언어 모델 이해

1.환경준비

(1) 라이브러리 설치

```
# !pip install transformers
# colab에서는 이미 설치되어 있음
```

(2) 라이브러리 Import

```
from transformers import pipeline
```

2.언어모델 사용 예제 코드 확인해보기

(1) 전체 코드

```
from transformers import PreTrainedTokenizerFast, BartForConditionalGeneration

#  Load Model and Tokenize
tokenizer = PreTrainedTokenizerFast.from_pretrained("ainize/kobart-news")
model = BartForConditionalGeneration.from_pretrained("ainize/kobart-news")

# Encode Input Text
input_text = '''
국내 전반적인 경기침체로 상가 건물주의 수익도 전국적인 감소세를 보이고 있는 것으로 나타났다.
수익형 부동산 연구개발기업 상가정보연구소는 한국감정원 통계를 분석한 결과 전국 중대형 상가
순영업소득(부동산에서 발생하는 임대수입, 기타수입에서 제반 경비를 공제한 순소득)이 1분기 ㎡당
3만4200원에서 3분기 2만5800원으로 감소했다고 17일 밝혔다.
수도권, 세종시, 지방광역시에서 순영업소득이 가장 많이 감소한 지역은 3분기 1만3100원을 기록한 울산으로
이어 대구(-27.7%), 서울(-26.9%), 광주(-24.9%), 부산(-23.5%), 세종(-23.4%), 대전(-21%), 경기(-19.2%),
지방 도시의 경우도 비슷했다.
경남의 3분기 순영업소득은 1만2800원으로 1분기 1만7400원 대비 26.4% 감소했으며
제주(-25.1%), 경북(-24.1%), 충남(-20.9%), 강원(-20.9%), 전남(-20.1%), 전북(-17%), 충북(-15.3%) 등도 감
조현택 상가정보연구소 연구원은 "올해 내수 경기의 침체된 분위기가 유지되며
상가, 오피스 등을 비롯한 수익형 부동산 시장의 분위기도 경직된 모습을 보였고
오피스텔, 지식산업센터 등의 수익형 부동산 공급도 증가해 공실의 위험도 늘었다"며
"실제 올 3분기 전국 중대형 상가 공실률은 11.5%를 기록하며 1분기 11.3% 대비 0.2% 포인트 증가했다"고 말
그는 "최근 소셜커머스(SNS를 통한 전자상거래), 음식 배달 중개 애플리케이션, 중고 물품 거래 애플리케이션
사용 증가로 오프라인 매장에 영향을 미쳤다"며 "향후 지역, 콘텐츠에 따른 상권 양극화 현상은 심화될 것으
'''

input_ids = tokenizer.encode(input_text, return_tensors="pt")

# Generate Summary Text Ids
summary_text_ids = model.generate(
    input_ids=input_ids,
    bos_token_id=model.config.bos_token_id,
    eos_token_id=model.config.eos_token_id,
    length_penalty=2.0,
    max_length=142,
    min_length=56,
    num_beams=4,
)

# Decoding Text
print(tokenizer.decode(summary_text_ids[0], skip_special_tokens=True))
```

```
You passed along `num_labels=3` with an incompatible id to label map: {'0': 'NEGATIVE', '1': 'PO
국내 국내 전반적인 경기침체로 상가 건물주의 수익도 전국적인 감소세를 보이고 있는 것으로 나타났고
```

(2) 코드를 나눠서 단계를 살펴봅시다.

- ① 모델, 토크나이저 다운로드

```
from transformers import PreTrainedTokenizerFast, BartForConditionalGeneration

# Load Model and Tokenize
tokenizer = PreTrainedTokenizerFast.from_pretrained("ainize/kobart-news")
model = BartForConditionalGeneration.from_pretrained("ainize/kobart-news")
```

```
You passed along `num_labels=3` with an incompatible id to label map: {'0': 'NEGATIVE', '1': 'PO
```

