Transformer의 큰 그림 이해: 기술적 복잡함 없이 핵심 아이디어 파악하기

Hugman Sangkeun Jung · Follow

23 min read · Apr 5, 2024

(You can find the English version of the post at this link.)

Transformer는 AI에서 다루어지는 중요한 네트워크 아키텍처 중 하나입니다. 복잡하고 다양한 AI 모델들 사이에서 Transformer는 그 뛰어난 성능과 다재다능함으로 많은 주목을 받고 있습니다. 그러나 모든 기술이 그렇듯, Transformer 역시 그 내부의 복잡한 구조와 원리를 처음부터 완벽히 이해하기는 쉽지 않습니다. 그리고 모든 사람이 그렇게 할 필요도 없죠.

이를 운전에 비유하면 이해가 쉬울 것입니다. 운전자가 안전하고 효율적으로 차량을 운전하기 위해서는 반드시 차량의 엔진 구성이나 각 부품의 작동 원리를 완벽히 알 필요는 없습니다. 중요한 것은 차량의 운용 방법, 즉 가속 페달과 브레이크, 핸들의 조작 방법 등 차량을 실제로 운전하는 데 필요한 기능적인 측면을 이해하는 것입니다.

마찬가지로, Transformer를 활용하는 데 있어서도 모든 내부 구조와 복잡한 작동 원리를 깊이 파악하는 것보다는, 어떻게 활용하면 좋을지에 대한 기능적인 측면과 응용 방법에 초점을 맞추는 것이 더욱 실질적일 수 있습니다.

이번 포스트에서는 Transformer의 기능적인 측면, 즉 Transformer가 AI 소프트웨어에서 어떠한 역할을 하는지, 그리고 우리가 이를 어떻게 활용할 수 있는지에 대해 살펴보고자 합니다. Transformer의 상세한 내부 구조나 복잡한 수학적 원리 등은 다른 강의 시간이나 전문적인 자료를 통해 파악하는 것을 권장드립니다(향후 제가 아주 깊이 있는 Transformer 101 시리즈를 올리도록 하겠습니다).

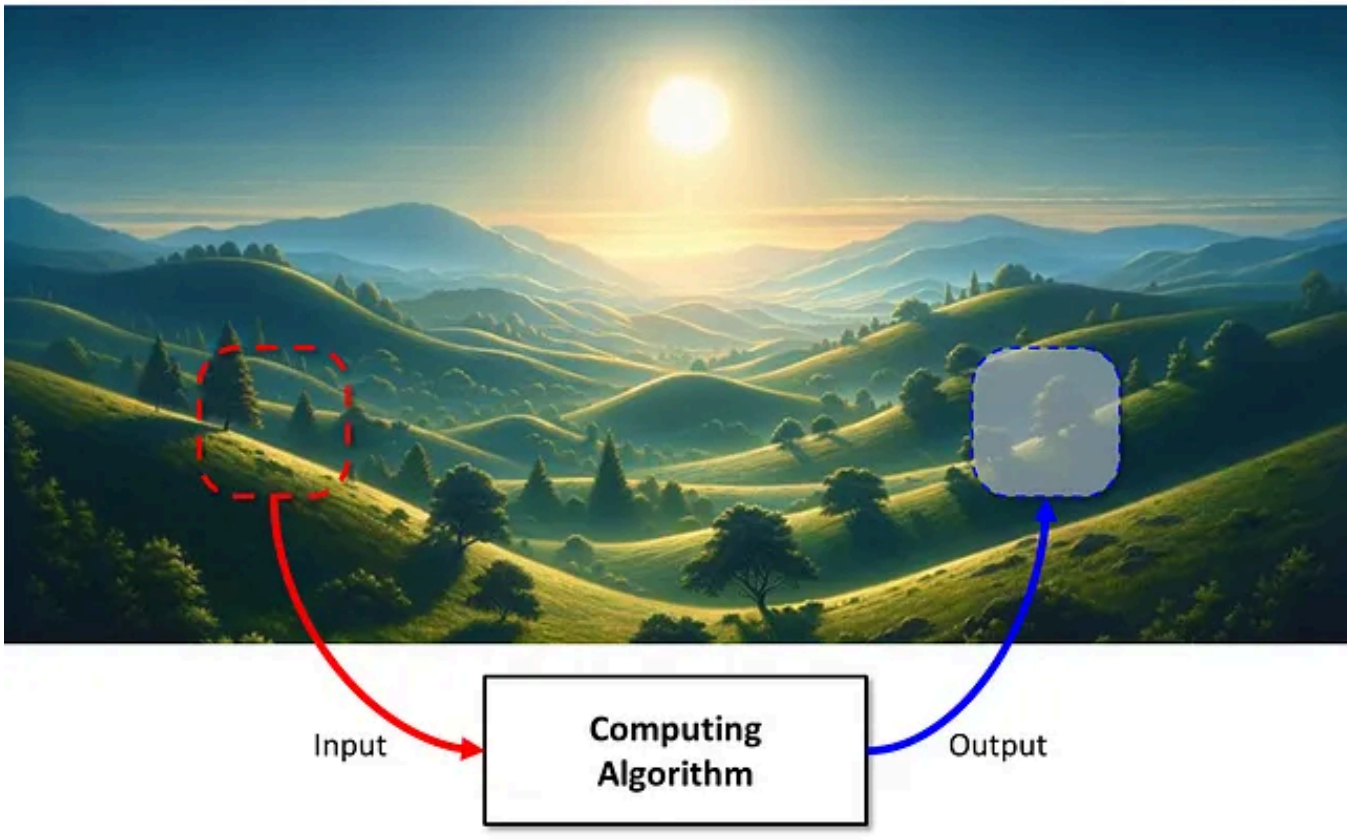
Transformer는 주로 자연어 처리(NLP) 분야에서 강력한 성능을 발휘하며, 문장의 의미를 파악하고, 새로운 문장을 생성하는 등의 작업에 뛰어난 능력을 보입니다. 그 외에도 이미지 처리, 음성 인식 등 다양한 AI 응용 분야에서도 그 가능성을 탐색하고 있습니다.

표현학습 및 Sequence to Sequence (Seq2Seq) 리마인드

Transformer 기술을 제대로 이해하려면 이 기술이 연계되어 있는 표현학습과 Sequence to Sequence learning(Seq2Seq)과의 연결점을 잘 파악해야 합니다. 잠시 시간을 내어 이에 대해 짚어보겠습니다.

세상을 표현하는 방법: 표현 학습

AI를 학습시키기 위해서는 먼저 세상을 기계가 이해할 수 있는 형태로 '표현'하는 것이 필수적입니다. 이 과정을 '표현 학습'이라고 하며, 데이터를 숫자, 벡터, 행렬 등의 수학적 객체로 변환해 기계가 이해할 수 있도록 만드는 것을 의미합니다. 이 과정을 통해 AI는 복잡한 현실 세계의 문제를 단순화된 형태로 인식하고 처리할 수 있게 됩니다.

