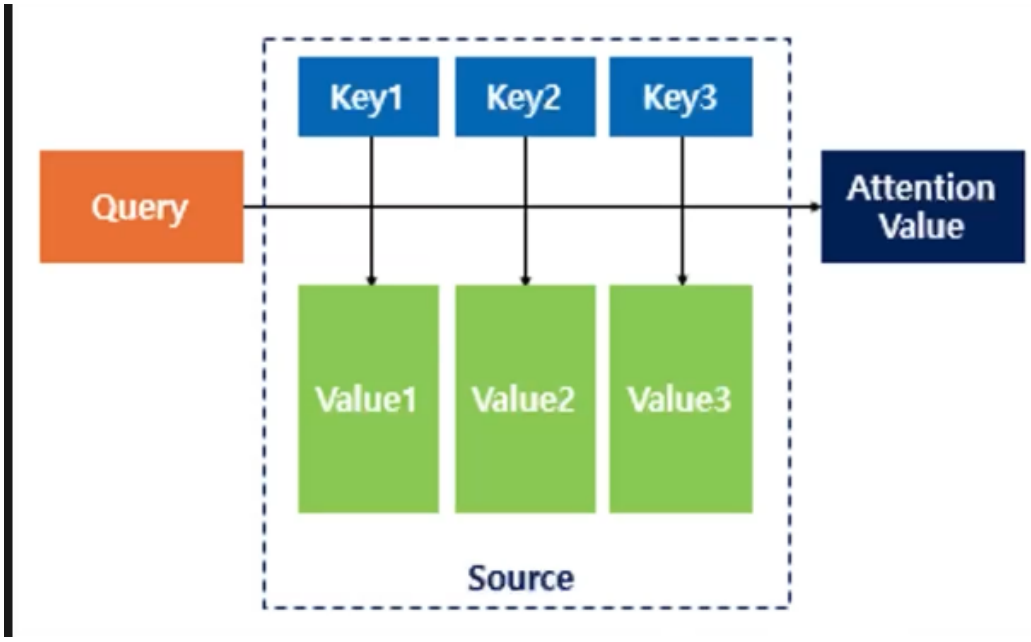
# LLM 작동 방식 및 원리

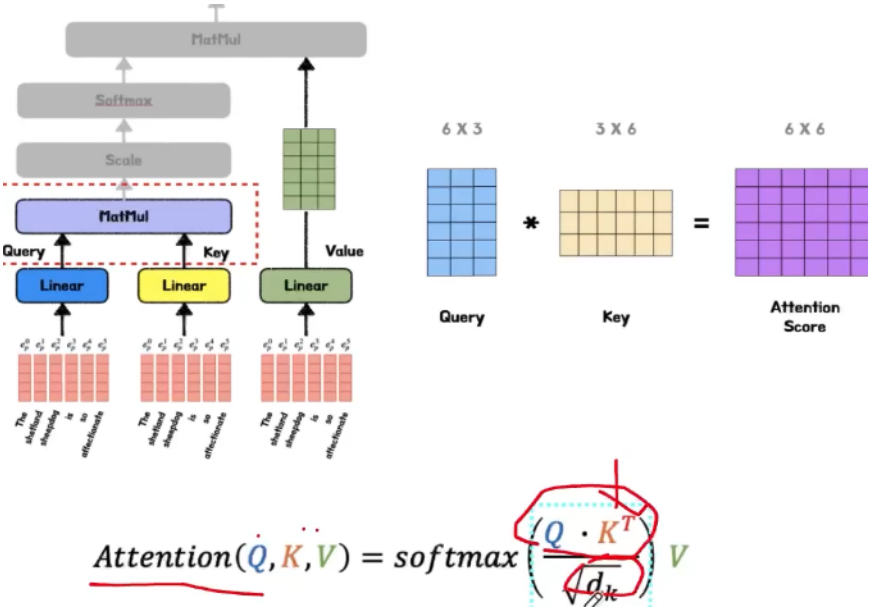
## LLM 구조 및 동작 방식

- **RNN 모델의 문제점**
  - 기울기 소실 / 폭발
    - 활성화 함수, 행렬 곱 연산 등에 의해 학습이 잘 진행되지 않음
  - 장기 의존성 문제
    - 은닉층의 과거 정보가 마지막까지 전달되지 않는 문제
- **Attention의 이해**
  - 입력 시퀀스에 대해서 각 부분에 대해서 유사도를 계산하여 어떤 부분이 더 중요한지 모델이 판단하게 함.
  - 어떤 부분에 집중할 지 찾아서 가중치를 계산해 전달 (혹은 활용)



