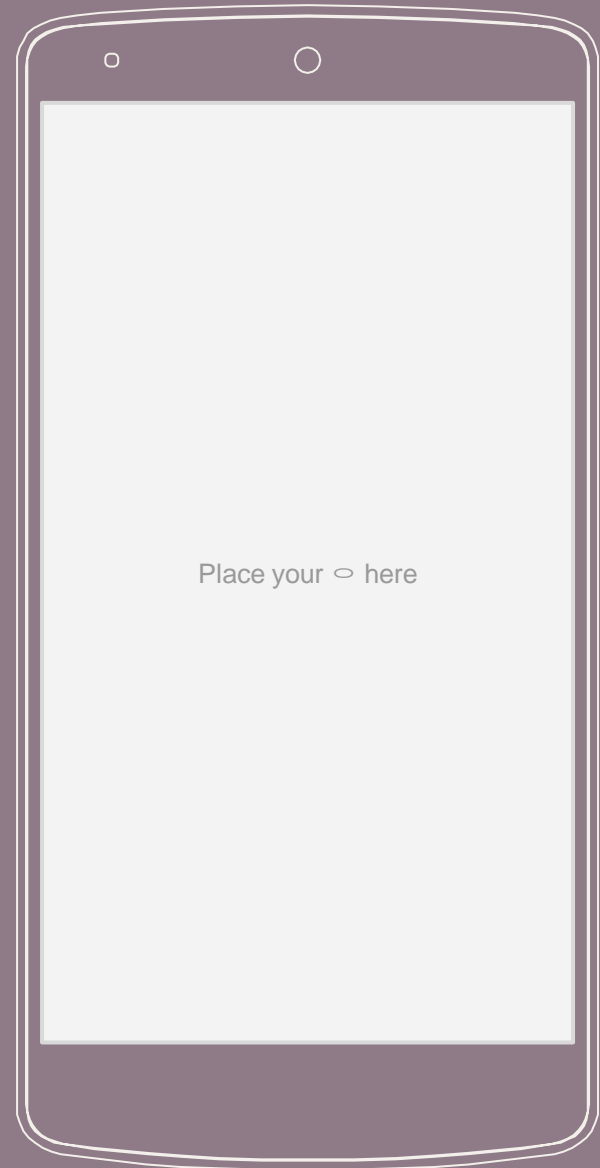


AI를 활용한 텍스트 분류기 개발



목차


- 학습 목표
- 개발 환경
- 설계도
- 코드
- 시각화



학습 목표

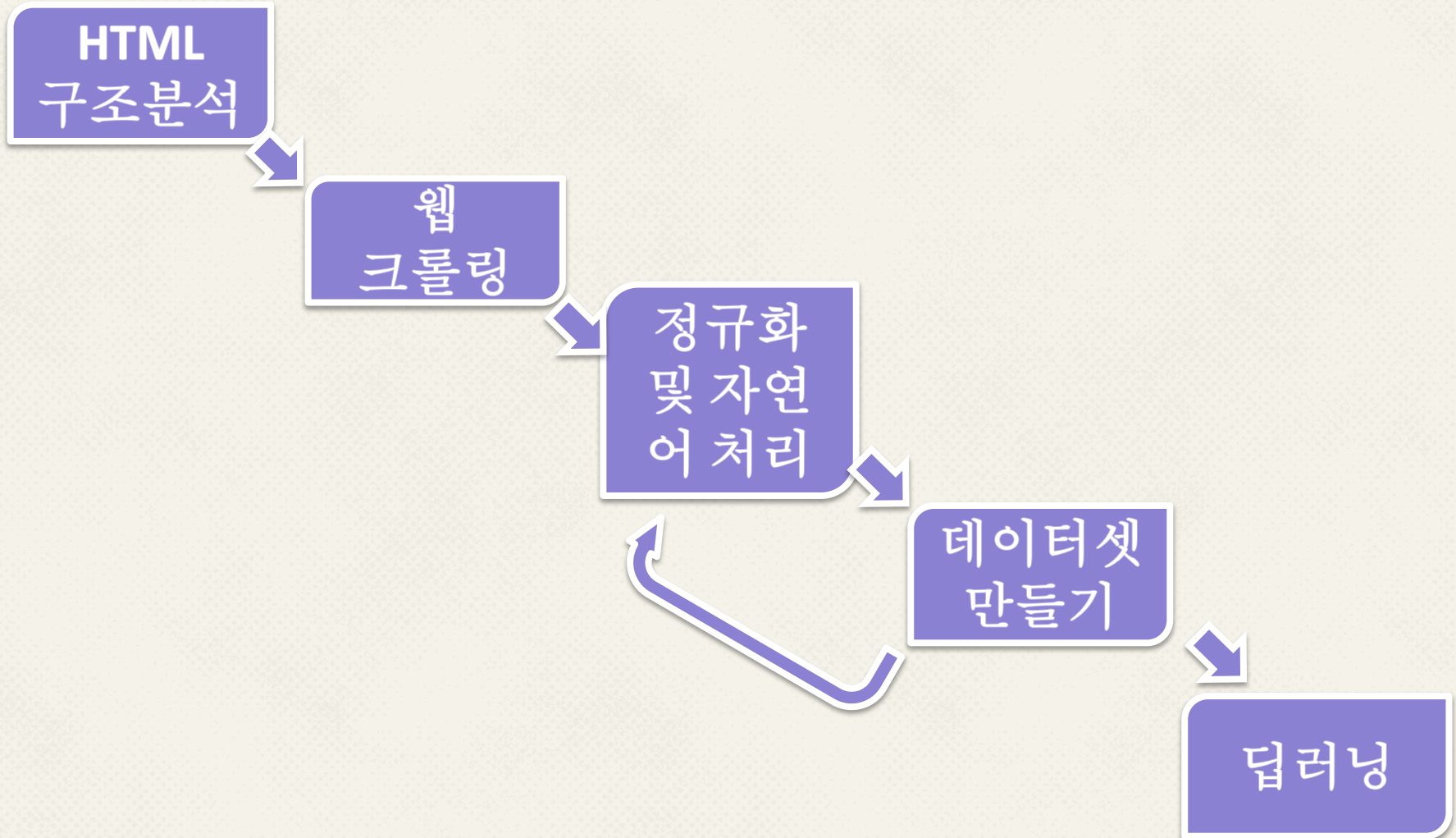
- 텍스트 웹 크롤링 실습
- 최신 딥러닝 기계학습 관련 이론 학습
- 다양한 AI 알고리즘 기반 텍스트 분석 프로그램 작성