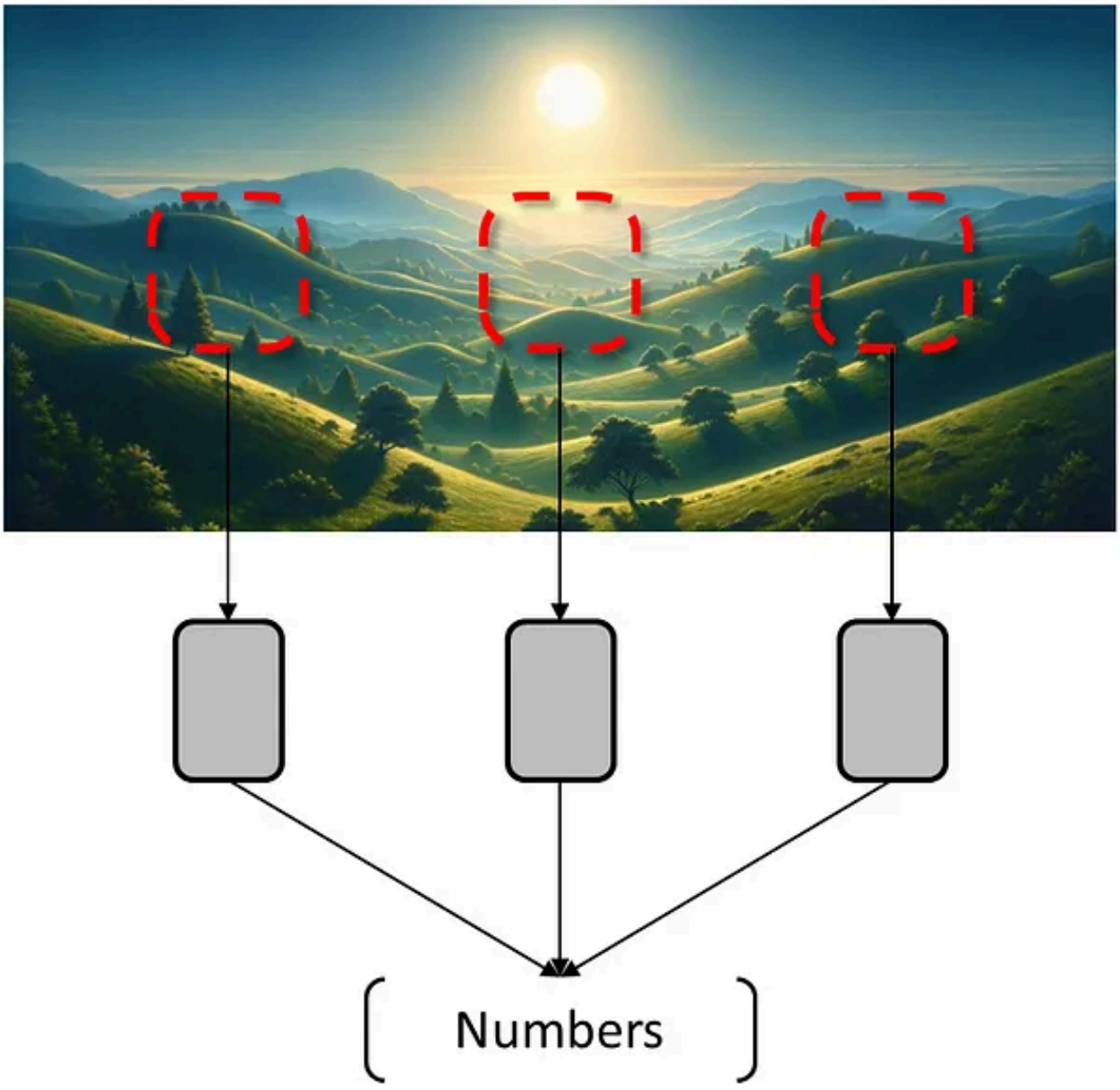


World Representation and Problem Solving

Seq2Seq: 복잡한 문제를 단순화하는 프레임워크

Seq2Seq는 이러한 '표현'을 바탕으로 하나의 시퀀스를 다른 시퀀스로 변환하는 과정을 학습하는 모델입니다. 이 프레임워크를 적용하면 일상생활의 대부분의 문제를 매우 단순화해서 인공지능을 통해 해결할 수 있습니다. 예를 들어, 한 언어로 된 문장을 다른 언어로 번역하는 과정에서 이 모델이 활용될 수 있습니다.

여기서 중요한 것은 복수의 아이템들(단어, 이미지의 픽셀, 음성 신호 등)을 하나의 유의미한 숫자들의 덩어리, 즉 '벡터'로 만드는 것입니다. 이 벡터는 해당 시퀀스의 '의미'를 담고 있으며, AI는 이를 바탕으로 새로운 시퀀스를 생성하게 됩니다.



Converting Multiple Items to Numerical Representations for Sequence-to-Sequence Learning

Seq2Seq 구현체의 역사

AI 기술의 발전에 따라, Seq2Seq 모델을 구현하는 방법도 크게 진화해왔습니다.

초기 Seq2Seq 구현체: RNN의 시대

초기 Seq2Seq 모델은 주로 순환 신경망(Recurrent Neural Network, RNN)을 사용하여 구현되었습니다. RNN은 시퀀스 데이터를 처리하는 데 강력한 능력을 보였으며, 많은 연구와 응용에서 중요한 역할을 했습니다.

그러나 RNN을 사용한 Seq2Seq 모델은 몇 가지 중대한 문제점에 직면했습니다. 이 중 가장 대표적인 문제는 바로 기울기 소실(Vanishing Gradient)과 기울기 폭발(Exploding Gradient) 문제였습니다. 이러한 문제는 신경망이 깊어질수록 또는 시퀀스가 길어질수록 그라디언트가 점점 소실되거나 폭발적으로 증가하는 현상을 말합니다. 이로 인해 모델의 학습이 어려워지거나 불안정해지는 문제가 발생했습니다. 또한, 장기 의존성(Long-Term Dependency) 문제로 인해 시퀀스의 앞 부분과 뒷부분 사이의 연관성을 효과적으로 학습하기 어렵다는 점도 큰 단점으로 지적되었습니다.

해결책: Attention 메커니즘의 등장

이러한 문제들을 해결하기 위해 **Attention Mechanism** 기술이 제안되었습니다. 어텐션은 모델이 입력 시퀀스의 중요 부분에 '집중'하여 필요한 정보를 선택적으로 추출하는 방법을 제공합니다. 이를 통해 모델은 전체 시퀀스를 일괄적으로 처리하는 대신 관련성이 높은 정보에 집중하여 효율적으로 처리할 수 있게 됩니다.

RNN + Attention: Seq2Seq 구현의 새로운 표준

Attention 메커니즘의 등장으로 RNN 기반 Seq2Seq 모델은 크게 개선되었습니다. RNN과 Attention이 결합된 형태는 2014년도부터 2017년도까지 Seq2Seq의 구현체로 광범위하게 사용되었습니다. 이 조합은 특히 기계 번역, 음성 인식, 텍스트 요약 등 다양한 분야에서 뛰어난 성능을 보였습니다.

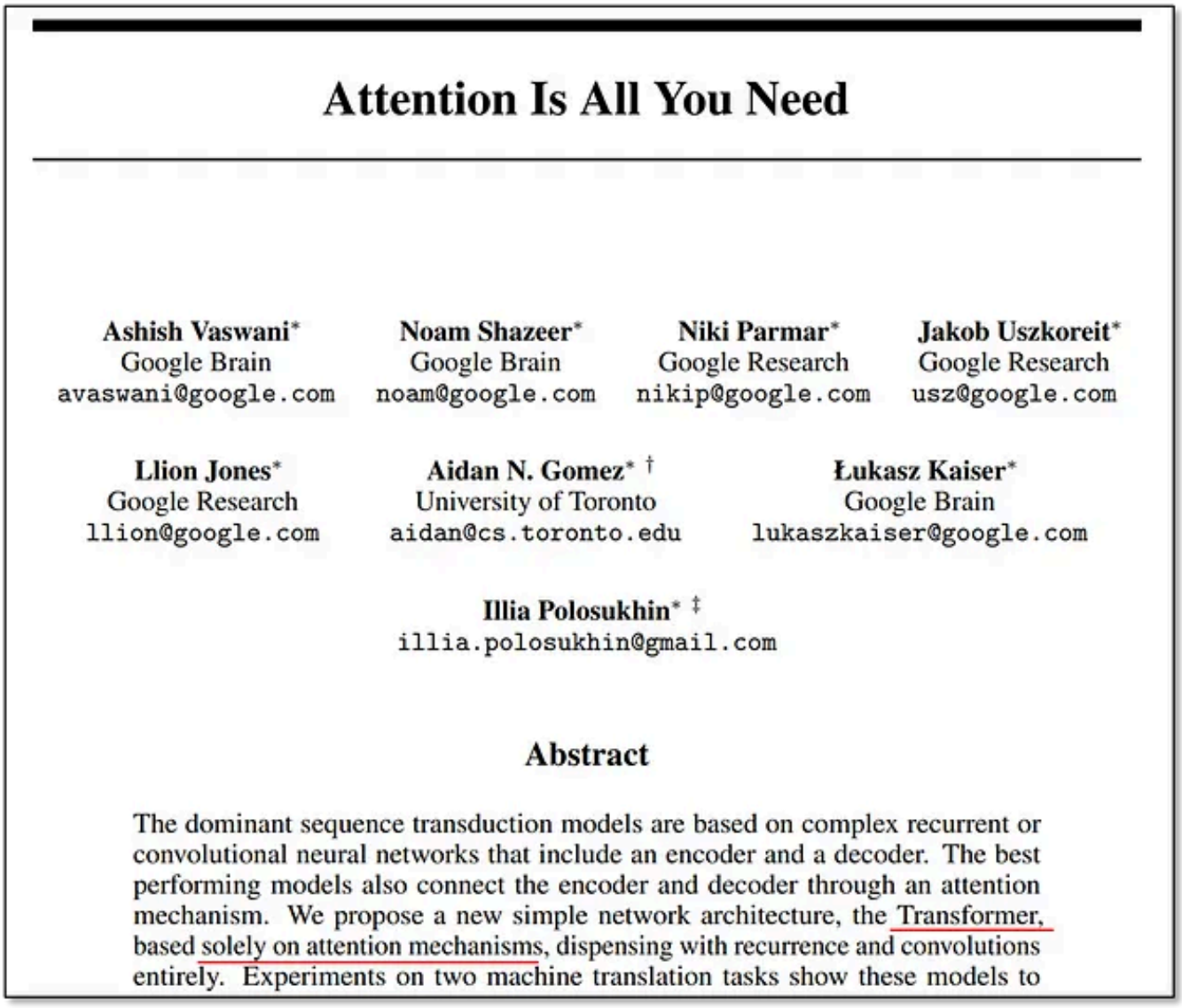
Transformer: Attention만으로 구현된 새로운 시대

Seq2Seq 구현의 패러다임을 완전히 바꾼 것은 바로 'Transformer' 아키텍처의 등장입니다. Transformer는 오로지 Attention 메커니즘만을 사용하여 Seq2Seq 문제를 해결하는 새로운 방식을 제시했습니다. 이 모델은 복잡한 RNN 구조를 제거하고, 전체 시퀀스를 한 번에 처리할 수 있는 구조로 설계되었습니다. 이로 인해 학습 속도가 크게 향상되었으며, 더 긴 시퀀스와 더 깊은 네트워크에 대한 처리가 가능해졌습니다. 특히 특정한 도메인이나 문제에 국한되는 네트워크 형태가 아니기 때문에 자연어 처리 문제 이외에도 다양한 문제에 범용적으로 적용 가능한 구조를 가지고 있습니다.

