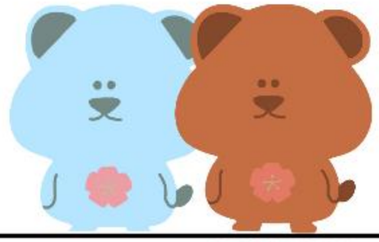
토픽 모델링

2팀 이채원

EWHA EURON

목차



EWHA EURON

#01 토픽 모델링

토픽 모델링이란?

- 문서 집합에 숨어 있는 주제를 찾아내는 것
- 머신러닝 기반의 토픽 모델링으로 숨어있는 중요 주제를 효과적으로 찾아낼 수 있음
- 숨겨진 주제를 효과적으로 표현할 수 있는 중심 단어를 함축적으로 추출
- LSA, LDA 기법을 사용

#02 LSA (Latent Semantic Analysis)

LSA란?

- Latent Semantic Analysis (잠재 의미 분석)의 약자
- 기존 DTM, TF-IDF의 단어의 의미를 고려하지 못한다는 단점을 개선
- **장점** : 쉽고 빠르게 구현 가능, 단어의 잠재적 의미 이끌어낼 수 있음
- **단점** : 이미 계산된 LSA에 새로운 데이터를 추가하기 위해서는 다시 계산해야 함.

LSA 적용 과정

- 텍스트 전처리
- TF-IDF 행렬 생성
- **절단된 SVD** 생성
- 토픽 모델링

#02 LSA (Latent Semantic Analysis)

SVD(특이값 분해)란?

- 행렬을 다음과 같은 3개의 행렬의 곱으로 분해하는 것

$$A = U\Sigma V^{\mathrm{T}}$$

U : $m \times m$ 직교행렬 $(AA^{\mathrm{T}} = U(\Sigma\Sigma^{\mathrm{T}})U^{\mathrm{T}})$

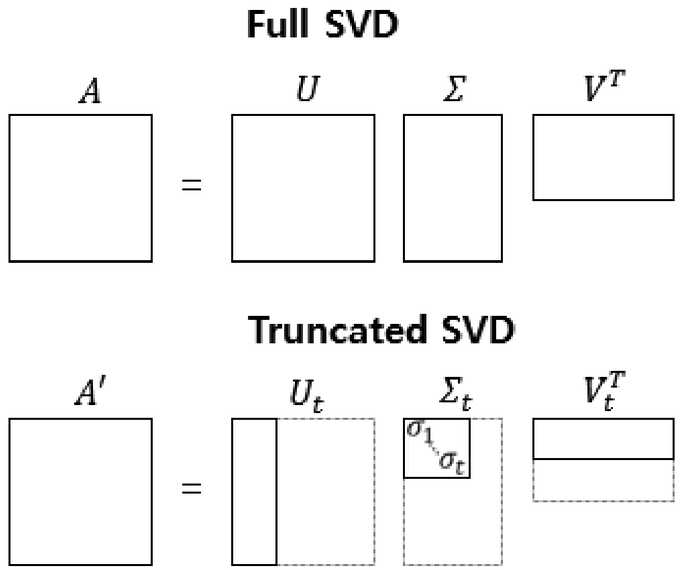
V : $n \times n$ 직교행렬 $(A^{\mathrm{T}}A = V(\Sigma^{\mathrm{T}}\Sigma)V^{\mathrm{T}})$

Σ : $m \times n$ 직사각 대각행렬

#02 LSA (Latent Semantic Analysis)

절단된 SVD란?

- 앞선 예시 : 풀 SVD (full SVD)
  풀 SVD에서 나온 3개의 행렬에서 일부 벡터를 삭제.



- 노이즈가 감소하고, 계산 비용이 낮아지는 효과를 얻을 수 있음.

#03 LDA (Latent Dirichlet Allocation)

LDA란?

- 문서의 집합으로부터 어떤 토픽이 존재하는지를 알아내기 위한 알고리즘
- 단어가 특정 토픽에 존재할 확률과 문서에 특정 토픽이 존재할 확률을 결합 확률로 추정하여 토픽을 추출.

LDA의 가정

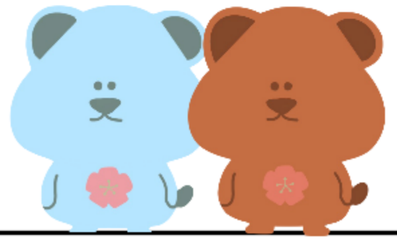
1. 문서에 사용할 단어의 개수 N을 정함.
2. 문서에 사용할 토픽의 혼합을 확률 분포에 기반하여 결정. (예 : 강아지 60%, 과일 40%)
3. 문서에 사용할 단어를 다음과 같이 정함.
    1. 토픽 분포에서 토픽 T를 확률적으로 고름
    2. 선택한 토픽 T에서 단어의 출현 확률 분포에 기반해 문서에 사용할 단어를 고름.

#03 LDA (Latent Dirichlet Allocation)

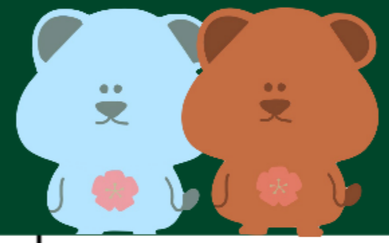
LDA의 수행 과정

1. 알고리즘에게 토픽의 개수 k를 알려줌.
2. 모든 단어를 k개 중 하나의 토픽에 할당
3. 모든 문서의 모든 단어에 대해 다음 과정을 반복
    1. 자신은 잘못된 토픽에 할당되어 있지만, 다른 단어들은 올바르게 할당되어 있다고 가정.
    2. 아래 두 가지 기준에 따라 토픽이 재할당
        1. p(topic t | document d) : 문서 d의 단어들 중 토픽 t에 해당하는 단어들의 비율
        2. p(word w | topic t) : 각 토픽들 에서 해당 단어 w의 분포

THANK YOU



EWHA EURON



문서 군집화 소개와 실습

문원정

EWHA EURON

#07 문서 군집화 소개와 실습

문서 군집화(Doucment Clustering) : 비슷한 텍스트 구성의 문서를 군집화 하는 것.

동일 군집에 속하는 문서를 같은 카테고리 소속으로 분류할 수 있으므로 텍스트 분류 기반의 문서 분류와 유사함.