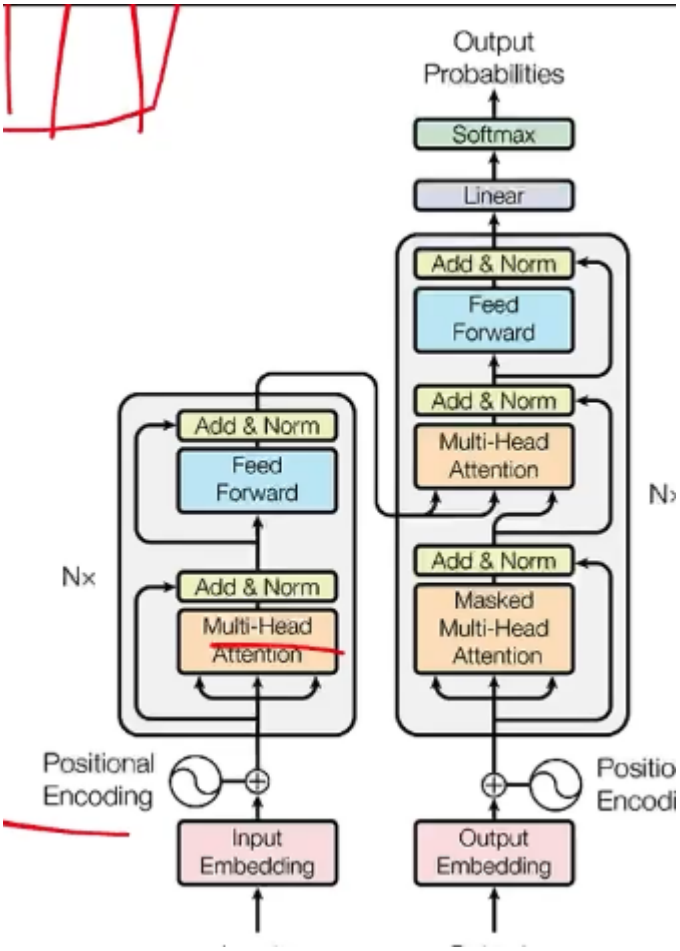

- **Encoder, Decoder**
  - 생성이란 영역에서는 Decoder가 필요
  - Encoder-Decoder는 구조가 복잡해짐
  - Decoder 만으로도 결과를 생성하는 것이 가능 → Decoder-olny 모델이 연구되고 있음 (LLM)

- **Transformer**
  - 다량의 데이터로 사전학습 시켰을 시 기존보다 향상된 성능을 보여줌.
  - 확장성이 뛰어남 (N 개의 계층을 쉽게 쌓을 수 있음 → 모델의 파라미터가 커진다.)
  - Attention Layer만을 활용해 모델을 구성
  - NLP, Vision 영역 등 다양한 영역에서 적용되고 뛰어난 성능을 보여줌.
- **어텐션 연산의 의미 파악**

