07_텍스트 분석

가장 유명: 챗GPT

텍스트 분석

• 텍스트 분석이란?

> 보편적으로 텍스트(문자)를 머신이 이해하게 만드는 기술

> 이전에는 NLP(National Language Processing)와 TA(Text Analysis)로 구분하였지만 현재는 구분의 의미가 없어졌다.

자연어처리 문서 분석

> NLP는 비정형 데이터인 문자 자체를 컴퓨터가 이해하기 위한 기술이고, TA는 텍스트에서 의미 있는 정보를 추출하는 기술이다.

> 텍스트 분석의 종류

- 텍스트 분류
- 감성 분석
- 텍스트 요약
- 텍스트 군집화, 유사도 측정

텍스트 분석

• 텍스트 분석의 종류

> 텍스트 분류(Text Classification / Text Categorization)

: 문서가 특정 분류(카테고리)에 속하는 것을 예측하는 기법

지도학습: 분류

> 감성 분석(Sentiment Analysis)

: 텍스트에서 나타나는 감정/판단/의견/기분 등의 주관적인 요소를 분석하는 기법

지도: 분류,회귀

예) 긍정/부정

> 텍스트 요약(Summariztion)

: 텍스트 내에서 중요한 주제나 중심 사상을 추출하는 기법

비지도학습 :워드클라우드

> 텍스트 군집화(Clustering)와 유사도 측정(Similarity)

: 비슷한 유형의 문서에 대한 군집화를 수행하는 기법

비지도: 군집

유사도 측정: 추천시스템의 핵심
논문 비교

텍스트 분석

• 텍스트 분석 프로세스

<전처리>
1.클렌징: 불필요한 내용 삭제
2.토큰화: (자르기) 문서->문장, 문장->단어
3.스톱워드(불용어)제거
4.어근 추출

> 텍스트 사전 준비작업(데이터 전처리)

: 클렌징, 토큰화, 스톱 워드 제거, 철자 수정, 어근 추출 등

> 피처 벡터화 / 추출

피처 벡터화: 숫자(변수)로 바꿈
/ 필요한 단어들만 남겨 숫자화

: 텍스트에서 피처를 추출하고 벡터 값을 할당

대표적인 방식으로 BOW(Bag Of Word), Word2Vec이 있다.

딥러닝에서 사용

> 머신러닝 모델 수립 및 학습/예측/평가

: 텍스트 분석에 맞는 머신러닝 모델을 적용해 학습/예측/평가를 진행하다.

• 텍스트 전처리

> 클렌징 : 텍스트에서 불필요한 문자, 기호 등을 사전에 제거한다.

```
with open('./stevejobs.txt', 'r', encoding='utf8') as f:
    rows = f.readlines()
    lines = [row for row in rows]
text = ' '.join(lines)
print(text[:200])
```

Out :

```
This is the text of the Commencement address by Steve Jobs, CEO of Apple Computer and of Pixar Animation Studios, delivered on June 12, 2005.


I am honored to be with you today at your commenceme
```

• 텍스트 전처리

> 클렌징 : 텍스트에서 불필요한 문자, 기호 등을 사전에 제거한다.

```
import re


compile = re.compile("[^ a-zA-Z0-9₩.]+")
text = compile.sub('',text).lower()
print(text[:200])
```

Out : this is the text of the commencement address by steve jobs ceo of apple computer and of pixar animation studios delivered on june 12 2005. i am honored to be with you today at your commencement fro

• 텍스트 전처리

> 텍스트 토큰화 : 문장을 분리하고, 단어를 분리하여 토큰으로 만드는 작업

```
# 문장 토큰화
import nltk
nltk.download('punkt')
sentences = nltk.sent_tokenize(text = text)
print(sentences[:3])
```

Out :

```
['this is the text of the commencement address by steve jobs ceo of apple computer and of pixar animatio
n studios delivered on june 12 2005.    i am honored to be with you today at your commencement from one
of the finest universities in the world.', 'i never graduated from college.', 'truth be told this is the
closest ive ever gotten to a college graduation.']
```

