

Term Project

# Transformer와 LSTM을 활용한 상품 카테고리 분류

---

2021198696 이현태

# 목차

1. 비즈니스 문제 정의
2. 연구배경
3. 활용 데이터
4. 데이터 EDA 및 전처리
5. 예측에 활용한 모델 (Self-Attention, LSTM)
6. 결과
7. 결과 해석 및 결론

# 01 비즈니스 문제 정의






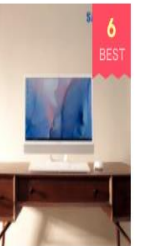
쇼핑하우 by kakao commerce 4 5단책장

카테고리 핫딜 베스트100 할인특가\* 기획전 #건강 식품 #마스크 #전기요 #갤럭시 워치3 #바람막이 점퍼 로그인 최근본 상품

인기 상품 BEST 100 (?) 쇼핑하우 베스트100은 사용자들의 상품클릭 현황을 반영합니다.

전체 패션 감화 뷰티 가전 컴퓨터 홈엔터테인먼트 생활/건강 식품 유아동 여행 캠핑대저

전체 노트북/태블릿pc 데스크탑/모니터/pc부품 프린터/pc주변/사무기기 게임/주변기기

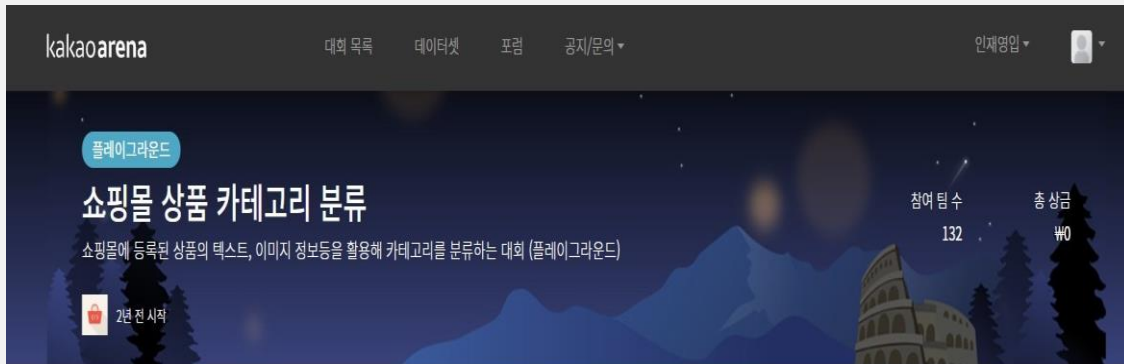
|   |  |  |   |  |   |
|---|--|--|---|--|---|
|  <p>1 BEST</p> <p>LG 그램 15Z960</p> <p>LG그램 15인치 15Z960 인기시리즈 윈도우10</p> <p>690,900원 G마켓</p> |  <p>2 BEST</p> <p>삼성전자 삼성 노트북 렌탈전시 80%할인 IS...</p> <p>579,830원 G마켓</p> |  <p>3 BEST</p> <p>삼성전자 올인원PC 윈도우10 정품 탑재 HDD500GB추가</p> <p>1,099,000원 인터파크</p> |  <p>4 BEST</p> <p>지패드2 10.1</p> <p>LG전자 지패드2 G패드2 10.1 정품</p> <p>155,990원 G마켓</p> |  <p>5 BEST</p> <p>배그/오버워치/게이밍/조립 PC/컴퓨터/본체/SSD장치/원...</p> <p>309,000원 11번가</p> |  <p>6 BEST</p> <p>DM530ABE-L14</p> <p>618,910원 최저가</p> |
|---|--|--|---|--|---|

- 다음 쇼핑 하우는 사용자에게 상품검색, 가격비교, 관심 상품 추천, 그 외 각종 쇼핑 정보를 제공해주는 서비스
- 카테고리 정보는 소비자들이 상품을 빠르게 검색하고 탐색할 때 반드시 필요한 정보
- 제공 업체마다 분류 기준이 다르거나 정보가 없는 경우가 있음
- 상품의 키워드 조합 및 상품명은 업체마다 다른 방식으로 부여됨
- 따라서 데이터 기반 AI 기술을 활용해 자동 분류기를 만들고 어떠한 알고리즘이 효과적일지 분석

## 02 연구 배경

- 연구 배경
  - 자연어 처리 분야(NLP)에서는 RNN 기반의 알고리즘이 많이 활용되고 있음
  - 텍스트 분류 문제에서도 마찬가지로 많은 RNN 기반 알고리즘이 활용됨
  - 최근에는 텍스트 길이에 상관없이 문맥 정보를 효과적으로 파악하기 위해 LSTM에서 더 발전된 다양한 알고리즘이 등장하고 있음 (Seq2Seq, Attention, Transformer, Bert)
  - 상품명 데이터 특징 파악해 상품 카테고리 분류에 있어서는 어떠한 알고리즘이 효과적인지 분석
  - 과연 최신 알고리즘 (Attention)이 상품 카테고리 분류에 효과적일지 분석
  - 상품 카테고리 분류 문제에 직면할 E-commerce 회사들에게 방향성을 제시

## 03 활용 데이터



메뉴

개요

참가하기

개요

데이터

리더보드

포럼

다른 대회 선택 ▼

### 대회 설명

다음쇼핑에는 수백개의 상품이 존재합니다. 사용자에게 효과적으로 상품을 노출하기 위해서는 체계적인 분류가 필요하지만, 상품을 제공하는 업체마다 기준이 다르거나 분류 정보가 없는 경우가 많기 때문에 일관된 분류 체계로 만드는 작업이 필요합니다.

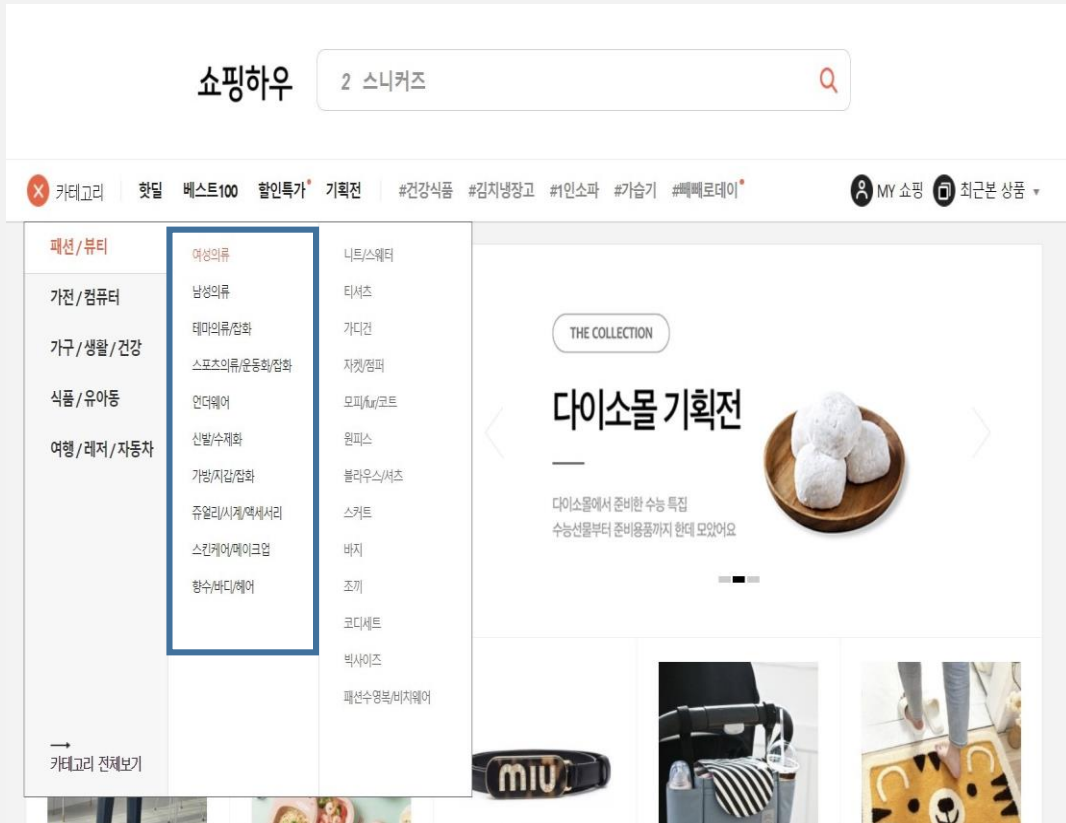
이 대회는 더 정확한 상품 분류기를 만드는 것이 목표입니다. 상품은 최대 4개까지의 분류 값을 갖는데, 각 분류는 계층적인 구조입니다. 예를 들어 아이디 `L3203227501` 상품은 `맛있는 제주차 3종세트 ...` 인데, 이 상품의 카테고리는 아래와 같습니다.

