7장(BERT~)

날짜	@2025년 5월 28일 → 2025년 6월 3일
선택	DL세션 과제
주차	13주차
진행 상태	완료

💡 BERT

📖 **BERT(Bidirectional Encoder Representations from Transformers)**

: 2018년 구글에서 발표한 트랜스포머 기반 양방향 인코더를 사용하는 자연어 처리 모델

- 인코더는 입력 문장의 단어들을 임베딩해 각 단어의 의미를 벡터화 → 순차 처리하여 문장 전체의 의미 추출

🧩 양방향 인코더

: 입력 시퀀스를 양쪽 방향에서 처리하여 이전과 이후의 단어를 모두 참조하면서 단어의 의미와 문맥을 파악함.

→ 더 정확하게 문맥을 파악하며 높은 성능을 보임

- 기존 언어 모델: 순차적 학습 → 이전 단어만 고려하여 문맥/의미 파악

- 대규모 데이터를 사용해 사전 학습되어 있어 전이학습에 주로 활용
- 다른 작업에서 재사용해 적은 양의 데이터로도 높은 정확도를 달성, 시간 단축 가능
- 사전 학습을 위해 마스킹된 언어 모델링(MLM)과 다음 문장 예측 방법(NSP)를 사용함.

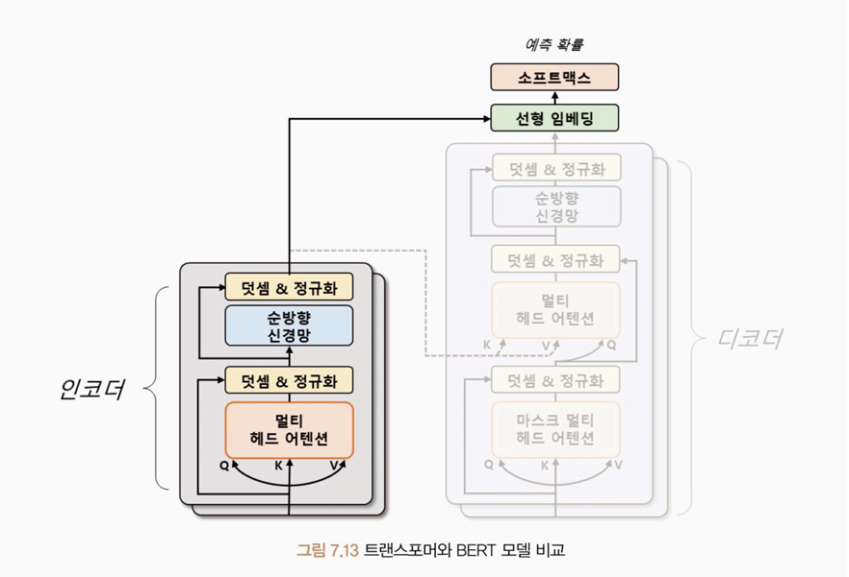

그림 7.13 트랜스포머와 BERT 모델 비교

📥사전 학습 방법

마스킹된 언어 모델 MLM(Masked Language Modeling)

: 입력 문장에서 임의로 일부 단어를 마스킹하고 해당 단어를 예측하는 방식

(ex) I'm learning PyTorch에서 'learning'을 마스킹 → I'm [MASK] PyTorch → BERT는 [MASK]를 예측

다음 문장 예측 NSP(Next Sentence Prediction)

: 두 개의 문장이 주어졌을 때, 두 번째 문장이 첫 번째 문장의 다음에 오는 문장인지 여부를 판단

(ex) I'm learning PyTorch와 PyTorch is a machine learning library 라는 두 문장 → 연속적인지 아닌지 판단

BERT 모델은 입력 문장에 특수 토큰을 추가해 모델이 학습하고 추론하는 과정에서 필요한 정보를 제공

- **[CLS] 토큰**
  - 입력 문장의 시작 부분에 추가되는 토큰
  - 이 토큰을 이용해 문장 분류 작업을 위해 어떤 유형의 문장인지 정보를 얻음

- (ex) 입력 문장이 긍정/부정인지, 문장 내에서 어떤 객체가 언급되는지 등

- **[SEP] 토큰**
  - 입력 문장 내에서 두 개 이상의 문장을 구분하기 위해 사용되는 토큰
  - (ex) 문장 분류 작업에서 두 개의 문장을 입력 받을 때 구분하는 용도로 사용
  - 입력 문장을 두 개의 독립적인 문장으로 인식, 각각 문장에 대한 정보를 정확하게 파악