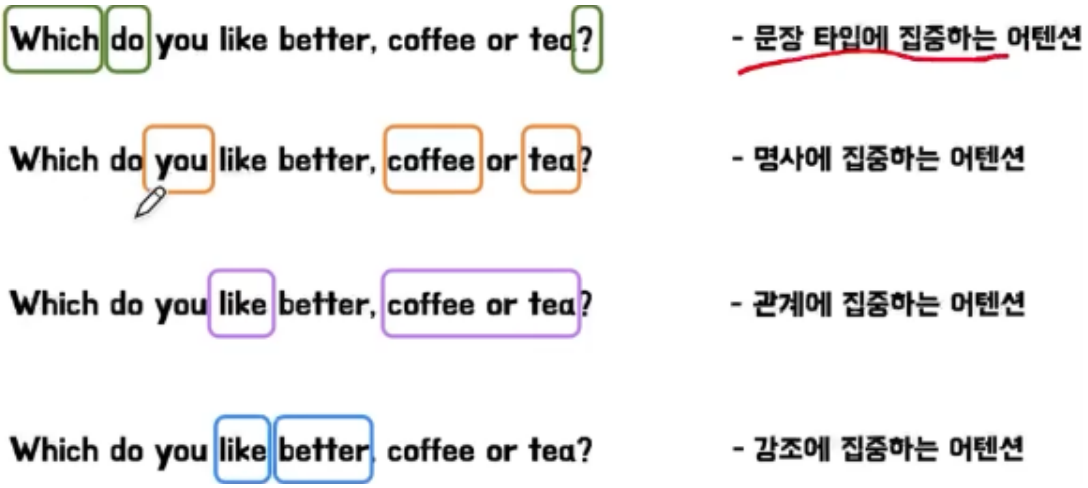



$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) V$$

  - 내적이란, 벡터의 크기를 반영한 유사도 연산이라고 볼 수 있음.
    즉, 둘의 연산은 각 토큰벡터 간의 유사도를 구하는 것이라 볼 수 있음.
  - 다만, 벡터의 내적을 그대로 활용하게 되면, 나중에 SoftMax 함수에서 특정 값만 과도하게 살아남는 것을 방지
    - SoftMax: Attention Score을 최종적으로 0~1의 값 사이로 변환해주기 위함.
- **Multi-haed attention**

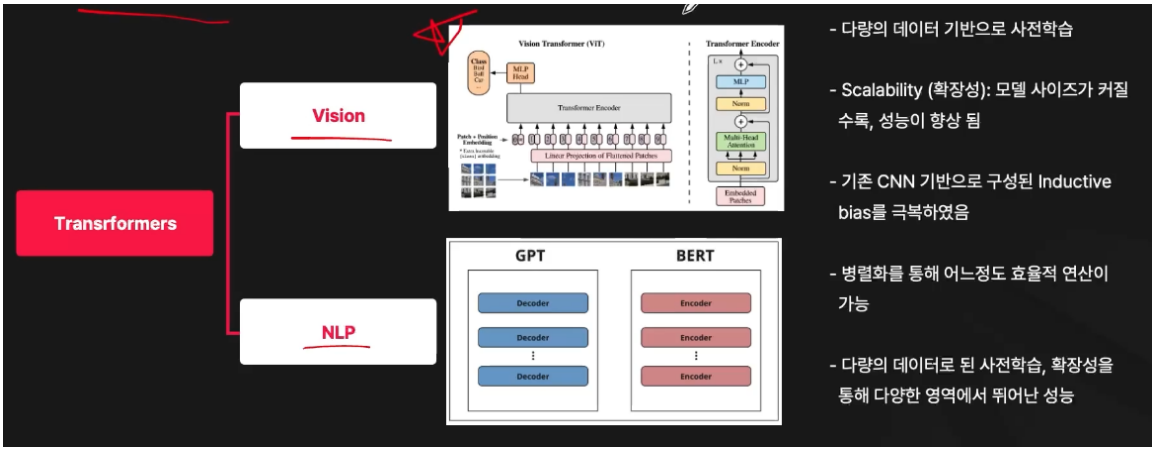



- 셀프 어텐션을 h번 병렬로 연산하는 구조
- 병렬로 연산하는 이유는, 문장 내에 다양한 관계와 의미를 파악해 문장을 이해하기 위해서
- 입력 토큰간의 복잡한 구조와 관계를 이해하는 데 매우 중요 (더 다양한 표현력을 위해 필요)



