

AI 딥러닝

# 서울시 맛집 추천 챗봇

여기가 챗봇 맛집이죠

Table of Contents

# 목차

01. 조원 소개
02. 프로젝트 일정
03. 주제 선정 배경
04. 데이터 준비
05. 데이터 전처리
06. 활용 기술
07. 모델링

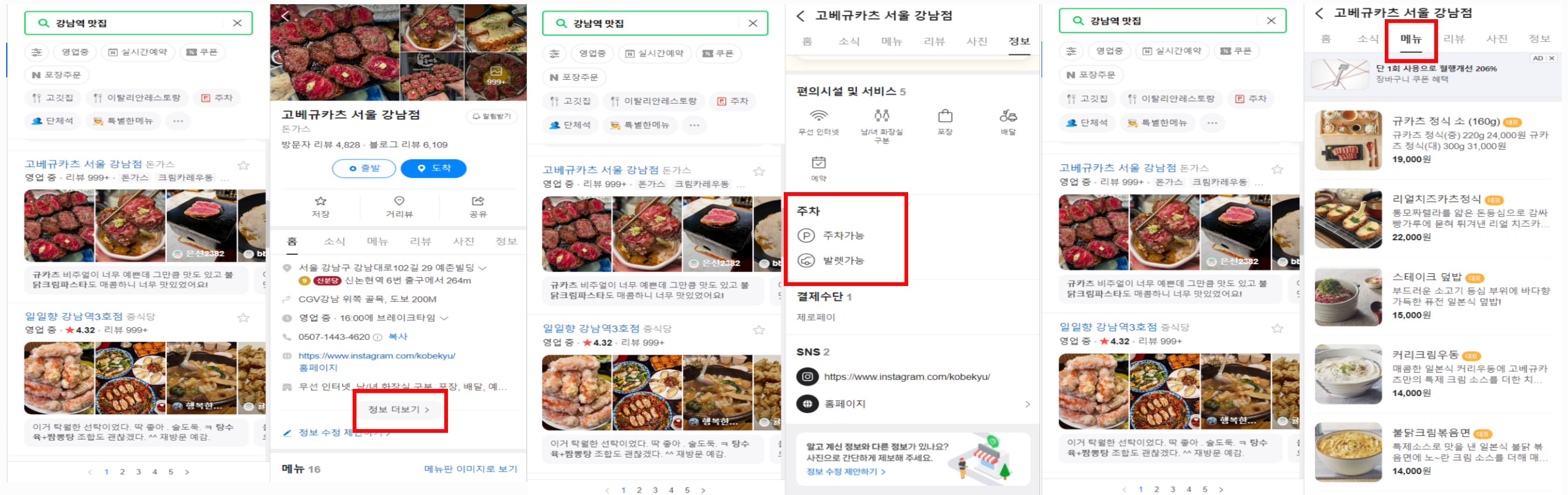
# 조원 소개

서유진, 차다훈, 이혁빈, 김한솔

# 프로젝트 일정

[illegible]

# 주제 선정 배경



주차시설, 메뉴, 주소 등 자세한 정보를 확인하기 위한 절차 복잡  
챗봇과 대화로 쉽게 정보 획득

# 데이터 준비

[AI 데이터찾기](#)[AI 허브소개](#)[참여하기](#)[커뮤니티](#)[AI 개발지원](#)[고객지원](#)[로그인](#)[회원가입](#)

데이터 찾기

[AI 데이터찾기](#)[데이터 찾기](#)

수도권

#관광

#POI

#이미지

#Triple

#QA

#KVQA

#Knowledge-Aware

#Visual

#관광타입

#관광지

NEW

## 관광 KVQA 데이터(수도권)

분야

문화관광

유형

텍스트, 이미지

구축년도 : 2022

갱신년월 : 2024-02

조회수 : 3,129

다운로드 : 135

용량 : 9.14 TB

다운로드



샘플 데이터



관심데이터 등록

11

소개

파일 목록 (API 다운로드)

# 데이터 전처리

```
{
  "question_id": 0,
  "question_type": "정보",
  "question": "대표 메뉴가 무엇인가요?",
  "question_wordNum": 3,
  "answer": "고기만두전골, 김치만두전골, 조랭이떡 만두국, 가래떡 만두국",
  "answer_wordNum": 6,
  "fact": [
    "개성만두 궁,메뉴,고기만두전골, 김치만두전골, 조랭이떡 만두국, 가래떡 만두국"
  ],
},
{
  "question_id": 1,
  "question_type": "정보",
  "question": "영업시간을 알 수 있나요?",
  "question_wordNum": 4,
  "answer": "11:30 ~21:00",
  "answer_wordNum": 2,
  "fact": [
    "개성만두 궁,영업시간,11:30 ~21:00"
  ]
},
{
  "question_id": 2,
  "question_type": "탐색",
  "question": "주차시설이 있는지 알 수 있나요?",
  "question_wordNum": 5,
  "answer": "없음",
  "answer_wordNum": 1,
  "fact": [
    "개성만두 궁,주차시설,없음"
  ]
}
```