• 텍스트 전처리

> 텍스트 토큰화 : 문장을 분리하고, 단어를 분리하여 토큰으로 만드는 작업

```
# 단어 토큰화
word_token = []
for sentence in sentences:
    words = nltk.word_tokenize(sentence)
    word_token.extend(words)
print(word_token[:10])
```

Out : ['this', 'is', 'the', 'text', 'of', 'the', 'commencement', 'address', 'by', 'steve']

• 텍스트 전처리

> 텍스트 토큰화 함수

```
def tokenize_text(text):
    sentences = nltk.sent_tokenize(text)
    word_tokens = [nltk.word_tokenize(sentense)
                   for sentense in sentences]
    return word_tokens
word_tokens = tokenize_text(text)
print(word_tokens[:2])
```

Out :

```
[['this', 'is', 'the', 'text', 'of', 'the', 'commencement', 'address', 'by', 'steve', 'jobs', 'ceo', 'o
f', 'apple', 'computer', 'and', 'of', 'pixar', 'animation', 'studios', 'delivered', 'on', 'june', '12',
'2005.', 'i', 'am', 'honored', 'to', 'be', 'with', 'you', 'today', 'at', 'your', 'commencement', 'from',
'one', 'of', 'the', 'finest', 'universities', 'in', 'the', 'world', '.'], ['i', 'never', 'graduated', 'f
rom', 'college', '.']]
```

• 텍스트 전처리

> 스톱 워드 제거 : 분석에 큰 의미 없는 단어 제거(this, is 등)

```
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
print('영어 stop words 개수: ',len(stopwords))
print(stopwords[:10])
```

Out : 영어 stop words 개수: 179
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]

• 텍스트 전처리

> 스톱 워드 제거 : 분석에 큰 의미 없는 단어 제거(this, is 등)

```
all_tokens = []
for sentence in word_tokens:
    filtered_words = []
    for word in sentence:
        if word not in stopwords:
            filtered_words.append(word)
    all_tokens.append(filtered_words)
print(all_tokens[:1])
```

Out :

```
[['text', 'commencement', 'address', 'steve', 'jobs', 'ceo', 'apple', 'compute
r', 'pixar', 'animation', 'studios', 'delivered', 'june', '12', '2005.', 'honore
d', 'today', 'commencement', 'one', 'finest', 'universities', 'world', '.']]
```

• 텍스트 전처리

> Stemming과 Lemmatizaion : 단어의 원형을 찾는 기법

```
from nltk.stem import LancasterStemmer
stemmer = LancasterStemmer()

print(stemmer.stem('working'))
print(stemmer.stem('works'))
print(stemmer.stem('worked'))
```

Out : work
work
work

• 텍스트 전처리

> Stemming과 Lemmatizaion : 단어의 원형을 찾는 기법

```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemma = WordNetLemmatizer()


print(lemma.lemmatize('amusing','v'))
print(lemma.lemmatize('amuses','v'))
print(lemma.lemmatize('amused','v'))
```

Out : amuse
amuse
amuse

• 텍스트 전처리

> Stemming과 Lemmatizaion : 단어의 원형을 찾는 기법

```
print(stemmer.stem('happier'), stemmer.stem('happiest'))
print(lemma.lemmatize('happier','a'))
print(lemma.lemmatize('happiest','a'))
```

Out : happy happiest
happy
happy

• 피처 벡터화

> BOW(Bag of Words)

: 단어에 대한 빈도 값을 부여해 피처 값을 추출하는 모델

	and	baseball	daughter	games	likes	my	play	to	too	watch	wife
문장 1	1	2	1	2	2	2		2	1	2	1
문장 2		1			1	1	1				1

> 단점

- 문맥 의미 부족 : 단어의 순서를 고려하지 않으므로 문맥적 의미가 무시됨
- 희소 행렬 문제 : 서로 다른 단어로 구성되어 빈도가 나타나지 않는 경우가 많아 예측 성능이 떨어진다.