- ② 입력 데이터 전처리(토크나이즈)

```
# 입력할 텍스트
input_text = '''
국내 전반적인 경기침체로 상가 건물주의 수익도 전국적인 감소세를 보이고 있는 것으로 나타났다.
수익형 부동산 연구개발기업 상가정보연구소는 한국감정원 통계를 분석한 결과 전국 중대형 상가
순영업소득(부동산에서 발생하는 임대수입, 기타수입에서 제반 경비를 공제한 순소득)이 1분기 ㎡당
3만4200원에서 3분기 2만5800원으로 감소했다고 17일 밝혔다.
'''

# 토크나이즈(encoding)
input_ids = tokenizer.encode(input_text, return_tensors="pt")

# 입력 토큰
input_ids
```

```
tensor([[    0,   230, 23436, 25623, 27091, 21067, 23895, 16442, 14453, 15342,
          9866, 14770, 14134, 15665, 14896, 16553, 14082, 14173, 15917, 18560,
         13679, 15821, 22164, 14451, 23895, 15107, 14867, 17047, 14188,  8995,
         22036, 14109, 15161, 22269, 14494, 14770, 22537, 13679, 23895,   230,
         11374, 18801, 16162,   239, 23625, 14030, 19668, 17249, 22540,   243,
         18453, 22540, 14030, 14060, 10773, 14051, 15541, 14061, 18010, 14408,
         16162, 15019, 20646,  1700, 17867,  9770,   230,   250, 10500,   251,
         18604, 15359, 20412, 18078,   252,   255, 14079, 15624, 15665, 14781,
         16826, 14278,     1]])
```

- ③ 예측(모델사용)

```
# 모델 사용
summary_text_ids = model.generate(
    input_ids=input_ids,                          # 입력으로 받는 데이터의 토큰
    bos_token_id=model.config.bos_token_id,       # Begin of String 문장이나 문단 등의 시작을 나타내는
    eos_token_id=model.config.eos_token_id,       # End of String 문장이나 문단 등의 종료 나타내는 특수
    length_penalty=2.0,                           # 생성된 텍스트의 길이에 대한 패널티. 1< 더 긴 문장을
    max_length=50,                                # 생성될 수 있는 최대 텍스트 길이(토큰 수)
    min_length=30,
    num_beams=4                                   # 빔 서치는 텍스트 생성 시 여러 가능성 있는 출력 중 최
)


# 요약된 문장 토큰
summary_text_ids
```

```
tensor([[    2,     0, 23436, 14454, 25623, 27091, 21067, 23895, 16442, 14453,
         15342,  9866, 14770, 14134, 15665, 14896, 16553, 14082, 14173, 19743,
         14297, 14454, 27091, 21067, 23895, 16442, 14453, 15342,  9866, 14770,
         14134, 15665, 14896, 16553, 14082, 14173, 19743,  9102, 14770, 12124,
         27091, 21067, 23895, 16442, 14453, 15342,  9866, 14770, 14134,     2]])
```

- ④ 결과 후처리

```
# 출력 토큰을 문장으로 변환(Decoding)
print(tokenizer.decode(summary_text_ids[0], skip_special_tokens=True))
```

```
ㅏ 건물주의 수익도 전국적인 감소세를 보이고 있는 것으로 나타났으며 국내 경기침체로 상가 건물주의 수
```

3.토크나이즈

- 텍스트에서 토큰으로

(1) 문자 토큰

- 텍스트 데이터셋

```
text_data = ["Tokenizers are one of the core components of the NLP.",
        "They serve one purpose: to translate text into data that can be processed by the model.",
        "Models can only process numbers, so tokenizers need to convert our text inputs to numerical
        "In this section, we will explore exactly what happens in the tokenization pipeline.",
        "However, models can only process numbers, so we need to find a way to convert the raw text t
        "That’s what the tokenizers do, and there are a lot of ways to go about this.",
        "The first type of tokenizer that comes to mind is word-based.",
        "It’s generally very easy to set up and use with only a few rules, and it often yields decent
        "Character-based tokenizers split the text into characters, rather than words.",
        "The vocabulary is much smaller.",
        "There are much fewer out-of-vocabulary (unknown) tokens, since every word can be built from
```

- 어휘사전 만들기 : 문자별로 생성

```
# 문자별로 쪼개기
tokenized_text = []
for i in range(len(text_data)):
    tokenized_text = tokenized_text + list(text_data[i])

print(tokenized_text[:20])

# 각 문자에 인덱스(id) 부여하기
token2idx = {ch: idx for idx, ch in enumerate(sorted(set(tokenized_text)))}
print(token2idx)
print('vocab size : ',len(token2idx))
```

```
['T', 'o', 'k', 'e', 'n', 'i', 'z', 'e', 'r', 's', ' ', 'a', 'r', 'e', ' ', 'o', 'n', 'e', ' ',
{' ': 0, '(': 1, ')': 2, ',': 3, '-': 4, '.': 5, ':': 6, 'C': 7, 'H': 8, 'I': 9, 'L': 10, 'M': 1
vocab size :  40
```