History of Sequence-to-Sequence Implementations (Image by the author)

Transformer가 풀고자 하는 문제

트랜스포머는 많은 이들에게 혁신적인 아키텍처로 인식되고 있습니다. 이 아키텍처는 "Attention is All You Need"라는 논문을 통해 처음 소개되었는데요, 논문 자체의 길이는 매우 짧지만 그 영향력은 매우 큽니다.



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

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to

Screenshot — "Attention is All You Need" (link)

논문의 저자들이 2017년 당시에 정확히 어떤 의도를 가지고 있었는지 알 수는 없지만, 지금 다시 이 논문을 자세히 뜯어보면서 미루어 짐작을 해보자면, 당시 저자들이 트랜스포머를 통해 해결하고자 했던 몇 가지 주요 문제는 아래와 같은 것 같습니다.

1. **Seq2Seq**를 위한 구조 제공: 트랜스포머는 시퀀스에서 다른 시퀀스로의 변환을 목표로 하는 Seq2Seq 모델링 문제에 대해 효과적인 구조를 제공하려고 했습니다.
2. 다수 항목 인코딩: 트랜스포머는 여러 항목을 어떻게 하나의 의미 있는 숫자 집합으로 인코딩할 수 있는가에 대한 해답을 제시합니다. 이는 자연어 처리뿐만 아니라 다양한 시퀀스 데이터를 다루는 데 있어 핵심적인 문제 중 하나입니다.
3. 장기 의존성 인코딩: 이전의 RNN이나 LSTM 같은 모델들이 장기 의존성 문제를 어느 정도 해결했지만, Transformer는 이를 효율적으로 처리하는 새로운 방법을 제시합니다.
4. 순차적 정보 인코딩: 시퀀스 내의 정보는 순서에 따라 의미가 달라질 수 있습니다. Attention 자체는 '순서'를 고려하는 메커니즘이 아니기 때문에, 이를 보완하는 방법을 트랜스포머에서는 이를 해결하는 방법을 제공합니다.
5. 빠른 인코딩: 트랜스포머는 병렬 처리를 통해 빠르게 데이터를 인코딩할 수 있도록 설계되었습니다. 이는 특히 대규모 데이터셋을 다룰 때 매우 중요한 특징입니다.
6. 단순한 아키텍처: 트랜스포머는 인코더와 디코더를 위한 단순하면서도 강력한 아키텍처를 제공합니다. 즉 한번의 코딩만 가지고도 소규모 모델 및 대규모 모델을 모두 대응할 수 있게 해두었습니다. 이 단순성은 모델의 이해와 확장, 그리고 적용을 용이하게 만들어줍니다.
7. 안정적인 학습: 트랜스포머는 모델 학습 중 발생할 수 있는 여러 문제들을 효과적으로 관리하고 해결하여 안정적인 학습 환경을 제공합니다. 이는 학습 과정의 수렴 속도를 개선하고 최적화 과정을 효율적으로 만들어 최종적인 모델 성능을 향상시킬 수 있습니다.

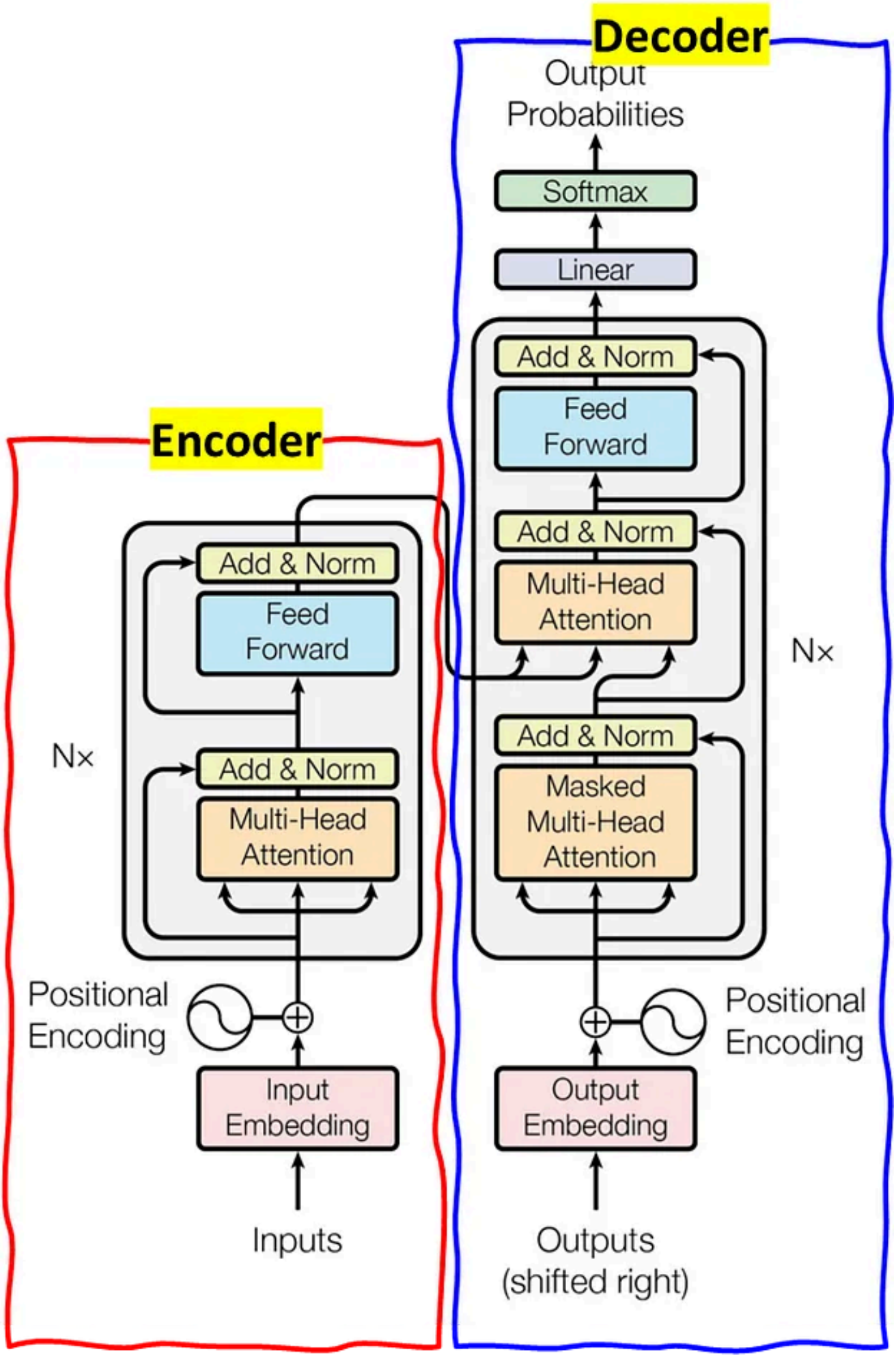
Transformer가 제시한 해법

자, 이제 위의 질문에 대한 트랜스포머의 내부 구조를 하나씩 살펴보겠습니다.

Seq2Seq를 위한 구조 제공

트랜스포머는 2017년 당시 RNN+Attention 기반의 seq2seq 구조를 대체하기 위해 만들어졌습니다. 특히 Encoder+Decoder 구조를 그대로 구현하도록 했습니다. 아래 그림을 살펴보세요. Encoder Part(왼쪽)와 Decoder Part(오른쪽)으로 구성된 것을 확인할

수 있습니다. 일단 내부에 어떤 블록들이 있는지는 우선 무시하고, 이 Encoder+Decoder 구조를 트랜스포머가 따르고 있다는 것만 이해하고 넘어가세요.



Transformer's Encoder-Decoder Structure for Sequence to Sequence Learning (as described in the paper)