• 피처 벡터화

> BOW 피처 백터화

: 모든 문서의 모든 단어를 칼럼 형태로 나열하고, 각 문서에서 해당 단어의 횟수 혹은 정규화된 빈도를 값으로 부여하여 데이터 세트 모델로 변경

> M개의 문서, N개의 단어 -> M x N 행렬로 구성

> Count 벡터화

: 개별 문서에 해당 단어가 나타나는 횟수가 많을수록 중요한 단어로 인식

> TF-IDF 벡터화

: 개별 문서에서 자주 나타나는 단어에 높은 가중치를 주되, 전체 문서에서 자주 나타나는 단어에 대해서 패널티를 부여

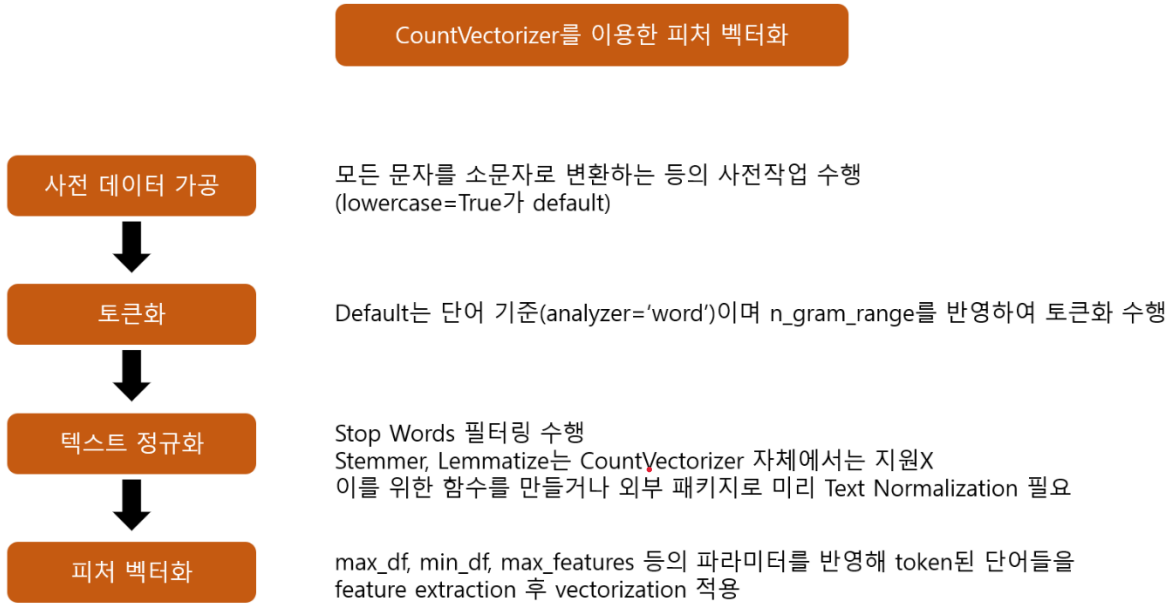
• 피처 벡터화

> 파라미터 종류

- max_df : 너무 높은 빈도수를 가지는 단어 피처를 제외, 정수 값을 가지면 전체 문서에 걸쳐 n개 이하로 나타나는 단어만 피처로 추출하고 부동 소수점 값을 가지면 전체 문서에 걸쳐 빈도수% 까지의 단어만 피처로 추출한다.
- min_df : 너무 낮은 빈도수를 가지는 단어 피처를 제외
- max_features : 높은 빈도를 가지는 단어 순으로 몇 개를 추출할 건지 입력
- stop_words : english로 지정하면 여어의 스톱 워드로 지정된 단어는 추출에서 제외한다.
- n_gram_range : BOW 모델이 문맥의 의미를 반영하지 못한다는 단점을 극복하기 위해 n_gram 범위를 설정한다. 튜플 형태로 (범위 최솟값, 범위 최댓값)을 지정
- analyzer : 피처 추출을 수행한 단위를 지정한다. default = 'word'
- token_pattern : 토큰화를 수행하는 정규 표현식 패턴을 지정합니다. default = '₩b₩w₩w+₩b'
- tokenizer : 토큰화를 별도의 커스텀 함수로 이용시 적용한다.