- 대분류: 음료/생수/커피
- 중분류: 차/티백
- 소분류: 차 선물세트

- 카카오 아레나에서 쇼핑 카테고리 분류 대회를 위한 상품 정보가 담긴 데이터를 제공
- 카테고리 매핑 정보 json 형식으로 제공
- 훈련과 테스트 데이터 셋은 hdf5 형식으로 제공

| 데이터 (84.98 GB)              |                               |            |
|-----------------------------|-------------------------------|------------|
| 파일 목록                       | 파일 상세                         | 컬럼         |
| ? cate1.json                | 왼쪽 파일 목록에서 파일을 선택해주세요.        | “ pid      |
| train.chunk.01 1000.0K x 12 |                               | “ product  |
| train.chunk.02 1000.0K x 12 | 파일명                           | “ brand    |
| train.chunk.03 1000.0K x 12 | train.chunk.01                | “ model    |
| train.chunk.04 1000.0K x 12 | 크기                            | “ maker    |
| train.chunk.05 1000.0K x 12 | 8.78 GB (9,422,749,432 Bytes) | “ price    |
| train.chunk.06 1000.0K x 12 | 데이터 개수                        | “ updtm    |
| train.chunk.07 1000.0K x 12 | 1,000,000 개                   | “ bcateid  |
| train.chunk.08 1000.0K x 12 |                               | “ mcateid  |
| train.chunk.09 134.8K x 12  |                               | “ scateid  |
| dev.chunk.01 507.8K x 12    |                               | “ dcateid  |
| test.chunk.01 1000.0K x 12  |                               | “ img_feat |
| test.chunk.02 526.5K x 12   |                               |            |

## 03 활용 데이터



Out[6]:

|        | pid         | product   | brand    | maker         | price | bcatenm       | mcatenm   | scatenm  | dcatenm |
|--------|-------------|---|----------|---------------|-------|---------------|-----------|----------|---------|
| 0      | O4486751463 | 직소퍼즐 - 1000조각 바다거북의 여행 (PL1275)                   | 퍼즐라 이프   | 상품상세설명 참조     | 16520 | 악기/취미/만들기     | 보드게임/퍼즐   | 직소/퍼즐    |         |
| 1      | P3307178849 | [모리케이스]아이폰6S/6S+ tree farm101 - 다 이어리케이스[바보사랑]... | 바보사랑     | MORY 해당없음     | 20370 | 휴대폰/액세서리      | 휴대폰액세서리   | 아이폰액세서리  |         |
| 2      | R4424255515 | 크리비아 기모 3부 속바지 GLG4314P                           | 크리비아     |               | -1    | 언더웨어          | 보정언더웨어    | 속바지/속치마  |         |
| 3      | F3334315393 | [하프클립/잭앤질]남성 솔리드 절개라인 포인트 포켓 팬츠 31133PT002_NA     | 잭앤질      | 쥘크리스패션        | 16280 | 남성의류          | 바지        | 일자면바지    |         |
| 4      | N731678492  | 코드프리혈당시험지50매/코드프리시험지/최장유효기간                       |          | 기타            | -1    | 건강관리/실비용품     | 건강측정용품    | 혈당지      |         |
| ...    | ...         | ...   | ...      | ...           | ...   | ...           | ...       | ...      | ...     |
| 999995 | Q4697414634 | 카렉스 노블맨 헬빙룩주시트                                    |          | 기타            | -1    | 자동차용품         | 시트커버/매트   | 여름 시트    |         |
| 999996 | N4402110844 | 아이숲 차밍래빗 자가드 편면내의                                 | 아이숲      |               | -1    | 언더웨어          | 아동언더웨어    | 아동 내복/내의 |         |
| 999997 | Q4029646876 | 앞포켓크로스백T-131 솔더백 패션크로스백 크로스가방                     | 크로바패션    | 크로바패션 / 크로바패션 | 15940 | 가방/지갑/잡화      | 캐주얼가방     | 캐주얼 토트백  |         |
| 999998 | O1952838815 | 정식판매처 QNAP TS-121 하드미포함 당일배송                      | 기타 (미입력) |               | -1    | 프린터/PC주변/사무기기 | 저장장치      | 외장하드     |         |
| 999999 | G2303841227 | 공식 쿠팡 자외선 햇병 식기살균건조기5인용 CSD-E051                  |          | 기타            | -1    | 주방가전/냉장고/전기밥솥 | 식기건조기/세척기 | 식기건조기    |         |

1000000 rows x 9 columns

- 카카오에서는 총 8,134,818개 훈련 데이터를 제공
- 개인 프로젝트에서는 1,000,000개의 데이터만 활용예정
- 상품 정보 데이터 (고유 ID (pid), 상품명 (product), 브랜드명 (brand), 제조사 (maker), 가격(price)) 제공
- 제품 정보 데이터를 활용해 대분류(bcatenm)에 있는 52개의 카테고리를 분류 예정
- 데이터 분석을 통해 분류에 도움이 될 수 있는 데이터 선정

## 04 데이터 EDA 및 전처리

- 예측에 도움이 될 컬럼 선택
  - 특정 컬럼에서 고유값 별로 등장 빈도수를 파악해 데이터 특징을 파악
  - Pandas의 value\_counts()를 이용해 고유값 별 빈도수와 전체 데이터에서 차지하는 비율 파악

```
# 특정 컬럼에서 고유값별로 등장 빈도수를 기록
def get_vc_df(df, col):
    vc_df = df[col].value_counts().reset_index()
    vc_df.columns = [col, 'count']
    vc_df['percentage'] = (vc_df['count'] / vc_df['count'].sum())*100
    return vc_df
```

- 브랜드명 (brand) 컬럼의 고유값 빈도수 파악

```
# train_df의 brand 컬럼의 빈도수 테이블을 가져와서 상단 10개의 행만 출력
# 비어있는 값이 많음
vc_df = get_vc_df(df, 'brand')
vc_df.head(10)
```

|   | brand     | count  | percentage |
|---|-----------|--------|------------|
| 0 |           | 253224 | 25.3224    |
| 1 | 바보사랑      | 14064  | 1.4064     |
| 2 | 상품상세설명 참조 | 9078   | 0.9078     |
| 3 | 아디다스      | 8300   | 0.8300     |
| 4 | 아트박스      | 6871   | 0.6871     |
| 5 | 오가닉맘      | 6867   | 0.6867     |
| 6 | 나이키       | 5087   | 0.5087     |
| 7 | 꾸밈        | 4629   | 0.4629     |
| 8 | 기타        | 3994   | 0.3994     |
| 9 | 1300K     | 3925   | 0.3925     |

## 04 데이터 EDA 및 전처리

- 예측에 도움이 될 컬럼 선택
  - 제조사 (maker) 컬럼의 고유값 별 빈도수 파악

```
# 제조사 고유값 빈도수
# 컬럼이 비어있거나 '상품상세설명 참조', '기타' 등으로 타겟 예측에 도움이 안 되는 데이터가 많음
vc_df = get_vc_df(df, 'maker')
vc_df.head(10)
```

|   | maker     | count  | percentage |
|---|-----------|--------|------------|
| 0 | 기타        | 200473 | 20.0473    |
| 1 |           | 141806 | 14.1806    |
| 2 | 상품상세설명 참조 | 25463  | 2.5463     |
| 3 | 아디다스      | 7530   | 0.7530     |
| 4 | LF        | 6389   | 0.6389     |
| 5 | 구름        | 6066   | 0.6066     |
| 6 | La Diosa  | 4180   | 0.4180     |
| 7 | [불명]      | 4140   | 0.4140     |
| 8 | 나이키       | 4042   | 0.4042     |
| 9 | 삼성물산 패션부문 | 3768   | 0.3768     |



