

# 생성형 AI

## Day 17

### 생성형AI I



## 목차

---

1. 생성형 AI – Generative AI
2. Transformer
3. Attention is All you need
4. 실습과제



# Generative AI

---

## 생성형 AI?

- 기존 데이터를 학습하여 새로운 데이터를 생성하는 인공지능 기술
- Diffusion, GAN, VAE, Transformer, ...

## 활용분야

- 이미지 생성: 예술 작품, 얼굴 이미지, 자연 경관 등
- 텍스트 생성: 소설, 시, 뉴스 기사 등
- 음악 생성: 새로운 음악 작곡, 편곡 등
- 비디오 생성: 새로운 영상, 애니메이션 등

## 생성형 AI가 주목받는 이유?

- 창의성 증대에 대한 기대 (디자인 및 예술 분야, 엔터테인먼트 산업)
- 데이터 증강 및 확장 (의료분야, 자율주행 등)
- 효율성 및 비용 절감 (데이터 수집 의존 하락 등)

# Generative AI

---

최근 생성형 AI?



ChatGPT



Midjourney

# Generative AI

---

## 대규모 언어 모델 (Large Language Model, LLM)

- 대규모 언어 모델(LLM, Large Language Model)은 방대한 양의 텍스트 데이터를 학습하여 자연어를 이해하고 생성할 수 있는 인공지능 모델
- OpenAI의 GPT-3, GPT-4

## 응용분야

- 자연어 생성(NLG)
- 질의응답(Q&A)
- 언어 번역
- 텍스트 요약
- 그외의 자연어와 관련된 작업



# Transformer

---

## Transformer 모델

- Transformer 모델은 2017년 Vaswani et al.에 의해 제안된 모델
- Attention is All You Need
- RNN이나 LSTM과 달리, Transformer는 순차적인 데이터 처리 없이 병렬로 처리 가능

## 기본 개념

### Self-Attention Mechanism

- 입력 데이터의 각 요소가 다른 모든 요소와의 관계를 고려하여 중요한 정보를 선택해 집중

### Encoder-Decoder 구조

- 두 개의 주요 구성 요소로 이루어져 있으며, 인코더는 입력을 처리하고 디코더는 출력을 생성

# Attention Is All You Need (2017)

---

## Introduction

### RNN과 LSTM

- 순환 신경망(RNN)과 장단기 기억 네트워크(LSTM), 게이트 순환 신경망(GRU)은 언어 모델링 및 기계 번역 같은 시퀀스 처리 문제에서 매우 효과적인 것으로 알려져 있음
- 입력 및 출력 시퀀스의 각 위치를 순차적으로 처리하여 hidden states를 생성
- 이러한 순차적 특성 때문에 병렬화가 어렵고, 긴 시퀀스를 처리할 때 메모리 제약이 발생

### 어텐션 메커니즘 (Attention Mechanism)

- 입력 및 출력 시퀀스 내의 요소 간의 의존성을 거리와 상관없이 모델링할 수 있게 함
- 대부분의 경우 어텐션 메커니즘은 순환 네트워크와 결합하여 사용되기 때문에 여전히 순차적 계산의 제약을 가지고 있음

### Transformer의 제안

- 본 논문에서는 순환 구조를 완전히 배제하고 어텐션 메커니즘에만 의존하는 Transformer 모델을 제안
- Transformer는 병렬화를 훨씬 더 많이 허용하여, 짧은 시간 내에 고품질의 출력물 획득 가능

# Attention Is All You Need

## Model Architecture

- 대부분의 신경 시퀀스 변환 모델과 마찬가지로, Transformer는 인코더-디코더 구조
- 인코더는 입력 시퀀스를 연속적인 표현 시퀀스로 맵핑하고, 디코더는 이를 바탕으로 출력 시퀀스를 생성합니다.
- 디코더는 이전에 생성된 기호를 추가 입력으로 사용하여 다음 기호를 생성

### 인코더 스택:

- 6개의 동일한 레이어로 구성
- 각 레이어는 멀티-헤드 self-attention 메커니즘과 위치별 완전 연결 피드-포워드 네트워크의 두 서브 레이어로 구성
- 각 서브 레이어는 잔차 연결을 통해 연결되며, 그 후 레이어 정규화 수행
- 모든 서브 레이어와 임베딩 레이어는 512차원의 출력을 생성