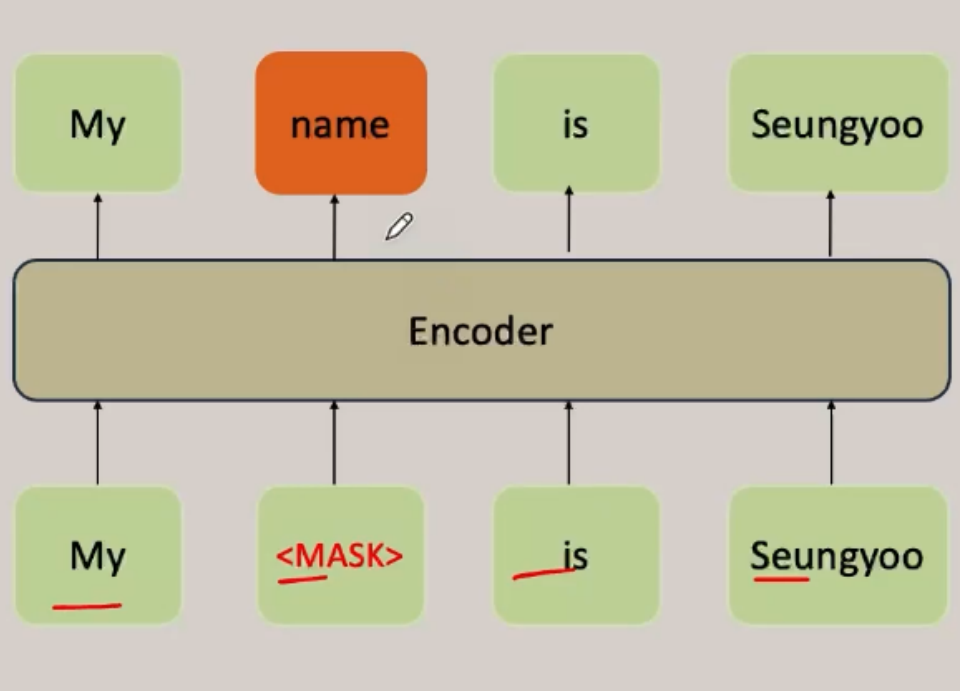

- **Transformer의 활용**

- 확장성: 모델 사이즈가 커질수록 성능이 향상됨
- Inductive bias를 극복
- 병렬화를 통해 상대적으로 효율적인 연산이 가능



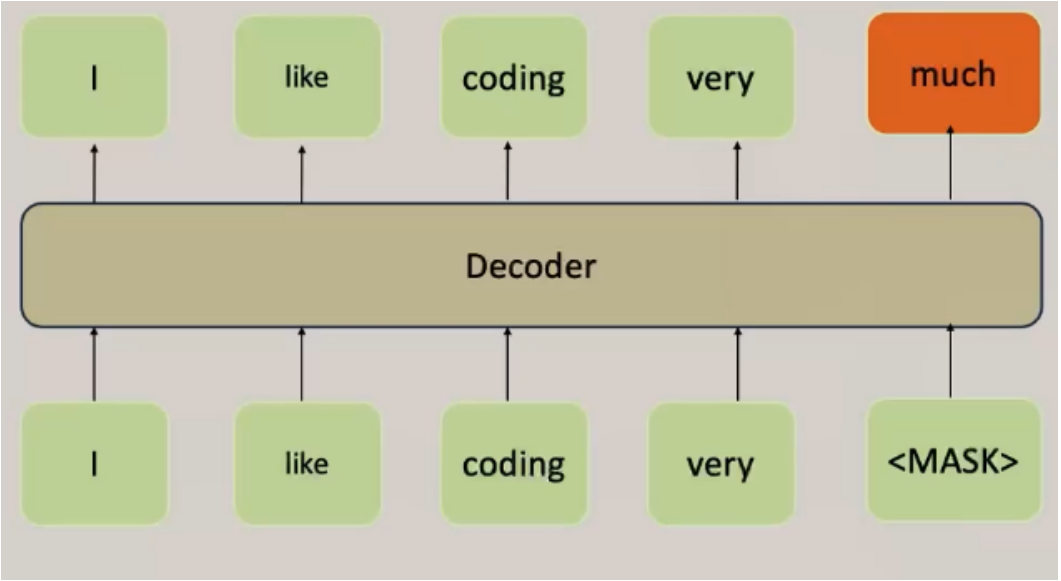

- **LLM 등장의 흐름**
  - Transformer
    - LLM의 Base
  - GPT-1
    - 디코더만으로 구성 (언어 생성 및 이해 능력이 뛰어남을 입증)
    - 사전훈련, 파인 튜닝
    - Auto Regressive Model
  - BERT
    - 인코더만으로 구성
    - 사전훈련, 파인 튜닝
    - Auto Encoding Model
  - GPT-2
  - XLNet
    - AR/AE 장점을 결합
    - 양방향 정보를 위해 모든 가능한 순열 학습
    - AR 방식으로 학습
  - RoBERTa

        - BERT의 훈련 절차 개선
        - NSP 방식의 훈련 제거
    - MASS
    - BART
        - AR / AE 장점 결합
        - 입력문장 변환, 원 문장을 복원하는 방식으로 훈련