개발 환경



Tool	Jupyter lab, Pycharm
Design	PowerPoint
Operation	Windows 10, Chrome
Language	Python 3.6
Library	Keras, Numpy, Pandas Konlpy, Nltk, bs4 ..etc

설계도

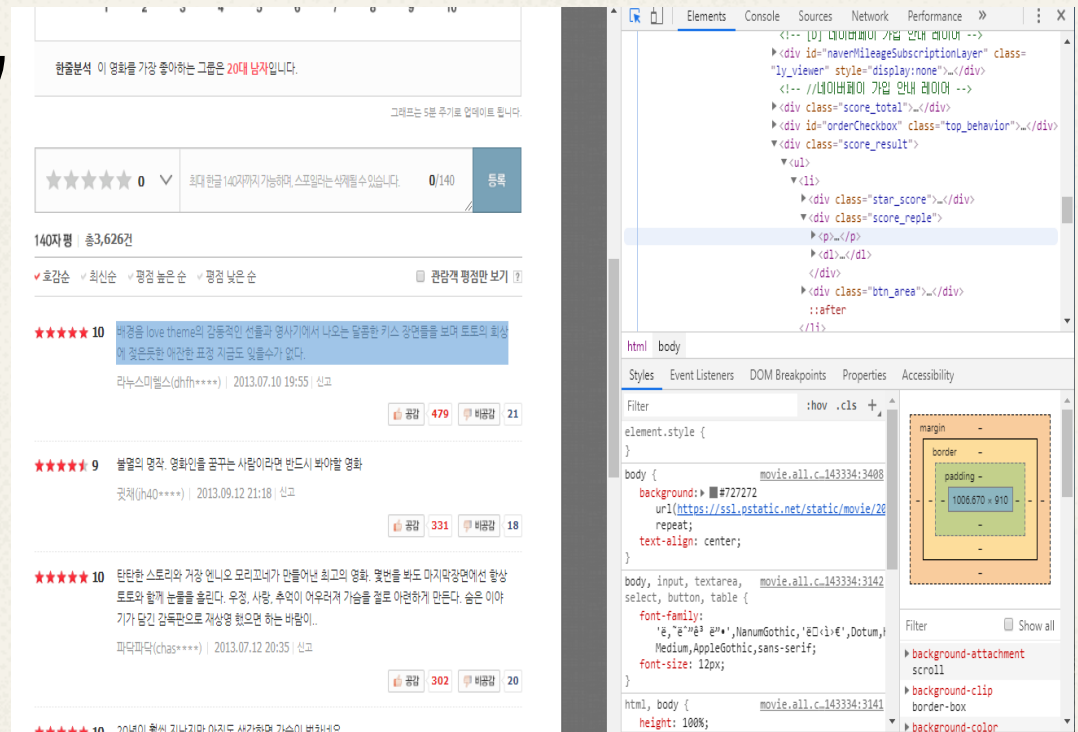


1. 사전 조사 하기



크롤링 하고 싶은 자료를 선정 한 후에 사이트를 선정한다. 우리는 네이버 영화 사이트에서 모든 댓글을 크롤링 하기로 결정하였다.

2. HTML 구조보기



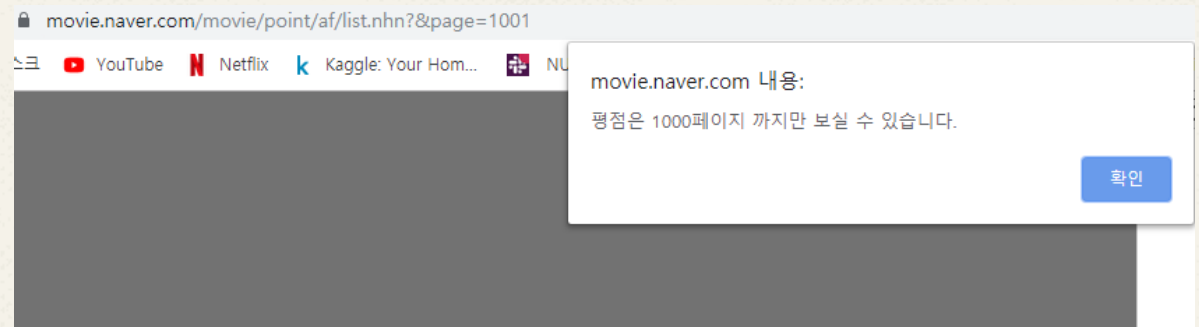
크롬의 개발자도구를 열면 친절하게도
HTML 트리구조가 나타나 있다.
`div class`의 `score_reple` 하위 `p`태그에
소속된 걸 볼 수 있다.

