RNN은 긴 시퀀스를 처리할 때, 시퀀스 초반의 정보를 끝까지 전달하는 데 어려움. 이는 **기울기 소실(vanishing gradient)** 및 **기울기 폭주(exploding gradient)** 문제 때문. 정보가 여러 단계를 거치면서 점차 희미해지거나(소실) 과도하게 증폭되어(폭주), 모델이 문장의 시작 부분에 있는 단어가 문장 끝에 있는 단어에 미치는 영향을 학습하기 매우 어려움.
- 2. **순차적 계산의 비효율성 (Inefficiency of Sequential Computation)** RNN의 가장 근본적인 한계는 본질적으로 순차적으로 계산을 수행. 즉, 이전 단계의 계산이 끝나야만 다음 단계의 계산을 시작할 수 있음. 이러한 구조는 **병렬 처리(parallel processing)**를 불가능하게 만들어, 긴 시퀀스에 대한 모델 훈련을 매우 느리게 만듦.

https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e

1.0 Transformer의 등장 배경: 순차적 처리의 한계



LSTM gates

1.3 LSTM과 발전된 모델들

- RNN의 장기 의존성 문제를 완화하기 위해 **LSTM(Long Short-Term Memory)**이라는 발전된 모델이 등장.
- LSTM은 '게이트(gate)'라는 정교한 메커니즘을 도입하여 어떤 정보를 기억하고, 어떤 정보를 잊을지를 선택적으로 제어함으로써 기울기 소실 문제를 상당 부분 해결.
- LSTM은 장기 의존성 처리 능력을 향상시켰지만, 여전히 데이터를 순차적으로 처리해야 한다는 근본적인 한계에서는 벗어나지 못함. 연구자들은 문맥을 더 잘 파악하기 위해 문장을 왼쪽에서 오른쪽, 그리고 오른쪽에서 왼쪽으로 모두 처리하는 **양방향 LSTM(bidirectional LSTM)**과 같은 모델도 개발했지만, 이 역시 순차적 처리라는 병목 현상을 해결하지는 못함.

1.4 결론 및 전환

- 결론적으로 RNN과 LSTM은 순차적 데이터 모델링의 기초를 다졌지만, 그들의 순차적 처리 방식은 성능과 효율성 모두에서 명백한 한계에 부딪힘. AI 연구 커뮤니티는 장거리 의존성을 더 효과적으로 학습하면서도, 병렬 처리를 통해 훈련 속도를 획기적으로 개선할 수 있는 새로운 아키텍처의 필요성을 절감→ **Transformer**!

2.0 Transformer의 설계 동기와 차별성: 순차성을 버리다

- **2.1 도입**
- RNN의 핵심적인 한계들—O(n)에 달하는 경로 길이 때문에 장기 의존성 학습에 어려움을 겪고, O(n)의 순차 연산 때문에 병렬화가 불가능하다는 점—은 단순히 개선해야 할 문제가 아님. 이는 '시퀀스 처리는 반드시 순차적이어야 한다'는 결함 있는 기본 가정에서 비롯된 증상
- Transformer의 설계는 이러한 가정을 정면으로 반박하며 다음과 같은 급진적인 해결책을 제안
- “만약 우리가 모든 단어를 다른 모든 단어와 O(1)의 경로 길이로 직접 연결하고, 이 모든 계산을 한 번에 수행할 수 있다면 어떨까? 이 패러다임 전환을 통해 Transformer는 순환(recurrence) 구조를 완전히 제거하고, 오직 **셀프 어텐션(self-attention)**이라는 메커니즘에만 의존하여 단어 간의 의존성을 파악하는 혁신을 이룸.