## 04 데이터 EDA 및 전처리

- 예측에 도움이 될 컬럼 선택
  - 가격 (price) 컬럼의 고유값 별 빈도수 파악
  - 데이터가 없는 경우가 너무 많고 가격 매기는 기준이 제공업체마다 다를 수 있어 예측에 어려움을 줄 수 있음

```
# price  
# -1 값 (가격 정보 없음)이 상당수 포함돼 있을  
vc_df = get_vc_df(df, 'price')  
vc_df.head(10)
```

|   | price | count  | percentage |
|---|-------|--------|------------|
| 0 | -1    | 628804 | 62.8804    |
| 1 | 14880 | 555    | 0.0555     |
| 2 | 12000 | 544    | 0.0544     |
| 3 | 19800 | 499    | 0.0499     |
| 4 | 9900  | 488    | 0.0488     |
| 5 | 14000 | 484    | 0.0484     |
| 6 | 19000 | 459    | 0.0459     |
| 7 | 28000 | 449    | 0.0449     |
| 8 | 18000 | 447    | 0.0447     |
| 9 | 14400 | 410    | 0.0410     |

## 04 데이터 EDA 및 전처리

- 예측에 도움이 될 컬럼 선택
  - 상품명 (product) 컬럼의 고유값 별 빈도수 파악
  - 중복 데이터 비율이 많지 않음
  - 상품명 컬럼에 이미 브랜드명 (brand), 제조사 (maker)에 대한 정보가 포함되어 있음
  - 예측에 가장 도움이 될 수 있는 컬럼

```
# product  
# product 컬럼은 비어있는 값이 존재하지 않고 다양한 제품명이 존재함  
vc_df = get_vc_df(df, 'product')  
vc_df.head(10)
```

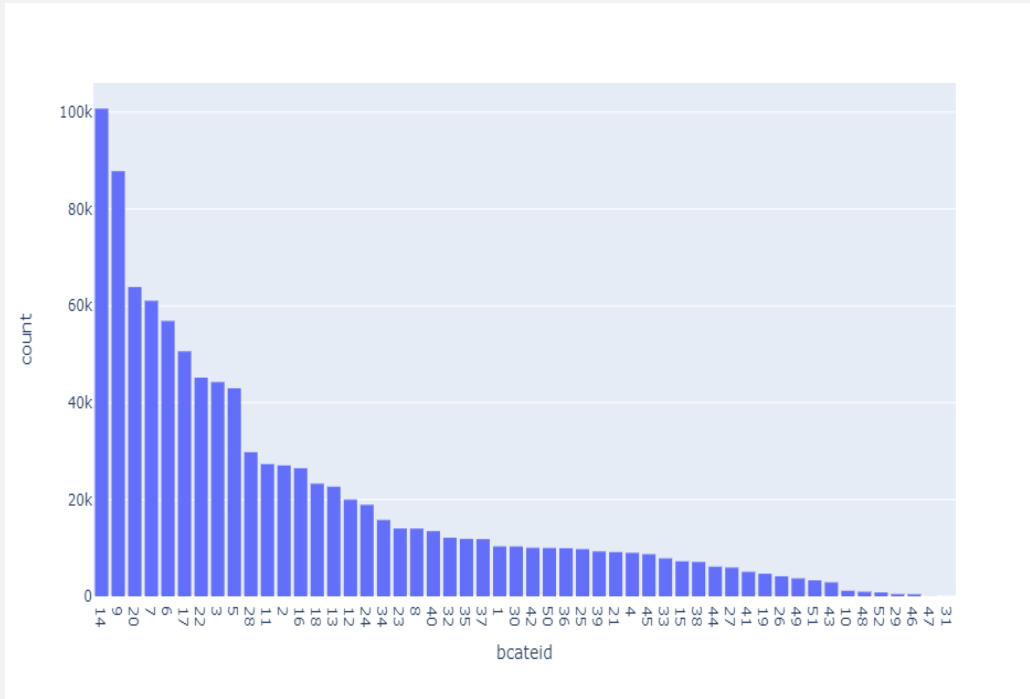
|   | product                             | count | percentage |
|---|-------------------------------------|-------|------------|
| 0 | DIY 돌하우스 로맨틱해변별장 미니어쳐 만들기           | 28    | 0.0028     |
| 1 | 아이폰7 플러스 저반사 지문방지 액정보호필름            | 26    | 0.0026     |
| 2 | 브리츠 휴대용 블루투스 스피커 BZ-A660 Sound Dome | 24    | 0.0024     |
| 3 | 퀸센스평첼주전자7L                          | 24    | 0.0024     |
| 4 | GB5011 대형 월넛 원형 벽시계 50cm 제조한국       | 24    | 0.0024     |
| 5 | [오로라] QM500S 호조 에디션 보조배터리 5000mah   | 24    | 0.0024     |
| 6 | 땡큐파머 미라클에이지 리페어크림50ml               | 23    | 0.0023     |
| 7 | 스테판조셉 스테인레스 물병 무당벌레                 | 22    | 0.0022     |
| 8 | DIY 돌하우스 엔젤하우스 미니어쳐 만들기             | 22    | 0.0022     |
| 9 | 클립형 (블루투스스피커) 이소닉 BT-102 무선스피커      | 22    | 0.0022     |

## 04 데이터 EDA 및 전처리

- 타깃 데이터 (대분류) 고유값 별 빈도수 파악
  - 가방/지갑/잡화 제품이 제일 많지만 퍼센트가 압도적으로 많은 건 아님
  - 다른 카테고리 제품도 다수 차지하고 있음 (예측에 방해될 만한 데이터 불균형은 아님)
  - 예측하는데 지장이 없는 데이터

```
# bcateid 대분류
# 타깃값, 가방/지갑/잡화 제품이 제일 많지만 퍼센트가 압도적으로 많은 건 아님
vc_df = get_vc_df(df, 'bcatenm')
vc_df.head(10)
```

|   | bcatenm      | count  | percentage |
|---|--------------|--------|------------|
| 0 | 가방/지갑/잡화     | 100729 | 10.0729    |
| 1 | 여성의류         | 87815  | 8.7815     |
| 2 | 주얼리/시계/액세서리  | 63878  | 6.3878     |
| 3 | 남성의류         | 61044  | 6.1044     |
| 4 | 스포츠의류/운동화/잡화 | 56880  | 5.6880     |
| 5 | 신발/수제화       | 50599  | 5.0599     |
| 6 | 유아동의류/신발/가방  | 45166  | 4.5166     |
| 7 | 휴대폰/액세서리     | 44249  | 4.4249     |
| 8 | 언더웨어         | 42974  | 4.2974     |
| 9 | 산업/공구/안전용품   | 29779  | 2.9779     |



## 04 데이터 EDA 및 전처리

- 특수 기호 제거
  - 제품명 컬럼에 있는 데이터에 -, \_ , [], () 와 같은 특수기호가 존재
  - 이는 예측에 불필요한 문자이므로 제거
  - 정규표현식을 사용해 제거
  - 특수기호를 공백문자로 치환
  - 두개 이상 연속된 빈공백은 하나의 빈공백으로 만듦

```
# 특수기호 제거
import re

# 특수기호를 나열한 패턴 문자열을 컴파일하여 패턴 객체를 얻는다.
p = re.compile('[!@#$%^&*~(){}-=[\]{}~./\?:~+~'"`|_~>×` ]')

# 문장의 특수기호 제거 함수
def remove_special_characters(sentence, lower=True):
    sentence = p.sub(' ', sentence) # 패턴 객체로 sentence 내의 특수기호를 공백문자로 치환한다.
    sentence = ' '.join(sentence.split()) # sentence 내의 두개 이상 연속된 빈공백들을 하나의 빈공백으로 만든다.
    if lower:
        sentence = sentence.lower()
    return sentence

# product 컬럼에 특수기호를 제거하는 함수를 적용한 결과를 반환한다.
train_df['product'] = train_df['product'].map(remove_special_characters)

train_df.head() # 특수기호가 제거된 train_df의 상단 5행만 출력
```

| product   |
|---|
| 직소퍼즐 - 1000조각 바다거북의 여행 (PL1275)                       |
| [모리케이스]아이폰6S/6S+ tree farm101 - 다<br>이어리케이스[바보사랑][... |
| 크리비아 기모 3부 속바지 GLG4314P                               |
| [하프클럽/잭앤질]남성 솔리드 절개라인 포인<br>트 포켓 팬츠 31133PT002_NA     |
| 코드프리혈당시험지50매/코드프리시험지/최<br>장유효기간                       |
| ...   |
| 카렉스 노블맨 웰빙목주시트  |
| 아이썬 차밍래빗 자가드 편면내의                                     |
| 앞포켓크로스백T-131 솔더백 패션크로스백 크<br>로스가방                     |
| 정식판매처 QNAP TS-121 하드미포함 당일배<br>송                      |
| 공식 쿠첸 자외선 젖병 식기살균건조기5인용<br>CSD-E051                   |



| product                                       |
|---|
| 직소퍼즐 1000조각 바다거북의 여행 pl1275                   |
| 모리케이스 아이폰6s 6s tree farm101 다이어리케이스 바보사랑 무료배송 |
| 크리비아 기모 3부 속바지 glg4314p                       |
| 하프클럽 잭앤질 남성 솔리드 절개라인 포인트 포켓 팬츠 31133pt002 na  |
| 코드프리혈당시험지50매 코드프리시험지 최장유효기간                   |
|   |