*텍스트 분류 기반 문서 분류와의 차이 : 텍스트 기반 문서 분류는 사전에 결정 카테고리 값을 가진 학습 데이터 세트가 필요함. 문서 군집화는 학습 데이터 세트가 필요없는 비지도학습 기반으로 동작함.

#07 문서 군집화 소개와 실습

```
import pandas as pd
import glob, os
import warnings

warnings.filterwarnings('ignore')
pd.set_option('display.max_colwidth', 700)
path = ~
# path로 지정한 디렉터리 밑에 있는 모든 .data 파일들의 파일명을 리스트로 취합
all_files = glob.glob(os.path.join(path, "*.data"))
filename_list = []
opinion_text = []

# 개별 파일들의 파일명은 filename_list 리스트로 취합,
# 개별 파일들의 파일 내용은 DataFrame 로딩 후 다시 string으로 변환하여 opinion_text 리스트로 취합
for file_ in all_files:
    # 개별 파일을 읽어서 DataFrame으로 생성
    df = pd.read_table(file_,index_col=None, header=0,encoding='latin1')

    # 절대경로로 주어진 file 명을 가공
    # 맨 마지막 .data 확장자도 제거
    filename_ = file_.split('\\')[-1]
    filename = filename_.split('.')[0]

    # 파일명 리스트와 파일 내용 리스트에 파일명과 파일 내용을 추가
    filename_list.append(filename)
    opinion_text.append(df.to_string())

# 파일명 리스트와 파일 내용 리스트를  DataFrame으로 생성
document_df = pd.DataFrame({'filename':filename_list, 'opinion_text':opinion_text})
document_df.head()
```

-먼저 해당 디렉터리 내의 모든 파일에 대해 각각 for 반복문으로 반복하면서 개별 파일명을 파일명 리스트에 추가

-개별 파일은 DataFrame으로 읽은 후 다시 문자열로 반환한 뒤 파일 내용 리스트에 추가

-만들어진 파일명 리스트와 파일 내용 리스트를 이용해 새롭게 파일명과 파일 매용을 칼럼으로 가지는 DataFrame을 생성

#07 문서 군집화 소개와 실습

	filename	opinion_text
0	accuracy_garmin_nuvi_255W_gps	, and is very, very accurate .\n0 but for the most part, we find that the Garmin software provides accurate directions, whereever we intend to go .\n1 This functi...
1	bathroom_bestwestern_hotel_sfo	The room was not overly big, but clean and very comfortable beds, a great shower and very clean bathrooms .\n0 The second room was smaller, with a very inconvenient bathroom layout, but at least it was quieter and we were able to sleep .\n1 ...
2	battery-life_amazon_kindle	After I plugged it in to my USB hub on my computer to charge the battery the charging cord design is very clever !\n0 After you have paged tru a 500, page book one, page, at, a, time to get from Chapter 2 to Chapter 15, see how excited you are about a low battery and all the time it took to get there !\n1 ...
3	battery-life_ipod_nano_8gb	short battery life I moved up from an 8gb .\n0 I love this ipod except for the battery life .\n1 ...
4	battery-life_netbook_1005ha	6GHz 533FSB cpu, glossy display, 3, Cell 23Wh Li, ion Battery , and a 1 .\n0 Not to mention that as of now...

각 파일 이름 자체만으로 의견의 텍스트가 어떠한 제품/서비스에 대한 리뷰인지 잘 알 수 있음

#07 문서 군집화 소개와 실습

TfidfVectorizer가 어근 변환을 직접 지원x -> tokenizer 인자에 커스텀 어근 변환 함수를 적용해 어근 변환 수행.

```
from nltk.stem import WordNetLemmatizer
import nltk
nltk.download('punkt')
nltk.download('wordnet')
import string

remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
lemmar = WordNetLemmatizer()

# 입력으로 들어온 token단어들에 대해서 lemmatization 어근 변환
def LemTokens(tokens):
    return [lemmar.lemmatize(token) for token in tokens]

# TfidfVectorizer 객체 생성 시 tokenizer인자로 해당 함수를 설정하여 lemmatization 적용
# 입력으로 문장을 받아서 stop words 제거-> 소문자 변환 -> 단어 토큰화 -> lemmatization 어근 변환
def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

#07 문서 군집화 소개와 실습

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vect = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english' , \
                             ngram_range=(1,2), min_df=0.05, max_df=0.85 )

#opinion_text 컬럼값으로 feature vectorization 수행
feature_vect = tfidf_vect.fit_transform(document_df['opinion_text'])
```

ngram(1, 2), min_df와 max_df 범위 설정해 피처 개수 제한.

TfidfVectorizer의 fit_transform() 인자로 option_text 칼럼 입력해 개별 텍스트 문서 대해 TF-IDF 변환된 피처 벡터화된 행렬 구함.

#07 문서 군집화 소개와 실습

```
from sklearn.cluster import KMeans

# 5개 집합으로 군집화 수행. 예제를 위해 동일한 클러스터링 결과 도출용 random_state=0
km_cluster = KMeans(n_clusters=5, max_iter=10000, random_state=0)
km_cluster.fit(feature_vect)
cluster_label = km_cluster.labels_
cluster_centers = km_cluster.cluster_centers_
document_df['cluster_label'] = cluster_label
```

문서별 텍스트가 tf-idf 변환된 피처 벡터화 행렬 데이터에 대해서 군집화를 수행해 어떤 문서끼리 군집되는지 확인

- 군집화 기법은 k-평균 적용
- 최대 반복 횟수는 10000으로 설정
- 5개의 중심기반으로 어떻게 군집화되는지 확인
- kmeans를 수행한 후에 군집의 label 값과 중심별로 할당된 데이터 세트의 좌표값도 구함

#07 문서 군집화 소개와 실습

```
document_df['cluster_label'] = cluster_label
document_df.head()
```

	filename	opinion_text	cluster_label
0	accuracy_garmin_nuvi_255W_gps	, and is very, very accurate .\n0but for the m...	4
1	bathroom_bestwestern_hotel_sfo	The room was not overly big, but clean and ve...	0
2	battery-life_amazon_kindle	After I plugged it in to my USB hub on my com...	1
3	battery-life_ipod_nano_8gb	short battery life I moved up from an 8gb	1
4	battery-life_netbook_1005ha	6GHz 533FSB cpu, glossy display, 3, Cell 23Wh...	1

각 데이터별로 할당된 군집의 레이블을 파일명과 파일 내용을 가지고 있는 document_df DataFrame에 'cluster_label' 칼럼을 추가해 저장