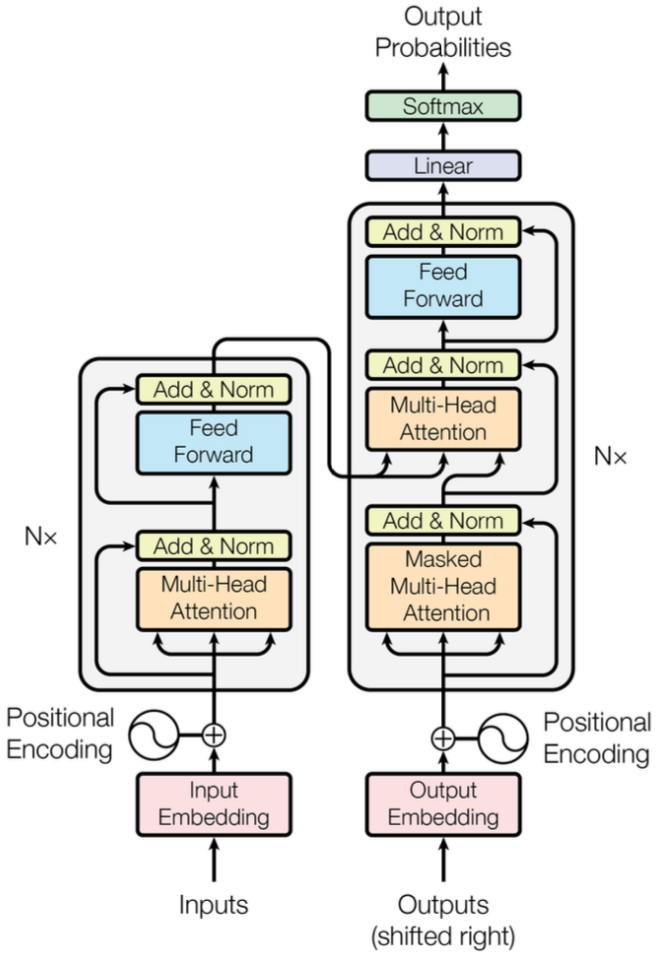


Figure 1: The Transformer - model architecture.

2.0 Transformer의 설계 동기와 차별성: 순차성을 버리다

- **2.2 핵심 아이디어: Attention 메커니즘**
- 어텐션(Attention)은 모델이 출력을 생성할 때 입력 시퀀스에서 가장 관련성이 높은 부분에 '집중'할 수 있도록 하는 메커니즘. 이를 통해 모델은 중요한 정보에 더 큰 가중치를 부여하고, 덜 중요한 정보는 무시.
- Transformer는 여기서 한 걸음 더 나아가 **셀프 어텐션(Self-Attention)** 또는 **인트라 어텐션(intra-attention)**이라는 개념을 도입.
- 이는 하나의 시퀀스가 자기 자신을 분석하여 시퀀스 내 모든 단어가 다른 모든 단어와 얼마나 관련이 있는지를 계산하는 방식.
- 예를 들어, "그 동물은 너무 피곤해서 길을 건너지 않았다(The animal didn't cross the street because **it** was too tired)"라는 문장이 있다면, 셀프 어텐션은 대명사 '**it**'이 '그 동물(The animal)'을 가리킨다는 것을 파악하는 데 결정적인 역할.

2.0 Transformer의 설계 동기와 차별성: 순차성을 버리다

- **2.3 순환(Recurrence) 방식과의 비교**
- Transformer의 셀프 어텐션 계층은 기존의 순환(Recurrent) 계층과 비교했을 때 뚜렷한 장점을 가짐. 아래 표는 'Attention Is All You Need' 논문의 Table 1을 기반으로 각 계층 유형의 특징을 비교
- *n: 시퀀스 길이, d: 표현 차원, r: 제한된 어텐션의 이웃 크기*

계층 유형 (Layer Type)	계층 당 계산 복잡도 (Complexity per Layer)	순차 연산 수 (Sequential Operations)	최대 경로 길이 (Maximum Path Length)
Self-Attention	O(n^2 * d)	O(1)	O(1)
Recurrent (RNN)	O(n * d^2)	O(n)	O(n)
Self-Attention (restricted)	O(r * n * d)	O(1)	O(n/r)