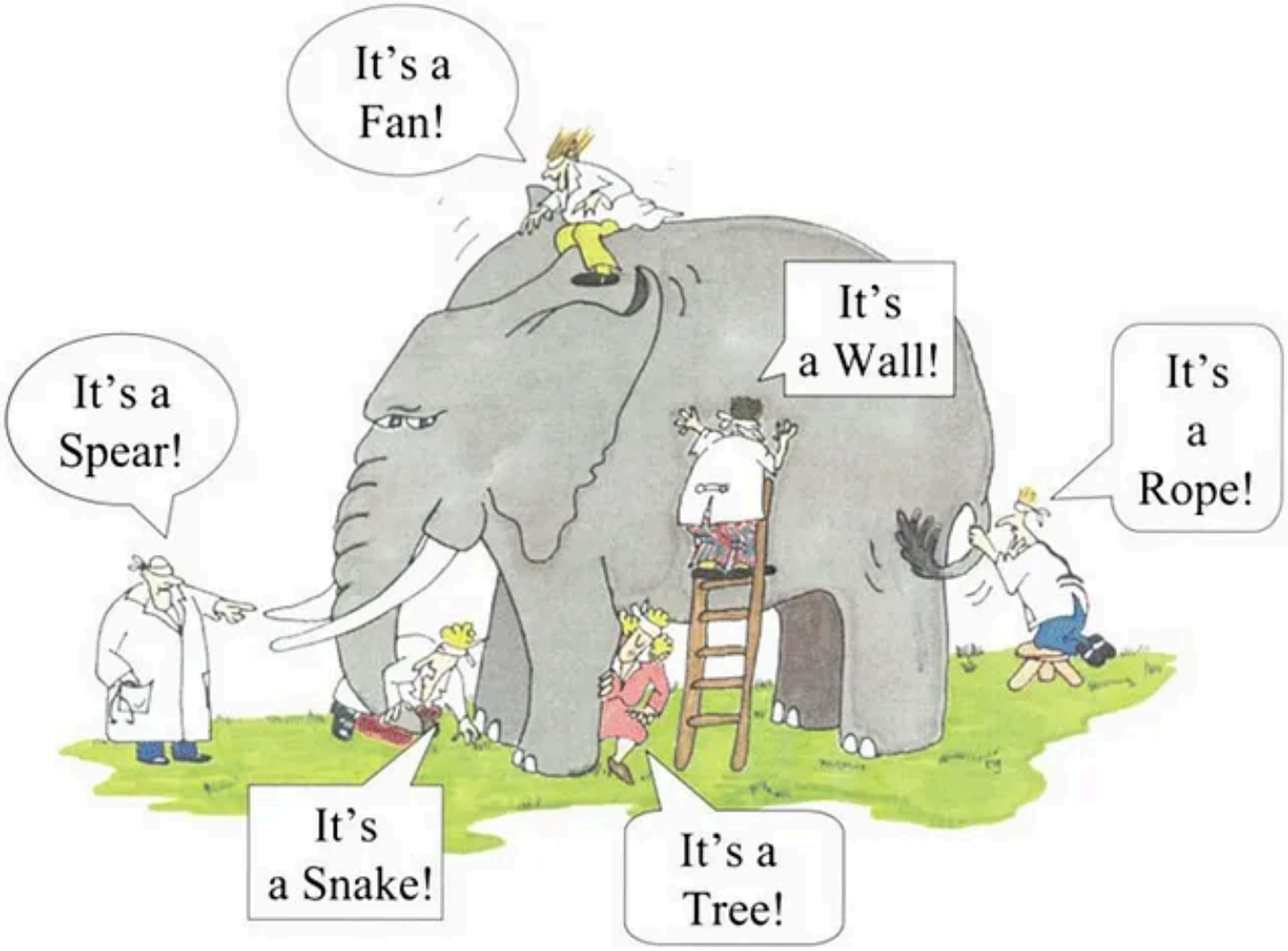
다수 항목 인코딩 + 장기 의존성 인코딩

우리가 살고 있는 4차원 공간(x, y, z, 시간)에서 발생하는 대부분의 데이터는 시계열 데이터라고 할 수 있습니다. 이러한 시계열 데이터는 여러 개의 연속된 데이터 포인트로 구성되며, 이 복수의 데이터 포인트를 효과적으로 인코딩하여 하나의 의미 있는 벡터로 표현하는 것이 매우 중요합니다. 이 과정은 다양한 AI 작업에서 기본이며 핵심적인 역할을 합니다.

이 복잡한 과제를 해결하기 위해 "Attention" 메커니즘이 이미 트랜스포머 이전에 개발되었습니다. Attention은 특정 데이터 포인트가 다른 데이터 포인트와 얼마나 관련되어 있는지를 파악하고, 이를 통해 입력 데이터의 전체적인 맥락을 파악할 수 있게 해줍니다. 이는 RNN과 같은 이전 모델들이 각 시점의 데이터를 순차적으로만 처리하는 것과는 대비됩니다. 특히 Attention 메커니즘은 전체 데이터 포인트를 고려하여(Global Encoding) 각 데이터 포인트의 중요도를 파악하고, 이를 기반으로 전체 데이터의 요약 벡터를 생성합니다. 트랜스포머는 이 Attention 메커니즘을 기반으로 하여, 복수의 데이터 포인트를 효과적으로 인코딩하고 장기 의존성을 처리하는 데 몇 가지 혁신적인 접근 방식을 제안합니다.

(아래 설명을 모두 이해하지 못해도 괜찮습니다. 이 포스트에서는 트랜스포머의 전체적인 활용 방법을 익히는 것이 중요합니다. 추후에 자세한 기술을 다루겠습니다.)

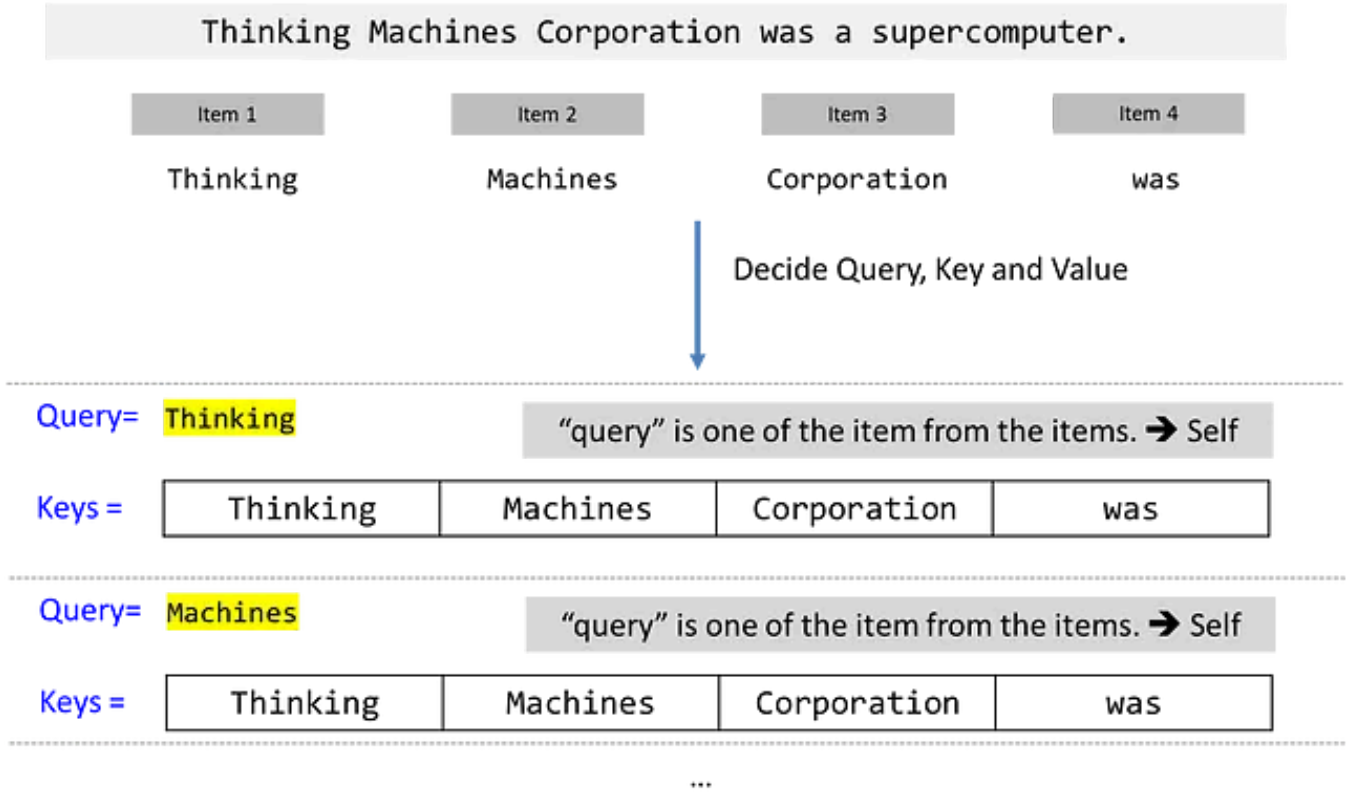
1. 멀티헤드 어텐션**(Multi-Head Attention):** 트랜스포머는 단일 어텐션 메커니즘 대신 멀티헤드 어텐션을 사용합니다. 이는 동일한 데이터에 여러 개의 어텐션 메커니즘을 병렬로 적용하여 다양한 방식으로 정보를 수집하고 분석할 수 있게 합니다. 이를 통해 모델은 데이터의 다양한 측면을 동시에 고려할 수 있으며, 결과적으로 더 풍부하고 다차원적인 데이터 표현을 얻을 수 있습니다.



Multi-Head Attention Provides 'Multiple Viewpoints' (Image Source : The Sufi elephant Parable)

2. 셀프 어텐션**(Self-Attention):** 셀프 어텐션은 입력 데이터 내의 각 요소가 서로 어떻게 상호 작용하는지를 모델링하는 방식입니다. 이 메커니즘은 입력 시퀀스 내의 각 요소(예: 단어나 픽셀 등)를 'Query'로 간주하고, 해당 'Query'와 입력 시퀀스 내의 모든 요소들('Keys')과의 관계를 평가합니다. 이 과정에서 각 'Key'에 대한 가중치(또는 '어텐션 점수')가 계산되며, 이를 통해 'Query'에 가장 관련이 높은 정보가 강조됩니다. 결과적으로, 각 요소는 다른 모든 요소와의 관계를 반영하여 자신의 표현을 조정하게 됩니다.