• 피처 벡터화

> 피처 벡터화 프로세스



• 피처 벡터화

> 텍스트 분류 실습 – 20 뉴스그룹

```
from sklearn.datasets import fetch_20newsgroups
import pandas as pd

news_data = fetch_20newsgroups(subset = 'all', random_state = 156)
print(news_data.data[0])
```

Out :

```
From: egreen@east.sun.com (Ed Green - Pixel Cruncher)
Subject: Re: Observation re: helmets
Organization: Sun Microsystems, RTP, NC
Lines: 21
Distribution: world
Reply-To: egreen@east.sun.com
NNTP-Posting-Host: laser.east.sun.com
```

• 피처 벡터화

> 텍스트 분류 실습 – 20 뉴스그룹

```
train_news= fetch_20newsgroups(subset='train',
                               remove=('headers', 'footers', 'quotes'),
                               random_state=156)
X_train = train_news.data
y_train = train_news.target
print(X_train[0])
```

Out :

```
What I did NOT get with my drive (CD300i) is the System Install CD you
listed as #1.  Any ideas about how I can get one?  I bought my IIvx 8/120
from Direct Express in Chicago (no complaints at all -- good price & good
service).

BTW, I've heard that the System Install CD can be used to boot the mac;
however, my drive will NOT accept a CD caddy is the machine is off.  How can
you boot with it then?

--Dave
```

• 피처 벡터화

> 텍스트 분류 실습 – 20 뉴스그룹

```
test_news= fetch_20newsgroups(subset='test',
                              remove=('headers', 'footers','quotes'),
                              random_state=156)
X_test = test_news.data
y_test = test_news.target
print(f'학습 데이터 크기 {len(train_news.data)}, 테스트 데이터 크기 {len(test_news.data)}')
```

Out : **학습 데이터 크기 11314 , 테스트 데이터 크기 7532**

• 피처 벡터화

> 텍스트 분류 실습 – 20 뉴스그룹

```
from sklearn.feature_extraction.text import CountVectorizer

cnt_vect = CountVectorizer()
cnt_vect.fit(X_train)
X_train_cnt_vect = cnt_vect.transform(X_train)
X_test_cnt_vect = cnt_vect.transform(X_test)

print('학습 데이터 Text의 CountVectorizer Shape:', X_train_cnt_vect.shape)
```

Out : 학습 데이터 Text의 CountVectorizer Shape: (11314, 101631)

• 피처 벡터화

> 텍스트 분류 실습 – 20 뉴스그룹

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')

lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train_cnt_vect , y_train)
pred = lr_clf.predict(X_test_cnt_vect)
print('예측 정확도는 {0:.3f}'.format(accuracy_score(y_test,pred)))
```

Out : 예측 정확도는 0.617

• 피처 벡터화

> 텍스트 분류 실습 – 20 뉴스그룹

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vect = TfidfVectorizer()
tfidf_vect.fit(X_train)
X_train_tfidf_vect = tfidf_vect.transform(X_train)
X_test_tfidf_vect = tfidf_vect.transform(X_test)
lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train_tfidf_vect , y_train)
pred = lr_clf.predict(X_test_tfidf_vect)
print('예측 정확도는 {0:.3f}'.format(accuracy_score(y_test ,pred)))
```

Out : 예측 정확도는 0.678

• 피처 벡터화

> 텍스트 분류 실습 – 20 뉴스그룹

```
tfidf_vect = TfidfVectorizer(stop_words='english', ngram_range=(1,2), max_df=300 )
tfidf_vect.fit(X_train)

X_train_tfidf_vect = tfidf_vect.transform(X_train)
X_test_tfidf_vect = tfidf_vect.transform(X_test)
lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train_tfidf_vect , y_train)
pred = lr_clf.predict(X_test_tfidf_vect)
print('예측 정확도는 {0:.3f}'.format(accuracy_score(y_test ,pred)))
```