3. 크롤링 준비

```
import urllib
import urllib.request
import urllib.parse
from bs4 import BeautifulSoup
import re
import os
import time
import requests
import pandas as pd
from multiprocessing import Pool
```

크롤링을 하기 위해서는 *html* 구조를 파싱하고 구조를 불러올 *beautifulsoup*이라는 라이브러리를 사용했다.
(*from bs4 import BeautifulSoup*)

4. 첫 실패



네이버 전체영화 페이지에 들어가보면
네이버에서 가져갈 수 있는 댓글 수를 한정 지었다.
그리하여 계획을 바꾸어 영화별로 모든 댓글을
가져오는 방법으로 바꾸었다.

5. Urllib, Request 활용하기

```
params = {'code': i,  
          'type': 'after',  
          'isActualPointWriteExecute': 'false',  
          'isMileageSubscriptionAlready': 'false',  
          'isMileageSubscriptionReject': 'false',  
          'page': page}  
url = base_url + urllib.parse.urlencode(params)  
title_url = 'https://movie.naver.com/movie/point/af/list.nhn?st=mcode&sword={}&target=after&page={}'  
res = requests.get(url)
```

Urllib.parse.urlencode는 url주소를 파싱해서 페이지 별로 크롤링 하기 위하여 사용하였다. 댓글 페이지마다 번호만 바뀌기 때문에 for문을 사용하여 모든 댓글을 가져올 수 있도록 하였다. 그리고 requests를 이용하여 html 을 가져왔다.

6. BeautifulSoup

```
def hangle(s):
    return re.sub('[^ㄱ-ㅣ가-힣]+', '', str(s)).strip()

def num(s):
    return re.sub('[^0-9]+', '', str(s)).strip()

title_res = requests.get(title_url.format(i, page))
clean_text = BeautifulSoup(re.sub("&#(?![0-9])", "", res.text), 'html5lib')
title_text = BeautifulSoup(re.sub("&#(?![0-9])", "", title_res.text), 'html5lib')
lis = clean_text.find('div', {'class', 'score_result'}).find('ul')
score = clean_text.select('div.star_score > em')
review = clean_text.select('div.score_reple > p')
title = title_text.select('table.list_netizen > tbody > tr > td.title > a.movie')
```

원하는 부분의 HTML 구조를 따와야 했다.

Title_res는 제목 타이틀을 따로 다와야 했기에 요청도 따로 필요했다.

정규식을 사용하여 되도록 필요한 문구만 나오도록 활용 하였고 html5lib를 이용하여 트리구조를 볼 수 있도록 하였다. Score, review, title 을 selec함수와 find함수로 트리구조를 뽑아내었다.

7. 병렬 구조로 데이터 얻기

```
if __name__ == '__main__':  
    pool = Pool(processes=10)  
    result = pool.map(get, [10002])  
result
```

Multiprocessing 라이브러리에서 프로세스 개수를 늘려 병렬로 작업하여 작업 속도를 개선 시킬 수 있었다. 많은 양의 데이터는 작업 시간이 너무 오래 걸리기 위하여 조금이라도 시간을 단축하기 위하여 사용했다.

8. 크롤링 전체 코드

```
def get(n):
    base_url = 'https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?'
    data = []

    def hangle(s):
        return re.sub('[^ㄱ-ㅣ가-힣]+', '', str(s)).strip()

    def num(s):
        return re.sub('[^ 0-9]+', '', str(s)).strip()

    for i in range(10001, n):
        for page in range(1, 100):
            params = {'code': i,
                      'type': 'after',
                      'isActualPointWriteExecute': 'false',
                      'isMileageSubscriptionAlready': 'false',
                      'isMileageSubscriptionReject': 'false',
                      'page': page}
            url = base_url + urllib.parse.urlencode(params)
            title_url = 'https://movie.naver.com/movie/point/af/list.nhn?st=mcode&sword={}&target=after&page={}'
            res = requests.get(url)
            title_res = requests.get(title_url.format(i, page))
            clean_text = BeautifulSoup(re.sub("&#(?![0-9])", "", res.text), 'html5lib')
            title_text = BeautifulSoup(re.sub("&#(?![0-9])", "", title_res.text), 'html5lib')
            lis = clean_text.find('div', {'class', 'score_result'}).find('ul')
            score = clean_text.select('div.star_score > em')
            review = clean_text.select('div.score_reple > p')
            title = title_text.select('table.list_netizen > tbody > tr > td.title > a.movie')
            for a, b, c in zip(score, review, title):
                re_score = num(a)
                re_review = hangle(b)
                re_title = hangle(c)
                data.append([re_score, re_review, re_title])
    df = pd.DataFrame(data, columns=['score', 'review', 'title'])
    return df

if __name__ == '__main__':
    pool = Pool(processes=10)
    result = pool.map(get, [10002])
    result
```