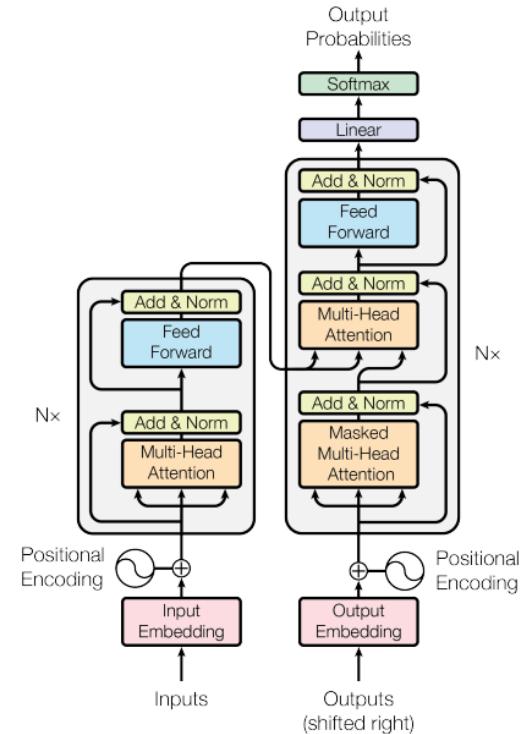


Figure 1: The Transformer - model architecture.

# Attention Is All You Need

## Model Architecture

### 디코더 스택

- 6개의 동일한 레이어로 구성
- 인코더 레이어의 두 서브 레이어 외에도, 인코더 스택의 출력을 대상으로 하는 세 번째 서브 레이어가 추가
- 디코더 스택의 self-attention 서브 레이어는 이후 위치를 참조하지 못하도록 마스킹되며, 출력 임베딩이 한 위치씩 오프셋되어 예측이 이전 위치의 출력에만 의존할 수 있도록 보장

### 어텐션 메커니즘

- 어텐션 함수는 쿼리, 키, 값 벡터( $Q$ ,  $K$ ,  $V$ )를 사용하여 쿼리와 키의 호환성 함수에 의해 계산된 가중치로 값들의 가중 합을 출력으로 매핑

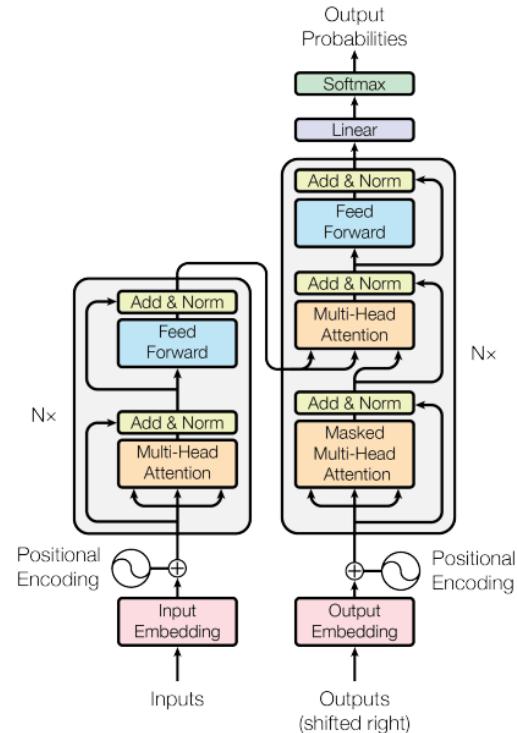


Figure 1: The Transformer - model architecture.

# Attention Is All You Need

## Model Architecture

### Scaled Dot-Product Attention

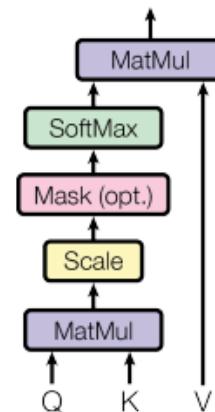
- 쿼리와 키의 내적을 계산하고, 이를  $\sqrt{d_k}$ 로 나누어 스케일링한 후, 소프트맥스 함수를 적용하여 값에 대한 가중치를 얻는 메커니즘
- 디코더 스택의 self-attention 서브 레이어는 이후 위치를 참조하지 못하도록 마스킹되며, 출력 임베딩이 한 위치씩 오프셋되어 예측이 이전 위치의 출력에만 의존할 수 있도록 보장
- 쿼리 집합 Q, 키 집합 K, 값 집합 V를 행렬로 묶어 동시에 처리
- $d_k$  값이 클수록 내적의 크기가 커져서 소프트맥스 함수가 매우 작은 기울기를 가지는 영역으로 밀려나게 되는데 이 문제를 해결하기 위해 내적을  $\sqrt{d_k}$ 로 스케일링하여 안정적인 학습을 보장