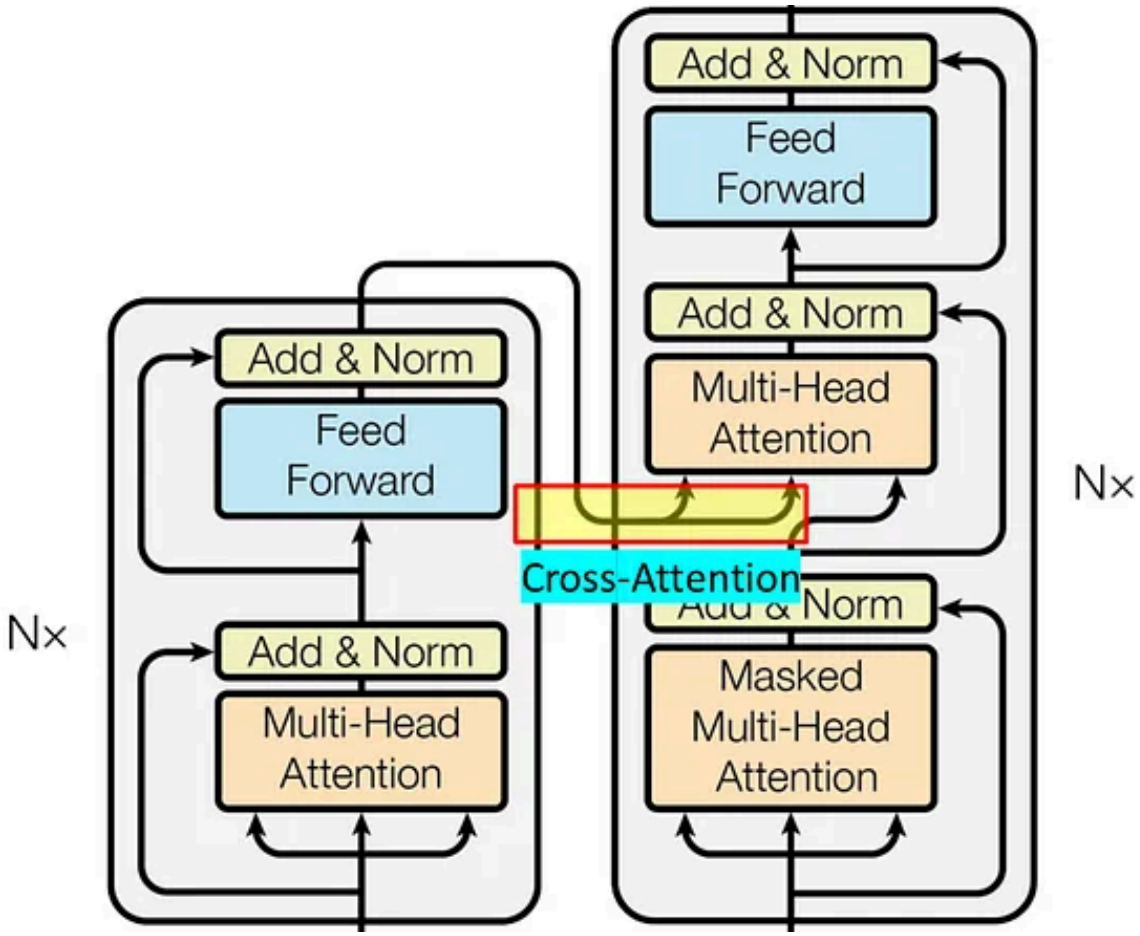
이러한 과정은 특히 장기 의존성 문제를 해결하는 데 유용하며, 시퀀스 내에서 멀리 떨어진 요소 간의 상호 작용도 효과적으로 포착할 수 있습니다. 셀프 어텐션은 'Query', 'Key', 'Value'라는 개념을 사용하여 내부적인 상호 작용을 모델링하며, 이러한 방식 때문에 '셀프 어텐션'이라는 명칭이 붙었습니다.



Self-Attention

3. 크로스 어텐션(**Cross-Attention**): 크로스 어텐션은 주로 트랜스포머 모델의 디코더 부분에서 활용되며, 인코더에서 생성된 표현과 디코더의 현재 입력 사이의 상호 작용을 모델링하는 데 사용됩니다. 이 메커니즘을 통해 디코더는 인코더가 제공하는 컨텍스트 정보를 바탕으로 출력 시퀀스의 다음 요소를 예측하며, 따라서 인코더의 정보를 효과적으로 활용할 수 있습니다. 이러한 과정은 특히 기계 번역과 같이 입력 시퀀스를 다른 형태의 출력 시퀀스로 변환하는 작업에 필수적입니다. 크로스 어텐션은 인코더와 디코더 사이의 '교차'되는 정보 흐름을 나타내며, 디코더의 'Query'가 인코더의 'Key'와 'Value'와 상호 작용하는 방식으로 구현됩니다.

이렇게 서로 다른 컴포넌트 간의 정보 교환 때문에 '크로스 어텐션'이라는 용어가 사용됩니다. 이러한 메커니즘은 다수의 데이터 포인트를 효과적으로 인코딩하고, 장기 의존성을 포함한 복잡한 시퀀스 구조를 이해하는 데 중요한 역할을 합니다.



Cross-Attention

순차적 정보 인코딩

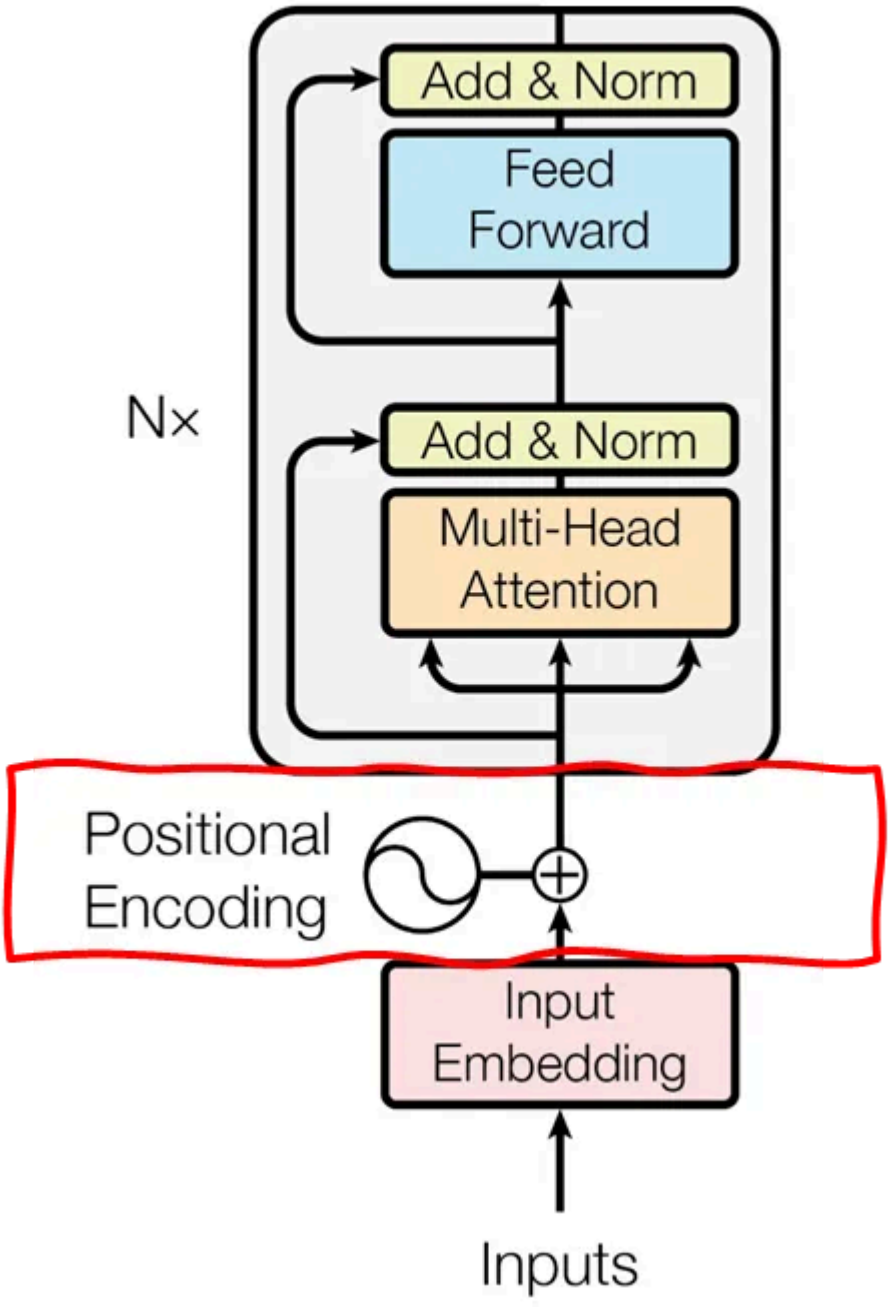
어텐션 메커니즘은 강력한 정보 처리 도구이지만, 그 자체만으로는 입력 데이터의 순서 정보를 반영하지 못하는 한계가 있습니다. 예를 들어, "나는 학교에 간다"와 "학교에 나는 간다"라는 두 문장이 주어졌을 때, 어텐션 메커니즘만을 사용하면 두 문장 사이의 순서 차이를 구분하지 못하고 유사한 결과를 도출할 가능성이 높습니다. 이는 특히 언어의 문법 구조와 같이 순서가 중요한 정보를 다룰 때 큰 문제가 될 수 있습니다.