        - Text 생성에서 SoTA
    - MT-DNN
    - T5
        - 모든 NLP 작업
        - Text-to-Text 변환
        - 다양한 벤치마크에서 SOTA 성능
    - GPT-3
        - LLM Boom!
        - GPT2를 확장
        - 대규모 사전학습으로 성능 향상
- **Auto Encoder Model**



- Encoder-Only
  - Feature를 추출하는 데 특화되어 있음
- Task
  - Classification
  - Sentiment Analysis
  - NER
  - Masked Prediction

- **Autoregressive Model**

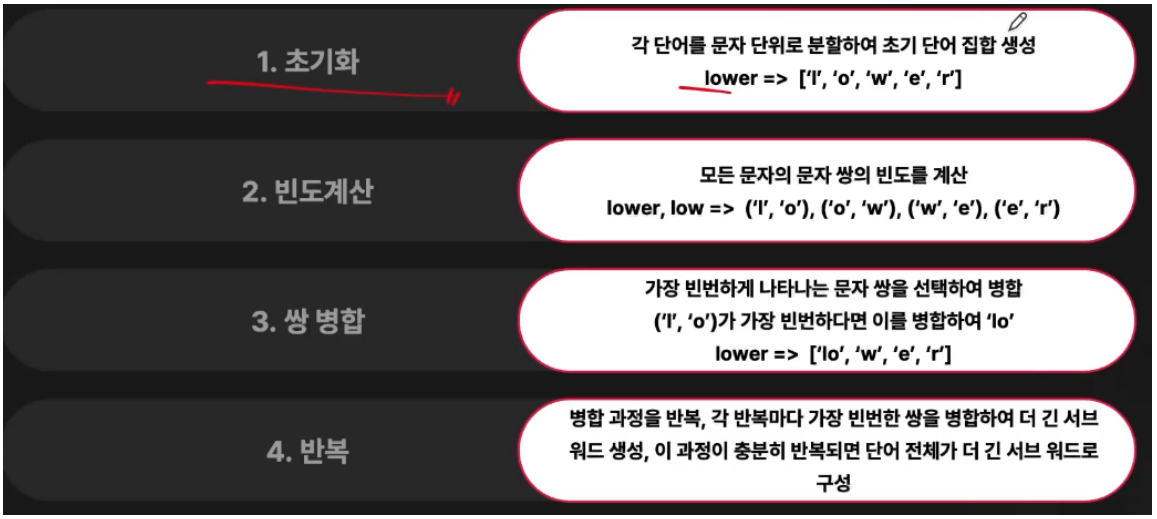



  - Decoder Only
  - Task
    - Text Generation
    - Next Token Prediction
  - 구조적으로 Decoder 만을 사용하기에 생성에 적합 (다음 토큰 예측에 적합)
- **LLM 작동 방식 및 원리**
  - Input
  - Tokenize
    - 텍스트를 숫자형 벡터로 변환
  - Token Embedding