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

### Additive Attention vs. Dot-Product Attention

- Additive Attention
  - 피드-포워드 네트워크와 단일 은닉층을 사용하여 호환성 함수를 계산
  - 이론적으로는 DPA와 복잡도가 비슷하지만, 실제로는 느리고 공간 효율이 떨어짐
- 'Dot-Product Attention'
  - 쿼리와 키의 내적을 계산하여 가중치 계산
  - 고도로 최적화된 행렬 곱셈 코드를 사용하여 빠르고 공간 효율적임

### Scaled Dot-Product Attention



# Attention Is All You Need

## Model Architecture

### Additive Attention vs. Dot-Product Attention

#### - Additive Attention

피드-포워드 네트워크와 단일 은닉층을 사용하여 호환성 함수를 계산

이론적으로는 DPA와 복잡도가 비슷하지만, 실제로는 느리고 공간 효율이 떨어짐

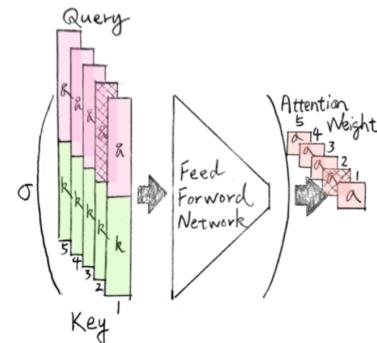
#### - 'Dot-Product Attention'

쿼리와 키의 내적을 계산하여 가중치 계산

고도로 최적화된 행렬 곱셈 코드를 사용하여 빠르고 공간 효율적임

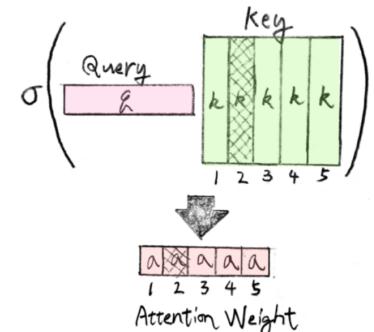
(Additive Attention)

$$\text{softmax}(\text{FFN}([Q; K]))$$



(Dot-Product Attention)

$$\text{softmax}(QK^T)$$



# Attention Is All You Need

## Model Architecture

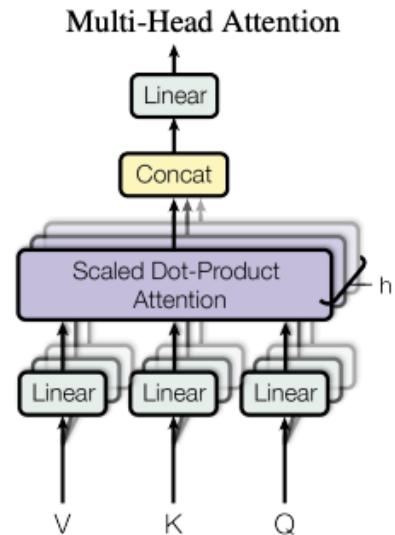
### 멀티-헤드 어텐션(Multi-Head Attention)

- 멀티-헤드 어텐션은 쿼리, 키, 값 벡터를 여러 번 선형 투영한 후 병렬로 어텐션 함수를 수행하여 다양한 표현 하위 공간의 정보를 통합하는 메커니즘
- 여러 어텐션 헤드를 사용하여 모델이 서로 다른 위치에서 다양한 정보를 볼 수 있게 함
- 단일 어텐션 헤드에서는 평균화가 이를 방해함

구성:

투영과 병렬 처리: 쿼리, 키, 값을 각각  $d_q, d_k, d_v$  차원으로 투영하고, 병렬로 어텐션 함수를 수행

출력 결합: 각 헤드의 출력을 연결하고 다시 투영한 값을 최종 출력 삼음



# Attention Is All You Need

## Model Architecture

Transformer에서의 멀티-헤드 어텐션 응용

### - 인코더-디코더 어텐션

- 쿼리는 이전 디코더 레이어에서 나오고, 메모리 키와 값은 인코더의 출력
- 디코더의 모든 위치가 입력 시퀀스의 모든 위치를 참조할 수 있음
- 기존 시퀀스-투-시퀀스 모델의 인코더-디코더 어텐션 메커니즘과 유사