9. 크롤링 결과

	score	review	title
0	10	배경음 의 감동적인 선율과 영사기에서 나오는 달콤한 키스 장면들을 보며 토토의 회...	시네마 천국
1	9	불멸의 명작 영화인을 꿈꾸는 사람이라면 반드시 봐야할 영화	시네마 천국
2	10	탄탄한 스토리와 거장 엔니오 모리코네가 만들어낸 최고의 영화 몇번을 봐도 마지막장면...	시네마 천국
3	10	년이 훨씬 지났지만 아직도 생각하면 가슴이 벅차네요	시네마 천국
4	10	오랜만에 다시 보아도 너무 예쁘고 사랑스런 눈물나게 하는 영화	시네마 천국
5	10	지금 극장에 걸려 있는 건 축약판분 축약판만 보신 분들은 꼭 감독판분도 구해서 보시...	시네마 천국
6	1	영화관에서 절대 보지말아라 이야기를 끌여가는 주요 이야기인 엘레나와 토토의 재회부분...	시네마 천국
7	10	대에 보았다 그때는 왜 눈물이 그치지 않는지 알수가 없었다 중반 이제서 다시 보고 ...	시네마 천국
8	10	아 근데 재개봉 버전 뭐이렇게 삭제된 장면이 많냐 후반 엘레나 시퀀스는 통째로 삭제됐네	시네마 천국
9	10	삶의 모든 것이 들어있었다 ㅠㅠ	시네마 천국
10	9	살바토레와 알프레도의 케미는 가히 역대급이다 명작의 가치는 역시 시대를 초월한다	시네마 천국
11	10	무슨 네오리얼리즘 영화도 아니고 구지 해석하고 분석하려 하지 말아라그냥 그대로 보고...	시네마 천국
12	6	신촌에서 예전시네마천국 기대했는데 너무영화가 똑똑 끊어지네 ㄴ누가 편집했는...	시네마 천국
13	10	만 들어도 눈물나는 영화	시네마 천국
14	10	아주 먼 훗날 이제 더는 흘릴 눈물이 남아 있을 것 같지 않을 그 날에도...	시네마 천국
15	10	관람객왜 죽기 전에 꼭 봐야 하는 영화인지 알겠습니다 오늘 시 분에 관람했는데 정말...	시네마 천국
16	10	살아가는 동안 누구나 하나쯤 시네마천국을 가질것이다 이영화는 나의 시네마천국을 생각...	시네마 천국
17	10	마지막 장면은 절대 잊을 수 없을 것이다	시네마 천국
18	10	제 인생 최고의 영화였습니다아직도 이보다 더한 감동과그리움을 느끼는 영화는 만나지 ...	시네마 천국
19	10	방금 감독판 봤는데 이런 걸작을 지금까지 못본게 후회가 될만큼 큰 감명을 받았네요 ...	시네마 천국
20	10	너무슬프다 말그대로 명작이다	시네마 천국
21	8	관람객시대는 변하고 기억은 변하지 않기때문에아름답게 추억할수있는 과거	시네마 천국
22	8	영화는 무엇일까 인생의 또다른 말이 아닐까	시네마 천국
23	8	글 남기려다 다른 글을 봤는데 감독판이 있나봐요 감독판은 얼마나 더 대단할까 감독판...	시네마 천국
24	10	년 고대 영화관에서 돈주고 번 분영화친척언니랑 여름방학때보고 넘감동적인 영화여서 친...	시네마 천국
25	10	단언컨대 시네마 천국은 최고의 영화입니다	시네마 천국
26	10	마지막 장면은 울지 않고는 볼 수 없었다는	시네마 천국
27	9	관람객최고의 영화를 지금 다시 볼 수 있어서 좋았습니다	시네마 천국
28	10	요즘 영화에 비하면 전개가 느리고 긴 시간이라 지루할 수 있지만 결말 여운이 평생감 진짜	시네마 천국
29	10	어떠한 긴 말보다 엄청난 감동이 밀려오는 영화입니다	시네마 천국
...
960	8	지금세대가 보기엔 명작은 아니다	시네마 천국
961	10	그냥 눈물이 왜나는지 모르는데 눈물이 흐르는 정말 최고의 영화	시네마 천국
962	10	소름이돋으면서 눈물이난다 뭐라고해야할지 정말최고다	시네마 천국
963	10	번도 넘게 본 영화	시네마 천국
964	10	역시 명작은 시간이 지나도 명작이네요ㅋㅋㅋ 마지막 장면은 정말 최고	시네마 천국

10. 자연어 처리 및 데이터 셋 준비하기

네이버 영화댓글 1200만개의 데이터를 받은 후에 일단 샘플로 10000개의 데이터를 자연어 처리하는 과정이 필요했다.
모든 데이터를 하기에는 시간적 제약때문에 일단 만개를 가지고하기로 했다.

11. 자연어 처리 및 데이터 셋 코드

```
import pandas as pd
import re
import numpy as np
df = pd.read_csv('./sample.csv')

content = df['contents']
train = content.loc[0:4998]
test = content.loc[5001:9999]

train.to_csv('train_data.csv', mode='w', index=False, header=True)
test.to_csv('test_data.csv', mode='w', index=False, header=True)

len(test)

4999

train_data = hangle(train)
test_data = hangle(test)

def read_data(filename):
    with open(filename, 'r') as f:
        data = [line.split('\t') for line in f.read().splitlines()]
        data = data[1:]
    return data

train_set = read_data('./train_data.csv')
test_set = read_data('./test_data.csv')

from konlpy.tag import Okt
okt = Okt()
import json
import os
from pprint import pprint

def tokenize(doc):
    return [' '.join(t) for t in okt.pos(doc, norm=True, stem=True)]

train_pos = []
test_pos = []
for row in train_set:
    try:
        train_pos0 = [tokenizer(row[1]), row[2]]
        train_pos.append(train_pos0)
    except:
        pass
for row in test_set:
    try:
        test_pos0 = [tokenizer(row[1]), row[2]]
        test_pos.append(test_pos0)
    except:
        pass

tokens = [t for d in train_pos for t in d[0]]
train_pos

[]

train_docs = [(tokenizer(row[0])) for row in train_set]
test_docs = [(tokenizer(row[0])) for row in test_set]

pprint(train_docs[0])

...

tokens = [t for d in train_docs for t in d]
tokens

...

import nltk
text = nltk.Text(tokens, name='NMSC')
pprint(text.vocab().most_common(20))
```

Konlpy 를 이용하여 조사별로 파싱을 하기로 했다. 일단 컬럼에서 필요한 것은 영화 댓글과 스코어를 분리하여 넘파이 배열로 만들기 위하여 댓글 한 개마다 조사별로 나누어 벡터화 시키는 과정이 필요 했다.