- **병렬 처리 (Parallelization)** 순차 연산 수는 어떤 연산이 이전 연산의 결과에 의존하는지를 나타냄. RNN은 시퀀스 길이 n에 비례하는 O(n)의 순차 연산이 필요하여 병렬화가 불가능했지만, 셀프 어텐션은 순차 연산 수가 O(1)로 상수. 이는 시퀀스 내 모든 단어에 대한 계산을 동시에 수행할 수 있음을 의미하며, GPU를 활용한 대규모 병렬 처리를 가능하게 하여 훈련 시간을 단축.
- **장기 의존성 학습 (Long-Range Dependency Learning)** 모델이 두 단어 간의 관계를 학습하기 위해 신호가 이동해야 하는 경로의 길이는 장기 의존성 학습 능력에 큰 영향을 미침. RNN에서는 멀리 떨어진 두 단어 간의 최대 경로 길이가 O(n)에 달했지만, 셀프 어텐션에서는 모든 단어가 직접 연결되므로 경로 길이가 O(1). 이론적으로 이는 모델이 문장 내 멀리 떨어진 단어들 간의 관계를 훨씬 더 쉽게 학습할 수 있게 만듦.

3.0 Transformer의 구조: Encoder와 Decoder

- **3.1 도입**
- 오리지널 Transformer는 기계 번역과 같은 시퀀스-투-시퀀스(sequence-to-sequence) 작업을 위해 설계
- **인코더-디코더(Encoder-Decoder)** 구조
- 인코더와 디코더 모두 여러 개의 동일한 계층(layer)을 쌓아 올린 구조로 되어 있지만, 각 계층을 구성하는 하위 요소에는 약간의 차이가 있음.

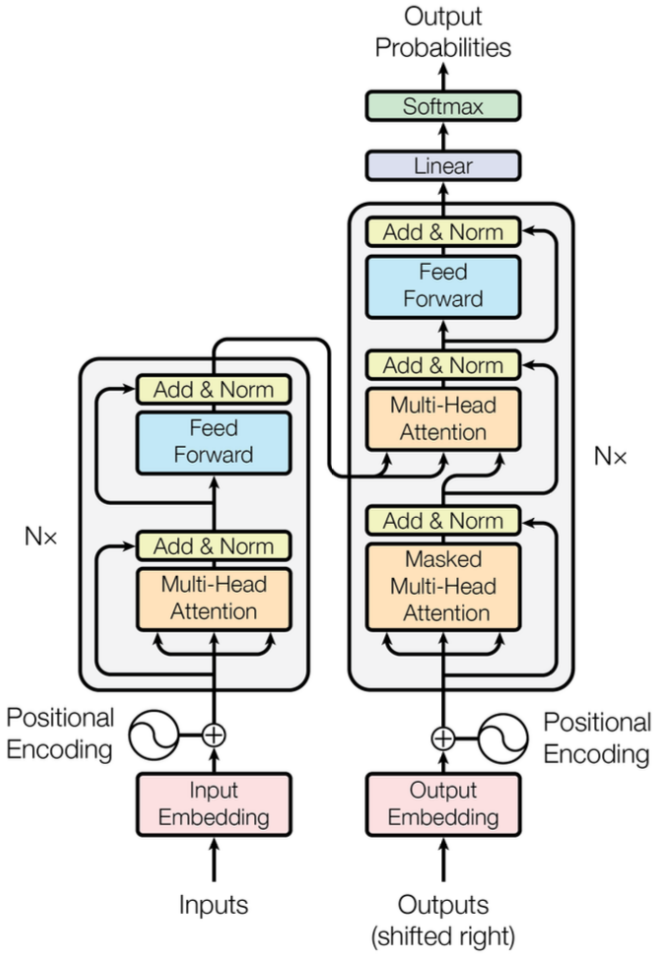


Figure 1: The Transformer - model architecture.

3.0 Transformer의 구조: Encoder와 Decoder

- **3.2 인코더 (Encoder) 스택**
- 인코더의 역할은 입력 시퀀스(예: 독일어 문장)를 처리하여, 의미와 문맥 정보가 풍부하게 담긴 연속적인 표현(representation)으로 변환. 오리지널 논문에서는 N=6개의 동일한 계층을 쌓아 인코더 스택을 구성.
- 각 인코더 계층은 두 개의 주요 하위 계층(sub-layer)으로 이루어져 있습니다.
- **멀티-헤드 셀프 어텐션 (Multi-Head Self-Attention):** 이 계층에서 입력 시퀀스 내 단어들 간의 관계를 파악하는 셀프 어텐션 메커니즘이 수행.
- **위치별 피드-포워드 신경망 (Position-wise Feed-Forward Network, FFN):** 각 위치의 표현에 독립적으로 적용되는 완전 연결(fully connected) 피드-포워드 네트워크.
- 각 하위 계층의 출력에는 **잔차 연결(Residual Connection)**과 **계층 정규화(Layer Normalization)**가 적용. 이는 깊은 신경망에서 기울기가 불안정해지는 것을 방지하고 훈련을 안정시키는 데 중요한 역할.