## 04 데이터 EDA 및 전처리

- 상품명 Word Count
  - 제품명(product) 컬럼의 단어 개수에 대한 통계량 파악
  - 평균적으로 상품명은 7개 단어로 구성
  - 최대 길이는 109개 단어 (추후에 토큰화, 패딩에 사용됨)

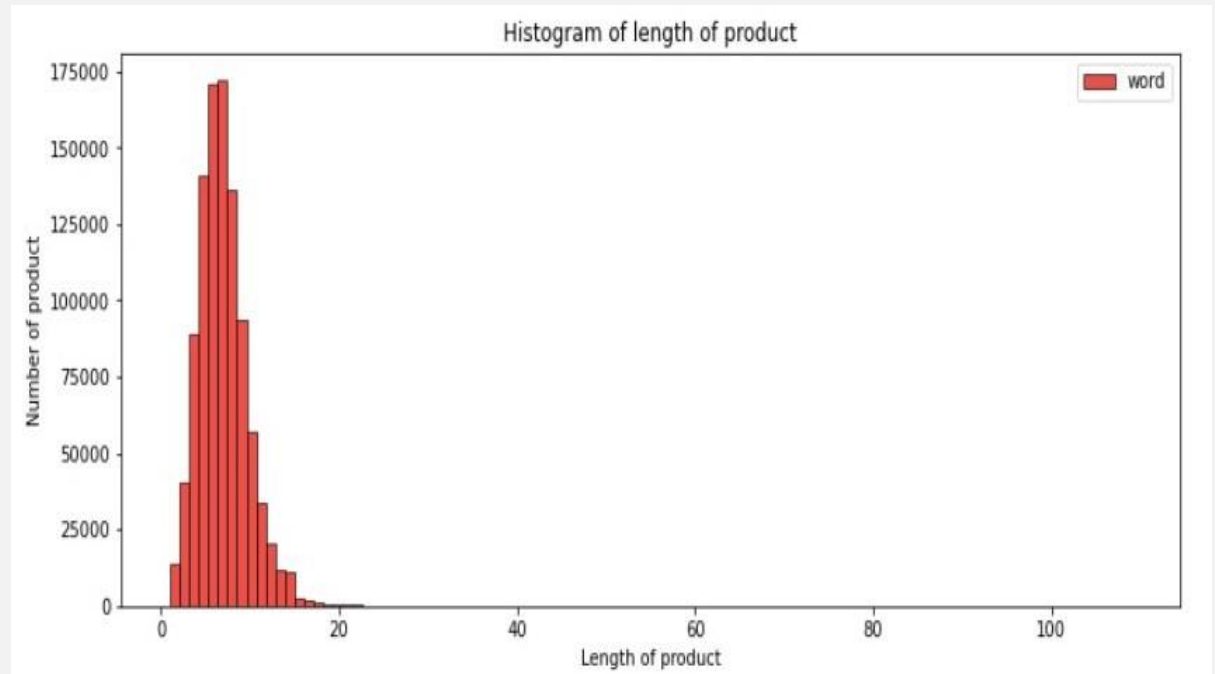
```
#word count
```

```
word_counts = train_df['product'].apply(lambda x: len(x.split(' ')))
```

```
word_counts.head()
```

```
word_counts.describe()
```

|                               |                |
|-------------------------------|----------------|
| count                         | 1000000.000000 |
| mean                          | 7.024293       |
| std                           | 2.594933       |
| min                           | 1.000000       |
| 25%                           | 5.000000       |
| 50%                           | 7.000000       |
| 75%                           | 8.000000       |
| max                           | 109.000000     |
| Name: product, dtype: float64 |                |



## 04 데이터 EDA 및 전처리

- SentencePiece로 Subword 토큰나이징 진행
  - 서브워드 분리(Subword segmentation) 작업은 하나의 단어를 여러 서브 워드로 분리해서 단어 임베딩을 위해 사용됨 (ex) birthplace = birth + place)
  - 이를 통해 단어 집합에 없는 단어로 인해 생기는 OOV(Out-Of-Vocabulary) 문제나 희귀단어, 신조어와 같은 문제를 완화시키는 효과가 있음
  - 이러한 서브워드 토큰나이징을 SentencePiece 라이브러리를 통해 구현이 가능함
  - 상품명은 명사 나열형 이므로 한국어 단어 분절기는 성능이 안 좋을 수 있음  
ex) 아이폰 7 플러스 저반사 지문방지 액정보호필름
  - 학습 기반의 문장 분절기이므로 상품명의 특성을 반영한 문장 분절이 가능함
  - 깃허브 주소: <https://github.com/google/sentencepiece>

## 04 데이터 EDA 및 전처리

- SentencePiece로 Subword 토큰나이징 진행
  - 대표적인 서브워드 알고리즘인 bpe 지정 (model\_type)
  - 문장의 최대 길이(max\_sentence\_length)는 상품명 컬럼의 단어 최대수인 109개로 지정
  - Vocab\_size는 BPE의 단어수를 얼마로 할 것인가를 지정 (25,000 ~ 35,000개가 적당)

### SentencePiece로 토큰나이징 진행

- 상품명으로 학습되는 분절기 사용
- input : 학습시킬 파일
- model\_prefix : 만들어질 모델 이름
- vocab\_size : 단어 집합의 크기
- model\_type : 사용할 모델 (unigram(default), bpe, char, word)
- max\_sentence\_length: 문장의 최대 길이

```
▶ import sentencepiece as spm
```

```
▶ # product 컬럼의 상품명들을 product.txt 파일명으로 저장한다.  
with open('product.txt', 'w', encoding='utf-8') as f:  
    f.write(train_df['product'].str.cat(sep='\n'))
```

```
▶ # 보통 학습을 진행하기 위해서 vocab_size 25,000 ~ 35,000 정도로 함  
# 문장 최대 길이는 분석한 결과 109개 단어임  
spm.SentencePieceTrainer.Train('--input=product.txt --model_type=bpe --model_prefix=product --vocab_size=35000 --  
                                --max_sentence_length=109 --shuffle_input_sentence=true')
```



## 04 데이터 EDA 및 전처리

- SentencePiece로 Subword 토큰나이징 진행
  - 토큰나이징을 진행한 데이터를 tokens 열에 따로 할당
  - 단어의 가장 앞 글자에 \_ 기호가 붙음
  - 단어의 첫 글자임을 표시하기 위함
  - 글자의 위치에 따라 의미가 달라지는 단어를 구분하기 위함  
ex) \_3부의 뒷글자 부와 부사장의 앞글자 부는 서로 다른 의미를 가짐

```
# 센텐스피스 모델을 로드한다.
sp = spm.SentencePieceProcessor()
sp.Load('product.model')

# product 칼럼의 상품명을 분절한 결과를 tokenized_product 칼럼에 저장한다.
train_df['tokens'] = train_df['product'].map(lambda x: " ".join(sp.EncodeAsPieces(x)) )

train_df[['product', 'tokens']].head()
```

|   | product                                       | tokens  |
|---|---|---|
| 0 | 직소퍼즐 1000조각 바다거북의 여행 pl1275                   | _직소퍼즐 _1000 조각 _바다거북 의 _여행 _pl 1275               |
| 1 | 모리케이스 아이폰6s 6s tree farm101 다이어리케이스 바보사랑 무료배송 | _모리케이스 _아이폰 6 s _6 s _tree _farm 101 _다이어리케이스 ... |
| 2 | 크리비아 기모 3부 속바지 glg4314p                       | _크리비아 _기모 _3 부 _속바지 _gl g 43 14 p                 |
| 3 | 하프클럽 잭앤질 남성 솔리드 절개라인 포인트 포켓 팬츠 31133pt002 na  | _하프클럽 _잭앤질 _남성 _솔리드 _절개라인 _포인트 _포켓 _팬츠 _311 33... |
| 4 | 코드프리혈당시험지50매 코드프리시험지 최장유효기간                   | _코드 프리 혈 당 시험지 50 매 _코드 프리 시험지 _최 장 유효 기간         |