12. 데이터 셋 레이블링 및 벡터화

```
selected_words = [f[0] for f in text.vocab().most_common(3000)]
selected_words
```

...

```
def term_frequency(doc):
    return [doc.count(word) for word in selected_words]
```

```
train_x = [term_frequency(d) for d in train_docs]
test_x = [term_frequency(d) for d in test_docs]
train_y = [c for c in train_docs]
test_y = [c for c in test_docs]
```

```
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
from tensorflow.keras import losses
from tensorflow.keras import metrics
```

```
x_train = np.asarray(train_x).astype('float32')
x_test = np.asarray(test_x).astype('float32')
```

딥러닝에 들어가기전에 벡터화 하고 레이블링 데이터,
*word2idx*를 만들어야 한다.

13. 입력 데이터 셋 생성

```
1 import pandas as pd
2 import operator
3 import re
4
5 df = pd.read_csv('test_set.csv')
6 patt = re.compile('[가-힣]+')
7
8 data = list() # list 초기 선언
9 data_dict = dict() # dict 초기 선언
10
11 # 문행초의 모든 단어를 list로 저장
12 for i in df.contents:
13     i = str(i)
14     if str(i) == '':
15         i = str(i)
16     else:
17         i = re.findall(patt, i)
18     data += i
19 print("data len : ", len(data)) # List 길이 확인
20 print(data)
21
22 # List의 값을 가져와 빈도수 체크 후 Dict 형태로 저장
23 for i in data:
24     if data_dict.__contains__(i):
25         data_dict[i] += 1
26     else:
27         data_dict[i] = 1
28 print("dict len : ", len(data_dict)) # dict 길이 확인
29
30 for i, j in data_dict.items():
31     print(i, ":", j)
32
33 # 생성된 dict 빈도수 기준으로 내림차순 정렬
34 dict_sorted = sorted(data_dict.items(), key=operator.itemgetter(1), reverse=True)
35 print(len(dict_sorted))
36 print(type(dict_sorted)) # 정렬된 후 List 형태로 저장되어 있음
37
38 for i in dict_sorted:
39     print(i)
40
41 test = pd.DataFrame(dict_sorted)
42 test.to_csv('word2idx.csv')
```

샘플데이터를 활용해

Word2idx 및 idx2Word Dictionary 생성

13. 입력 데이터 셋 생성

1	0,1
2	영화,1
3	너무,2
4	정말,3
5	진짜,4
6	이,5
7	더,6
8	잘,7
9	수,8
10	그냥,9
11	보고,10
12	좀,11
13	영화를,12
14	꼭,13
15	다,14
16	그,15
17	최고의,16
18	본,17
19	있는,18
20	봤는데,19
21	영화가,20
22	영화는,21
23	많이,22
24	왜,23
25	점,24
26	재밌게,25
27	좋은,26
28	역시,27
29	이런,28
30	연기,29

31	것,30
32	보는,31
33	그리고,32
34	완전,33
35	최고,34
36	하는,35
37	다시,36
38	한,37
39	이렇게,38
40	또,39
41	없는,40
42	스토리,41
43	영화입니다,42
44	평점,43
45	내가,44
46	봤습니다,45
47	불,46
48	재미있게,47
49	내내,48
50	참,49
51	하지만,50
52	마지막,51
53	내,52
54	보면,53
55	말이,54
56	난,55
57	많은,56
58	그래도,57
59	조금,58
60	없다,59

61	감동,60
62	연기가,61
63	봤어요,62
64	오랜만에,63
65	보세요,64
66	이건,65
67	아,66
68	솔직히,67
69	넌,68
70	스토리가,69
71	가장,70
72	배우들의,71
73	연기도,72
74	감사합니다,73
75	영화다,74
76	대한,75
77	봐도,76
78	재밌어요,77
79	같다,78
80	배우들,79
81	있고,80
82	끝까지,81
83	할,82
84	함께,83
85	평점이,84
86	같은,85
87	뭔가,86
88	근데,87
89	지금,88
90	계속,89