        - 토큰을 고정된 길이의 벡터로 변환
    - Encoding
        - 문맥적 의미가 반영된 벡터로 변환
    - LLM
    - Output (Logit)
    - Sampling & Decoding

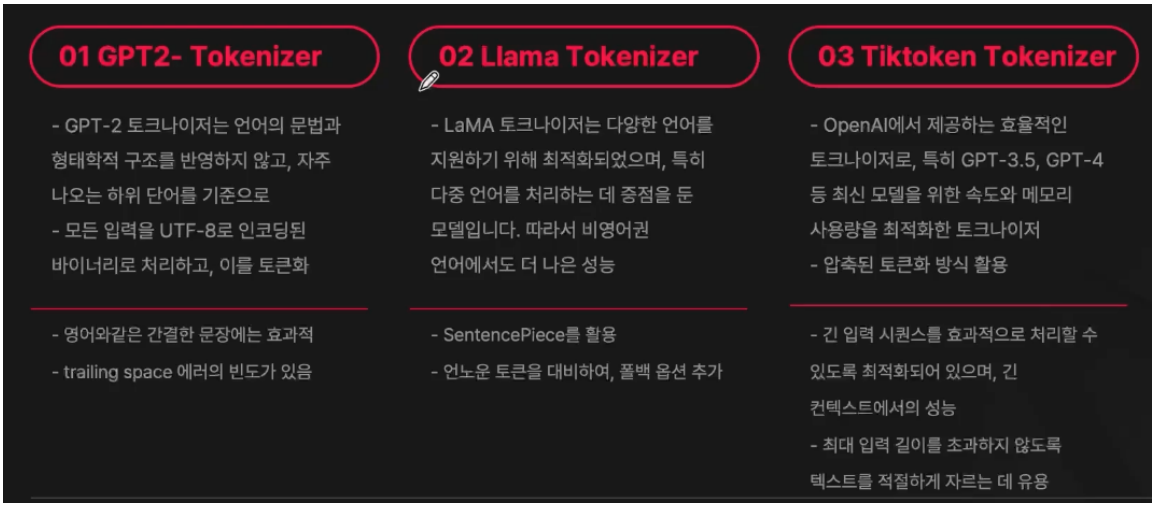
---

- **Tokenizer: BPE의 원리**




    - 처음 보는 단어도 처리할 수 있다.