## 04 데이터 EDA 및 전처리

- SentencePiece로 Sub-word 토큰나이징 진행
  - SentencePiece vocab 파일을 통해 인덱스 값 확인
  - 단어 집합의 크기는 35000개
  - encoded\_as\_pieces 함수를 통해 서브 워드 시퀀스 변환
  - encodes\_as\_ids를 통해 정수 시퀀스 확인 (단어를 벡터로 임베딩하기 위해서는 정수 시퀀스를 입력해줘야 함)

```
import csv
vocab_list = pd.read_csv('product.vocab', sep='\\t', header=None, quoting=csv.QUOTE_NONE)
vocab_list[:10]
```

|   | 0     | 1  |
|---|-------|----|
| 0 | <unk> | 0  |
| 1 | <s>   | 0  |
| 2 | </s>  | 0  |
| 3 | _1    | 0  |
| 4 | 00    | -1 |
| 5 | _스    | -2 |
| 6 | _s    | -3 |
| 7 | _아    | -4 |
| 8 | _2    | -5 |
| 9 | _a    | -6 |

```
len(vocab_list)
```

35000

```
lines = [
    "직소퍼즐 1000조각 바다거북의 여행 pl1275",
    "모리케이스 아이폰6s 6s tree farm101 다이어리케이스 바보사랑 무료배송",
]
for line in lines:
    print(line)
    print(sp.encode_as_pieces(line))
    print(sp.encode_as_ids(line))
    print()
```

직소퍼즐 1000조각 바다거북의 여행 pl1275  
['\_직소퍼즐', '\_1000', '조각', '\_바다거북', '의', '\_여행', '\_pl', '1275']  
[1643, 827, 1600, 30502, 34027, 1660, 495, 24537]

모리케이스 아이폰6s 6s tree farm101 다이어리케이스 바보사랑 무료배송  
['\_모리케이스', '\_아이폰', '6', 's', '\_6', 's', '\_tree', '\_farm', '101', '\_다이어리케이스', '\_바보사랑', '\_무료배송']  
[1806, 335, 33882, 33871, 26, 33871, 12876, 27511, 665, 1378, 146, 224]

## 04 데이터 EDA 및 전처리

```
from sklearn.model_selection import train_test_split
import tensorflow as tf
```

```
data = product_name
target = train_df['bcateid']
```

```
X_train, X_test, y_train, y_test = train_test_split(data, target, test_size=0.2, shuffle=True, stratify=target)
```

```
max_len = 109
```

```
X_train = tf.keras.preprocessing.sequence.pad_sequences(X_train, maxlen=max_len)
```

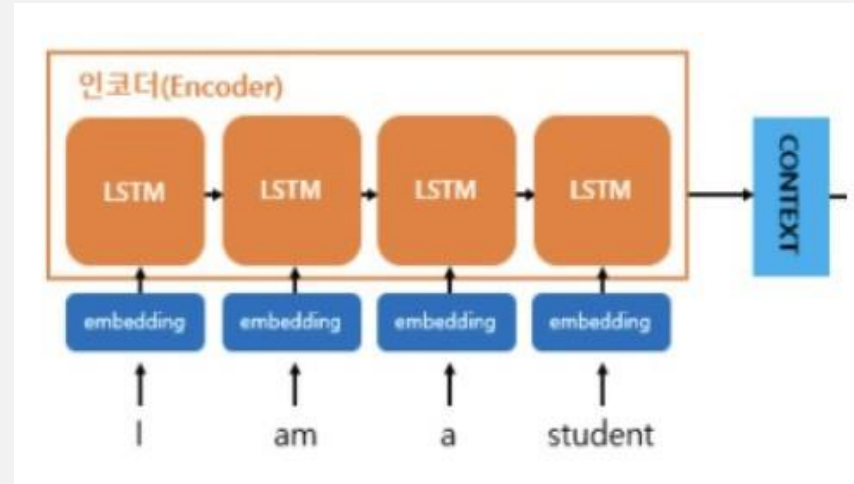
```
X_test = tf.keras.preprocessing.sequence.pad_sequences(X_test, maxlen=max_len)
```

X\_train

```
array([[ 0,  0,  0, ..., 362, 237, 8816],
       [ 0,  0,  0, ..., 48, 33871, 11122],
       [ 0,  0,  0, ..., 15561, 773, 33895],
       ...,
       [ 0,  0,  0, ..., 4280, 8, 33887],
       [ 0,  0,  0, ..., 9813, 1461, 6840],
       [ 0,  0,  0, ..., 33926, 33980, 2605]])
```

- Sklearn을 사용해 훈련, 테스트 세트 분리
  - SentencePiece를 활용해 인덱스로 변환된 상품명 데이터를 독립변수로 할당
  - 대분류 아이디는 타겟 데이터로 할당
  - 20% 비율로 테스트를 가지게 함
- Tensorflow의 pad\_sequences 함수를 사용해 패딩
  - 텍스트 데이터를 모델에 입력하기 위해서는 모두 같은 길이를 가져야 함
  - 따라서 최대 길이 109로 지정함
  - 최대 길이보다 짧은 텍스트는 앞에 0이 채워지도록 함
  - padding='post'로 지정해서 0을 뒤에 채우면 LSTM 학습이 잘 안됨 (정확도 10%에서 향상이 안 됨)

## 05 예측에 활용한 모델 (LSTM)



- 기존 RNN에서 장기 의존성 문제를 해결하기 위해 LSTM이 등장
- LSTM에 들어가기 전에 embedding 층을 통해 텍스트 데이터를 벡터로 임베딩함
- 가장 마지막에 출력되는 hidden state가 context vector가 됨
- Context vector는 상품명의 각 단어에 대한 모든 정보를 함축적으로 포함한 벡터가 됨
- 마지막 출력층에는 클래스 개수만큼 뉴런 개수를 둠
- Softmax를 통해 각 클래스에 대한 확률을 출력

## 05 예측에 활용한 모델 (LSTM)

```
vocab_size = 35000 # sentencepiece와 동일
embedding_dim = 128
hidden_units = 128
num_classes = 53 # 카테고리 ID는 1부터 시작하므로 카테고리 개수 + 1 해줘야 함

model2 = Sequential()
model2.add(Embedding(vocab_size, embedding_dim))
model2.add(LSTM(hidden_units))
model2.add(Dense(num_classes, activation='softmax'))

es = EarlyStopping(monitor='val_loss', mode = "auto", verbose=1, patience=4)
mc = ModelCheckpoint('best_model2.h5', monitor='val_loss', mode = "auto", verbose=1, save_best_only=True)

model2.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history2 = model2.fit(X_train, y_train, batch_size=32, epochs=30, callbacks=[es, mc], validation_split=0.2)
```

- Embedding 층을 통해 35000개의 인덱스로 변환된 단어(데이터 사전 수)들을 128차원을 가진 벡터로 변환
- 클래스 개수를 카테고리 개수 + 1을 해줘야 함
- 카테고리 ID는 0이 아닌 1부터 시작하기 때문에 +1을 해줘야 오류가 안 생김
- 과적합을 방지하기 위해 earlystopping 적용
- 최적화 함수는 Adam을 사용, 다중 분류이기 때문에 손실값은 'sparse\_categorical\_crossentropy'로 지정 (하이퍼파라미터는 모든 모델에 동일하게 적용)