- **[MASK] 토큰**
  - 입력 문장 내에서 임의로 선택된 단어를 가리키는 특별한 토큰
  - 주어진 문장에서 일부 단어를 가린 후 모델의 학습/예측에 활용
  - (ex) 언어 모델링 작업에서 마스킹된 실제 단어를 예측하는데 사용

→ 입력 시퀀스: [CLS] 문장-1 [SEP] 문장-2 [MASK] [SEP} 등의 구조를 가짐

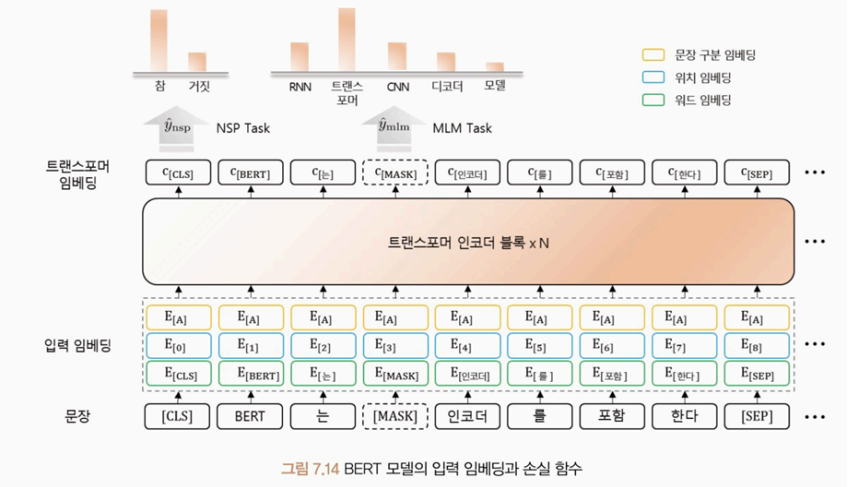

그림 7.14 BERT 모델의 입력 임베딩과 손실 함수

- 입력 문장: BERT는 트랜스 포머 인코더를 포함한다

→ [CLS] BERT는 트랜스 포머 인코더를 포함한다 [SEP]로 변환

- [MASK] 토큰 적용 방법

텍스트 토큰 중 15%에 해당하는 단어를 대상으로 마스킹을 수행

이 중 80%는 [MASK] 토큰으로 대체하고, 10%는 어휘 사전에 존재하는 무작위 단어로 변경, 나머지 10%는 실제 토큰을 사용

BERT는 사전학습 수행 후, **미세 조정 기법**을 통해 다양한 자연어 처리 작업에 적용할 수 있음(미세 조정 과정에서는 해당 작업에 맞는 계층을 추가하고, 손실 함수를 정이해 학습함)

(ex) 문장 분류, 감성 분석, 질문 응답, 기계 번역 등

분류 문제에서는 일반적으로 [CLS] 토큰 벡터를 사용하지만, 각 토큰마다 예측이 필요한 경우에는 모든 토큰 벡터를 사용

💡BART

📖 **BART**(Bidirectional Auto-Regressive Transformer)

: 2019년 메타의 FAIR 연구소에서 발표한 트랜스포머 기반의 모델

- BERT의 인코더 + GPT 디코더를 결합한 시퀀스-시퀀스(Seq2Seq) 구조
- 노이즈 제거 오토인코더(Denosing Autoencoder)로 사전학습됨
  - 학습방법은 입력 데이터에 잡음을 추가하고, 잡음이 없는 원본 데이터로 복원하도록 학습하는 방식

- BERT 인코더 : 입력 문장에서 일부 단어를 무작위로 마스킹 처리하고, 마스킹 단어를 맞추도록 학습
  - 문장 전체의 맥락을 이해하고, 문맥 내 단어 간 상호작용 파악
- GPT: 언어 모델을 이용해 문자으이 이전 토큰들을 입력으로 받고, 다음에 올 토큰을 맞추도록 학습
  - 문장 내 단어들의 순서와 문맥을 파악하고, 다음에 올 단어를 예측하는 능력
- BART: 사전 학습 시 BERT의 인코더와 GPT의 디코더가 학습하는 방법을 일반화해 학습