음식점의 정보에 대한 질문

질문에 대한 답변이  
[상호명, 키워드, 답변]의  
형태임을 확인



# 데이터 전처리

```
import json
import pandas as pd
import os
import re
import random

data_dir = 'data'
file_list = os.listdir(data_dir)
questions = []
facts = []
for file_name in random.sample(file_list, 2000):
    with open(os.path.join(data_dir, file_name), 'r') as file:
        data = json.load(file)
        for annotation in data['annotations']:
            for question in annotation['question']:
                questions.append(question['question'])
                facts.append(question['fact'][0].split(','))
```

```
df = pd.DataFrame({'question': questions, 'answer': facts})
df
```

	question	answer
0	이 곳의 대표적인 메뉴를 알 수 있나요?	[대성각, 메뉴, 짜장면, 짬뽕]
1	영업시간은 어떻게 되나요?	[대성각, 영업시간, 11:00 - 15:00/21:00]
2	연락처를 알 수 있나요?	[대성각, 연락처, 02-356-2194]
3	이 곳에 인접한 시설을 알 수 있나요?	[대성각, 인접, 역촌역 3번출구, 그림나라아동미술, 메리피아노음악교습소]
4	이 곳에 인접한 시설을 알 수 있나요?	[대성각, 인접, 역촌역 3번출구, 그림나라아동미술, 메리피아노음악교습소]
...	...	...
39995	이 곳에 인접한 시설이 있나요?	[목동양대창, 인접, 목동역 7번 출구, 양목초등학교, 돈카츠마켄 목동점]
39996	휴무일을 알 수 있나요?	[목동양대창, 휴무일, 없음]
39997	영업시간을 알 수 있나요?	[목동양대창, 영업시간, 17:00-24:00]
39998	이 곳의 대표적인 메뉴를 알 수 있나요?	[목동양대창, 메뉴, 양대창, 소곱창]
39999	이 곳에 인접한 시설이 있나요?	[목동양대창, 인접, 목동역 7번 출구, 양목초등학교, 돈카츠마켄 목동점]
40000 rows × 2 columns		

파일 중 랜덤으로 2000개만 뽑아서 데이터 사용



# 데이터 전처리

```
# 키워드 추출하여 도수표  
df['answer'].apply(lambda x: x[1]).value_counts()
```

메뉴	6273
주차시설	5767
영업시간	5754
인접	5494
주소	5383
연락처	5319
휴무일	5281
이다	212
전문	164
가능	89
사용	53
위치	42
판매	33
제공	24

```
# ['메뉴', '영업시간', '주차시설', '인접', '주소', '연락처', '휴무일']에 관한 질문-답변만 사용  
df = df[df['answer'].apply(lambda x: x[1] in ['메뉴', '영업시간', '주차시설', '인접', '주소', '연락처', '휴무일'])]  
df
```

	question	answer
0	이 곳의 대표적인 메뉴를 알 수 있나요?	[대성각, 메뉴, 짜장면, 짬뽕]
1	영업시간은 어떻게 되나요?	[대성각, 영업시간, 11:00 - 15:00/21:00]
2	연락처를 알 수 있나요?	[대성각, 연락처, 02-356-2194]
3	이 곳에 인접한 시설을 알 수 있나요?	[대성각, 인접, 역촌역 3번출구, 그림나라아동미술, 메리피아노음악교습소]
4	이 곳에 인접한 시설을 알 수 있나요?	[대성각, 인접, 역촌역 3번출구, 그림나라아동미술, 메리피아노음악교습소]
...	...	...
39995	이 곳에 인접한 시설이 있나요?	[목동양대창, 인접, 목동역 7번 출구, 양목초등학교, 돈카츠마케 목동점]
39996	휴무일을 알 수 있나요?	[목동양대창, 휴무일, 없음]
39997	영업시간을 알 수 있나요?	[목동양대창, 영업시간, 17:00-24:00]
39998	이 곳의 대표적인 메뉴를 알 수 있나요?	[목동양대창, 메뉴, 양대창, 소곱창]
39999	이 곳에 인접한 시설이 있나요?	[목동양대창, 인접, 목동역 7번 출구, 양목초등학교, 돈카츠마케 목동점]

39271 rows x 2 columns

상위 7개 키워드와 나머지의 데이터의 차이가 커 상위 7개만 사용

# 데이터 전처리

```
df = df[~df['answer'].apply(lambda x: bool(re.search(r'[0-9a-zA-Z]', x[0])))]  
df = df[df.apply(lambda row: row['answer'][1] in row['question'], axis=1)]  
df
```

	question	answer
0	이 곳의 대표적인 메뉴를 알 수 있나요?	[대성각, 메뉴, 짜장면, 짬뽕]
1	영업시간은 어떻게 되나요?	[대성각, 영업시간, 11:00 - 15:00/21:00]
2	연락처를 알 수 있나요?	[대성각, 연락처, 02-356-2194]
3	이 곳에 인접한 시설을 알 수 있나요?	[대성각, 인접, 역촌역 3번출구, 그림나라아동미술, 메리피아노음악교습소]
4	이 곳에 인접한 시설을 알 수 있나요?	[대성각, 인접, 역촌역 3번출구, 그림나라아동미술, 메리피아노음악교습소]
...	...	...
39995	이 곳에 인접한 시설이 있나요?	[목동양대창, 인접, 목동역 7번 출구, 양목초등학교, 돈카츠마켄 목동점]
39996	휴무일을 알 수 있나요?	[목동양대창, 휴무일, 없음]
39997	영업시간을 알 수 있나요?	[목동양대창, 영업시간, 17:00-24:00]
39998	이 곳의 대표적인 메뉴를 알 수 있나요?	[목동양대창, 메뉴, 양대창, 소곱창]
39999	이 곳에 인접한 시설이 있나요?	[목동양대창, 인접, 목동역 7번 출구, 양목초등학교, 돈카츠마켄 목동점]