## 05 예측에 활용한 모델 (LSTM)

```
vocab_size = 35000 # sentencepiece와 동일
embedding_dim = 128
hidden_units = 128
num_classes = 53 # 카테고리 ID는 1부터 시작하므로 카테고리 개수 + 1 해줘야 함

model1 = Sequential()
model1.add(Embedding(vocab_size, embedding_dim))
model1.add(LSTM(hidden_units, return_sequences=True))
model1.add(LSTM(hidden_units))
model1.add(Dropout(0.2))
model1.add(Dense(hidden_units, activation='tanh'))
model1.add(Dense(num_classes, activation='softmax'))

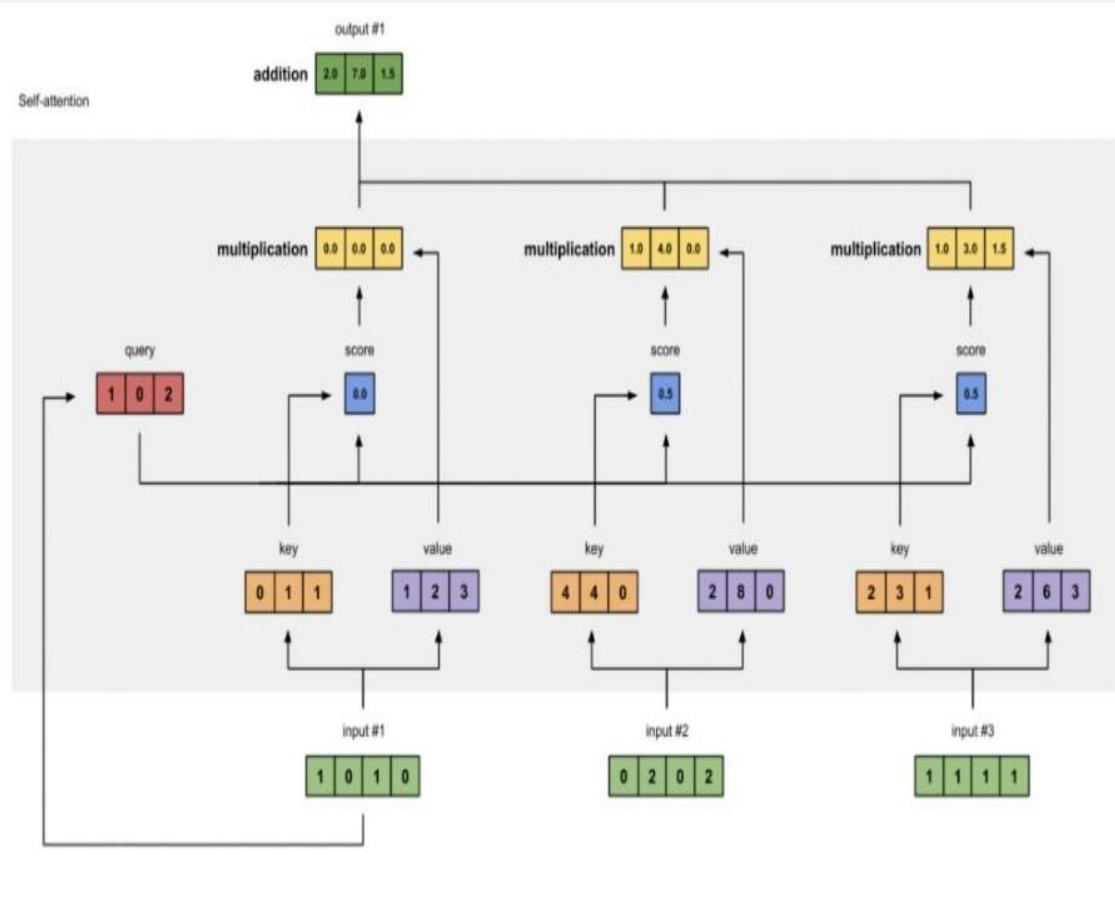
es = EarlyStopping(monitor='val_loss', mode = "auto", verbose=1, patience=4)
mc = ModelCheckpoint('best_model1.h5', monitor='val_loss', mode = "auto", verbose=1, save_best_only=True)

model1.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history1 = model1.fit(X_train, y_train, batch_size=32, epochs=30, callbacks=[es, mc], validation_split=0.2)
```

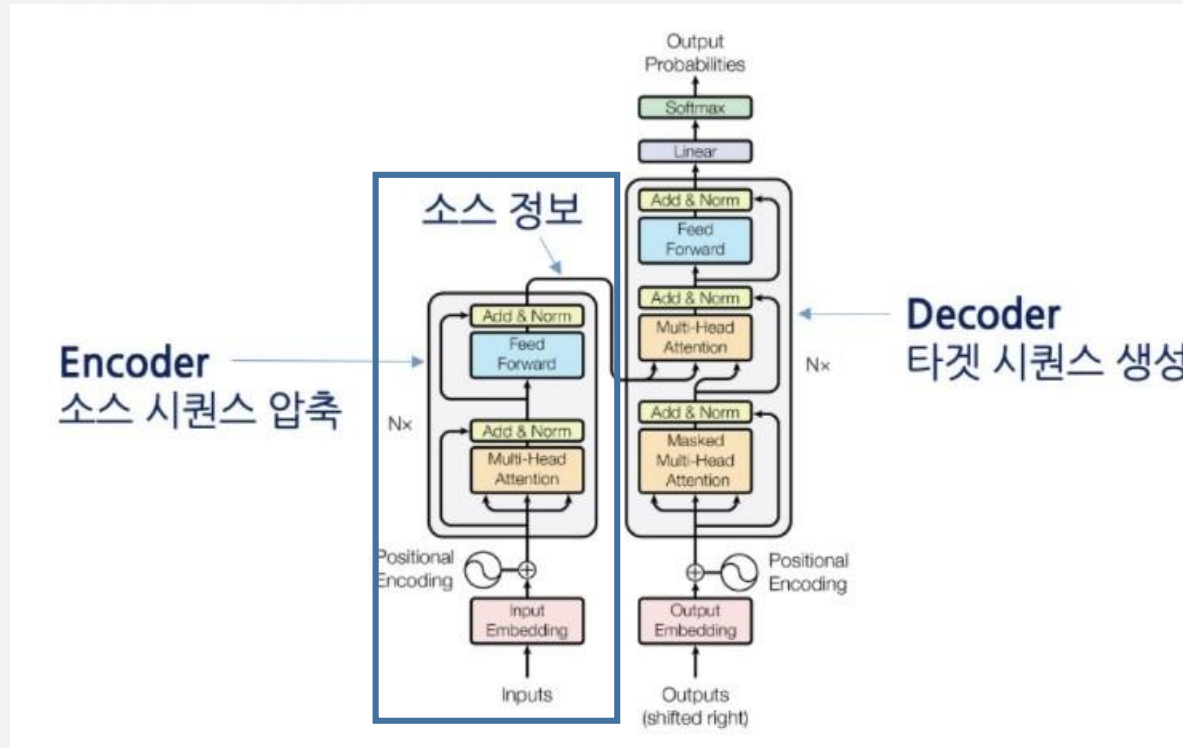
- 2개 층으로 이루어진 LSTM 모델
- 과적합 방지를 위해 dropout 적용
- 마지막 출력층 전에 또 다른 하나의 은닉층을 사용
- 하이퍼파라미터는 이전 LSTM 모델과 동일

## 05 예측에 활용한 모델 (Self-Attention)



- 기존 LSTM 인코더는 하나의 텍스트를 하나의 context vector로 압축하는 과정에서 입력 시퀀스의 정보가 일부 손실될 수 있음
- 각 단어와 다른 단어 간의 관계에 대한 정보를 가진 attention을 활용해 모든 단어에 대한 정보를 활용하고자 하는 것이 기본적인 개념
- 학습을 통해 각 단어에 대한 Query (물어보는 주체), Key(물어보는 대상), Value(각 단어의 값) 값을 얻음
- 이러한 값들을 이용해 각 단어마다 attention 값을 구해줌
- Self-attention을 통해 문맥정보를 학습하도록 함

## 05 예측에 활용한 모델 (Self-Attention)



- Input Embedding과 Positional Encoding을 활용해 각 단어의 위치가 담긴 벡터 추출
- 마지막 Position Feed Forward 네트워크를 거쳐서 각 단어의 문맥 정보가 담긴 벡터를 추출
- Attention is all you need 논문 참고