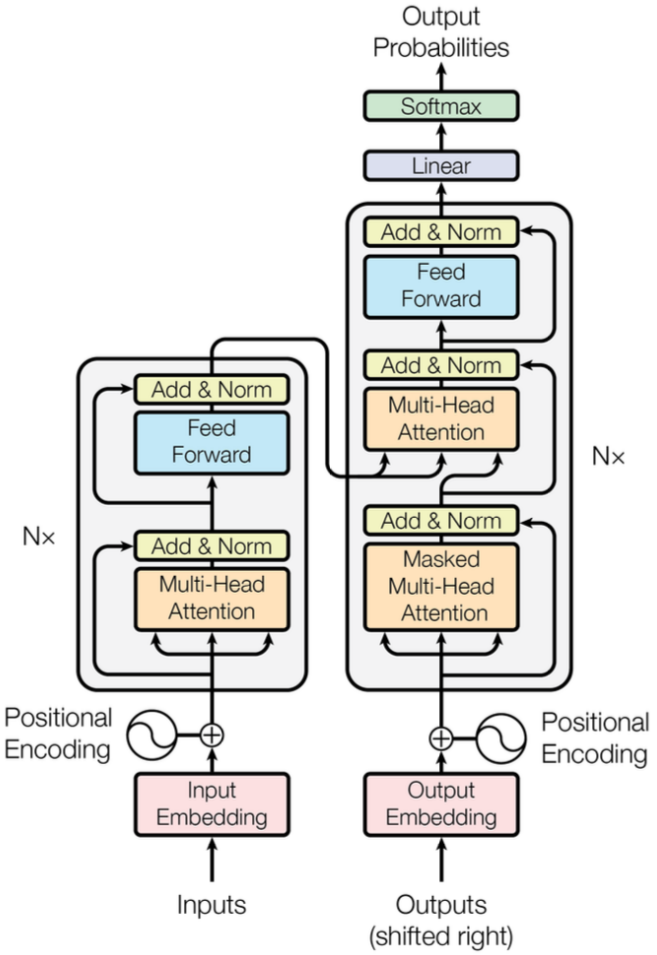


Figure 1: The Transformer - model architecture.

3.0 Transformer의 구조: Encoder와 Decoder

- **3.3 디코더 (Decoder) 스택**
- 디코더의 역할은 인코더가 생성한 표현을 입력으로 받아, 출력 시퀀스(예: 영어 문장)를 한 번에 한 토큰씩 생성
- 이 과정은 **자기회귀(auto-regressive)**라고 불리는데, 이는 모델이 다음 단어를 예측한 결과를 다시 입력으로 사용하여 그 다음 단어를 예측하는 방식. 마치 우리가 문장을 한 단어씩 쓸 때, 이미 쓴 단어들을 바탕으로 다음 단어를 결정하는 것과 유사. 디코더 역시 N=6개의 동일한 계층으로 구성.
- 각 디코더 계층은 세 개의 하위 계층으로 구성되며, 인코더보다 하나 더 많음.
- **마스크드 멀티-헤드 셀프 어텐션 (Masked Multi-Head Self-Attention):** 이 계층은 인코더의 셀프 어텐션과 유사하지만, 한 가지 중요한 차이점이 있음. 바로 '마스크(mask)'를 사용하여 특정 위치가 그 이후의 위치를 참조하지 못하도록 막음. 이는 디코더가 정답을 미리 보고 생성하는 것을 방지, 자기회귀 속성을 유지하기 위해 필수적.
- **인코더-디코더 어텐션 (Encoder-Decoder Attention):** 이 계층은 인코더와 디코더를 연결하는 다리 역할. 디코더는 이 계층을 통해 인코더가 출력한 입력 시퀀스의 표현 중에서 현재 토큰을 생성하는 데 가장 관련성이 높은 부분에 집중.
- **위치별 피드-포워드 신경망 (FFN):** 인코더에 있는 것과 동일한 기능을 수행.
- 디코더 역시 각 하위 계층마다 잔차 연결과 계층 정규화가 적용.