36470 rows × 2 columns

kobert 사용 예정으로 상호명에 숫자나 영어가 들어가있으면 제외

# 데이터 전처리

이 곳에 인접한 시설이 있나요?  
이 곳에 인접한 시설이 있나요?  
이 곳의 주요 메뉴를 알 수 있나요?  
이 곳의 대표적인 메뉴를 알 수 있나요?  
영업시간은 어떻게 되나요?  
이 곳의 주요 메뉴를 알 수 있나요?  
휴무일이 있나요?  
이 곳의 주소는 어떻게 되나요?  
휴무일을 알 수 있나요?  
이 곳에 인접한 시설이 있나요?  
이 곳에 인접한 시설을 알 수 있나요?  
이 곳의 주소를 알 수 있나요?

```
new_df['questions'] = new_df.apply(lambda row:  
    row['question'].replace('이 곳', row['fact'][0])  
    if isinstance(row['fact'], list) and '이 곳' in row['question']  
    else row['fact'][0] + '의 ' + row['question'], axis=1)
```

'이 곳'이 포함된 질문에서 '이 곳'을 음식점 상호명으로 대체

# 데이터 전처리

answers	questions
봉구비어 서울시립대점의 휴무일은 알수없습니다	봉구비어 서울시립대점의 휴무일은 어떻게 되나요?
몽크2의 주차시설은 없습니다	몽크2의 주차시설이 있나요?
포대포의 주소는 서울 용산구 청파로 332입니다	포대포의 주소를 알 수 있나요?
79파운야드 가산디지털단지점의 주차시설은 있습니다	79파운야드 가산디지털단지점의 주차시설이 있는지 알 수 있나요?
써브웨이 종암점의 영업시간은 08:00-22:00입니다	써브웨이 종암점의 영업시간을 알 수 있나요?
...	...
오제제 서울역의 주소는 서울 용산구 한강대로 363-2 1층입니다	오제제 서울역의 주소를 알 수 있나요?
한양숯불갈비의 주차시설은 없습니다	한양숯불갈비의 주차시설이 있나요?
동흥관의 메뉴는 삼선짬뽕, 짜장면, 탕수육입니다	동흥관의 대표 메뉴가 무엇인가요?
까치네분식 당산역점의 영업시간은 00:00 - 24:00입니다	까치네분식 당산역점의 영업시간을 알 수 있나요?
배스킨라빈스 마포구청역의 영업시간은 10:00 - 23:00입니다	배스킨라빈스 마포구청역의 영업시간을 알 수 있나요?

# 데이터 전처리

```
new_df = df[df['answer'].apply(lambda x: '주소' in x[1])]
new_df['location'] = new_df['answer'].apply(lambda x: x[2].split(' ')[1])
new_df
```

	question	answer	location
6	이 곳의 주소를 알 수 있나요?	[대성각, 주소, 서울 은평구 녹번로 7]	은평구
10	이 곳의 주소를 알 수 있나요?	[대성각, 주소, 서울 은평구 녹번로 7]	은평구
15	이 곳의 주소를 알 수 있나요?	[대성각, 주소, 서울 은평구 녹번로 7]	은평구
17	이 곳의 주소를 알 수 있나요?	[대성각, 주소, 서울 은평구 녹번로 7]	은평구
24	이 곳의 주소를 알 수 있나요?	[가문의우동, 주소, 서울 용산구 청파로47다길 5-1]	용산구
...	...	...	...
39972	이 곳의 주소를 알 수 있나요?	[짬지계란말이김밥, 주소, 서울 구로구 디지털로 236-1]	구로구
39979	이 곳의 주소를 알 수 있나요?	[짬지계란말이김밥, 주소, 서울 구로구 디지털로 236-1]	구로구
39981	이 곳의 주소를 알 수 있나요?	[목동양대창, 주소, 서울 양천구 오목로46길 32 1층]	양천구
39985	이 곳의 주소를 알 수 있나요?	[목동양대창, 주소, 서울 양천구 오목로46길 32 1층]	양천구
39989	이 곳의 주소는 어떻게 되나요?	[목동양대창, 주소, 서울 양천구 오목로46길 32 1층]	양천구

5002 rows × 3 columns

```
new_df['location'].value_counts()
```

영등포구	659
송파구	599
용산구	593
은평구	404
광진구	327
강서구	306
동대문구	284
구로구	265
관악구	222
양천구	199
동작구	183
금천구	179
마포구	136
서초구	116
성동구	104
강남구	81
강동구	68
서대문구	58
성북구	50
노원구	47
강북구	42
도봉구	34
종구	24
종로구	19
중랑구	3