## 05 예측에 활용한 모델 (Self-Attention)

```
embedding_dim = 128 # 각 단어의 임베딩 벡터의 차원
num_heads = 2 # 어텐션 헤드의 수
dff = 128 # 포지션 와이즈 피드 포워드 신경망의 은닉층의 크기
vocab_size = 35000 # sentencepiece와 동일
max_len = 109 # 샘플명 최대 길이는 109개 단어

inputs = tf.keras.layers.Input(shape=(max_len,))
embedding_layer = TokenAndPositionEmbedding(max_len, vocab_size, embedding_dim)
x = embedding_layer(inputs)
transformer_block = TransformerBlock(embedding_dim, num_heads, dff)
x = transformer_block(x)
x = tf.keras.layers.GlobalAveragePooling1D()(x)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.Dense(20, activation="relu")(x)
x = tf.keras.layers.Dropout(0.1)(x)
outputs = tf.keras.layers.Dense(53, activation="softmax")(x) # 카테고리 ID는 1부터 시작하므로 카테고리 개수 + 1 해줘야 함

model = tf.keras.Model(inputs=inputs, outputs=outputs)

from keras.callbacks import ModelCheckpoint, EarlyStopping

es = EarlyStopping(monitor='val_loss', mode = "auto", verbose=1, patience=4)
mc = ModelCheckpoint('best_model.h5', monitor='val_loss', mode = "auto", verbose=1, save_best_only=True)

model.compile("adam", "sparse_categorical_crossentropy", metrics=["accuracy"])
history = model.fit(X_train, y_train, batch_size=32, epochs=30, callbacks=[es, mc], validation_split=0.2)
```

- Self-attention mechanism을 구현하기 위한 과정을 class로 지정함
- 하이퍼파라미터는 이전과 동일함
- <https://wikidocs.net/103802> (코드 참고)

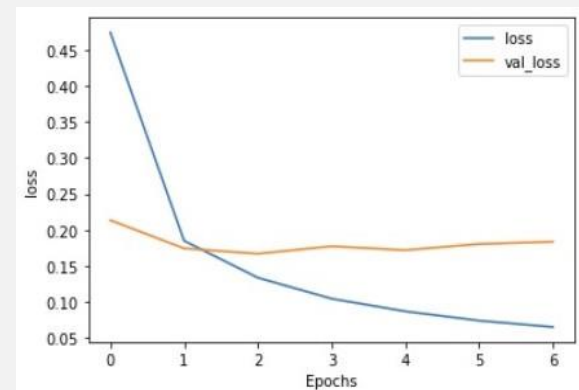
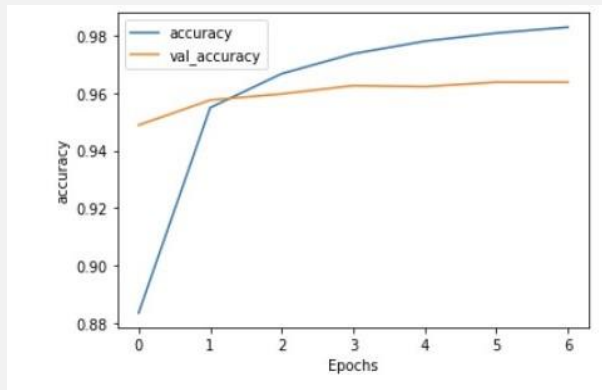


## 06 결과

### ➤ Self-Attention

```
print("테스트 정확도: %.4f" % (model.evaluate(X_test, y_test)[1]))
```

```
6250/6250 [=====] - 30s 5ms/step - loss: 0.1835 - accuracy: 0.9637  
테스트 정확도: 0.9637
```

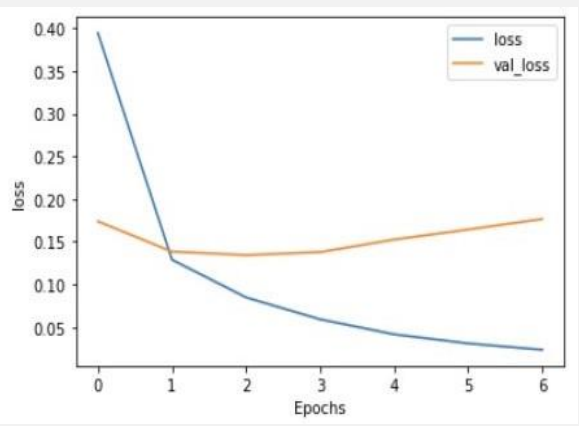
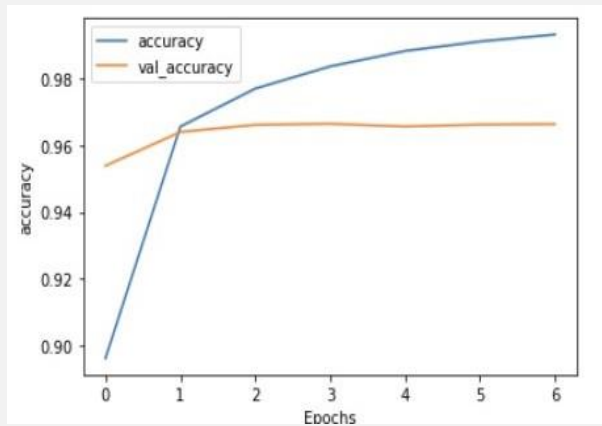


### ➤ Multi-layered LSTM

```
print("\n 테스트 정확도: %.4f" % (model1.evaluate(X_test, y_test)[1]))
```

```
6250/6250 [=====] - 44s 7ms/step - loss: 0.1796 - accuracy: 0.9656
```

테스트 정확도: 0.9656

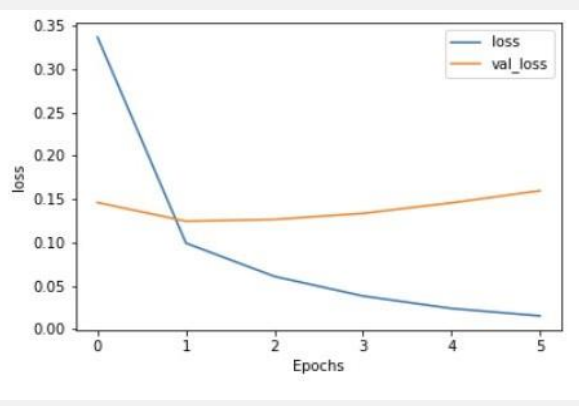
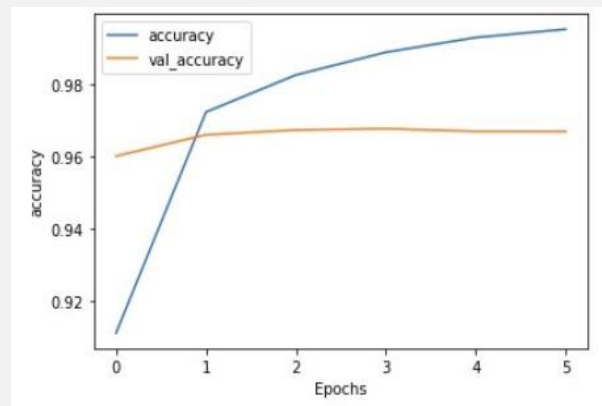


### ➤ 일반 LSTM

```
print("\n 테스트 정확도: %.4f" % (model2.evaluate(X_test, y_test)[1]))
```

```
6250/6250 [=====] - 26s 4ms/step - loss: 0.1600 - accuracy: 0.9666
```