생성된 데이터 셋 형식

14. 신경망 모델 구현 및 생성

```
1 import tensorflow
2 from tensorflow.keras.models import Sequential
3 from tensorflow.keras.layers import Dense, Embedding, GlobalAveragePooling1D, Dropout
4 from tensorflow.keras.preprocessing.sequence import pad_sequences
5 # keras.layers.Dropout(rate, noise_shape=None, seed=None)
6
7 import numpy as np
8 import pandas as pd
9 import operator
10 import re
11 import matplotlib.pyplot as plt
12
13 patt = re.compile('[A-Za-z]+') # 한글 추출 정규표현식
14
15 # word2idx 정의 하기
16 vocab_size = 10000
17 df = pd.read_csv('word2idx.csv')
18 df.columns = ['word', 'idx']
19 word2idx = dict()
20 word2idx['<PAD>'] = 0 # InputData의 길이를 맞추기 위한 패딩
21 word2idx['<START>'] = 1 # text의 시작을 의미
22 word2idx['<UNSEEN>'] = 3 #
23
24 for i, j in zip(df.word, df.idx):
25     if int(j) > (vocab_size-4):
26         word2idx[i] = 2 # vocab_size를 초과하는 단어는 모두 2로 처리 <UNK>
27     else:
28         word2idx[i] = int(j)+3
29
30 #idx2word
31 idx2word = dict()
32 for k, v in word2idx.items():
33     idx2word[v] = '<UNK>'
34     if v != 2:
35         idx2word[v] = k
36
37 data = pd.read_csv('test_set.csv')
38 x_list = []
39 y_list = []
40 # x_train, y_train 정의하기
41 for i, j in zip(data.contents, data.score):
42     li = []
43     i = str(i)
44     i = re.findall(patt, i)
45     li.append(i)
46     for x in i:
47         li.append(word2idx[x])
48     x_list.append(li)
49     y_list.append(j)
50
51 # 공백 부정 제거하기
52 # if int(j) > 7:
53 #     y_list.append(1)
54 # else:
55 #     y_list.append(0)
56
57 x_train = np.array(x_list)
58 y_train = np.array(y_list)
59
60 x_train = x_train.reshape(len(x_train)) # x_train 0이던 reshape
61
62 x_train = pad_sequences(x_train, value=word2idx['<PAD>'], padding='post', maxlen=64) # 입력 0이던 패딩
63
64 #create model
65 model = Sequential()
66 model.add(Embedding(vocab_size, 128))
67 model.add(Dropout(0.5, noise_shape=None, seed=None))
68 model.add(Dense(128, activation='relu'))
69 model.add(Dense(128, activation='relu'))
70 model.add(Dense(128, activation='relu'))
71 model.add(GlobalAveragePooling1D())
72 model.add(Dense(128, activation='relu'))
73 model.add(Dense(128, activation='relu'))
74 model.add(Dense(128, activation='relu'))
75 model.add(Dense(11, activation='sigmoid'))
76
77 Adam = tensorflow.keras.optimizers.Adam # cost function
78
79 model.compile(optimizer=Adam(lr=0.0005), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
80 history = model.fit(x_train, y_train, epochs=10, validation_split=0.33)
81
82 model.save('test_model.h5')
83 model.evaluate(x_train, y_train)
```

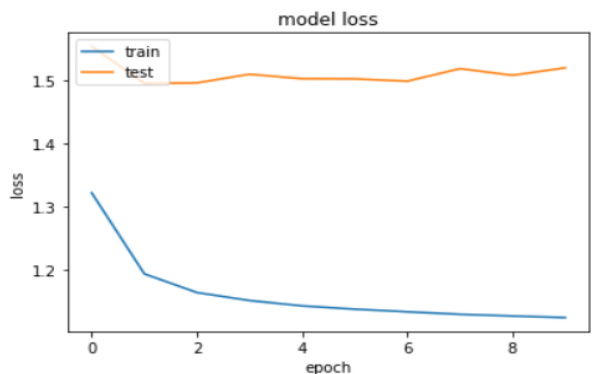
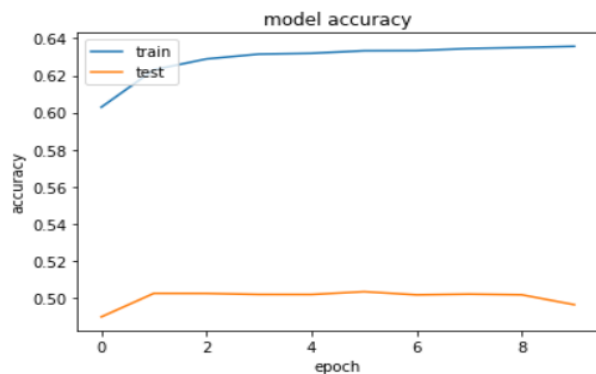
Keras를 활용하여 신경망 모델 생성 및 학습 진행

14. 생성한 모델 학습 결과

Train on 167499 samples, validate on 82501 samples

```
W0922 16:29:01.411008 4612 deprecation.py:323] From C:\Users\S340\Anaconda3\lib\site-packages\tensorflow\python\ops\math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

```
Epoch 1/10
167499/167499 [=====] - 272s 2ms/sample - loss: 1.3221 - acc: 0.6029 - val_loss: 1.5538 - val_acc: 0.4901
Epoch 2/10
167499/167499 [=====] - 226s 1ms/sample - loss: 1.1943 - acc: 0.6231 - val_loss: 1.4952 - val_acc: 0.5027
Epoch 3/10
167499/167499 [=====] - 220s 1ms/sample - loss: 1.1647 - acc: 0.6289 - val_loss: 1.4960 - val_acc: 0.5027
Epoch 4/10
167499/167499 [=====] - 198s 1ms/sample - loss: 1.1521 - acc: 0.6315 - val_loss: 1.5096 - val_acc: 0.5021
Epoch 5/10
167499/167499 [=====] - 212s 1ms/sample - loss: 1.1437 - acc: 0.6319 - val_loss: 1.5025 - val_acc: 0.5021
Epoch 6/10
167499/167499 [=====] - 217s 1ms/sample - loss: 1.1384 - acc: 0.6333 - val_loss: 1.5023 - val_acc: 0.5036
Epoch 7/10
167499/167499 [=====] - 216s 1ms/sample - loss: 1.1343 - acc: 0.6333 - val_loss: 1.4986 - val_acc: 0.5019
Epoch 8/10
167499/167499 [=====] - 224s 1ms/sample - loss: 1.1304 - acc: 0.6345 - val_loss: 1.5184 - val_acc: 0.5023
Epoch 9/10
167499/167499 [=====] - 339s 2ms/sample - loss: 1.1277 - acc: 0.6350 - val_loss: 1.5080 - val_acc: 0.5019
Epoch 10/10
167499/167499 [=====] - 310s 2ms/sample - loss: 1.1251 - acc: 0.6357 - val_loss: 1.5197 - val_acc: 0.4966
```



신경망 학습 결과를 시각화한 자료

15. 구현한 신경망 모델 테스트