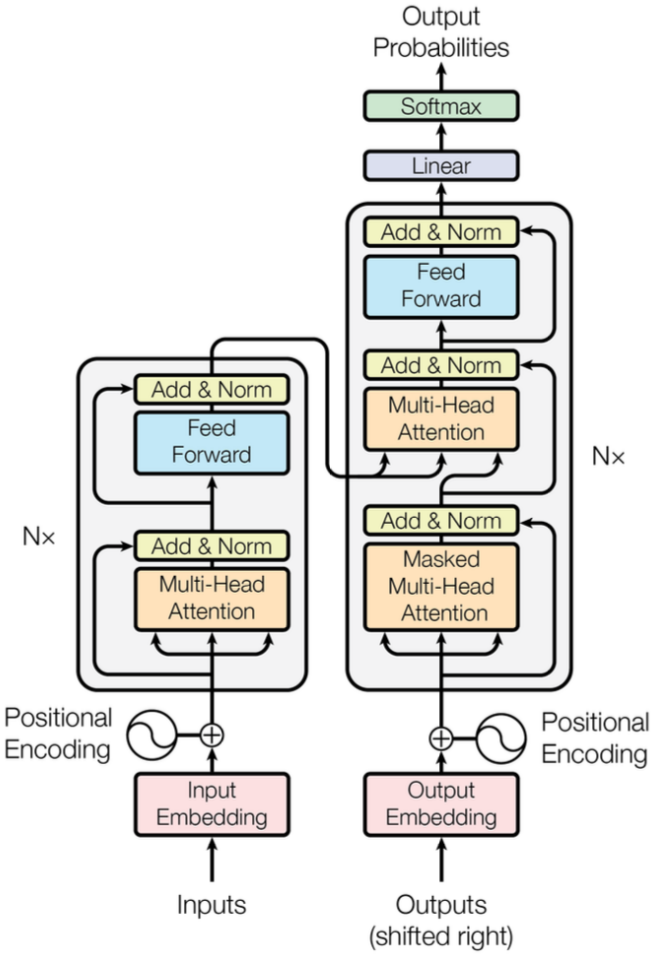


Figure 1: The Transformer - model architecture.

4.0 Transformer의 동작 상세 설명: Attention의 작동 방식

- **4.1 도입**
- 어텐션의 작동 원리는 세 가지 주요 개념으로 나누어 이해 :
  - **스케일드 닷-프로덕트 어텐션(Scaled Dot-Product Attention),**
  - **멀티-헤드 어텐션(Multi-Head Attention),**
  - **위치 인코딩(Positional Encoding)**