#07 문서 군집화 소개와 실습

```
document_df[document_df['cluster_label']==0].sort_values(by='filename')
```

-> 판다스 DataFrame의 sort_values(by=정렬칼럼명)을 수행하면 인자로 입력된 '정렬칼럼명 ' 으로 데이터를 정렬

	filename	opinion_text	cluster_label
1	bathroom_bestwestern_hotel_sfo	The room was not overly big, but clean and ve...	0
13	food_holiday_inn_london	The room was packed to capacity with queues ...	0
14	food_swissotel_chicago	The food for our event was delicious .\n0 Th...	0
15	free_bestwestern_hotel_sfo	The wine reception is a great idea as it is ...	0
20	location_bestwestern_hotel_sfo	Good Value good location , ideal choice .\n...	0
21	location_holiday_inn_london	Great location for tube and we crammed in a f...	0
24	parking_bestwestern_hotel_sfo	Parking was expensive but I think this is comm...	0
28	price_holiday_inn_london	All in all, a normal chain hotel on a nice lo...	0
32	room_holiday_inn_london	We arrived at 23,30 hours and they could not r...	0
30	rooms_bestwestern_hotel_sfo	Great Location , NiceRooms , Helpless Conci...	0
31	rooms_swissotel_chicago	The Swissotel is one of our favorite hotels in...	0
38	service_bestwestern_hotel_sfo	Both of us having worked in tourism for over 1...	0
39	service_holiday_inn_london	not customer, oriented hotelvery low service ...	0
40	service_swissotel_hotel_chicago	Mediocre room and service for a very extravaga...	0
45	staff_bestwestern_hotel_sfo	Staff are friendly and helpful.\n0The staff i...	0
46	staff_swissotel_chicago	The staff at Swissotel were not particularly...	0

cluster 0은 호텔에 대한 리뷰로 군집화 되어 있음

#07 문서 군집화 소개와 실습

document_df[document_df['cluster_label']==1].sort_values(by='filename')

	filename	opinion_text	cluster_label
2	battery-life_amazon_kindle	After I plugged it in to my USB hub on my com...	1
3	battery-life_ipod_nano_8gb	short battery life I moved up from an 8gb	1
4	battery-life_netbook_1005ha	6GHz 533FSB cpu, glossy display, 3, Cell 23Wh...	1
11	features_windows7	I had to uninstall anti, virus and selected ot...	1
19	keyboard_netbook_1005ha	, I think the new keyboard rivals the great...	1
26	performance_netbook_1005ha	The Eee Super Hybrid Engine utility lets user...	1
34	screen_garmin_nuvi_255W_gps	It is easy to read and when touching the scre...	1
35	screen_ipod_nano_8gb	As always, the video screen is sharp and brigh...	1
36	screen_netbook_1005ha	Keep in mind that once you get in a room full ...	1
41	size_asus_netbook_1005ha	A few other things I'd like to point out is th...	1
42	sound_ipod_nano_8gb	headphone jack i got a clear case for it and ...	1
49	video_ipod_nano_8gb	I bought the 8, gig Ipod Nano that has the bu...	1

cluster 1은 킨들, 아이팟, 넷북 등 전자기기에 대한 리뷰로 군집화 되어 있음

	filename	opinion_text	cluster_label
6	comfort_honda_accord_2008	Drivers seat not comfortable, the car itself ...	2
7	comfort_toyota_camry_2007	Ride seems comfortable and gas mileage fairly ...	2
16	gas_mileage_toyota_camry_2007	Ride seems comfortable and gas mileage fairly...	2
17	interior_honda_accord_2008	I love the new body style and the interior is ...	2
18	interior_toyota_camry_2007	First of all, the interior has way too many ch...	2
22	mileage_honda_accord_2008	It's quiet, get good gas mileage and looks cle...	2
25	performance_honda_accord_2008	Very happy with my 08 Accord, performance is...	2
29	quality_toyota_camry_2007	I previously owned a Toyota 4Runner which had...	2
37	seats_honda_accord_2008	Front seats are very uncomfortable .\n0 No mem...	2
47	transmission_toyota_camry_2007	After slowing down, transmission has to be ...	2

cluster 2는 토요타, 혼다 등 자동차에 대한 리뷰로 군집화 되어 있음

#07 문서 군집화 소개와 실습

	filename	opinion_text	cluster_label
5	buttons_amazon_kindle	I thought it would be fitting to christen my ...	3
10	eyesight-issues_amazon_kindle	It feels as easy to read as the K1 but doesn't...	3
12	fonts_amazon_kindle	Being able to change the font sizes is awesom...	3
23	navigation_amazon_kindle	In fact, the entire navigation structure has...	3
27	price_amazon_kindle	If a case was included, as with the Kindle 1,...	3
44	speed_windows7	Windows 7 is quite simply faster, more stable...	3

cluster 3은 킨들에 대한 리뷰로 군집화 되어 있음.

	filename	opinion_text	cluster_label
0	accuracy_garmin_nuvi_255W_gps	, and is very, very accurate .\n0but for the m...	4
8	directions_garmin_nuvi_255W_gps	You also get upscale features like spoken di...	4
9	display_garmin_nuvi_255W_gps	3 quot widescreen display was a bonus .\n0 ...	4
33	satellite_garmin_nuvi_255W_gps	It's fast to acquire satellites .\n0 If you'v...	4
43	speed_garmin_nuvi_255W_gps	Another feature on the 255w is a display of t...	4
48	updates_garmin_nuvi_255W_gps	Another thing to consider was that I paid $50 ...	4
50	voice_garmin_nuvi_255W_gps	The voice prompts and maps are wonderful espe...	4