### - 인코더의 Self-Attention

- 키, 값, 쿼리가 모두 인코더의 이전 레이어의 출력을 사용
- 인코더의 각 위치가 이전 레이어의 모든 위치를 참조할 수 있음

### - 디코더의 Self-Attention

- 디코더의 각 위치가 해당 위치까지의 모든 위치를 참조할 수 있음
- 디코더에서 왼쪽으로의 정보 흐름을 방지하기 위해 소프트맥스에 입력되는 값에 마스킹
- Scaled Dot-Product Attention 내에서 마스킹을 적용하여 구현

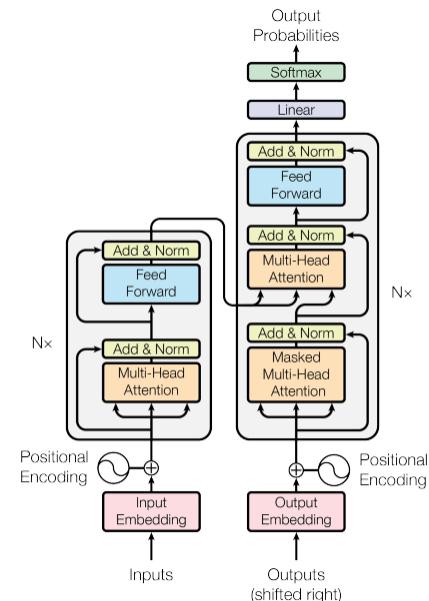


Figure 1: The Transformer - model architecture.

# Attention Is All You Need

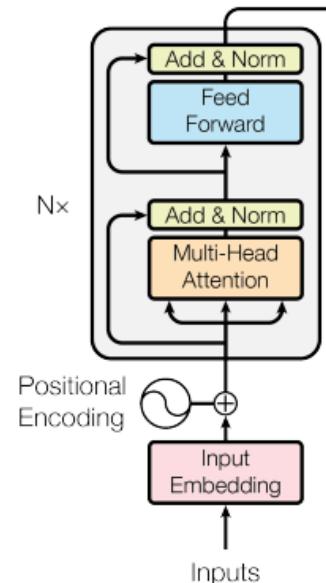
## Model Architecture

### Position-wise Feed-Forward Networks:

- 인코더와 디코더의 각 레이어에는 각 위치에 대해 동일하게 적용되는 완전 연결 Feed-Forward Networks 가 포함
- 두 개의 선형 변환과 ReLU 활성화 함수로 구성, 입력과 출력의 차원은  $d_{model} = 512$ , 내부 레이어의 차원은  $d_{ff} = 2048$

### Learned Embedding:

- 입력 토큰과 출력 토큰을  $d_{model}$  차원의 벡터로 변환
- 디코더 출력을 예측된 다음 토큰 확률로 변환하기 위해 선형 변환과 소프트맥스 함수를 사용
- 두 임베딩 레이어와 사전 소프트맥스 선형 변환 사이에 동일한 가중치 행렬을 공유하며, 임베딩 레이어에서는 이 가중치를  $\sqrt{d_{model}}$ 로 곱합니다.



# Attention Is All You Need

## Model Architecture

### Positional Encoding

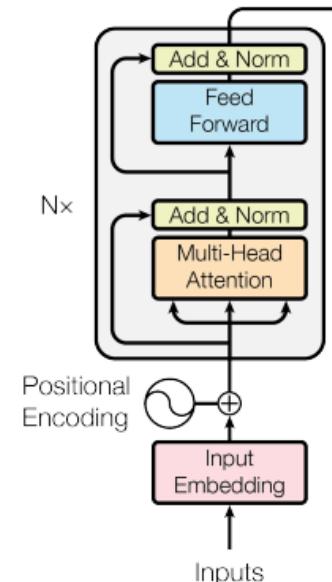
- 모델이 시퀀스의 순서 정보를 활용할 수 있음
- 인코더와 디코더의 입력 임베딩에 추가
- 위치 인코딩은 임베딩과 동일한 차원  $d_{model}$

### 사인 및 코사인 함수 사용

- 위치 인코딩은 사인함수와 코사인함수로 계산
- 파장은  $2\pi$ 에서  $10000 \cdot 2\pi$ 까지 기하급수적으로 증가
- 상대적 위치에 따라 주목(attend)하는 것을 쉽게 학습할 수 있도록 설계됨