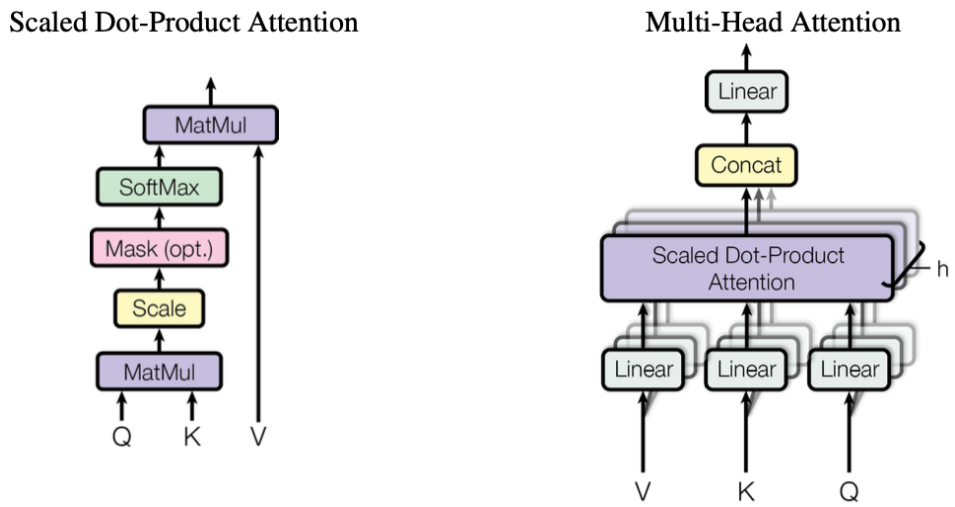


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

4.0 Transformer의 동작 상세 설명: Attention의 작동 방식

- **4.2 스케일드 닷-프로덕트 어텐션 (Scaled Dot-Product Attention)**
- 어텐션 메커니즘의 가장 기본적인 구성 요소는 **쿼리(Query, Q)**, **키(Key, K)**, ****밸류(Value, V)****라는 세 가지 벡터
- 도서관에서 연구하는 상황에 비유.
  - **쿼리(Q)**는 구체적인 '질문'. 책들은 '제목'을 가지고 있는데, 이것이 바로 **키(K)**, 책의 실제 '내용'은 **밸류(V)**에 해당
  - 질문(Q)과 각 책의 제목(K)을 비교하여 관련성 점수를 계산. 이 점수를 바탕으로, 가장 관련성이 높은 책들의 내용(V)에 가장 큰 가중치를 두어 종합적인 답변을 생성.

- 이 과정은 아래의 수식으로 표현.

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

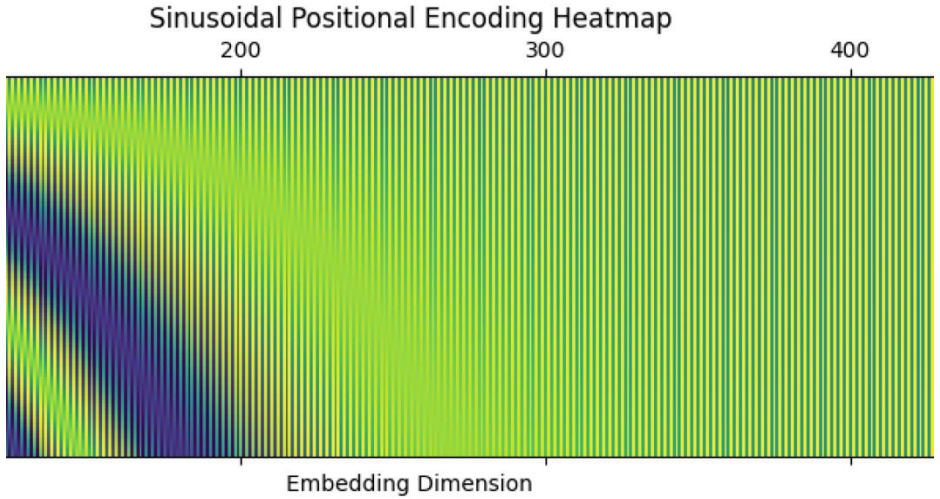
  - Attention(Q, K, V) = softmax((Q * K^T) / sqrt(dk)) * V
  - 쿼리(Q)와 모든 키(K)의 **내적(dot product)**을 계산 (Q * K^T). 이는 쿼리와 각 키의 유사도를 측정하는 단계.
  - 결과를 키 벡터의 차원(dk)의 제곱근(sqrt(dk))으로 나눔. 이 **스케일링(scaling)** 과정은 dk 값이 클 때 내적 값이 너무 커져 기울기가 매우 작아지는 현상을 방지하여 훈련을 안정시키는 중요한 역할.
  - **소프트맥스(softmax)** 함수를 적용하여 어텐션 가중치(attention weights)를 얻음. 이 가중치들은 총합이 1이 되는 확률값으로, 각 밸류를 얼마나 참조할지를 결정.
  - 계산된 가중치를 각 밸류(V)에 곱하여 **가중합(weighted sum)**을 구함으로써 최종 출력 벡터를 얻음.