Name: location, dtype: int64

키워드가 주소인 행만 가져와서 구 정보 획득

# 데이터 전처리

```
new_df['restaurant'] = new_df['answer'].apply(lambda x: x[0])
new_df
```

location	restaurant
은평구	대성각
은평구	대성각
은평구	대성각
은평구	대성각
용산구	가문의우동
...	...
구로구	짬지계란말이김밥
구로구	짬지계란말이김밥
양천구	목동양대창
양천구	목동양대창
양천구	목동양대창

```
question_list = ['맛집을 알고싶어', '맛집 추천해줘', '맛집 추천좀', '맛집 알려줄래?', '맛집 알려줘', '맛집 추천 부탁해', '맛집 추천해줄래?']
answer_list = ['가 있습니다', '를 추천드립니다', '가 알려져 있습니다', '에 가보세요']
new_df['questions'] = new_df['location'].apply(lambda x: x + ' ' + random.choice(question_list))
new_df['answers'] = new_df['location'].apply(lambda x: x + '의 맛집으로는 ') + new_df['restaurant'].apply(lambda x: x + random.choice(answer_list))
new_df
```

	location	restaurant	questions	answers
6	은평구	대성각	은평구 맛집 추천해줄래?	은평구의 맛집으로는 대성각가 있습니다
10	은평구	대성각	은평구 맛집 추천해줄래?	은평구의 맛집으로는 대성각를 추천드립니다
15	은평구	대성각	은평구 맛집 추천좀	은평구의 맛집으로는 대성각를 추천드립니다
17	은평구	대성각	은평구 맛집 추천해줄래?	은평구의 맛집으로는 대성각가 알려져 있습니다
24	용산구	가문의우동	용산구 맛집 추천해줄래?	용산구의 맛집으로는 가문의우동을 추천드립니다
...	...	...	...	...
39972	구로구	짬지계란말이김밥	구로구 맛집 알려줄래?	구로구의 맛집으로는 짬지계란말이김밥에 가보세요
39979	구로구	짬지계란말이김밥	구로구 맛집을 알고싶어	구로구의 맛집으로는 짬지계란말이김밥를 추천드립니다
39981	양천구	목동양대창	양천구 맛집 알려줘	양천구의 맛집으로는 목동양대창에 가보세요
39985	양천구	목동양대창	양천구 맛집 추천 부탁해	양천구의 맛집으로는 목동양대창가 있습니다
39989	양천구	목동양대창	양천구 맛집 알려줄래?	양천구의 맛집으로는 목동양대창를 추천드립니다

상호명과 구 정보를 토대로 질문 랜덤작성

# 데이터 전처리

```
df['questions'] = df['questions'].apply(lambda x: x.replace('주소', '위치'))
```

```
def process_answer(answer):
    if isinstance(answer, list):
        if '주차시설' in answer[1]:
            return f"{answer[0]}의 {answer[1]}은 {answer[2]}습니다"
        elif '휴무일' in answer[1]:
            return f"{answer[0]}의 {answer[1]}은 {'', '.join(answer[2:])}입니다"
        elif '주소' in answer[1]:
            return f"{answer[0]}의 위치는 {'', '.join(answer[2:])}입니다"
        elif '연락처' in answer[1]:
            return f"{answer[0]}의 {answer[1]}는 {'', '.join(answer[2:])}입니다"
        elif '영업시간' in answer[1]:
            return f"{answer[0]}의 {answer[1]}은 {answer[2]}입니다"
        elif '메뉴' in answer[1]:
            return f"{answer[0]}의 {answer[1]}에는 {'', '.join(answer[2:])}가 있습니다"
        elif '인접' in answer[1]:
            return f"{answer[0]}에 {answer[1]}한 시설에는 {'', '.join(answer[2:])}가 있습니다"
```

```
result_df = pd.concat([df, new_df], axis=0)
result_df = result_df.drop_duplicates()
result_df
```

	questions	answers
0	대성각의 대표적인 메뉴를 알 수 있나요?	대성각의 메뉴에는 짜장면, 짬뽕가 있습니다
1	대성각의 영업시간은 어떻게 되나요?	대성각의 영업시간은 11:00 - 15:00/21:00입니다
2	대성각의 연락처를 알 수 있나요?	대성각의 연락처는 02-356-2194입니다
3	대성각에 인접한 시설을 알 수 있나요?	대성각에 인접한 시설에는 역촌역 3번출구, 그림나라아동미술, 메리피아노음악교습소...
5	대성각의 휴무일을 알 수 있나요?	대성각의 휴무일은 매주 일요일입니다
...	...	...
39972	구로구 맛집 알려줄래?	구로구의 맛집으로는 짬지계란말이김밥에 가보세요
39979	구로구 맛집을 알고싶어	구로구의 맛집으로는 짬지계란말이김밥을 추천드립니다
39981	양천구 맛집 알려줘	양천구의 맛집으로는 목동양대창에 가보세요
39985	양천구 맛집 추천 부탁해	양천구의 맛집으로는 목동양대창가 있습니다
39989	양천구 맛집 알려줄래?	양천구의 맛집으로는 목동양대창을 추천드립니다