- 첫번째 문장을 id로 변환

```
input_text = text_data[0]
print('Input : ', input_text)
print(list(text_data[0]))

input_ids = [token2idx[token] for token in list(text_data[0])]
print(input_ids)
```

```
Input :  Tokenizers are one of the core components of the NLP.
['T', 'o', 'k', 'e', 'n', 'i', 'z', 'e', 'r', 's', ' ', 'a', 'r', 'e', ' ', 'o', 'n', 'e', ' ', '
[14, 28, 24, 19, 27, 23, 38, 19, 30, 31, 0, 15, 30, 19, 0, 28, 27, 19, 0, 28, 20, 0, 32, 22, 19,
```

(2) 단어 토큰화

- 어휘사전 만들기 : 문자별로 생성

```
# 문자별로 쪼개기
tokenized_text = []
for i in range(len(text_data)):
    tokenized_text = tokenized_text + text_data[i].split()

print(tokenized_text[:2])

# 각 문자에 인덱스(id) 부여하기
token2idx = {ch: idx for idx, ch in enumerate(sorted(set(tokenized_text)))}
print(token2idx)
print('vocab size : ',len(token2idx))
```

```
['Tokenizers', 'are']
{'(unknown)': 0, 'Character-based': 1, 'However,': 2, 'In': 3, 'It’s': 4, 'Models': 5, 'NLP.': 6
vocab size :  105
```

- 첫번째 문장을 id로 변환

```
input_text = text_data[0]
print('Input : ', input_text)
print(text_data[i].split())

input_ids = [token2idx[token] for token in text_data[i].split()]
print(input_ids)
```

```
Input :  Tokenizers are one of the core components of the NLP.
['There', 'are', 'much', 'fewer', 'out-of-vocabulary', '(unknown)', 'tokens,', 'since', 'every',
[9, 15, 51, 35, 61, 0, 88, 73, 31, 101, 19, 16, 17, 38, 21]
```

(3) 부분단어(Subwords) 토큰화

- 문자토큰과 단어토큰의 결합
- BERT 에서 사용되는 토크나이즈 WordPiece
- tokenization ==> token, #ization

```
from transformers import AutoTokenizer

model_ckpt = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_ckpt)
```

```
tokenizer_config.json: 100%  28.0/28.0 [00:00<00:00, 544B/s]
config.json: 100%  483/483 [00:00<00:00, 6.21kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 3.16MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 6.27MB/s]
```

```
text = text_data[0]
encoded_text = tokenizer(text)
print(text)
print(encoded_text)
```

```
Tokenizers are one of the core components of the NLP.
{'input_ids': [101, 19204, 17629, 2015, 2024, 2028, 1997, 1996, 4563, 6177, 1997, 1996, 17953, 2
```

- input_ids : 각 토큰의 고유 정수값
- attention_mask :

```
# id를 다시 토큰으로 변
tokens = tokenizer.convert_ids_to_tokens(encoded_text.input_ids)
print(tokens)
```

```
['[CLS]', 'token', '##izer', '##s', 'are', 'one', 'of', 'the', 'core', 'components', 'of', 'the'
```

```
print(tokenizer.convert_tokens_to_string(tokens))
```

```
[CLS] tokenizers are one of the core components of the nlp. [SEP]
```

- 어휘 사전 크기

```
tokenizer.vocab_size
```

```
30522
```

- 모델의 최대 문맥 크기

```
tokenizer.model_max_length
```

```
512
```

- 모델 입력을 위한 필드 이름

```
tokenizer.model_input_names
```

```
['input_ids', 'attention_mask']
```

4.임베딩

- 자주 사용되는 Word2Vec 임베딩 벡터를 사용해 봅시다.

```
!pip install gensim
from gensim.models import Word2Vec
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages (4.3.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from ge
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from gen
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.10/dist-packages (fro
```

- 각 문장들을 토크나이즈 하기

```
token_sent = [tokenizer.convert_ids_to_tokens(tokenizer(text).input_ids) for text in text_data]
token_sent[0]
```

```
['[CLS]',
 'token',
 '##izer',
 '##s',
 'are',
 'one',
 'of',
 'the',
 'core',
 'components',
 'of',
 'the',
 'nl',
 '##p',
 '.',
 '[SEP]']
```

```
# Word2Vec 임베딩 생성
```