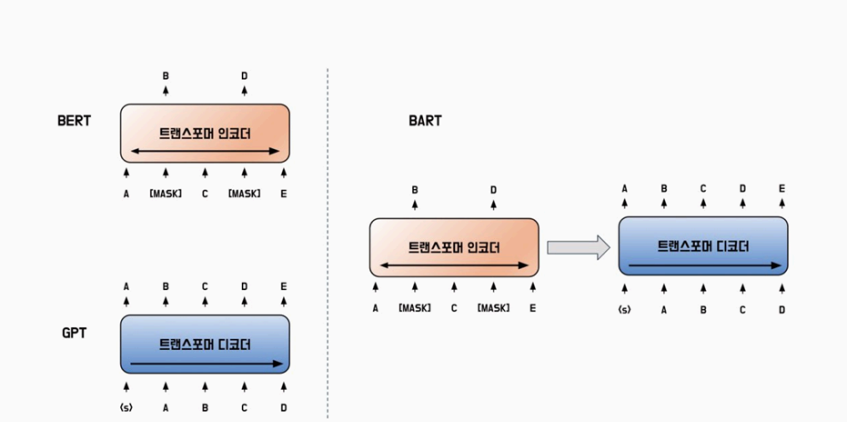

그림 7.15 BART 사전 학습 시각화

- 인코더&디코더 사용 → 트랜스포머와 유사한 구조

차이점	BART	트랜스포머
어텐션 연산 수행 위치	인코더의 마지막 계층과 디코더의 각 계층 사이에만 어텐션 연산 수행	인코더의 모든 계층과 디코더의 모든 계층 사이의 어텐션 연산 수행

BART의 인코더에서는 입력 문장의 각 단어를 임베딩

→ 여러 층의 인코더 → 마지막 계층에서 입력 문장 전체의 의미를 잘 반영하는 벡터 생성 → 디코더가 문장 생성 시 참고 → 디더의 각 계층에서 이 벡터&이전 계층에서 생성된 출력 문장의 정보를 활용해 출력 문장 생성

- BART에서는 인코더의 마지막 계층과 디코더의 각 계층 사이에서만 어텐션 연산을 수행 → 정보 전달을 최적화& 메모리용량 줄일 수 있음.

📦 사전 학습 방법

BART 인코더는 BERT의 MLM 이외에도 다양한 노이즈 기법을 사용한다.