cluster 4는 주로 차량용 네비게이션으로 군집이 되어 있음

#07 문서 군집화 소개와 실습

	filename	opinion_text	cluster_label
0	accuracy_garmin_nuvi_255W_gps	, and is very, very accurate .\n0but for the m...	0
2	battery-life_amazon_kindle	After I plugged it in to my USB hub on my com...	0
3	battery-life_ipod_nano_8gb	short battery life I moved up from an 8gb	0
4	battery-life_netbook_1005ha	6GHz 533FSB cpu, glossy display, 3, Cell 23Wh...	0
5	buttons_amazon_kindle	I thought it would be fitting to christen my ...	0
8	directions_garmin_nuvi_255W_gps	You also get upscale features like spoken di...	0
9	display_garmin_nuvi_255W_gps	3 quot widescreen display was a bonus .\n0 ...	0
10	eyesight-issues_amazon_kindle	It feels as easy to read as the K1 but doesn't...	0
11	features_windows7	I had to uninstall anti, virus and selected ot...	0
12	fonts_amazon_kindle	Being able to change the font sizes is awesom...	0
19	keyboard_netbook_1005ha	, I think the new keyboard rivals the great...	0
23	navigation_amazon_kindle	In fact, the entire navigation structure has...	0
26	performance_netbook_1005ha	The Eee Super Hybrid Engine utility lets user...	0
27	price_amazon_kindle	If a case was included, as with the Kindle 1,...	0
33	satellite_garmin_nuvi_255W_gps	It's fast to acquire satellites .\n0 If you'v...	0
34	screen_garmin_nuvi_255W_gps	It is easy to read and when touching the scre...	0
35	screen_ipod_nano_8gb	As always, the video screen is sharp and brigh...	0
36	screen_netbook_1005ha	Keep in mind that once you get in a room full ...	0
41	size_asus_netbook_1005ha	A few other things I'd like to point out is th...	0
42	sound_ipod_nano_8gb	headphone jack i got a clear case for it and ...	0
43	speed_garmin_nuvi_255W_gps	Another feature on the 255w is a display of t...	0
44	speed_windows7	Windows 7 is quite simply faster, more stable...	0
48	updates_garmin_nuvi_255W_gps	Another thing to consider was that I paid $50 ...	0
49	video_ipod_nano_8gb	I bought the 8, gig Ipod Nano that has the bu...	0
50	voice_garmin_nuvi_255W_gps	The voice prompts and maps are wonderful espe...	0

	filename	opinion_text	cluster_label
1	bathroom_bestwestern_hotel_sfo	The room was not overly big, but clean and ve...	1
13	food_holiday_inn_london	The room was packed to capacity with queues ...	1
14	food_swissotel_chicago	The food for our event was delicious .\n0 Th...	1
15	free_bestwestern_hotel_sfo	The wine reception is a great idea as it is ...	1
20	location_bestwestern_hotel_sfo	Good Value good location , ideal choice .\n...	1
21	location_holiday_inn_london	Great location for tube and we crammed in a f...	1
24	parking_bestwestern_hotel_sfo	Parking was expensive but I think this is comm...	1
28	price_holiday_inn_london	All in all, a normal chain hotel on a nice lo...	1
32	room_holiday_inn_london	We arrived at 23,30 hours and they could not r...	1
30	rooms_bestwestern_hotel_sfo	Great Location , NiceRooms , Helpless Conci...	1
31	rooms_swissotel_chicago	The Swissotel is one of our favorite hotels in...	1
38	service_bestwestern_hotel_sfo	Both of us having worked in tourism for over 1...	1
39	service_holiday_inn_london	not customer, oriented hotelvery low service ...	1
40	service_swissotel_hotel_chicago	Mediocre room and service for a very extravaga...	1
45	staff_bestwestern_hotel_sfo	Staff are friendly and helpful.\n0The staff i...	1
46	staff_swissotel_chicago	The staff at Swissotel were not particularly...	1

	filename	opinion_text	cluster_label
6	comfort_honda_accord_2008	Drivers seat not comfortable, the car itself ...	2
7	comfort_toyota_camry_2007	Ride seems comfortable and gas mileage fairly ...	2
16	gas_mileage_toyota_camry_2007	Ride seems comfortable and gas mileage fairly...	2
17	interior_honda_accord_2008	I love the new body style and the interior is ...	2
18	interior_toyota_camry_2007	First of all, the interior has way too many ch...	2
22	mileage_honda_accord_2008	It's quiet, get good gas mileage and looks cle...	2
25	performance_honda_accord_2008	Very happy with my 08 Accord, performance is...	2
29	quality_toyota_camry_2007	I previously owned a Toyota 4Runner which had...	2
37	seats_honda_accord_2008	Front seats are very uncomfortable .\n0 No mem...	2
47	transmission_toyota_camry_2007	After slowing down, transmission has to be ...	2

cluster 0은 전자기기에 대한 리뷰만으로 잘 군집화 되어 있음

cluster 1은 호텔에 대한 리뷰만으로 잘 군집화 되어 있음

cluster 2는 자동차에 대한 리뷰만으로 잘 군집화 되어 있음

#07 문서 군집화 소개와 실습

-KMeans 객체는 각 군집을 구성하는 단어 피처가 군집의 중심을 기준으로 얼마나 가깝게 위치해 있는지를 cluster_centers_라는 속성으로 제공

-cluster_centers_는 배열 값으로 제공되며, 행은 개별 군집을, 열은 개별 피처를 의미

-각 배열 내의 값은 개별 군집 내의 상대 위치를 숫자 값으로 표현한 일종의 좌표값

Ex) cluster_centers[0,1]은 0번 군집에서 두 번째 피처의 위치 값

```
cluster_centers = km_cluster.cluster_centers_
print('cluster_centers shape :',cluster_centers.shape)
print(cluster_centers)
```

```
cluster_centers shape : (3, 4613)
[[0.00760308 0.00777633 0.         ... 0.0067567  0.         0.        ]
 [0.00263248 0.         0.0017299  ... 0.         0.00190972 0.00146615]
 [0.00334841 0.         0.         ... 0.         0.         0.        ]]
```

cluster_centers_는 (3, 4611) 배열

이는 군집이 3개, word 피처가 4611개로 구성되었음을 의미함.

각 행의 배열 값은 각 군집 내의 4611개 피처의 위치가 개별 중심과 얼마나 가까운가를 상대 값으로 나타낸 것으로 0에서 1까지의 값을 가질 수 있으며 1에 가까울수록 중심과 가까운 값을 의미함.