26704 rows × 2 columns

## 맛집 질문 답변 데이터와 기존 전처리된 데이터 결합



# 활용 기술

언어 Python

토큰화 Kober / Sentencepiece

모델링

- Pytorch
- seq2seq with attention
- transformer

배포 flask / botpress

# 모델링

```
!pip install 'git+https://github.com/SKTBrain/KoBERT.git#egg=kobert_tokenizer&subdirectory=kobert_hf'
```

Collecting kobert\_tokenizer

Cloning <https://github.com/SKTBrain/KoBERT.git> to /tmp/pip-install-441rjnbo/kobert-tokenizer\_54fa972c48bf4ff8b7e1c5acb7d66db4

Running command git clone --filter=blob:none --quiet <https://github.com/SKTBrain/KoBERT.git> /tmp/pip-install-441rjnbo/kobert-tokenizer\_54fa972c48bf4ff8b7e1c5acb7d66db4

Resolved <https://github.com/SKTBrain/KoBERT.git> to commit 47a69af87928fc24e20f571fe10c3cc9dd9af9a3

Preparing metadata (setup.py) ... done

Building wheels for collected packages: kobert\_tokenizer

Building wheel for kobert\_tokenizer (setup.py) ... done

Created wheel for kobert\_tokenizer: filename=kobert\_tokenizer-0.1-py3-none-any.whl size=4633 sha256=06a6d8981df125c0fcad31451650d7bded356a81835a8930b13845c62a927d97

Stored in directory: /tmp/pip-ephem-wheel-cache-ctzk44wb/wheels/e9/1a/3f/a864970e8a169c176bfa3c4a1e07aa612f69195907a4045fe

Successfully built kobert\_tokenizer

Installing collected packages: kobert\_tokenizer

Successfully installed kobert\_tokenizer-0.1


질문-답변 토큰화를 위해 skt-brain에서 제공하는 kobert 다운로드


# 모델링


```
import torch
from kobert_tokenizer import KoBERTTokenizer

tokenizer = KoBERTTokenizer.from_pretrained('skt/kobert-base-v1')
tokenizer.encode("하르달 목동역점에 인접한 시설은 뭐가 있어?")

/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens)
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(

tokenizer_config.json: 100%  432/432 [00:00<00:00, 14.6kB/s]

spiece.model: 100%  371k/371k [00:00<00:00, 1.89MB/s]

special_tokens_map.json: 100%  244/244 [00:00<00:00, 2.36kB/s]

The tokenizer class you load from this checkpoint is not the same type as the class this function is called from. It may
The tokenizer class you load from this checkpoint is 'XLNetTokenizer'.
The class this function is called from is 'KoBERTTokenizer'.
[2,
 4924,
 6113,
 5804,
 2070,
 6926,
 7220,
 6896,
 3758,
 7225,
 7828,
 2981,
 7086,
 2145,
 5330,
 3868,
 258,
 3]
```

## 토크나이저 생성 및 성능 확인

# 모델링

```
questions = my_data['questions'][:20000]
answers = my_data['answers'][:20000]

# # 토큰화
question_tokens = [tokenizer.encode(q, add_special_tokens=True) for q in questions]
answer_tokens = [tokenizer.encode(a, add_special_tokens=True) for a in answers]

from torch.nn.utils.rnn import pad_sequence

cls_token_id = tokenizer.cls_token_id
sep_token_id = tokenizer.sep_token_id
pad_token_id = tokenizer.pad_token_id

# 토큰화된 리스트를 PyTorch 텐서로 변환
questions_tensors = [torch.tensor(t, dtype=torch.long) for t in question_tokens]
answers_tensors = [torch.tensor(t, dtype=torch.long) for t in answer_tokens]

# 패딩을 위한 최대 길이 결정
questions_max_length = max(tensor.size(0) for tensor in questions_tensors)
answers_max_length = max(tensor.size(0) for tensor in answers_tensors)
max_length = max(questions_max_length, answers_max_length)

# 패딩 함수
def pad_or_truncate(tensors, max_length, pad_token_id):
    padded_tensors = []
    for tensor in tensors:
        if tensor.size(0) < max_length:
            padded_tensor = torch.cat([tensor, torch.tensor([pad_token_id] * (max_length - tensor.size(0)), dtype=torch.long)])
        else:
            padded_tensor = tensor[:max_length]
        padded_tensors.append(padded_tensor)
    return pad_sequence(padded_tensors, batch_first=True)

# 패딩 및 트렁케이팅 적용
questions_padded = pad_or_truncate(questions_tensors, max_length, pad_token_id)
answers_padded = pad_or_truncate(answers_tensors, max_length, pad_token_id)

# 패딩된 텐서의 모양
print("Padded train_Questions Shape:", questions_padded.shape)
print("Padded train_Answers Shape:", answers_padded.shape)

Padded train_Questions Shape: torch.Size([20000, 72])
Padded train_Answers Shape: torch.Size([20000, 72])
```