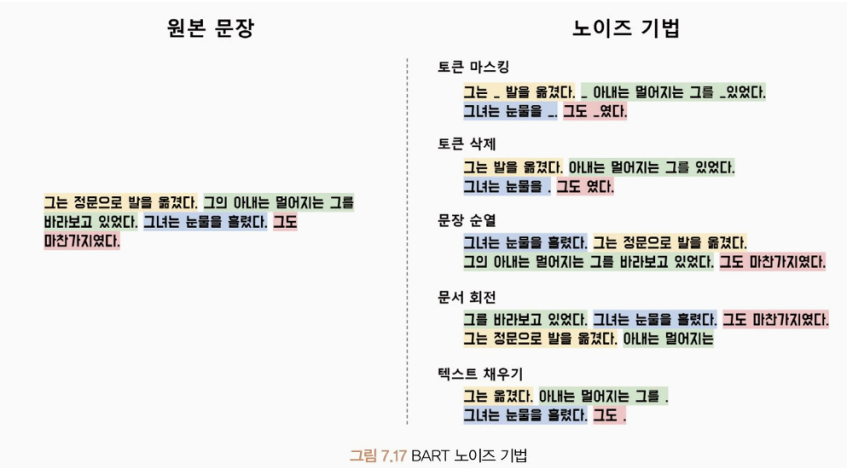

그림 7.17 BART 노이즈 기법

- **토큰 마스킹(Token Masking)**
  - BERT의 MLM과 동일한 기법
  - 입력 문장의 일부 토큰을 마스크 토큰으로 치환
  - 문장 내에서 어떤 단어가 중요한 역할을 하는지 학습
  - 문맥 이해 & 중요한 정보 추출 능력 향상
- **토큰 삭제(Token Deletion)**
  - 입력 문장의 일부 토큰을 삭제
  - 토큰 마스킹과 다르게 어떤 위치의 토큰이 삭제되었는지 맞춰야 함.
  - 입력 문장에서 불필요/중요하지 않은 정보를 자동으로 필터링 처리 가능
  - 모델 학습&예측 시간 감소, 모델의 일반화 성능 향상
- **문장 순열(Sentence Permutation)**
  - 마침표(.)를 기준으로 문장을 나눈 뒤, 문장의 순서를 섞으면 원래 순서를 맞춰야 함
  - 문장 내에서 단어들이 연결되있는 구조를 파악하게 되며, 다른 순서로 주어져도 잘 처리 가능
- **문서 회전(Document Rotation)**
  - 임의의 토큰으로 문서가 시작
  - 문장 순열과 다르게 문장의 순서는 유지
  - 문서의 원래 시작 토큰을 맞춰야 함
  - 문서의 시작점을 인식하게 함.
- **텍스트 채우기(Text Infilling)**
  - 몇 개의 토큰을 하나의 구간(span)으로 묶고, 일부 구간을 마스크 토큰으로 대체
  - 묶은 구간의 길이는 0~임의의 값까지 설정 가능(일반적으로 포아송 분포 사용, lambda=3으로 설정해 계산)
  - 구간의 길이가 0인 경우, 해당 위치에 변환된 토큰이 없음을 의미
  - 모델은 역속된 마스크 토큰을 복구하되, 실제로는 마스킹되지 않은 토큰도 구분해야 함
  - 누락된 단어를 예측 유도 → 더 많은 정보를 활용하여 입력 문장을 더 잘 이해할 수 있음

⚙️ 미세 조정 방법

BART는 인코더와 디코더를 모두 사용하는 구조를 가짐

→ 미세 조정 시 각 다운스트림 작업에 맞게 입력 문장을 구성해야 함

→ 인코더/디코더에 다른 문장 구조로 입력

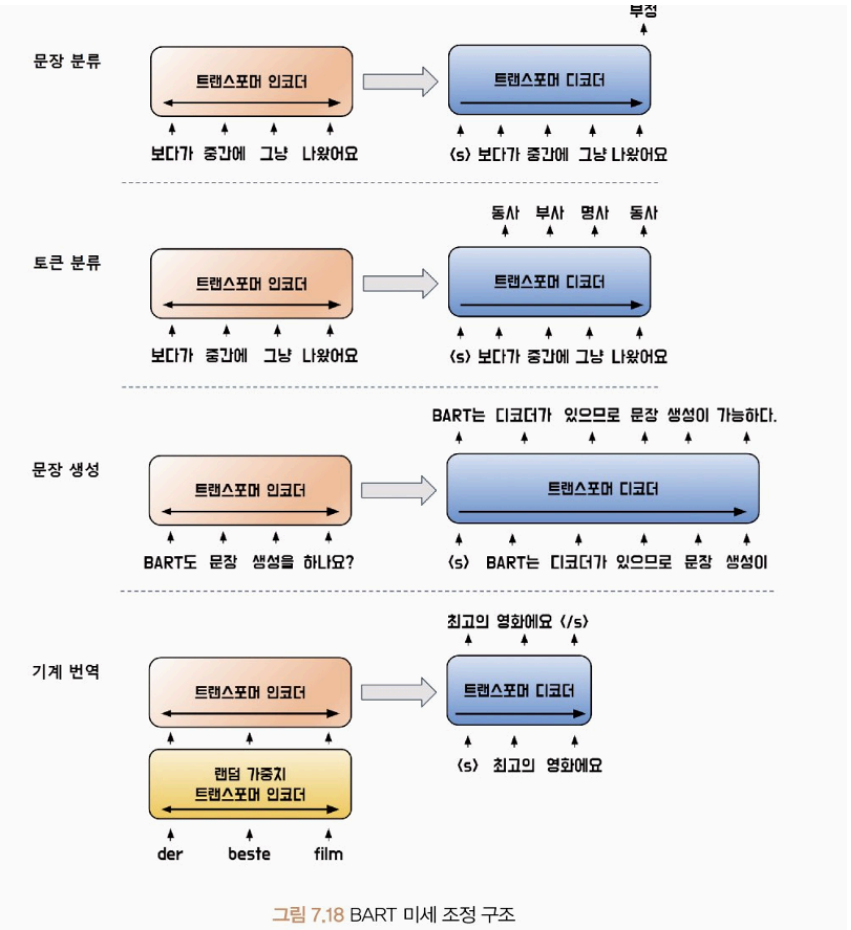

그림 7.18 BART 미세 조정 구조

(문장 분류 작업)