#07 문서 군집화 소개와 실습

```
def get_cluster_details(cluster_model, cluster_data, cluster_nums,
                        feature_names, top_n_features=10):

    # 핵심 단어 등 정보를 담을 사전 생성
    cluster_details = {}

    # word 피처 중심과의 거리 내림차순 정렬시 값들의 index 반환
    center_info = cluster_model.cluster_centers_         # 군집 중심 정보
    center_descend_ind = center_info.argsort()[:, ::-1] # 행별(군집별)로 역순 정렬

    # 군집별 정보 담기
    for i in range(cluster_nums):
        # 군집별 정보를 담을 데이터 초기화
        cluster_details[i] = {} # 사전 안에 사전

        # 각 군집에 속하는 파일명
        filenames = cluster_data[cluster_data["cluster_label"] == i]["filename"]
        filenames = filenames.values.tolist()

        # 군집별 중심 정보
        top_feature_values = center_info[i, :top_n_features].tolist()

        # 군집별 핵심 단어 피처명
        top_feature_indexes = center_descend_ind[i, :top_n_features]
        top_features = [feature_names[ind] for ind in top_feature_indexes]

        # 각 군집별 정보 사전에 담기
        cluster_details[i]["cluster"] = i                          # i번째 군집
        cluster_details[i]["top_features"] = top_features              # 군집별 핵심 단어
        cluster_details[i]["top_feature_values"] = top_feature_values  # 군집별 중심 정보
        cluster_details[i]["filenames"] = filenames                    # 군집 속 파일명

    return cluster_details
```

ndarray의 argsort()[:,::-1]를 이용하면 cluster_centers 배열 내 값이 큰 순으로 정렬된 위치 인덱스를 반환

-> 이 위치 인덱스는 핵심 단어 피처의 이름을 출력하기 위해서 필요.

새로운 함수 get_cluster_details()를 생성해 cluster_centers_ 배열 내에서 가장 값이 큰 데이터의 위치 인덱스를 추출한 뒤, 해당 인덱스를 이용해 핵심 단어 이름과 그때의 상대 위치 값을 추출해 cluster_details라는 Dict 객체 변수에 기록하고 반환.

-> cluster_details에는 개별 군집번호, 핵심 단어, 핵심단어 중심 위치 상댓값, 파일명 속성 값 정보가 있음.

#07 문서 군집화 소개와 실습

```
def print_cluster_details(cluster_details):
    for cluster_num, cluster_detail in cluster_details.items():
        print('####### Cluster {0}'.format(cluster_num))
        print('Top features:', cluster_detail['top_features'])
        print('Reviews 파일명 :',cluster_detail['filenames'][:7])
        print('================================================')



feature_names = tfidf_vect.get_feature_names()

cluster_details = get_cluster_details(cluster_model=km_cluster,
cluster_data=document_df,\
                                      feature_names=feature_names,
clusters_num=3, top_n_features=10 )
print_cluster_details(cluster_details)
```

get_cluster_details(), print_cluster_details()를 호출.

get_cluster_details() 호출 시 인자는 KMeans 군집화 객체, 파일명 추출을 위한 document_df DataFrame, 핵심 단어 추출을 위한 피처명 리스트, 전체 군집 개수, 그리고 핵심 단어 추출 개수.

피처명 리스트는 앞에서 TF-IDF 변환된 tfidf_vect 객체에서 get_feature_names()로 추출.

#07 문서 군집화 소개와 실습

```
####### Cluster 0
Top features: ['screen', 'battery', 'keyboard', 'battery life', 'life',
'kindle', 'video', 'direction', 'size', 'voice']
Reviews 파일명: ['accuracy_garmin_nuvi_255W_gps', 'battery-
life_amazon_kindle', 'battery-life_ipod_nano_8gb']
------------------------------------------------------------
------------------------------------------------------
####### Cluster 1
Top features: ['room', 'hotel', 'service', 'staff', 'food', 'location',
'bathroom', 'clean', 'price', 'parking']
Reviews 파일명: ['bathroom_bestwestern_hotel_sfo',
'food_holiday_inn_london', 'food_swissotel_chicago']
------------------------------------------------------------
------------------------------------------------------
####### Cluster 2
Top features: ['interior', 'seat', 'mileage', 'comfortable', 'gas', 'gas
mileage', 'car', 'transmission', 'performance', 'quality']
Reviews 파일명: ['comfort_honda_accord_2008',
'comfort_toyota_camry_2007', 'gas_mileage_toyota_camry_2007']
------------------------------------------------------------
------------------------------------------------------
```

cluster 0: ‘screen’, ‘battery’ m ‘battery life’ 등과 같은 화면과 배터리 수명 등이 핵심 단어로 군집화

cluster 1: ‘room’, ‘hotel’, ‘service’, ‘staff’ 등 같은 방과 서비스 등이 핵심 단어로 군집화

cluster 2: ‘interior’, ’ seat’, ‘mileage’, ‘comfortable’ 등과 같은 실내 인테리어, 좌석, 연료 효율 등이 핵심 단어로 군집화