Out : 예측 정확도는 0.690

• 피처 벡터화

> 텍스트 분류 실습 – 20 뉴스그룹

```
from sklearn.model_selection import GridSearchCV
params = { 'C':[0.01, 0.1, 1, 5, 10]}
grid_cv_lr = GridSearchCV(lr_clf, param_grid=params, cv=3,
                                    scoring='accuracy', verbose=True )
grid_cv_lr.fit(X_train_tfidf_vect , y_train)
print('Logistic Regression best C parameter :', grid_cv_lr.best_params_ )
pred = grid_cv_lr.predict(X_test_tfidf_vect)
print('예측 정확도는 {0:.3f}'.format(accuracy_score(y_test ,pred)))
```

Out : Logistic Regression best C parameter : {'C': 10}

예측 정확도는 0.704

• 피처 벡터화

> 텍스트 분류 실습 – 20 뉴스그룹

```
from sklearn.pipeline import Pipeline

pipeline = Pipeline([('tfidf_vect', TfidfVectorizer(stop_words='english')),
                     ('lr_clf', LogisticRegression(solver='liblinear'))])

params = {'tfidf_vect__ngram_range': [(1,1), (1,2), (1,3)],
          'tfidf_vect__max_df': [100, 300, 700],
          'lr_clf__C': [1, 5, 10]}

grid_cv_pipe = GridSearchCV(pipeline, param_grid=params, cv=3,
                            scoring='accuracy', verbose=True)
```

- 피처 벡터화

> 텍스트 분류 실습 – 20 뉴스그룹

```
grid_cv_pipe.fit(X_train , y_train)
print(grid_cv_pipe.best_params_, grid_cv_pipe.best_score_)


pred = grid_cv_pipe.predict(X_test)
print('예측 정확도는 {0:.3f}'.format(accuracy_score(y_test ,pred)))
```

Out :

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
{'lr_clf__C': 10, 'tfidf_vect__max_df': 700, 'tfidf_vect__ngram_range': (1, 2)}
0.7550828826229531
예측 정확도는 0.702
```

• 한글 텍스트 분석

> 텍스트 분석은 언어의 특성마다 다르게 처리를 해주어 한다.

> 한글은 다양한 조사, 띄어쓰기에 의해 영어 텍스트 처리보다 까다롭다.

> 이를 처리하기 위한 라이브러리로 KoNLPy라는 한글 형태소 패키지가 있다.

> KoNLPy에서 제공하는 패키지로 꼬꼬마(Kkma), 한나눔(Hananum), Komoran, 은전한잎 프로젝트(Mecab), Twitter 등이 존재한다.

> KoNLPy는 C/C++, Java 기반 패키지로 구성되어 있어 설치시 Java 1.7 버전 이상이 설치되어 있어야 한다.

• KoNLPy 설치

> **NLP 라이브러리만 바뀌고 텍스트 분석의 순서와 방법은 동일하다.**

- 텍스트 사전 준비작업

  : 클렌징, 토큰화, 스톱 워드 제거, 철자 수정, 어근 추출 등

- 피처 벡터화 / 추출

  : BOW, Word2Vec 등

- 머신러닝 모델 학습/예측/평가

• 한글 텍스트 분석

> 클렌징

```
with open('./소나기.txt', 'r', encoding='utf8') as f:
    text = f.read()
print(text[:200])
```

Out :

```
소년은 개울가에서 소녀를 보자 곧 윤 초시네 증손녀(曾孫女)딸이라는 걸 알 수
있었다. 소녀는 개울에다 손을 잠그고 물장난을 하고 있는 것이다. 서울서는 이런
개울물을 보지 못하기나 한 듯이.
벌써 며칠째 소녀는, 학교에서 돌아오는 길에 물장난이었다. 그런데, 어제까지 개울
기슭에서 하더니, 오늘은 징검다리 한가운데 앉아서 하고 있다.
소년은 개울둑에 앉아
```

• 한글 텍스트 전처리

> 클렌징