1. 입력 문장을 인코더/디코더에 동일하게 입력
2. 디코더의 마지막 토큰 은닉 상태를 선형 분류기의 입력값으로 사용 (BERT의 CLS 토큰과 비슷하지만, BART는 전체 입력과 어텐션 연산이 적용된 은닉 상태를 사용)

(토큰 분류 작업)

1. 입력 문장을 인코더/디코더에 동일하게 입력
2. 디코더의 각 시점별 마지막 은닉 상태를 토큰 분류기의 입력값으로 사용
3. 전체 입력과 디코더의 각 시점별 은닉 상태와의 어텐션 연산을 수행

BART는 트랜스포머 디코더를 사용 → BERT가 해결하지 못했던 문장 생성 작업을 수행 가능

- 입력값을 조작해 출력을 생성하는 추상적 질의응답(Abstractive Question Answering)과 문장 요약(Summarization)과 같은 작업에 적합
- BART의 사전 학습 방식과 유사하기 때문에 높은 성능을 보임
- → 이를 통해 문장 의미 파악 & 기반 새로운 문장 생성 능력을 가짐

(EX) 기계 번역 작업에서 BART는 사전 학습된 인코더에 기계 번역을 위한 인코더를 추가해 수행 가능.

추가된 인코더는 기존의 단어사전을 사용하지 않아도 됨

디코더는 사전 학습된 가중치와 단어 사전 사용

(학습 단계)

1. 새로 추가된 트랜스포머 인코더의 가중치와 위치 임베딩, 첫 번째 인코더 층의 셀프 어텐션 입력 행렬 가중치만 학습
2. 모든 신경망의 가중치를 작은 반복으로 학습

💡 ELECTRA(451

📖 ELECTRA(Efficiently Learning an Encoder that Classifies Token Replacements Accurately)

2020년 구글에서 발표한 트랜스포머 기반의 모델

- 입력을 마스킹하는 대신 생성자(Generator)와 판별자(Discriminator)를 사용하여 사전학습 수행
  - 생성적 적대 신경망(GAN)과 유사한 방식으로 학습을 수행 → 효율적 학습
    - 생성 모델: 실제 데이터와 비슷하게 토큰을 생성해 다른 토큰으로 대체
    - 판별모델: 생성 모델의 데이터와 실제 데이터를 입력 받아 어떤 데이터가 실제 데이터 or 생성된 데이터인지 구분
- BERT 와 비교했을 매개 변수의 수가 적음 → 빠르고, 적은 메모리 사용

🪫 사전 학습 방법

: ELECTRA의 생성자 모델과 판별자 모델은 모두 트랜스포머 인코더 구조를 따름

RTD(Replaced Token Detection)

- 생성자 모델: 입력 문장의 일부 토큰을 마스크 처리 → 어떤 토큰이었는지 예측하며 학습
  - BERT의 마스킹딘 언어 모델(MLM)과 동일
  - 입력 문장의 15%를 마스크 처리
- 판별자 모델: 각 입력 토큰이 원본 문장의 토큰인지, 생성자 모델로 인해 바뀐 토큰인지 맞히며 학습
  - 생성자 모델이 복원한 값을 입력받음

그림 7.20 ELECTRA 모델 학습 방법

생성자/판별자의 구조로 유사하지만, 생성적 적대 신경망(GAN)과의 차이점

1. 생성자 모델이 원래 토큰을 정확히 예측한 경우,

   이 토큰은 생성된 토큰이 아닌 원본 토큰으로 인식

2. GAN 모델은 생성 모델과 판별 모델을 적대적으로 학습

   : 생성 모델은 판별 모델이 구분하기 어렵게 학습하고, 판별 모델은 더욱 더 잘 구분하게 학습함

3. GAN 모델은 완전한 노이즈 벡터를 입력 받아 생성

   ↔ ELECTRA 생성 모델은 일부 토큰이 마스크 처리된 텍스트를 입력으로 받음

- 생성 모델 & 판별 모델은 모두 트랜스포머 인코더 구조를 따름 → 같은 개수의 계층으로 구성되어 있다면 가중치를 공유할 수 있어, 빠른 학습이 가능
- 가중치를 완전히 공유 → 생성 모델의 성능이 너무 높아져 판별 모델이 학습할 수 없음
  - → 생성 모델을 판별 모델의 1/2에서 1/4 크기로 바꿔 설정
  - 모든 가중치를 공유하는 대신 임베딩 계층의 가중치만 공유