이러한 문제를 해결하기 위해 트랜스포머는 입력 임베딩 단계에서 순서 정보를 추가하는 방식을 도입했습니다. 이 과정에서 사용되는 기술이 바로 포지셔널 인코딩(Positional Encoding)입니다.

포지셔널 인코딩

포지셔널 인코딩은 시퀀스의 각 요소 위치에 대한 정보를 임베딩 벡터에 추가함으로써 모델이 단어의 순서를 인식할 수 있도록 합니다. 원래 트랜스포머에서 제안된 포지셔널 인코딩 방법은 사인 함수와 코사인 함수의 조합을 사용하여 각 위치에 고유한 값을 할당합니다. 이 방식은 모델이 임의의 길이의 시퀀스를 처리할 수 있게 하며, 위치 정보를 효율적으로 모델에 통합할 수 있는 장점이 있습니다. 간단히 말하자면, 포지셔널 임

베딩은 위치 정보 t 가 주어졌을 때, 해당 위치에 대응하는 임베딩 벡터 E 를 반환하는 메커니즘입니다.



Positional Encoding

포지셔널 인코딩의 다양한 변형

그러나 이후의 연구에서는 기존의 포지셔널 인코딩 방식 외에도 여러 가지 대안적인 방법들이 제안되었습니다. 이러한 대안적인 방법들은 각기 다른 방식으로 순서 정보를 임베딩 벡터에 녹여내며, 특정 상황이나 데이터 유형에 더 적합할 수 있습니다.

1. 학습 가능한 포지셔널 인코딩(**Learnable Positional Encoding**): 이 방식에서는 포지셔널 인코딩 값을 모델 학습 과정에서 함께 학습하도록 설계합니다. 이를 통해 모델이 데이터에 특화된 위치 정보를 스스로 학습할 수 있게 됩니다.

2. 상대적 포지셔널 인코딩(**Relative Positional Encoding**): 단어 간의 절대적 위치 대신 상대적 위치 정보를 사용하는 방식입니다. 이는 특히 긴 시퀀스를 다룰 때 효과적일 수 있으며, 문맥에 따라 단어 사이의 관계를 더 유연하게 모델링할 수 있게 합니다.

이와 같은 다양한 포지셔널 인코딩 방법들은 트랜스포머 모델이 시퀀스 데이터의 순서 정보를 더 정확하게 이해하고 반영할 수 있도록 도와줍니다. 따라서 모델의 성능을 향상시키고, 보다 정교한 언어 처리 작업을 수행할 수 있게 됩니다.

빠른 인코딩

트랜스포머 아키텍처는 특히 그 빠른 인코딩 능력으로 주목받습니다. 기존의 순환 신경망(RNN)과 같은 모델들은 시퀀스의 각 요소를 순차적으로 처리해야 했기 때문에, 시간이 길어질수록 처리 시간이 기하급수적으로 증가했습니다. 이는 특히 긴 문서나 대화에서 매우 비효율적이었습니다.

RNN(Recurrent Neural Network)은 시퀀스 데이터를 처리할 때 각 시점(time step)의 데이터를 순차적으로 처리합니다. 이 구조 때문에 RNN은 한 번에 하나의 요소만 처리할 수 있으며, 다음 요소를 처리하기 전에 이전 요소의 처리가 완료되어야 합니다. 예를 들어, 문장 “The cat sat on the mat”을 처리할 때, “The”를 처리한 후 “cat”을 처리하고, 이후 “sat”을 처리하는 식으로 순차적으로 진행됩니다. 이 과정에서 각 단계마다 이전 단계의 정보가 다음 단계로 전달되며, 이로 인해 발생하는 시간 지연이 RNN의 주요 단점 중 하나입니다. 또한, 이러한 순차적 처리 방식은 GPU와 같은 병렬 처리 장치의 성능을 최대한 활용하기 어렵게 만듭니다.

반면에, 트랜스포머는 ‘어텐션 메커니즘’을 사용하여 시퀀스 내의 모든 요소를 동시에, 병렬적으로 처리합니다. 이는 각 요소가 독립적으로 다른 모든 요소와의 관계를 계산하고, 이 정보를 바탕으로 각 요소의 새로운 표현을 생성할 수 있음을 의미합니다. 따라서, 트랜스포머는 전체 문장이나 시퀀스를 한꺼번에 처리할 수 있으며, 이로 인해 처리 속도가 대폭 향상됩니다.

예를 들어, “The cat sat on the mat” 문장을 트랜스포머로 처리할 때, “The”, “cat”, “sat”, “on”, “the”, “mat”의 모든 단어를 동시에 처리할 수 있습니다. 각 단어는 다른 모든 단어와의 관계를 동시에 계산하고, 이를 바탕으로 문장 내에서의 자신의 역할과 의미를 파악할 수 있습니다. 이와 같은 병렬 처리 방식은 특히 GPU와 같은 병렬 처리 장치에서 트랜스포머 모델의 성능을 극대화할 수 있게 해줍니다.

단순한 아키텍처

트랜스포머는 인코더와 디코더를 위한 단순하면서도 강력한 아키텍처를 제공합니다. 즉 한번의 코딩만 가지고도 소규모 모델 및 대규모 모델을 모두 대응할 수 있게 해두었습니다. 이 단순성은 모델의 이해와 확장, 그리고 적용을 용이하게 만들어줍니다.

트랜스포머 아키텍처의 핵심은 '어텐션 메커니즘'을 중심으로 구성되어 있으며, 인코더와 디코더의 각 레이어가 이 메커니즘을 기반으로 동작합니다. 트랜스포머의 각 레이어는 크게 두 부분으로 나눌 수 있는데, 하나는 '멀티-헤드 어텐션(Multi-Head Attention)'이고 다른 하나는 '포지셔널 와이즈 피드 포워드 네트워크(Position-wise Feed-Forward Network)'입니다. 그리고 이 구조가 계속 레이어별로 반복되는 구조입니다.

이러한 구조는 다음과 같은 이유로 단순하다고 할 수 있습니다:

1. 모듈화: 트랜스포머의 각 부분(인코더와 디코더)은 재사용 가능한 모듈로 구성되어 있어, 필요에 따라 조합하거나 확장하기 쉽습니다. 예를 들어, 어텐션 레이어나 피드 포워드 네트워크를 추가하거나 줄이는 것만으로 모델의 용량을 조절할 수 있습니다.

2. 표준화된 레이어: 각 레이어는 크게 어텐션과 피드 포워드 네트워크로 이루어져 있으며, 이 레이어들은 비슷한 패턴을 따릅니다. 이로 인해 모델의 구조를 이해하기 쉬워지고, 개발자들이 쉽게 구현하고 수정할 수 있습니다.

3. 병렬 처리: 트랜스포머는 병렬 처리에 최적화된 구조를 가지고 있으며, 이는 모델의 학습과 추론 속도를 높여줍니다. RNN과 같은 순차적인 모델과 달리, 트랜스포머는 데이터의 각 부분을 동시에 처리할 수 있습니다.

4. 유연성: 트랜스포머는 다양한 크기와 형태의 입력 데이터를 처리할 수 있는 유연성을 제공합니다. 이는 포지셔널 인코딩을 통해 구현되며, 시퀀스의 길이에 상관없이 모델이 순서 정보를 효과적으로 활용할 수 있게 해줍니다.

이러한 특징들은 트랜스포머를 단순하면서도 강력한 도구로 만들어, 다양한 분야와 문제에 적용할 수 있는 범용성을 부여합니다.

안정적인 학습

트랜스포머 모델은 안정적인 학습을 가능하게 하는 여러 기술적 요소를 내포하고 있습니다. 특히, 트랜스포머 아키텍처는 '레이어 정규화(Layer Normalization)'와 '잔차 연결(Residual Connection)'을 통해 학습 과정을 안정화합니다. 이러한 구성 요소들은 다음과 같은 방식으로 안정적인 학습을 지원합니다:

1. 레이어 정규화(**Layer Normalization**): 각 레이어의 출력을 정규화하여 학습 과정에서의 파라미터 스케일과 이동 변화에 강인하게 만듭니다. 레이어 정규화는 각 레이어의 입력을 평균과 분산을 사용하여 정규화함으로써, 학습 과정에서 발생할 수 있는 수치적 불안정성을 줄이고 모델이 더 빠르게 수렴하도록 돕습니다.