4.0 Transformer의 동작 상세 설명: Attention의 작동 방식

- **4.3 멀티-헤드 어텐션 (Multi-Head Attention)**
- 하나의 '큰' 어텐션을 한 번 계산하는 것보다, 여러 개의 '작은' 어텐션을 병렬적으로 계산하는 것이 더 효과적이라는 아이디어에서 출발한 것이 멀티-헤드 어텐션.
- 과정.
  - 기존의 Q, K, V 벡터를 h개(오리지널 논문에서는 h=8)의 서로 다른 저차원 부분 공간(subspace)으로 선형 투영(linearly project)합니다.
  - 이렇게 생성된 h개의 Q, K, V 세트 각각에 대해 스케일드 닷-프로덕트 어텐션을 독립적으로, 그리고 병렬적으로 수행합니다.
  - 각 '헤드(head)'에서 나온 h개의 출력 벡터를 모두 연결(concatenate)합니다.
  - 연결된 벡터를 다시 한번 선형 투영하여 원래 모델의 차원으로 되돌립니다.
- 이는 마치 하나의 문장을 여러 가지 관점(예: 한 헤드는 문법적 관계에 집중하고, 다른 헤드는 의미적 관계에 집중)에서 동시에 바라보는 것과 같음. 이를 통해 모델은 다양한 표현 부분 공간의 정보를 종합적으로 활용하여 문맥을 더 풍부하게 이해

4.0 Transformer의 동작 상세 설명: Attention의 작동 방식

- **4.4 위치 인코딩 (Positional Encoding)**
- 만약 Transformer가 순차적인 단계 없이 모든 입력 토큰을 동시에 처리한다면, 어떻게 문장 속 단어의 순서를 파악할 수 있을까?
- 이 정보가 없다면 '개가 사람을 물었다'와 '사람이 개를 물었다'는 모델에게 똑같이 보임
- 이 문제에 대한 모델의 해결책이 바로 **위치 인코딩(Positional Encoding)**.
- 위치 인코딩은 각 단어의 시퀀스 내 상대적 또는 절대적 위치 정보를 담고 있는 벡터로, 입력 임베딩에 더해짐
- 오리지널 논문에서는 서로 다른 주기를 가진 사인(sine)과 코사인(cosine) 함수를 사용하여 위치 인코딩을 구현.
- 이러한 함수를 사용하면 모델이 절대적인 위치에 얽매이지 않고, 상대적인 위치 관계를 훨씬 쉽게 학습할 수 있음



https://www.ibm.com/think/topics/positional-encoding

In this work, we use sine and cosine functions of different frequencies:

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

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

5.0 Transformer의 성능: 새로운 시대의 서막

- **5.1 도입**
- Transformer의 혁신적인 아키텍처는 곧바로 최첨단(state-of-the-art) 성능으로 이어졌음
- **5.2 기계 번역 분야에서의 성과**
- 'Attention is All You Need' 논문은 기계 번역 분야에서 Transformer의 놀라운 성능을 입증
- 특히 가장 권위 있는 데이터셋 중 하나인 **WMT 2014**의 영어-독일어 및 영어-프랑스어 번역 과제에서 전례 없는 결과를 달성. 영어-독일어 번역 데이터는 약 37,000개의 토큰으로 구성된 공유 어휘 사전을 사용했고, 영어-프랑스어 번역 데이터는 32,000개의 단어 조각(word-piece) 어휘 사전을 사용.
- **BLEU 점수**: 기계가 번역한 문장이 사람이 번역한 고품질의 문장 집합과 얼마나 유사한지를 측정하는 지표로, 점수가 높을수록 번역 품질이 좋다고 평가.
- **영어-독일어 번역 (English-to-German): 28.4**라는 새로운 최고 BLEU 점수를 기록. 이는 기존 앙상블 모델을 포함한 최고 성능 모델들을 2.0 BLEU 점수 이상으로 능가하는 결과. WMT 2014와 같이 성숙한 벤치마크에서 단일 모델이 이 정도의 성능 향상을 이룬 것은 단순한 점진적 개선이 아닌, 근본적인 아키텍처의 우월성을 입증하는 거대한 도약.
- **영어-프랑스어 번역 (English-to-French):** 단일 모델로서 **41.0**의 새로운 최고 BLEU 점수를 달성하며 당시 최첨단 성능을 경신.