질문-답변 쌍 토큰화

텐서로 변환

질문-답변 중 긴 쪽의  
길이를 패딩 추가

# 모델링

```
class PositionalEncoding(nn.Module):
    def __init__(self, d_model, max_len=5000):
        super(PositionalEncoding, self).__init__()

        pe = torch.zeros(max_len, d_model)
        position = torch.arange(0, max_len, dtype=torch.float).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, d_model, 2).float() * (-np.log(10000.0) / d_model))

        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)

        pe = pe.unsqueeze(0)
        self.register_buffer('pe', pe)

    def forward(self, x):
        x = x + self.pe[:, :x.size(1), :]
        return x
```

```
class TransformerModel(nn.Module):
    def __init__(self, d_model, nhead, num_encoder_layers, num_decoder_layers, dim_feedforward, vocab_size, max_len=512):
        super(TransformerModel, self).__init__()
        self.transformer = nn.Transformer(d_model=d_model, nhead=nhead,
                                           num_encoder_layers=num_encoder_layers,
                                           num_decoder_layers=num_decoder_layers,
                                           dim_feedforward=dim_feedforward,
                                           batch_first=True)

        self.embedding = nn.Embedding(num_embeddings=vocab_size, embedding_dim=d_model)
        self.pos_encoder = PositionalEncoding(d_model, max_len)
        self.fc_out = nn.Linear(d_model, vocab_size)

    def generate_square_subsequent_mask(self, sz):
        mask = (torch.triu(torch.ones(sz, sz)) == 1).transpose(0, 1)
        mask = mask.float().masked_fill(mask == 0, float('-inf')).masked_fill(mask == 1, float(0.0))
        return mask

    def forward(self, src, tgt, src_key_padding_mask=None, tgt_key_padding_mask=None):
        src = self.embedding(src)
        tgt = self.embedding(tgt)
        src = self.pos_encoder(src)
        tgt = self.pos_encoder(tgt)

        tgt_mask = self.generate_square_subsequent_mask(tgt.size(1)).to(tgt.device)
        memory_mask = torch.zeros((tgt.size(1), src.size(1))).to(tgt.device).bool()

        output = self.transformer(src, tgt,
                                   tgt_mask=tgt_mask,
                                   memory_mask=memory_mask,
                                   src_key_padding_mask=src_key_padding_mask,
                                   tgt_key_padding_mask=tgt_key_padding_mask)

        return self.fc_out(output)
```

# 모델링

```
vocab_size = tokenizer.vocab_size
d_model = 512
nhead = 8
num_encoder_layers = 6
num_decoder_layers = 6
dim_feedforward = 2048

# 입력 텐서 준비
src = questions_padded
tgt = answers_padded

class EarlyStopping:
    def __init__(self, patience=5, verbose=False):
        self.patience = patience
        self.verbose = verbose
        self.best_loss = float('inf')
        self.counter = 0
        self.stopped_early = False

    def __call__(self, train_loss, model):
        if train_loss < self.best_loss:
            self.best_loss = train_loss
            self.counter = 0
            # 끝나면 모델 저장
            torch.save(model.state_dict(), 'best_model.pth')
            if self.verbose:
                print(f'Training loss decreased to {train_loss:.6f}.')
        else:
            self.counter += 1
            if self.verbose:
                print(f'Training loss did not improve. Counter: {self.counter}/{self.patience}')
            if self.counter >= self.patience:
                self.stopped_early = True
                if self.verbose:
                    print('Early stopping triggered.')
```

loss 개선이 5번 되지 않을시 종료 후 모델 저장(Early stopping)

# 모델링

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print("Using device:", device)

# 데이터셋 생성
dataset = TensorDataset(src, tgt) # src는 입력, tgt는 타겟
dataloader = DataLoader(dataset, batch_size=256, shuffle=True, num_workers=2, pin_memory=True)

# 모델, 옵티마이저 생성
model = TransformerModel(d_model=d_model, nhead=nhead, num_encoder_layers=num_encoder_layers,
                          num_decoder_layers=num_decoder_layers, dim_feedforward=dim_feedforward,
                          vocab_size=vocab_size, max_len=512)

model.to(device)
optimizer = optim.Adam(model.parameters(), lr=0.001)
criterion = nn.CrossEntropyLoss(ignore_index=pad_token_id)

early_stopping = EarlyStopping(patience=5, verbose=True)
```

모델, 옵티마이저, 얼리스탑 객체 생성



# 모델링

```
from torch.cuda.amp import GradScaler, autocast

scaler = GradScaler()

# Training
num_epochs = 50

for epoch in range(num_epochs):
    model.train()
    epoch_loss = 0

    for batch in dataloader:
        src_batch, tgt_batch = batch

        src_batch = src_batch.to(device)
        tgt_batch = tgt_batch.to(device)

        tgt_input = tgt_batch[:, :-1]
        tgt_output = tgt_batch[:, 1:]

        src_key_padding_mask = (src_batch == pad_token_id)
        tgt_key_padding_mask = (tgt_input == pad_token_id)

        src_key_padding_mask = src_key_padding_mask.to(device)
        tgt_key_padding_mask = tgt_key_padding_mask.to(device)

        optimizer.zero_grad()

        with autocast():
            output = model(src_batch, tgt_input,
                           src_key_padding_mask=src_key_padding_mask,
                           tgt_key_padding_mask=tgt_key_padding_mask)
            loss = criterion(output.contiguous().view(-1, vocab_size), tgt_output.contiguous().view(-1))

        scaler.scale(loss).backward()