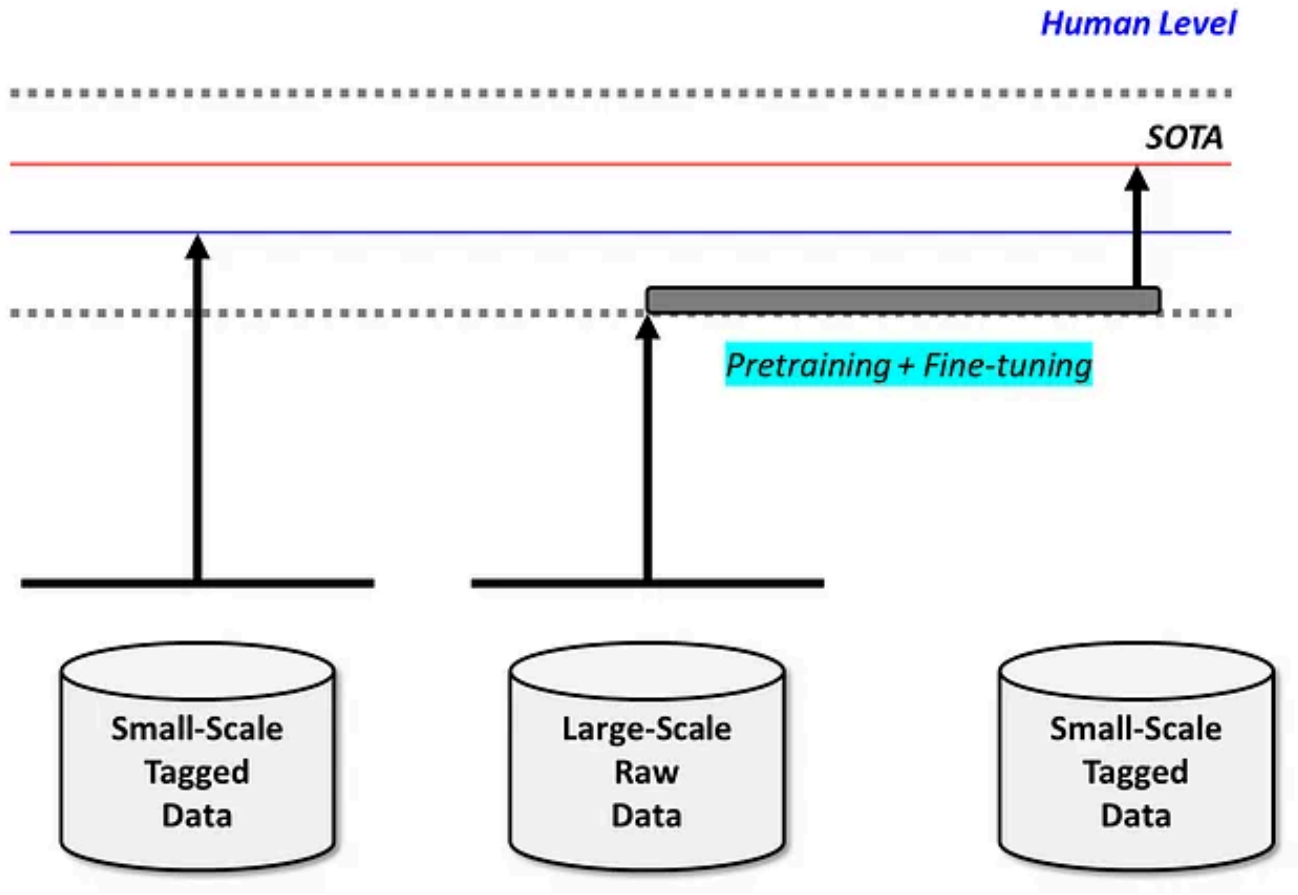
2. 잔차 연결(**Residual Connection**): 트랜스포머의 각 레이어는 입력값에 대한 출력값을 더하는 잔차 연결을 포함합니다. 이는 레이어를 거치면서 발생할 수 있는 정보의 손실을 방지하고, 깊은 네트워크에서 발생하기 쉬운 '소실된 기울기(Vanishing Gradient)' 문제를 완화합니다. 잔차 연결 덕분에, 깊은 네트워크에서도 각 레이어의 학습이 보다 원활하게 이루어질 수 있습니다.

3. 어텐션 메커니즘(**Attention Mechanism**): 트랜스포머의 핵심인 어텐션 메커니즘은 입력 시퀀스 내의 각 요소가 전체 시퀀스와 어떻게 상호 작용하는지를 동적으로 결정합니다. 이는 모델이 중요한 정보에 더 많은 '주의'를 기울이도록 하여, 불필요한 정보로 인한 학습의 방해를 줄이고, 학습 과정의 효율성을 높입니다.

이러한 요소들은 트랜스포머를 안정적으로 학습시키는 데 중요한 역할을 하며, 따라서 트랜스포머는 다양한 규모의 데이터셋과 복잡한 문제에 적용될 때 일관된 성능을 보여줄 수 있습니다.

Pretraining + Fine-Tuning 학습 패턴

트랜스포머 아키텍처를 더욱 강력하고 유력한 메커니즘으로 만들어준 것 중 하나는 사전 학습(Pretraining)과 미세 조정(Fine-Tuning)의 패턴 일반화입니다. 이 학습 패턴은 트랜스포머 모델을 다양한 문제에 적용하고 높은 성능을 달성하는 데 크게 기여했습니다.



Pretraining + Fine-tuning learning pattern (Image by the author)

트랜스포머의 범용성은 개발자들이 복잡한 신경망 아키텍처를 직접 설계하지 않고도, 기존에 사전 학습된 트랜스포머 모델을 적용하여 다양한 문제를 해결할 수 있도록 합니다. 이는 특히 자원과 시간이 제한된 상황에서 효율적인 방법을 제공합니다. 더욱이, 누군가가 대규모 데이터셋으로 오랜 시간에 걸쳐 사전 학습한 트랜스포머 모델을 공유하면, 이를 다운로드하여 자신의 관심 분야에 특화된 소규모 태그가 붙은 데이터셋으로 모델을 미세 조정함으로써, 안정적으로 높은 성능을 달성할 수 있습니다.

이러한 사전 학습 및 미세 조정 패턴은 매우 익숙한 학습 방법으로 자리 잡았으며, 실질적으로 인공 신경망의 표준화를 이루는 결과를 가져왔습니다. 이는 사실상의 표준(De facto standard)으로서, 트랜스포머가 다양한 분야에서 널리 채택되는 주요 이유 중 하나입니다.

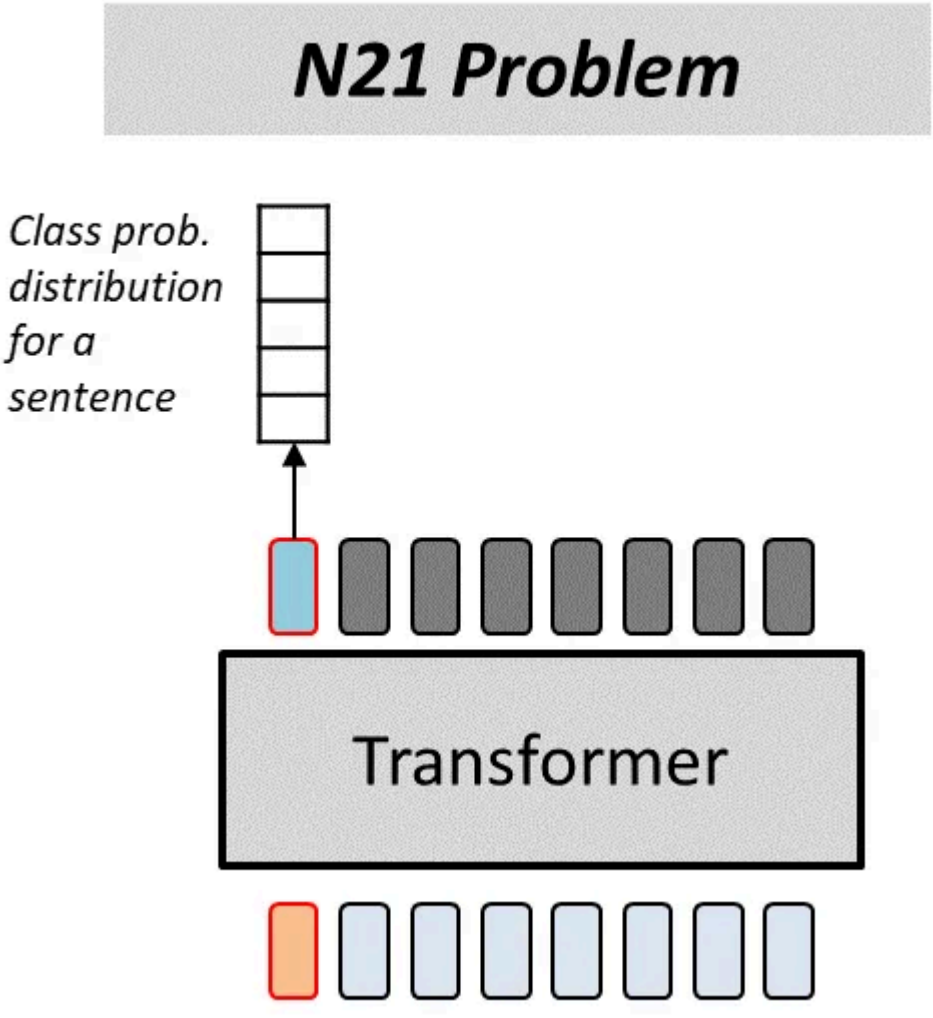
트랜스포머의 적용: 다양한 문제 해결을 위한 범용 아키텍처

트랜스포머는 그 유연성과 강력한 성능 덕분에 다양한 유형의 문제 해결에 적용될 수 있는 범용 아키텍처로 자리매김했습니다. 특히 N21, N2N, N2M과 같은 다양한 형태의 입력과 출력을 다루는 시퀀스 작업에 매우 효과적입니다.

N21 문제의 적용

N21 문제는 여러 개의 입력에서 단일 출력을 생성하는 경우를 말합니다. 예를 들어, 문장의 감정 분석은 각 단어(입력 시퀀스)를 분석하여 전체 문장의 감정(긍정적인지 부정적인지 등의 단일 출력)을 판단하는 N21 문제입니다.