```
1 from tensorflow.keras.models import load_model
2 import pandas as pd
3 import numpy as np
4 import re
5
6 patt = re.compile('[가-힣]+')
7 model = load_model('test_model.h5')
8
9 # word2idx 정의 하기
10 vocab_size = 10000
11 df = pd.read_csv('word2idx.csv')
12 df.columns = ['word', 'idx']
13 word2idx = dict()
14 word2idx['<PAD>'] = 0 # InputData의 길이를 맞추기 위한 패딩
15 word2idx['<START>'] = 1 # text의 시작을 의미
16 word2idx['<UNUSED>'] = 3 #
17 for i, j in zip(df.word, df.idx):
18     if int(j) > (vocab_size-4):
19         word2idx[i] = 2 # vocab_size를 초과하는 단어는 모두 2로 처리 <UNK>
20     else:
21         word2idx[i] = int(j)+3
22
23 #idx2word
24 idx2word = dict()
25 for k, v in word2idx.items():
26     idx2word[2] = '<UNK>'
27     if v != 2:
28         idx2word[v] = k
29
```

```
29
30 input_contents = input('영화 리뷰를 입력하세요 : ')
31 word_li = re.findall(patt, input_contents)
32 code_li = list()
33 for i in word_li:
34     code_li.append(1)
35     if word2idx.__contains__(i):
36         code_li.append(word2idx[i])
37     else:
38         code_li.append(2)
39
40 while len(code_li) < 64:
41     code_li.append(0)
42
43 code_li = np.array(code_li)
44
45 out = model.predict(code_li.reshape(1,64))
46 print("Predict score : ", out.argmax())
47
48
49
50
51
52
53
54
55
56
57
58
59
```

생성된 모델에 임의의 텍스트를 입력하여
예측 결과 확인하기

15. 구현한 신경망 모델 테스트

[테스트 CASE1]

```
input_contents = input('영화 리뷰를 입력하세요 : ')
```

영화 리뷰를 입력하세요 : 조정석 백수 역할 진짜 잘 연기했네요 ㅋㅋㅋ 윤아도 잘어울리고 ㅋㅋㅋㅋ전 시사회를 통해서 봤는데 웃기기도 하지만 은근 째내나고 가족들 친구들이랑 보기 부담없어서 추천해요

```
out = model.predict(code_li.reshape(1,64))  
print("Predict score : ", out.argmax())
```

Predict score : 10

★★★★★ 10 조정석 백수 역할 진짜 잘 연기했네요 ㅋㅋㅋ 윤아도 잘어울리고 ㅋㅋㅋㅋ전 시사회를 통해서 봤는데 웃기기도 하지만 은근 째내나고 가족들 친구들이랑 보기 부담없어서 추천해요

NN(gndu****) | 2019.07.31 09:05 | 신고

공감 4717 비공감 468

생성된 모델에 임의의 텍스트를 입력하여
예측 결과 확인하기

15. 구현한 신경망 모델 테스트

[테스트 CASE2]

```
input_contents = input('영화 리뷰를 입력하세요 : ')
```

영화 리뷰를 입력하세요 : 진짜 네이버 평점 못믿겟음 이제 영화 전체적으로 너무 루즈하고 중간중간 웃음 포인트도 없음 뻔한 전개는 예상했지만 코미디라고 해서 재미를 기대했는데 재미는 무슨 대구지하철참사 재연한 부분만 슬펐음 .. 결국 전체적인 영화는 별로

```
out = model.predict(code_li.reshape(1,64))
print("Predict score : ", out.argmax())
```

Predict score : 5

★★★★★ 1 진짜 네이버 평점 못믿겟음 이제 영화 전체적으로 너무 루즈하고 중간중간 웃음 포인트도 없음 뻔한 전개는 예상했지만 코미디라고 해서 재미를 기대했는데 재미는 무슨 대구지하철참사 재연한 부분만 슬펐음 .. 결국 전체적인 영화는 별로

zinnn(sck0****) | 2019.09.16 00:43 | 신고

공감 34 비공감 9

생성된 모델에 임의의 텍스트를 입력하여
예측 결과 확인하기

15. 구현한 신경망 모델 테스트

[테스트 CASE3]

```
input_contents = input('영화 리뷰를 입력하세요 : ')
```

영화 리뷰를 입력하세요 : `ㅠㅠ 댓글 다 알바들인가 우리 가족들 왜만한거는 다 재밌다고 하는데 이건 진짜 아니라고 그러시더라 살면서 돈이 아까웠던 영화 중 한개... 영화를 욕하고 싶은게 아니라 너무 재미없었음 내용도 너무 질질 끌고`

```
out = model.predict(code_li.reshape(1,64))  
print("Predict score : ", out.argmax())
```

Predict score : 1

★★★★★ 1 `ㅠㅠ 댓글 다 알바들인가 우리 가족들 왜만한거는 다 재밌다고 하는데 이건 진짜 아니라고 그러시더라 살면서 돈이 아까웠던 영화 중 한개... 영화를 욕하고 싶은게 아니라 너무 재미없었음 내용도 너무 질질 끌고`

Korea(wndk****) | 2019.09.13 10:01 | 신고

공감 64 비공감 39

생성된 모델에 임의의 텍스트를 입력하여
예측 결과 확인하기

• Project시 어려웠던 점 •

1. 데이터 정제

네이버영화의 천 만개가 넘는 리뷰와 평점을 데이터로 가져올 때 어려움을 느꼈습니다.
팀원들끼리 많은 코드를 짜보고 회의하고, 인터넷에서 도움도 얻어가며 끝내 네이버의 모든 영화 평점 리뷰를 가져오는 코드를 작성했습니다.