        # Gradient clipping
        torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)

        scaler.step(optimizer)
        scaler.update()

        epoch_loss += loss.item()

    avg_loss = epoch_loss / len(dataloader)

    print(f"Epoch {epoch+1}/{num_epochs}, Train Loss: {avg_loss:.6f}")

    early_stopping(avg_loss, model)
    if early_stopping.stopped_early:
        break

print("Training complete!")
```

# 모델링

```
dataloader = DataLoader(dataset, batch_size=128, shuffle=True, num_workers=2, pin_memory=True)
optimizer = optim.Adam(model.parameters(), lr=0.0005)
```

```
Epoch 39/50, Train Loss: 0.072367
Training loss decreased to 0.072367.
Epoch 40/50, Train Loss: 0.066747
Training loss decreased to 0.066747.
Epoch 41/50, Train Loss: 0.065436
Training loss decreased to 0.065436.
Epoch 42/50, Train Loss: 0.064395
Training loss decreased to 0.064395.
Epoch 43/50, Train Loss: 0.063817
Training loss decreased to 0.063817.
Epoch 44/50, Train Loss: 0.063563
Training loss decreased to 0.063563.
Epoch 45/50, Train Loss: 0.062880
Training loss decreased to 0.062880.
Epoch 46/50, Train Loss: 0.062685
Training loss decreased to 0.062685.
Epoch 47/50, Train Loss: 0.061808
Training loss decreased to 0.061808.
Epoch 48/50, Train Loss: 0.061605
Training loss decreased to 0.061605.
Epoch 49/50, Train Loss: 0.061216
Training loss decreased to 0.061216.
Epoch 50/50, Train Loss: 0.060257
Training loss decreased to 0.060257.
Training complete!
```

```
Epoch 15/20, Train Loss: 0.048769
Training loss decreased to 0.048769.
Epoch 16/20, Train Loss: 0.048742
Training loss decreased to 0.048742.
Epoch 17/20, Train Loss: 0.048389
Training loss decreased to 0.048389.
Epoch 18/20, Train Loss: 0.047964
Training loss decreased to 0.047964.
Epoch 19/20, Train Loss: 0.048106
Training loss did not improve. Counter: 1/5
Epoch 20/20, Train Loss: 0.047983
Training loss did not improve. Counter: 2/5
Training complete!
```

# 모델링

```
def generate_text_from_predictions(predicted_indices, tokenizer):
    # 입력을 토큰화
    tokens = tokenizer.convert_ids_to_tokens(predicted_indices)

    # [sep] 토큰이 나오면 예측을 멈춤
    filtered_tokens = []
    for token in tokens:
        if token in [tokenizer.sep_token]:
            break
        if token not in [tokenizer.pad_token, tokenizer.cls_token]:
            filtered_tokens.append(token)

    # 토큰을 문자로 변환
    sentence = tokenizer.convert_tokens_to_string(filtered_tokens)
    return sentence

def decode_output(output_tokens, tokenizer):
    return [generate_text_from_predictions(tokens, tokenizer) for tokens in output_tokens]
```

```
# 평가
model.eval()
with torch.no_grad():
    for i, batch in enumerate(data_loader):
        src_batch, tgt_batch = batch
        src_batch = src_batch.to(device)
        tgt_batch = tgt_batch.to(device)

        tgt_input = tgt_batch[:, :-1]
        tgt_output = tgt_batch[:, 1:]

        src_key_padding_mask = (src_batch == pad_token_id)
        tgt_key_padding_mask = (tgt_input == pad_token_id)

        output = model(src_batch, tgt_input, src_key_padding_mask=src_key_padding_mask, tgt_key_padding_mask=tgt_key_padding_mask)

        # 소프트맥스 층 통과
        output_probs = F.softmax(output, dim=-1)

        _, predicted = torch.max(output_probs, dim=-1)
        predicted = predicted.cpu().numpy()
        src_batch = src_batch.cpu().numpy()
        tgt_batch = tgt_batch.cpu().numpy()

        # Decoding examples
        for j in range(min(20, len(predicted))):
            input_text = tokenizer.decode(src_batch[j], skip_special_tokens=True)
            predicted_text = decode_output([predicted[j]], tokenizer)[0]
            target_text = tokenizer.decode(tgt_batch[j], skip_special_tokens=True)

            print(f"Input: {input_text}")
            print(f"Predicted: {predicted_text}")
            print(f"Target: {target_text}")

        break
```

데이터셋을 이용해 예측이 제대로 되는지 확인

# 모델링

```
Input: 런트베이크에 인접한 시설을 알 수 있나요?  
Predicted: 아돗치베이크에 인접한 시설에는 대명빌라, GS25가 있습니다  
Target: 런트베이크에 인접한 시설에는 대명빌라, GS25가 있습니다  
  
Input: 아건 구디점의 대표적인 메뉴가 어떻게 되나요?  
Predicted: 아건 구디점의 메뉴에는 치킨 티카 마살라, 치킨 카다이어 있습니다  
Target: 아건 구디점의 메뉴에는 치킨 티카 마살라, 치킨 카다이어 있습니다  
  