테스트 정확도: 0.9666



## 06 결과

### ➤ Self-Attention

|    |      |      |      |      |
|----|------|------|------|------|
| 24 | 0.97 | 0.96 | 0.96 | 3780 |
| 25 | 0.92 | 0.94 | 0.93 | 1948 |
| 26 | 0.96 | 0.97 | 0.96 | 829  |
| 27 | 0.89 | 0.93 | 0.91 | 1188 |
| 28 | 0.97 | 0.98 | 0.98 | 5956 |
| 29 | 0.68 | 0.81 | 0.74 | 91   |
| 30 | 0.97 | 0.96 | 0.96 | 2059 |
| 31 | 0.00 | 0.00 | 0.00 | 7    |
| 32 | 0.97 | 0.96 | 0.96 | 2419 |
| 33 | 0.98 | 0.96 | 0.97 | 1567 |
| 34 | 0.97 | 0.97 | 0.97 | 3157 |
| 35 | 0.96 | 0.93 | 0.95 | 2374 |
| 36 | 0.95 | 0.95 | 0.95 | 1988 |
| 37 | 0.94 | 0.94 | 0.94 | 2363 |
| 38 | 0.93 | 0.96 | 0.94 | 1418 |
| 39 | 0.94 | 0.95 | 0.94 | 1855 |
| 40 | 0.94 | 0.95 | 0.95 | 2693 |
| 41 | 0.94 | 0.94 | 0.94 | 1012 |
| 42 | 0.94 | 0.96 | 0.95 | 2005 |
| 43 | 0.89 | 0.90 | 0.90 | 576  |
| 44 | 0.96 | 0.92 | 0.94 | 1225 |
| 45 | 0.96 | 0.93 | 0.94 | 1738 |
| 46 | 0.96 | 0.92 | 0.94 | 85   |
| 47 | 0.89 | 0.73 | 0.80 | 11   |
| 48 | 0.93 | 0.88 | 0.90 | 189  |
| 49 | 0.91 | 0.94 | 0.93 | 740  |
| 50 | 0.96 | 0.94 | 0.95 | 1998 |
| 51 | 0.93 | 0.94 | 0.94 | 661  |
| 52 | 0.93 | 0.92 | 0.93 | 162  |

### ➤ Multi-layered LSTM

|    |      |      |      |      |
|----|------|------|------|------|
| 25 | 0.95 | 0.94 | 0.95 | 1948 |
| 26 | 0.96 | 0.96 | 0.96 | 829  |
| 27 | 0.91 | 0.94 | 0.92 | 1188 |
| 28 | 0.97 | 0.98 | 0.98 | 5956 |
| 29 | 0.70 | 0.89 | 0.78 | 91   |
| 30 | 0.97 | 0.96 | 0.96 | 2059 |
| 31 | 0.00 | 0.00 | 0.00 | 7    |
| 32 | 0.98 | 0.94 | 0.96 | 2419 |
| 33 | 0.98 | 0.97 | 0.98 | 1567 |
| 34 | 0.99 | 0.96 | 0.98 | 3157 |
| 35 | 0.92 | 0.96 | 0.94 | 2374 |
| 36 | 0.95 | 0.94 | 0.94 | 1988 |
| 37 | 0.94 | 0.94 | 0.94 | 2363 |
| 38 | 0.95 | 0.95 | 0.95 | 1418 |
| 39 | 0.95 | 0.95 | 0.95 | 1855 |
| 40 | 0.96 | 0.95 | 0.96 | 2693 |
| 41 | 0.96 | 0.91 | 0.94 | 1012 |
| 42 | 0.96 | 0.96 | 0.96 | 2005 |
| 43 | 0.92 | 0.87 | 0.90 | 576  |
| 44 | 0.98 | 0.93 | 0.95 | 1225 |
| 45 | 0.96 | 0.94 | 0.95 | 1738 |
| 46 | 0.98 | 0.98 | 0.98 | 85   |
| 47 | 1.00 | 0.73 | 0.84 | 11   |
| 48 | 0.87 | 0.84 | 0.86 | 189  |
| 49 | 0.92 | 0.93 | 0.93 | 740  |
| 50 | 0.95 | 0.94 | 0.94 | 1998 |
| 51 | 0.93 | 0.94 | 0.93 | 661  |
| 52 | 0.94 | 0.95 | 0.95 | 162  |

### ➤ 일반 LSTM

|    |      |      |      |      |
|----|------|------|------|------|
| 25 | 0.94 | 0.94 | 0.94 | 1948 |
| 26 | 0.97 | 0.97 | 0.97 | 829  |
| 27 | 0.93 | 0.93 | 0.93 | 1188 |
| 28 | 0.98 | 0.98 | 0.98 | 5956 |
| 29 | 0.94 | 0.90 | 0.92 | 91   |
| 30 | 0.97 | 0.96 | 0.97 | 2059 |
| 31 | 0.00 | 0.00 | 0.00 | 7    |
| 32 | 0.97 | 0.96 | 0.96 | 2419 |
| 33 | 0.96 | 0.98 | 0.97 | 1567 |
| 34 | 0.97 | 0.98 | 0.97 | 3157 |
| 35 | 0.96 | 0.95 | 0.96 | 2374 |
| 36 | 0.94 | 0.96 | 0.95 | 1988 |
| 37 | 0.95 | 0.94 | 0.95 | 2363 |
| 38 | 0.95 | 0.95 | 0.95 | 1418 |
| 39 | 0.93 | 0.95 | 0.94 | 1855 |
| 40 | 0.97 | 0.95 | 0.96 | 2693 |
| 41 | 0.96 | 0.93 | 0.94 | 1012 |
| 42 | 0.97 | 0.96 | 0.96 | 2005 |
| 43 | 0.92 | 0.89 | 0.90 | 576  |
| 44 | 0.96 | 0.94 | 0.95 | 1225 |
| 45 | 0.93 | 0.96 | 0.94 | 1738 |
| 46 | 0.98 | 0.95 | 0.96 | 85   |
| 47 | 1.00 | 0.64 | 0.78 | 11   |
| 48 | 0.85 | 0.88 | 0.87 | 189  |
| 49 | 0.94 | 0.93 | 0.93 | 740  |
| 50 | 0.93 | 0.94 | 0.94 | 1998 |
| 51 | 0.94 | 0.93 | 0.94 | 661  |
| 52 | 0.96 | 0.95 | 0.95 | 162  |

## 07 결과 해석 및 결론

### • 결과해석

- 3가지 모델 모두 데이터가 적은 카테고리(카테고리 ID 7) 경우에는 분류 예측을 아무것도 하지 못함
- 데이터 수가 11개로 비교적 적은 카테고리 ID인 47번 같은 경우에는 다층 LSTM이 제일 높은 성능을 보여줌
- 데이터 수가 많은 다른 카테고리 ID는 3가지 모델 모두 좋은 성능을 보여줌
- 하지만 전체적으로 보면, 상품명을 활용한 카테고리 분류에서는 최신 기법인 self-attention은 LSTM과 유의미한 성능차이를 보이지 않았음
- 상품명 같은 경우에는 문맥적인 의미를 가지기 보다는 명사 나열형이 많음  
ex) 아이폰 7 플러스 저반사 지문방지 액정보호필름
- 순서를 바꿔어도 상품명에 텍스트의 의미가 크게 달라지지 않음  
ex) 직소퍼즐 1000조각 바다거북의 여행 -> 1000조각 바다거북의 여행 직소퍼즐
- 이러한 이유와 더불어 상품명 컬럼 데이터는 대부분 10개 단어 이하로 이루어져 있기 때문에 LSTM 인코더처럼 하나의 context vector로 압축해도 이전 입력 시퀀스의 정보가 크게 사라지지 않음
- 따라서 간단한 LSTM으로도 충분한 좋은 분류 성능을 낼 수 있음

### • 결론

- 상품 카테고리 분류기가 필요한 e-commerce 회사들은 복잡한 attention 기법을 사용하기 보다는 간단한 구조의 LSTM 구조를 사용하는 것이 더 효율적일 수 있다.
- 하지만 명사 나열형이 아닌 상품에 대한 정보를 문장으로 표현하는 텍스트로 분류한다고 했을 때는 attention 기법이 더 정확한 성능을 낼 수 있다.
- 상품에 대한 데이터가 어떠한 방식으로 이루어지는지에 따라서 달라짐

# Appendix

- **참고자료**

- 최규민 외, *카카오 아레나 데이터 경진대회 1등 노하우*, 파주:위키북스, 2021
- “셀프 어텐션(Self-Attention)”, *ratsgo's blog*, [https://ratsgo.github.io/nlpbook/docs/language\\_model/tr\\_self\\_attention/](https://ratsgo.github.io/nlpbook/docs/language_model/tr_self_attention/)
- “셀프 어텐션을 이용한 텍스트 분류(Multi-head Self Attention for Text Classification)”, *WikiDocs*, <https://wikidocs.net/103802>
- 전창욱 외, *텐서플로 2와 머신러닝으로 시작하는 자연어 처리 로지스틱 회귀부터 BERT와 GPT2까지*, 파주:위키북스, 2020