EWHA EURON

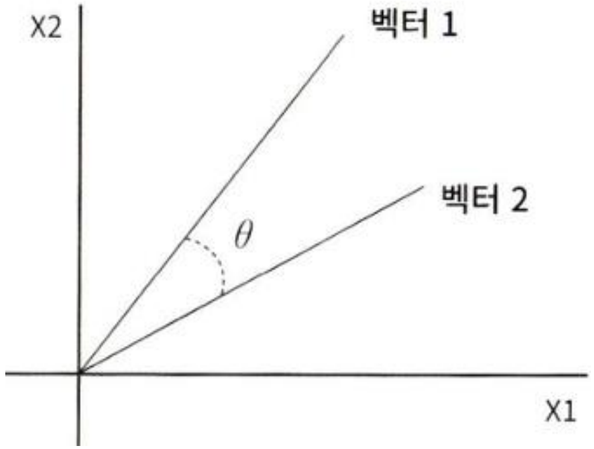
1. 문서 유사도

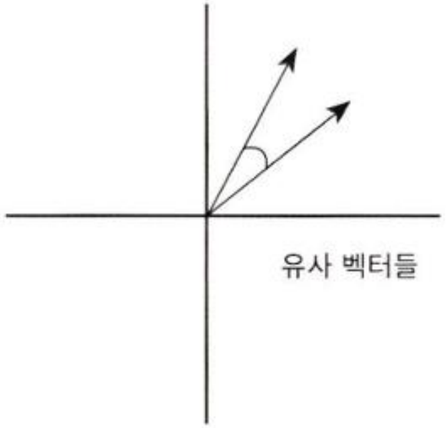
초급 세션 2팀 조주현

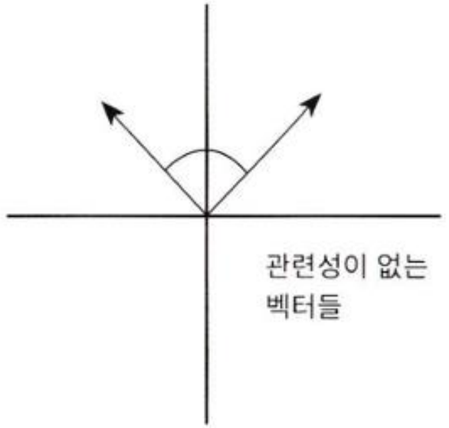
EWHA EURON

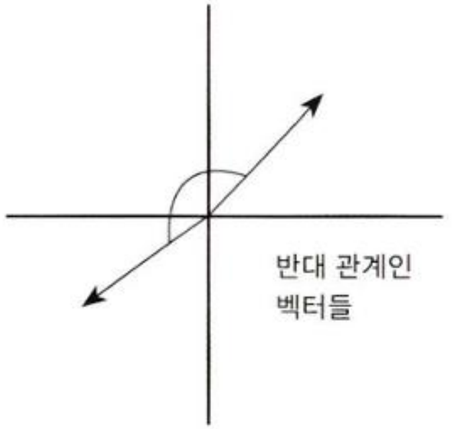
코사인 유사도

코사인 유사도: 두 벡터 사이의 사잇각을 구하여 얼마나 유사한지를 수치로 적용한 것









$$\text{similarity} = cos\theta = \frac{A \cdot B}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

코사인 유사도 예제

```python
import numpy as np

def cos_similarity(v1, v2):
    dot_product = np.dot(v1, v2)
    l2_norm = (np.sqrt(sum(np.square(v1))) * np.sqrt(sum(np.square(v2))))
    similarity = dot_product / l2_norm

    return similarity
```

```python
from sklearn.feature_extraction.text import TfidfVectorizer

doc_list = ['if you take the blue pill, the story ends' ,
            'if you take the red pill, you stay in Wonderland',
            'if you take the red pill, I show you how deep the rabbit hole goes']

tfidf_vect_simple = TfidfVectorizer()
feature_vect_simple = tfidf_vect_simple.fit_transform(doc_list)
print(feature_vect_simple.shape)
```

```
(3, 18)
```

코사인 유사도 예제

```
feature_vect_dense = feature_vect_simple.todense()

vect1 = np.array(feature_vect_dense[0]).reshape(-1,)
vect2 = np.array(feature_vect_dense[1]).reshape(-1,)

similarity_simple = cos_similarity(vect1, vect2 )
print('문장 1, 문장 2 Cosine 유사도: {0:.3f}'.format(similarity_simple))
```

```
문장 1, 문장 2 Cosine 유사도: 0.402
```

```
vect1 = np.array(feature_vect_dense[0]).reshape(-1,)
vect3 = np.array(feature_vect_dense[2]).reshape(-1,)
similarity_simple = cos_similarity(vect1, vect3 )
print('문장 1, 문장 3 Cosine 유사도: {0:.3f}'.format(similarity_simple))

vect2 = np.array(feature_vect_dense[1]).reshape(-1,)
vect3 = np.array(feature_vect_dense[2]).reshape(-1,)
similarity_simple = cos_similarity(vect2, vect3 )
print('문장 2, 문장 3 Cosine 유사도: {0:.3f}'.format(similarity_simple))
```

```
문장 1, 문장 3 Cosine 유사도: 0.404
문장 2, 문장 3 Cosine 유사도: 0.456
```

코사인 유사도 예제

```
from sklearn.metrics.pairwise import cosine_similarity

similarity_simple_pair = cosine_similarity(feature_vect_simple[0] , feature_vect_simple)
print(similarity_simple_pair)
```

```
[[1.         0.40207758 0.40425045]]
```

```
from sklearn.metrics.pairwise import cosine_similarity

similarity_simple_pair = cosine_similarity(feature_vect_simple[0] , feature_vect_simple[1:])
print(similarity_simple_pair)
```

```
[[0.40207758 0.40425045]]
```

```
similarity_simple_pair = cosine_similarity(feature_vect_simple , feature_vect_simple)
print(similarity_simple_pair)
print('shape:',similarity_simple_pair.shape)
```

```
[[1.         0.40207758 0.40425045]
 [0.40207758 1.         0.45647296]
 [0.40425045 0.45647296 1.        ]]
shape: (3, 3)
```

Opinion Review 데이터 세트를 이용한 문서 유사도 측정

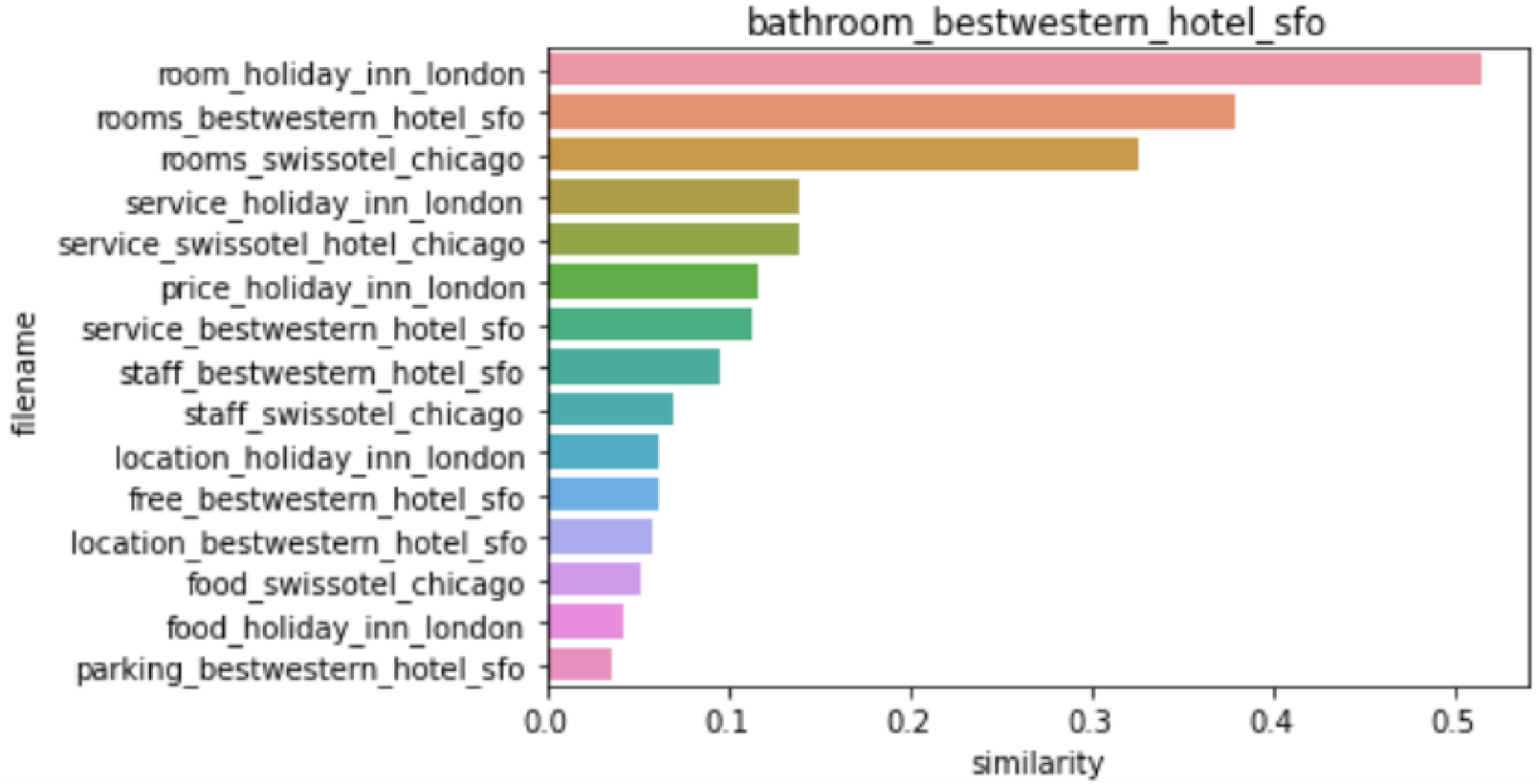
```
from sklearn.metrics.pairwise import cosine_similarity

similarity_simple_pair = cosine_similarity(feature_vect_simple[0] , feature_vect_simple)
print(similarity_simple_pair)
```