Input: 놀잇터의 주차시설이 있나요?  
Predicted: 놀부터의 주소시설이 없습니다  
Target: 놀잇터의 주차시설이 없습니다  
  
Input: 핏제리아오 동부이촌점의 주차시설이 있는지 알 수 있나요?  
Predicted: 핏제리아오 동부이촌점의 주차시설이 있습니다  
Target: 핏제리아오 동부이촌점의 주차시설이 있습니다  
  
Input: 진도배기 가락본점의 영업시간은 어떻게 되나요?  
Predicted: [UNK] 진도배기 가락본점의 영업시간은 08:00 - 22:00입니다  
Target: 진도배기 가락본점의 영업시간은 08:00 - 22:00입니다  
  
Input: 꼬꼬스토리 신도림점의 주요 메뉴를 알 수 있나요?  
Predicted: 꼬부스토리 신도림점의 메뉴에는 왕박스, 대박스가 있습니다  
Target: 꼬꼬스토리 신도림점의 메뉴에는 왕박스, 대박스가 있습니다  
  
Input: 송파구 맛집 추천해줘  
Predicted: 송파구 맛집은 가네칼국수를 추천드립니다  
Target: 송파구 맛집은 원가네칼국수를 추천드립니다  
  
Input: 그라츠커피랩 위례점의 영업시간을 알 수 있나요?  
Predicted: 그라츠커피랩 위례점의 영업시간은 10:00 - 21:00입니다  
Target: 그라츠커피랩 위례점의 영업시간은 10:00 - 21:00입니다
```

완벽하진 않지만 어느정도 예측이 잘 되는 것을 확인

# 모델링

```
def generate_response(input_text):
    inputs = tokenizer(input_text, return_tensors='pt', padding=True, truncation=True, max_length=max_length)
    src_batch = inputs['input_ids'].to(device)

    # 디코딩을 위한 초기 입력 설정 (CLS 토큰)
    tgt_input = torch.tensor([[cls_token_id]], device=device)

    for _ in range(100): # 최대 생성 길이 설정
        src_key_padding_mask = (src_batch == pad_token_id)
        tgt_key_padding_mask = (tgt_input == pad_token_id)

        output = model(src_batch, tgt_input, src_key_padding_mask=src_key_padding_mask, tgt_key_padding_mask=tgt_key_padding_mask)
        output_probs = F.softmax(output[:, -1, :], dim=-1)
        _, next_token = torch.max(output_probs, dim=-1)

        next_token = next_token.unsqueeze(0)
        tgt_input = torch.cat((tgt_input, next_token), dim=1)

        if next_token.item() == sep_token_id:
            break

    predicted = tgt_input[:, 1:].cpu().numpy() # CLS 토큰을 제외하고 디코딩
    predicted_text = decode_output(predicted, tokenizer)[0]

    return predicted_text

# 입력을 받아서 응답 생성
while True:
    input_text = input("질문을 입력하세요 (종료하려면 '초기화' 입력): ")
    if input_text == '초기화':
        break

    response = generate_response(input_text)
    print(f"챗봇의 응답: {response}")
```

질문을 입력받아 성능을 확인

# 모델링

질문을 입력하세요 (종료하려면 '초기화' 입력): 동대문구 맛집 알려줄 수 있어?

챗봇의 응답: 동대문구 맛집은 카페어반테라스를 추천합니다

질문을 입력하세요 (종료하려면 '초기화' 입력): 카페어반테라스에 인접한 시설은?

챗봇의 응답: 카페어반테라스에 인접한 시설에는 해법중국어, 모든기타교습소가 있습니다

질문을 입력하세요 (종료하려면 '초기화' 입력): 카페어반테라스 대표 메뉴 추천좀

챗봇의 응답: 카페어반테라스의 메뉴에는 카페모카, 허니브레드이 있습니다

질문을 입력하세요 (종료하려면 '초기화' 입력): 노원구 맛집 알려줘

챗봇의 응답: 노원구 맛집은 커피나무 노원역점가 있습니다

질문을 입력하세요 (종료하려면 '초기화' 입력): 은평구 맛집도 알려줄래?

챗봇의 응답: 은평구 맛집은 연서콩나물국밥에 가보세요

질문을 입력하세요 (종료하려면 '초기화' 입력): 연서콩나물국밥은 휴무일이 언제야?

챗봇의 응답: 연서콩나물국밥의 휴무일은 매주 일요일입니다

질문을 입력하세요 (종료하려면 '초기화' 입력): 연서콩나물국밥 주소 좀 알려줘

챗봇의 응답: 연서콩나물국밥의 주소는 서울 은평구 통일로 850입니다

질문을 입력하세요 (종료하려면 '초기화' 입력): 은평구 다른 맛집도 추천해줘

챗봇의 응답: 은평구 맛집은 도화원에 가보세요

질문을 입력하세요 (종료하려면 '초기화' 입력): 도화원은 메뉴가 뭐가 있을까?

챗봇의 응답: 도화원의 메뉴에는 짬뽕, 해물쟁반짜장이 있습니다

질문을 입력하세요 (종료하려면 '초기화' 입력): 초기화

들어주셔서 감사합니다.

---

**THANK  
YOU**

---

[GITHUB.COM//GRAPEFRUITSS/CHATBOT-PROJECT](https://github.com/GRAPFRUITSS/CHATBOT-PROJECT)

여기가 챗봇 맛집이조