### 학습된 위치 임베딩과 비교(learned positional embeddings)

- 사인함수 위치 인코딩과 학습된 위치 임베딩이 거의 동일한 결과를 나타냄
- 사인함수 위치 인코딩이 더 긴 시퀀스 길이에 대해 외삽할 수 있음



# Attention Is All You Need

---

## Why Self-Attention

Self-Attention vs. 순환 및 합성곱 레이어 비교

- 레이어당 계산 복잡성에 이점
- 최소한의 순차적 연산 수로 계산 과정을 병렬화 가능
- 장거리 의존성 경로 길이에 큰 이점

Self-Attention의 장점:

- Self-Attention 레이어는 모든 위치를 일정한 수의 순차적 연산으로 연결, 순환 레이어는  $O(n)$  순차적 연산 필요
- 시퀀스 길이 n이 표현 차원 d보다 작을 때 Self-Attention 레이어의 계산 속도가 순환 레이어보다 빠름
- 매우 긴 시퀀스에서 Self-Attention은 크기 r의 이웃만을 고려하도록 제한 하는 것이 가능 -> future work

합성곱 레이어의 한계:

- 커널 폭이 작은 단일 합성곱 레이어는 모든 입력 및 출력 위치 쌍을 연결하지 않기 때문에  $O(n/k)$  합성곱 레이어 스택 필요
- 일반적으로 합성곱 레이어는 순환 레이어보다 커널만큼 더 비쌈

Self-Attention의 추가 이점

- Self-Attention은 더 해석 가능한 모델을 제공할 수 있어보임
- 모델의 어텐션 분포를 검사하여 문장의 구문 및 의미 구조 파악

# Attention Is All You Need

---

## Training

### 학습 데이터 및 배치

- WMT 2014 영어-독일어 및 영어-프랑스어 데이터셋 사용
- byte-pair encoding 및 word-piece 어휘 사용
- 각 배치당 25000개의 소스 및 타겟 토큰 포함

### 하드웨어 및 일정

- 8개의 NVIDIA P100 GPU에서 학습
- 기본 모델: 총 100,000steps 또는 12시간 학습
- 큰 모델: 총 300,000steps 또는 3.5일 학습

### 최적화기

- Adam optimizer 사용
- 첫 warmup\_steps 동안 학습률 선형 증가, 이후 역제곱근에 비례하여 감소

### 정규화

- Residual Dropout: 0.1 사용
- Label Smoothing: 0.1 사용

# Attention Is All You Need

---

## Results

기계 번역:

- 영어-독일어 번역: BLEU 점수 28.4를 달성
- 영어-프랑스어 번역: BLEU 점수 41.0을 기록하여 이전의 모든 단일 모델을 능가, 학습 비용은 이전 모델의 1/4에 불과

모델 변형(Model Variations):

- 구성 요소 평가: 어텐션 헤드 수와 어텐션 키 및 값의 차원을 변경하여 성능 변화를 관찰
  - 단일 헤드 어텐션은 최상의 설정보다 0.9 BLEU 낮았으며, 너무 많은 헤드가 있는 경우 오히려 품질이 저하
- 어텐션 키 크기: 어텐션 키 크기  $d_k$  를 줄이면 모델 품질이 저하
- 모델 크기: 큰 모델이 더 좋은 성능을 보이며, 드롭아웃이 과적합을 피하는 데 매우 유용
- 위치 인코딩: 사인파 위치 인코딩을 학습된 위치 임베딩으로 교체했을 때 기본 모델과 거의 동일한 결과

영어 구문 분석:

- Penn Treebank의 WSJ 부분에서 4-layer 트랜스포머를 학습, 약 4만 개의 학습 문장 사용
- 작업별 튜닝 없이도 놀라운 성능을 보여주며, RNN Grammar를 제외한 모든 이전 모델보다 더 나은 결과

# Attention Is All You Need

---

## Conclusion

### 트랜스포머 모델 소개:

- 완전한 어텐션 기반 모델: 트랜스포머는 인코더-디코더 아키텍처에서 일반적으로 사용되는 순환 레이어를 다중 헤드 self-attention으로 대체한 최초의 시퀀스 변환 모델