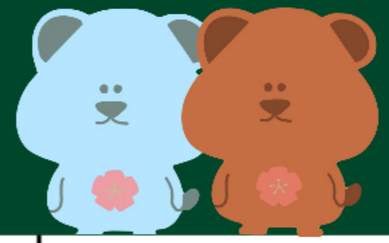
```
[[1.         0.40207758 0.40425045]]
```

```
from sklearn.metrics.pairwise import cosine_similarity

hotel_indexes = document_df[document_df['cluster_label']==2].index
print('호텔로 클러스터링 된 문서들의 DataFrame Index:', hotel_indexes)

comparison_docname = document_df.iloc[hotel_indexes[0]]['filename']
print('##### 비교 기준 문서명 ',comparison_docname,' 와 타 문서 유사도######')

similarity_pair = cosine_similarity(feature_vect[hotel_indexes[0]] , feature_vect[hotel_indexes])
print(similarity_pair)
```

```
호텔로 클러스터링 된 문서들의 DataFrame Index: Int64Index([1, 13, 14, 15, 20, 21, 24, 28, 30, 31,
32, 38, 39, 40, 45, 46], dtype='int64')
##### 비교 기준 문서명  bathroom_bestwestern_hotel_sfo  와 타 문서 유사도######
[[1.         0.0430688  0.05221059 0.06189595 0.05846178 0.06193118
  0.03638665 0.11742762 0.38038865 0.32619948 0.51442299 0.11282857
  0.13989623 0.1386783  0.09518068 0.07049362]]
```

Opinion Review 데이터 세트를 이용한 문서 유사도 측정



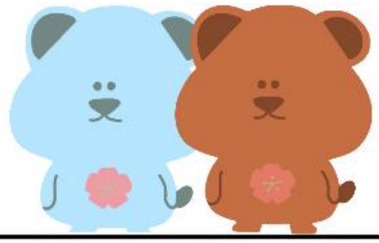


한글 텍스트 처리 – 네이버 영화 평점 감성 분석

2조 김정은

EWHA EURON

목차



EWHA EURON

#01 한글 NLP 처리의 어려움

#1 띄어쓰기

➔ 띄어쓰기를 잘못하면 의미가 왜곡되어 전달될 수 있기 때문임
➔ 영어의 경우 띄어쓰기를 잘못하면 의미가 왜곡되는 게 아니라 잘못된 또는 없는 단어로 인식됨
➔ 또한 한글의 띄어쓰기는 고등 교육을 받은 사람이라도 틀리는 경우 종종 발생

#2 조사

➔ 주어나 목적어를 위해 추가되며, 경우의 수가 많아 어근 추출 등의 전처리 시 제거하기가 까다로움
➔ ‘너희 집은 어디 있니?’ 에서 ‘집은’ 의 ‘은’ 이 뜻하는 것이 조사인지, 아니면 금속 은인지 구분이 까다로움

#02 KoNLPy

#1 KoNLPy

➔ 파이썬의 대표적인 한글 형태소 패키지
➔ 형태소의 사전적인 의미는 '단어로서 의미를 가지는 최소 단위'

➔ ***형태소 분석(Morphological analysis)*** : 말뭉치를 이러한 형태소 어근 단위로 쪼갠 후 각 형태소에 품사 태깅(POS tagging)을 부착하는 작업
➔ KoNLPy 이전에는 파이썬 기반의 형태소 분석 프로그램이 거의 전무했었음
➔ 대부분의 형태소 분석은 C/C++과 Java 기반 패키지로 개발되었음

➔ ***KoNLPy***
➔ 기존의 C/C++, Java로 잘 만들어진 한글 형태소 엔진을 파이썬 래퍼(Wrapper) 기반으로 재작성한 패키지
➔ 기존의 엔진은 그대로 유지한 채 파이썬 기반에서 인터페이스 제공, 검증된 패키지의 안정성 유지 가능
➔ 꼬꼬마(Kkma), 한나눔(Hannanum), Komoran, 은전한닢 프로젝트(Mecab), Twitter 사용 가능

#03 네이버 영화 평점 감성 분석

#1 데이터 로딩

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

train_df = pd.read_csv('ratings_train.txt', sep='₩t')
train_df.head(3)
```

Out[1]:

	id	document	label
0	9976970	아 더빙.. 진짜 짜증나네요 목소리	0
1	3819312	흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나	1
2	10265843	너무재밋었다그래서보는것을추천한다	0

- 한글로 된 문서를 DataFrame으로 로딩할 때 인코딩 이슈가 발생할 수 있음 -> encoding을 cp949로 설정