5.0 Transformer의 성능: 새로운 시대의 서막

5.3 훈련 효율성

- 더욱 놀라운 점은 이러한 혁신적인 결과가 이전의 최고 성능 모델들에 비해 훨씬 적은 훈련 시간으로 달성.
- 논문에 따르면, 대형 Transformer 모델은 **단 3.5일 동안 8개의 P100 GPU**에서 훈련
- 이는 이전 모델들이 소요했던 훈련 비용의 극히 일부에 불과. 이러한 훈련 효율성의 비약적인 향상은 이론에만 그치지 않았던 아키텍처의 장점, 즉 2장에서 논의했던 O(1)의 순차 연산을 통한 대규모 병렬 처리 덕분.
- 이 효율성은 결국 더 큰 모델을 탐색할 수 있는 문을 열어줌.

5.4 결론 및 전환

- Transformer의 압도적인 성능과 뛰어난 훈련 효율성은 자연어 처리(NLP) 분야에 새로운 시대가 도래했음을 알리는 신호탄. 이 모델이 단지 하나의 성공 사례에 그치지 않고, AI 생태계 전체에 거대한 파급 효과를 가져옴.

6.0 Transformer의 파급 효과: NLP 생태계의 변화

6.1 도입

- Transformer의 영향력은 단 하나의 최첨단 성능 기록을 넘어 훨씬 더 광범위하게 퍼져나감.
- 이 모델은 하나의 foundational '레고 블록'처럼 작용하여, AI 분야를 재편하고 완전히 새로운 세대의 모델들을 탄생시키는 기폭제

6.2 Transformer 기반 모델의 탄생

- Transformer 아키텍처의 모듈성(modularity) 덕분에 연구자들은 인코더와 디코더 같은 구성 요소를 다양한 방식으로 조합하여 사용할 수 있게됨.
- **인코더-온리 (Encoder-only) 모델:** 이 모델들은 텍스트의 의미를 깊이 있게 파악하는 자연어 이해(NLU) 과제에 최적화. 문장 전체를 양방향에서 동시에 바라보는 구조는 문맥을 완벽하게 이해해야 하는 과제에 이상적. 대표적인 예로 **BERT(Bidirectional Encoder Representations from Transformers)**.
- **디코더-온리 (Decoder-only) 모델:** 이 모델들은 텍스트 생성(NLG) 과제에 특화되어 있습니다. 한 번에 한 단어씩 왼쪽에서 오른쪽으로 생성해 나가는 자기회귀(auto-regressive) 방식은 일관성 있는 문장을 만드는 데 자연스럽게 부합. 가장 유명한 예는 **GPT(Generative Pre-trained Transformer)**와 그 후속 모델들.
- **인코더-디코더 (Encoder-Decoder) 모델:** 이 모델들은 입력 시퀀스를 출력 시퀀스로 변환하는 번역, 요약과 같은 시퀀스-투-시퀀스 과제에서 뛰어난 성능을 발휘. **T5**와 **BART**가 대표적인 예.

6.0 Transformer의 파급 효과: NLP 생태계의 변화

- **6.3 NLP를 넘어 다른 분야로의 확장**
- Transformer가 순차 데이터 내의 관계를 모델링하는 능력은 특정 도메인에 국한되지 않는 범용성을 가지고 있음이 입증.
- **컴퓨터 비전 (Computer Vision):** Transformer는 **비전 트랜스포머(Vision Transformer, ViT)**의 등장과 함께 컴퓨터 비전 분야에도 성공적으로 적용. ViT는 이미지를 여러 개의 작은 패치(patch)로 나눈 뒤, 이를 단어 시퀀스처럼 처리하여 이미지 분류 및 인식에서 놀라운 성능을 보여줌.
- **기타 분야:** 이 외에도 Transformer 아키텍처는 **오디오 및 음성 처리(Audio and Speech Processing)**, **신호 처리(Signal Processing)** 등 다양한 분야로 확장되어, 각 분야의 기존 모델들을 대체하며 새로운 표준으로 자리 잡음.