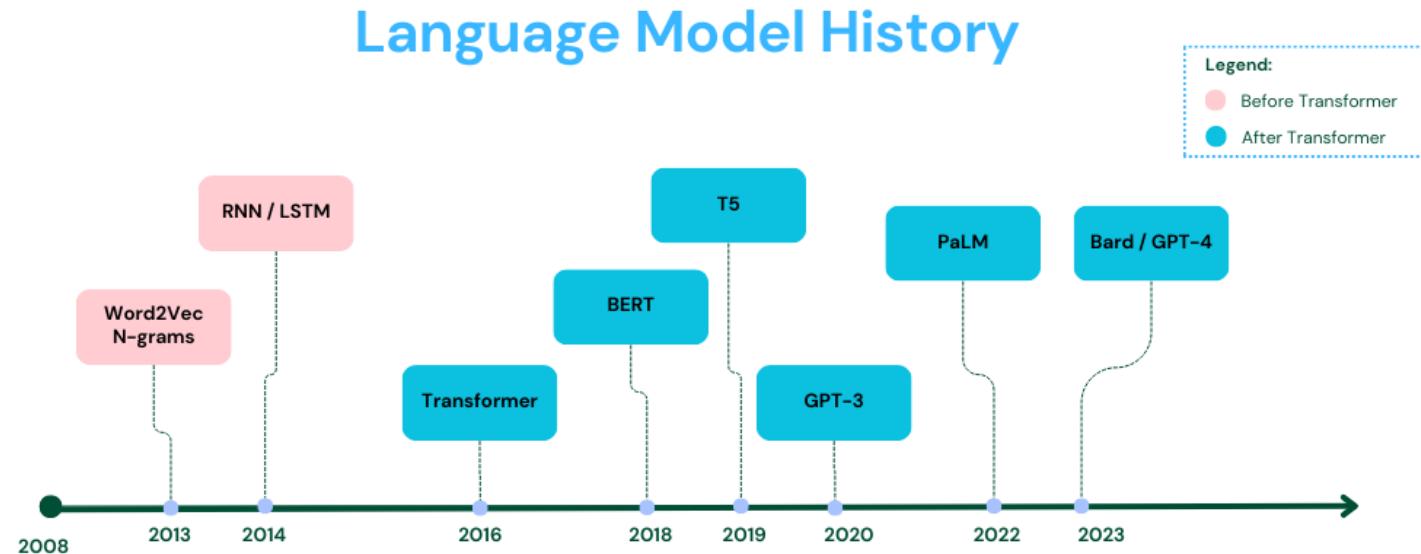
### 번역 작업에서의 성능

- 더 빠른 학습: 트랜스포머는 순환 또는 합성곱 레이어 기반 아키텍처보다 훨씬 빠르게 학습 가능
- 최첨단 성능: WMT 2014 영어-독일어 및 WMT 2014 영어-프랑스어 번역 작업 모두에서 새로운 최고 성능 달성
- 최고 모델 성능: 최고 성능의 Transformer모델은 이전에 보고된 모든 양상들보다도 뛰어난 성능을 보임

### Future work

- 다양한 작업에 적용: 어텐션 기반 모델을 다른 작업에도 적용할 계획
- 다양한 모달리티 처리: 텍스트 외의 입력 및 출력 모달리티를 포함하는 문제로 트랜스포머를 확장할 계획
- 지역적으로 제한된 어텐션: 이미지, 오디오 및 비디오와 같은 대형 입력 및 출력을 효율적으로 처리하기 위해 지역적으로 제한된 어텐션 메커니즘을 조사할 계획
- 덜 순차적인 생성: 생성 과정을 덜 순차적으로 만드는 것도 우리의 연구 목표 중 하나

# Attention Is All You Need



## 실습 과제

---

### 1. Transformer 모델의 구조 이해

- 트랜스포머 모델의 인코더와 디코더의 구조를 도식화하고, 각 구성 요소의 역할을 요약하여 설명해보기

### 2. Attention 메커니즘의 역할

- Attention 메커니즘이 필요한 이유를 설명하고, Scaled Dot-Product Attention과 Multi-Head Attention의 차이점을 설명해보기

### 3. Positional Encoding

- Positional Encoding의 원리를 이해하고, 다른 위치 인코딩 방법들을 조사해보고 비교하여 설명해보기

### 4. 트랜스포머 모델의 학습 및 최적화

- Adam Optimizer, 학습률 스케줄링, 레이블 스무딩의 개념을 조사하고, 각 기법이 모델 학습에 어떻게 기여하는지 설명해보기

### 5. LLM모델의 평가 방법에 대한 이해

- BLEU scores이 무엇인지 이해하고, 다른 평가 방법에 대해 조사해보기

# 실습 진행