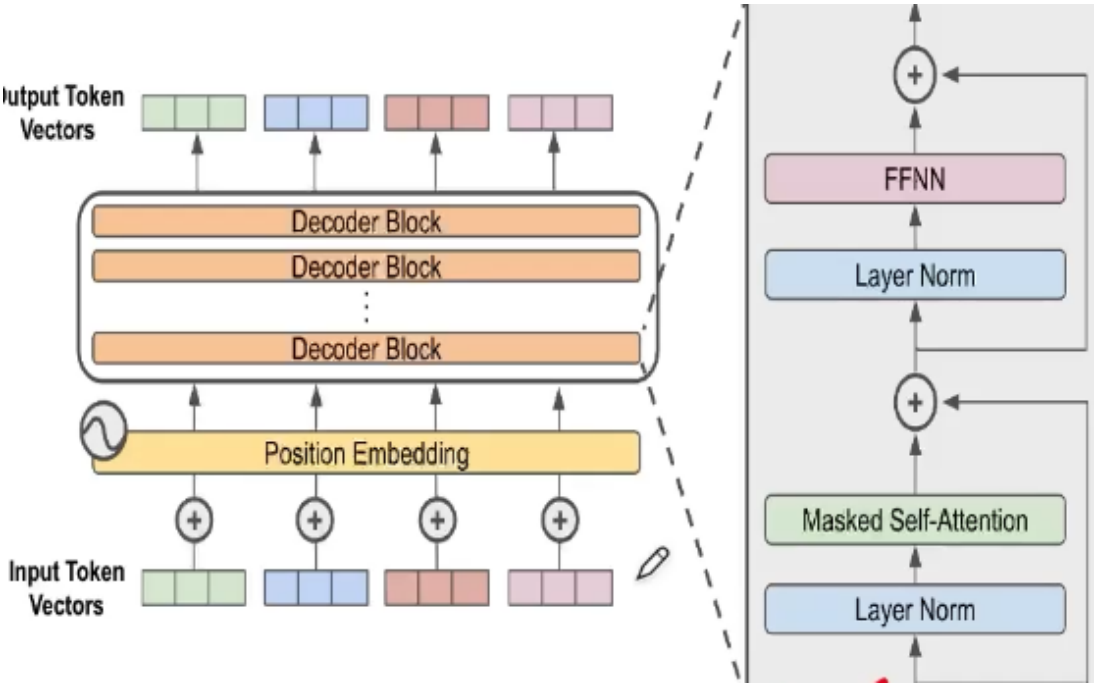
- **Tokenizer 종류별 특징**



- **Embedding Layer**
  - 언어 모델이 텍스트 데이터를 수치형 벡터로 변환하는 중요한 역할
  - 텍스트를 고차원 벡터 공간으로 변환하여 각 단어의 의미를 벡터 형태로 표현
  - 모델이 텍스트 데이터를 학습하며 행렬의 가중치도 학습
    이를 통해 임베딩 벡터가 단어의 의미적 유사성을 반영하도록 최적화

- **Encoding**
  - 텍스트 데이터를 신경망이 이해하고 처리하는 형태.
  - 문맥정보를 유지하고, 언어의 다양한 구조적 특성을 반영하여 문법적 의미적 규칙을 학습할 수 있도록 함.
  - 고정된 길이 벡터로 변환.
  - 입력 정규화
    - 데이터를 정규화하여 학습 안정성을 향상 시킴

- **Transformer Decoder Layer**
  - 각 토큰이 다른 모든 토큰들과 관계 학습 (Self-Attention Mechanism)
  - 어텐션 메커니즘 후에 위치한 완전 연결 층 (FFNN)
    - 두 개의 선형 변환과 비선형 활성화 함수 사용
    - 다음 단계로 전달하기 위해 변환
  - 학습 안정성 및 훈련 속도 향상 (Layer Normalization)
    - 각 서브 레이어 뒤에 적용됨
  - Gradient 소실 문제를 완화 (Residual Connection)
    - 입력과 출력 간의 직접 경로를 제공 (정보 소실 방지)




- **Output Layer**
  - Linear Transformation
    - 출력 벡터를 어휘의 크기만큼의 차원을 가진 벡터로 변환
  - SoftMax Activation
    - 각 차원의 값을 [0, 1] 범위의 확률로 변환, 모든 차원의 합이 1이되도록
    - 각 단어가 다음 위치에 등장할 확률
  - Logits and Probabilities
    - 모델의 선형 변환 출력
    - 로그 확률과 유사한 값을 가지며, 확률로 변환되기 전에 모델의 예측 점수를 나타냄
    - 이후 SoftMax를 적용하면 각 단어가 선택될 확률, 다음 단어 예측 및 생성에 활용

- **Sampling & Decoding**
  - 모델이 출력한 확률 분포를 사용하여 다음 토큰을 선택
  - 선택된 토큰을 시퀀스에 추가하고, 이를 반복
    이후 이 숫자형 벡터를 디코딩하여 텍스트 데이터로 변환