- 예제: 고객 리뷰 감정 분석
- 입력: “이 제품은 정말 마음에 듭니다. 다음에도 구매할 예정입니다.”
- 출력: 긍정적

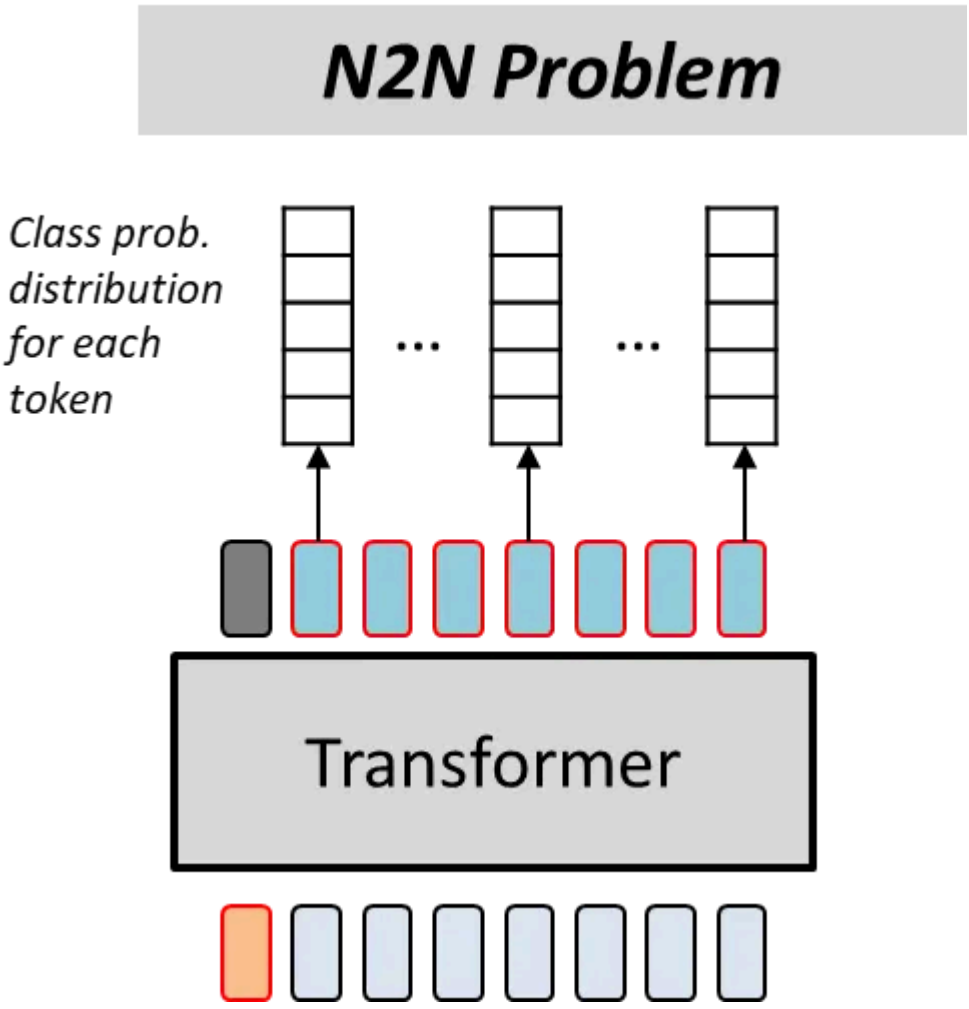


Transformer(Encoder) for N21 Problem (Image by the author)

N2N 문제의 적용 예

N2N 문제는 입력 시퀀스와 동일한 길이의 출력 시퀀스를 생성하는 경우입니다. 영어 문장의 품사 태깅(POS tagging)이 대표적인 예로, 하나의 문장을 구성하는 각 단어에 해당하는 품사를 태깅합니다.

- 예제: 품사 태깅(POS Tagging)
- 입력: “The transformer architecture is crucial for NLP.”
- 출력: [“The/DT”, “transformer/NN”, “architecture/NN”, “is/VBZ”, “crucial/JJ”, “for/IN”, “NLP/NNP”]



Transformer(Encoder) for N2N Problem (Image by the author)

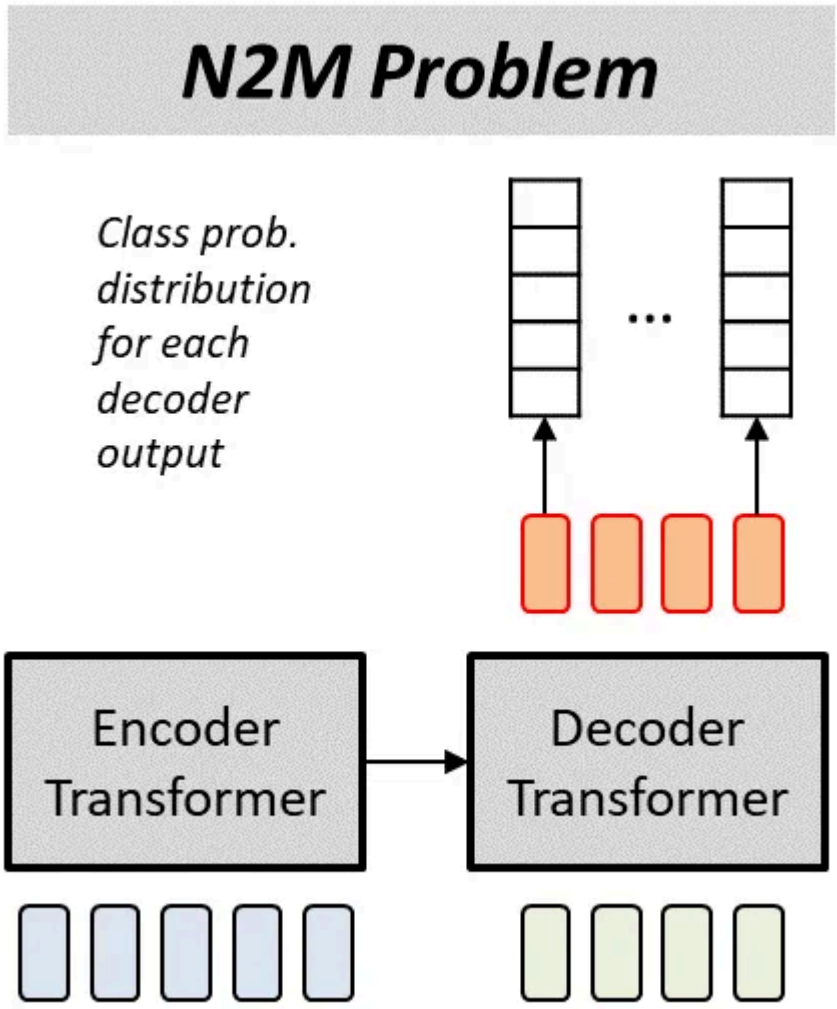
N2M 문제의 적용

N2M 문제는 입력 시퀀스의 길이와 출력 시퀀스의 길이가 다른 경우를 말합니다. 요약 작업 같은 것이 이에 해당되며, 긴 입력 문서를 짧은 요약문으로 변환합니다.

- 예제: 뉴스 기사 요약
- 입력: 긴 뉴스 기사 본문
- 출력: 뉴스의 핵심 내용을 담은 짧은 요약문

트랜스포머는 이러한 다양한 형태의 문제에 적용될 수 있도록 설계된 인코더와 디코더 구조를 가지고 있으며, 이를 통해 입력 데이터의 복잡한 패턴을 학습하고 적절한 출력

을 생성할 수 있습니다. 이러한 유연성이 트랜스포머를 NLP 분야를 넘어 다양한 분야에서 활용될 수 있는 강력한 도구로 만들어 줍니다.



Transformer(Encoder+Decoder) for N2M Problem(Image by the author)

결론

지금까지 우리는 트랜스포머 모델의 다양한 측면과 핵심 아이디어들을 깊게 살펴보았습니다. 이 글을 통해 명확해진 사실 하나는 트랜스포머가 현재까지 등장한 수많은 네트워크 아키텍처들 중에서도, 특히 시퀀스-투-시퀀스 문제 해결에 있어서 놀라운 성공을 거두고 있다는 점입니다.

트랜스포머 모델의 성공은 단순히 성능의 우수성에만 기인한 것이 아닙니다. 모델의 구조적인 간결함, 확장성, 그리고 다양한 문제에 대한 적용 가능성이 이 모델을 현 시점에서 가장 매력적인 선택지 중 하나로 만들어주고 있습니다. 또한, 사전 학습

(pretraining)과 미세 조정(fine-tuning)의 패러다임은 모델 학습 방법론에 혁명을 가져왔으며, 이는 트랜스포머를 넘어서 다양한 분야에까지 영향을 미치고 있습니다.

Transformers | Sequence To Sequence | Attention Mechanism | Encoder Decoder | Pretraining Finetuning

Follow

Written by Hugman Sangkeun Jung

139 Followers · 1 Following

Hugman Sangkeun Jung is a professor at Chungnam National University, with expertise in AI, machine learning, NLP, and medical decision support.

More from Hugman Sangkeun Jung