```
train_df['label'].value_counts( )
```

```
0    75173
1    74827
Name: label, dtype: int64
```

- 학습 데이터 세트의 0과 1의 Label 값 비율
  1이 긍정 감성, 0이 부정 감성
- 어느 한 쪽으로 치우치지 않은 균등한 분포

#03 네이버 영화 평점 감성 분석

#2 데이터 가공

```
import re

train_df = train_df.fillna(' ')
# 정규 표현식을 이용하여 숫자를 공백으로 변경(정규 표현식으로 ₩d 는 숫자를 의미함.)
train_df['document'] = train_df['document'].apply( lambda x : re.sub(r"₩d+", " ", x) )

# 테스트 데이터 셋을 로딩하고 동일하게 Null 및 숫자를 공백으로 변환
test_df = pd.read_csv('ratings_test.txt', sep='₩t')
test_df = test_df.fillna(' ')
test_df['document'] = test_df['document'].apply( lambda x : re.sub(r"₩d+", " ", x) )

# id 칼럼 삭제 수행
train_df.drop('id', axis=1, inplace=True)
test_df.drop('id', axis=1, inplace=True)
```

- ‘document’ 칼럼에 Null이 일부 존재, 공백으로 변환
- 숫자의 경우 단어적인 의미 부족으로 파이썬 정규 표현식 모듈인 re를 이용해 공백으로 변환
- 테스트 데이터 세트의 경우도 파일 로딩, 동일한 데이터 가공 수행

#03 네이버 영화 평점 감성 분석

#3 단어 벡터화

```
from konlpy.tag import Twitter

twitter = Twitter()
def tw_tokenizer(text):
    # 입력 인자로 들어온 text 를 형태소 단어로 토큰화 하여 list 객체 반환
    tokens_ko = twitter.morphs(text)
    return tokens_ko
```

```
C:₩Users₩yong₩anaconda3₩envs₩pymldgrev2₩lib₩site-packages₩konlpy₩tag₩_okt.py:17: UserWarning: "Twi
y v0.4.5.
  warn('"Twitter" has changed to "Okt" since KoNLPy v0.4.5.')
```

- TF-IDF 방식으로 단어를 벡터화
- 먼저 각 문장을 한글 형태소 분석을 통해 형태소 단어로 토큰화
- 한글 형태소 엔진 -> SNS 분석에 적합한 Twitter 클래스 이용
- Twitter 객체의 morphs( ) 메서드를 이용하면 입력 인자로 들어온 문장을 형태소 단어 형태로 토큰화 -> list 객체로 반환
- 형태소 단어 형태로 변환하는 별도의 tokenizer 함수 생성
- 뒤에서 사이킷런의 TfidfVectorizer 클래스의 tokenizer 사용

#03 네이버 영화 평점 감성 분석

#4 TF-IDF 피처 모델 생성

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# Twitter 객체의 morphs( ) 객체를 이용한 tokenizer를 사용, ngram_range는 (1,2)
tfidf_vect = TfidfVectorizer(tokenizer=tw_tokenizer, ngram_range=(1,2), min_df=3, max_df=0.9)
tfidf_vect.fit(train_df['document'])
tfidf_matrix_train = tfidf_vect.transform(train_df['document'])
```

```
C:₩Users₩yong₩anaconda3₩envs₩pymldgrev2₩lib₩site-packages₩sklearn₩feature_extraction₩text.py:516: UserWarning:
ern' will not be used since 'tokenizer' is not None'
  warnings.warn(
```

- 사이킷런의 TfidfVectorizer을 이용해 TF-IDF 피처 모델 생성
- Tokenizer는 tw_tokenizer( ) 함수를 이용

EWHA EURON

#03 네이버 영화 평점 감성 분석

#5 로지스틱 회귀 , 분류 기반의 감성 분석

```
# Logistic Regression 을 이용하여 감성 분석 Classification 수행.
lg_clf = LogisticRegression(random_state=0, solver='liblinear')

# Parameter C 최적화를 위해 GridSearchCV 를 이용.
params = { 'C': [1 ,3.5, 4.5, 5.5, 10 ] }
grid_cv = GridSearchCV(lg_clf , param_grid=params , cv=3 ,scoring='accuracy', verbose=1 )
grid_cv.fit(tfidf_matrix_train , train_df['label'] )
print(grid_cv.best_params_ , round(grid_cv.best_score_,4))
```

```
Fitting 3 folds for each of 5 candidates, totalling 15 fits
{'C': 3.5} 0.8593
```

- 로지스틱 회귀를 이용해 분류 기반의 감성 분석 수행
- 로지스틱 회귀의 하이퍼 파라미터 C 최적화를 위해 GridSearchCV 이용
- C가 3.5일 때 최고 0.8593의 정확도

#03 네이버 영화 평점 감성 분석

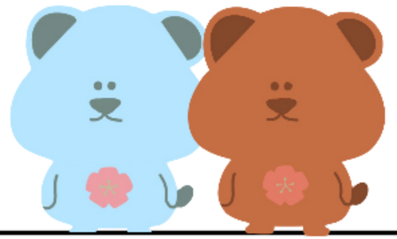
#6 테스트 세트를 이용한 최종 감성 분석 예측

```
from sklearn.metrics import accuracy_score

# 학습 데이터를 적용한 TfidfVectorizer를 이용하여 테스트 데이터를 TF-IDF 값으로 Feature 변환함.
tfidf_matrix_test = tfidf_vect.transform(test_df['document'])

# classifier 는 GridSearchCV에서 최적 파라미터로 학습된 classifier를 그대로 이용
best_estimator = grid_cv.best_estimator_
preds = best_estimator.predict(tfidf_matrix_test)

print('Logistic Regression 정확도: ',accuracy_score(test_df['label'],preds))
```

```
Logistic Regression 정확도:  0.86172
```

- 학습 때 적용한 TfidfVectorizer 그대로 사용
- 그래야만 학습 시 설정된 TfidfVectorizer 피처 개수와 테스트 데이터를 TfidfVectorizer로 변환할 피처 개수가 같아짐
- TfidfVectorizer 객체 변수인 tfidf_vect를 이용해 transform( )을 테스트 데이터의 document 칼럼에 수행

THANK YOU



EWHA EURON