```
In [1]: import urllib
import urllib.request
import urllib.parse
import bs4
import re
import os
import time
import pandas as pd
from concurrent.futures import ThreadPoolExecutor
```

```
In [ ]: def deleteTag(x):
    return re.sub("<[>]*>", "", x)

def getComments(code):
    def makeArgs(code, page):
        params = {
            'code': code,
            'type': 'after',
            'isActualPointWriteExecute': 'false',
            'isMileageSubscriptionAlready': 'false',
            'isMileageSubscriptionReject': 'false',
            'page': page
        }
    return urllib.parse.urlencode(params)

def innerHTML(s, sl=0):
    ret = ''
    for i in s.contents[sl:]:
        if i.is_string():
            ret += i.strip()
        else:
            ret += str(i)
    return ret

def fText(s):
    if len(s): return innerHTML(s[0]).strip()
    return ''

retList = []
colSet = set()
print("Processing: %d" % code)
page = 1
```

```
# code데이터로 영화 제목 추출
title = "https://movie.naver.com/movie/bi/mi/basic.nhn?code=" + str(code)
html = urllib.request.urlopen(title).read()
soup_title = bs4.BeautifulSoup(html, 'html.parser')
name = soup_title.find('h3', {'class': 'h_movie'}).find('a').text

while 1:
    try:
        f = urllib.request.urlopen(
            "http://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=" + str(code) + "&page=" + str(page))
        data = f.read().decode('utf-8')
    except:
        break
    soup = bs4.BeautifulSoup(re.sub("<?([0-9])>", "", data), "html.parser")
    cs = soup.select(".score_result li")

    if not len(cs): break
    for link in cs:
        try:
            url = link.select(".score_reple em a")[0].get('onclick')
        except:
            print(page)
            print(data)
            raise ""
        m = re.search("[0-9]+", url)
        if m:
            url = m.group(0)
        else:
            url = ''
        if url in colSet: return retList
        colSet.add(url)
        cat = fText(link.select(".star_score em"))
        cont = fText(link.select(".score_reple p"))
        cont = re.sub("<span [0-9]+>.*</span>", "", cont)
        retList.append((name, cont, cat))
        page += 1

return retList
```

```
def fetch(i):
    outname = 'comments/{}.csv'.format(i)
    data = []
    try:
        if os.stat(outname).st_size > 0: return
    except:
        None
    rs = getComments(i)
    # 해당 영화코드에 데이터가 없으면 스킵
    if not len(rs): return

    for idx, r in enumerate(rs):
        data.append([i, r[0], r[1], r[2].replace("\n", "").replace("##", "###")])

    df = pd.DataFrame(data)
    df.to_csv('comments/{}.csv'.format(i), encoding='utf-8')

    time.sleep(1)
```

```
with ThreadPoolExecutor(max_workers=5) as executor:
    # 10000 ~ 200000 영화 고유코드로 전체 스캔
    for i in range(10000, 200000):
        executor.submit(fetch, i)
```

• Project시 어려웠던 점 •

2. 신경망 입력 데이터 만들기

전체 댓글의 단어별 빈도수를 기준으로 정렬한 데이터 만들기

```
In [1]: import pandas as pd
import operator
import re

In [2]: df = pd.read_csv('test_set.csv')
patt = re.compile('[가-힣]+')

In [3]: data = list() # list 초기 선언
data_dict = dict() # dict 초기 선언

In [4]: # 콘텐츠의 모든 단어를 List로 저장
for i in df.contents:
    i = str(i)
    if str(i) == '':
        i = str(i);
    else:
        i = re.findall(patt, i)
        data += i
    print("data len : ", len(data)) # List 길이 확인
data len : 85613

In [5]: # List의 값들을 가져와 빈도수 체크후 Dict 형태로 저장
for i in data:
    if data_dict.__contains__(i):
        data_dict[i] += 1;
    else:
        data_dict[i] = 1;
    print("dict len : ", len(data_dict)) # dict 길이 확인
dict len : 36685

In [6]: # 생성된 dict 빈도수 기준으로 내림차순 정렬
dict_sorted = sorted(data_dict.items(), key=operator.itemgetter(1), reverse=True)
print(len(dict_sorted))
print(type(dict_sorted)) # 정렬된 후 List형태로 저장되어있음
36685
<class 'list'>

In [7]: test = pd.DataFrame(dict_sorted)
test.to_csv('test02.csv')
```

word2idx 만들기 step2

- 빈도수별 idx값 매긴후 파일로 저장

```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('test02.csv')

In [3]: df.columns = ['idx', 'word', 'count']

In [4]: df.idx = {i+1 for i in df.idx}

In [5]: data = []
for i,j in zip(df.word, df.idx):
    data.append([i,j])

In [6]: df = pd.DataFrame(data)

In [7]: df.to_csv('word2idx.csv', index=False)
```