💡 T5(457)

📖 T5(Text-to-Text Transfer Transformer)

: 자연어 처리 분야의 딥러닝 모델로 트랜스포머 구조를 기반으로 함

- 인코더-디코더 모델 구조를 바탕으로 GLUE, superGLUE, CNN/DM 등의 데이터세트에서 SOTA(State-of-the-art)를 달성
- 입력&출력 모두 토큰 시퀀스로 처리하는 텍스트-텍스트(Text-to-Text) 구조
  ⇒ 입력/출력 형태를 자유롭게 다룰 수 잇음

  ↔ 기존 자연어 처리 모델: 대부분 입력 문장을 벡터나 행렬로 변환한 뒤, 이를 이용해 출력 문장을 생 or 출력값이 클래스 또는 입력값의 일부를 반환
- 문장 번역, 요약, 질의응답, 텍스트 분류 등

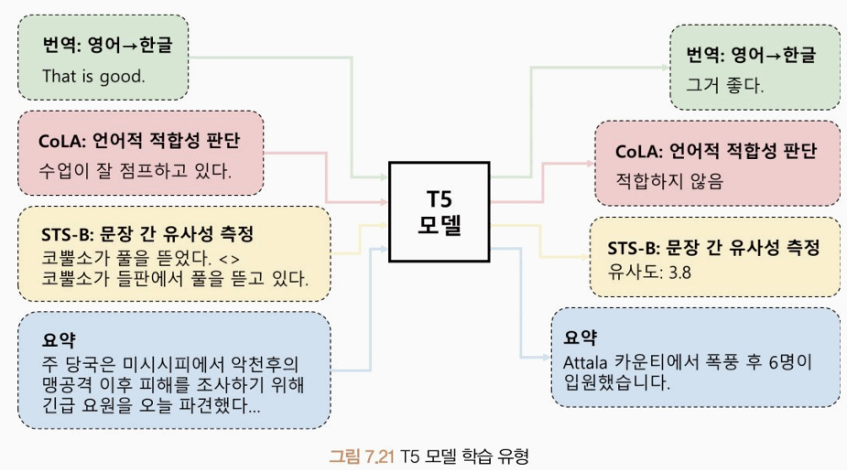

그림 7.21 T5 모델 학습 유형

입력과 출력이 모두 토큰(텍스트) 시퀀스 → 입력&출력간 관계를 세밀하게 다룰 수 있음.

사전학습 후 미세 조정 단계에서 해당 작업의 데이터를 이용해 모델을 조정해 최적의 성능을 얻을 수 있음.

- CoLA dataset: 문법적으로 허용 가능한 문장/그렇지 않은 문장을 구분할 수 있게 됨
- STS-B(The Semantic Textual Similarity Benchmark) dataset: 문장 간 의미적 유사성 측정 가능

T5는 텍스트-텍스트 모델이므로, 학습을 위한 데이터세트는 원본 문장&대상 문장 사용

- C4(Colossa Clean Crawled Corpus) 데이터 세트를 활용해 다양한 자연어 처리 작업을 수행할 수 있게 사전 학습됐음
- 사전 학습 방식:
  - 비지도 학습 방식
  - 입력 문장의 일부 구간을 마스킹해 입력 시퀀스를 처리
  - 출력 시퀀스는 실제 마스킹된 토큰과 마스크 토큰의 연결로 구성
  - 문장마다 유일한 마스크 토큰을 의미하는 센티널 토큰(Sentinel Token)이 사용됨: <extra_id_0>, <extra_id_1>과 같이 0~99까지의 100개의 기본값을 사용

    (ex) '인코더-디코더 모델 구조'라는 문장에서 '인코더'와 '디코더'를 마스킹해 처리할 경우, [<extra_id_0>, -,<extra_id_1>, 모델 구조]로 생성되며, 출력 토큰은 [인코더, <extra_id_0>, 디코더, <extra_id_1>로 정의됨
    - 이 과정에서 인코딩되기 전에 'translate English to German:' 또는 'summerize:'와 같은 작업 토큰을 문장 앞에 추가 → 작업 토큰도 함께 학습해 다양한 자연어 처리 작업에서 높은 성능을 발휘

T5 모델 실습

- 토큰 인덱스(input_ids)
- 어텐션 마스크(attention_mask)
- 디코더 토큰 인덱스(decoder_input_ids)
- 라벨(labels)