```
import re


compile = re.compile("[^ ㄱ-ㅣ가-힣₩.]+")
text = compile.sub('',text)
print(text[:200])
```

Out : 소년은 개울가에서 소녀를 보자 곧 윤 초시네 증손녀딸이라는 걸 알 수있었다. 소녀는 개울에다 손을 잠그고 물장난을 하고 있는 것이다. 서울서는 이런개울물을 보지 못하기나 한 듯이.벌써 며칠째 소녀는 학교에서 돌아오는 길에 물장난이었다. 그런데 어제까지 개울기슭에서 하더니 오늘은 징검다리 한가운데 앉아서 하고 있다.소년은 개울둑에 앉아 버렸다. 소녀가 비키기

• 한글 텍스트 전처리

> 텍스트 토큰화

```
# 문장 토큰화
import nltk


sentences = nltk.sent_tokenize(text)
print(sentences[:3])
```

Out : ['소년은 개울가에서 소녀를 보자 곧 윤 초시네 증손녀딸이라는 걸 알 수있었다.',
'소녀는 개울에다 손을 잠그고 물장난을 하고 있는 것이다.', '서울서는 이런개울물을
보지 못하기나 한 듯이.벌써 며칠째 소녀는 학교에서 돌아오는 길에 물장난이었다.']

• 한글 텍스트 전처리

> 텍스트 토큰화

```
# 단어 토큰화
from konlpy.tag import Okt
okt = Okt()
words = []
for sentence in sentences:
    word = okt.morphs(sentence)
    words.append(word)
print(words[:2])
```

Out :

```
[['소년', '은', '개울가에서', '소녀', '를', '보자', '곧', '윤', '초시', '네',
'증손녀', '딸', '이라는', '걸', '알', '수', '있었다', '.'], ['소녀', '는', '개
울', '에다', '손', '을', '잠그고', '물장난', '을', '하고', '있는', '것', '이다',
'.']]
```

• 한글 텍스트 전처리

> 텍스트 토큰화

```
test_text = '나는 정말로 파이썬을 좋아한다. 아니 머신러닝을 더 좋아한다.'


print('normalize :', okt.normalize(test_text))
print('morphs    :', okt.morphs(test_text))
print('nouns     :', okt.nouns(test_text))
print('phrases   :', okt.phrases(test_text))
print('pos       :', okt.pos(test_text))
```

Out :

```
normalize : 나는 정말로 파이썬을 좋아한다. 아니 머신러닝을 더 좋아한다.
morphs    : ['나', '는', '정말로', '파이썬', '을', '좋아한다', '.', '아니', '머
신', '러닝', '을', '더', '좋아한다', '.']
nouns     : ['나', '파이썬', '머신', '러닝', '더']
phrases   : ['파이썬', '머신러닝', '머신', '러닝']
pos       : [('나', 'Noun'), ('는', 'Josa'), ('정말로', 'Adverb'), ('파이썬', 'N
oun'), ('을', 'Josa'), ('좋아한다', 'Adjective'), ('.', 'Punctuation'), ('아니',
'Adjective'), ('머신', 'Noun'), ('러닝', 'Noun'), ('을', 'Josa'), ('더', 'Nou
n'), ('좋아한다', 'Adjective'), ('.', 'Punctuation')]
```

• 한글 텍스트 전처리

> 텍스트 토큰화 함수

```
okt = Okt()

def okt_tokenizer(text):
    tokens_ko = okt.morphs(text, stem=True)
    return tokens_ko
word_tokens = tw_tokenizer(text)
print(word_tokens[:20])
```

Out : ['소년', '은', '개다', '소녀', '를', '보다', '곧', '윤', '초시', '네', '증손녀', '딸', '이라는', '걸', '알', '수', '있다', '.', '소녀', '는']

• 한글 텍스트 전처리

> **스톱 워드 제거**
  - KoNLPy에서 제공하지 않는다.

```
with open('./stopword.txt','r',encoding='utf-8') as f:
    word = f.read()
stopwords = word.split('₩n')
print(stopwords[:10])
```

Out : ['않다', '되어다', '되다', '하다', '어떻다', '이렇다', '이다', '어제', '매일', '아']

• 한글 텍스트 전처리

> 스톱 워드 제거

```
filtered_words = []
for word in word_tokens:
    if word not in stopwords:
        filtered_words.append(word)
print(filtered_words[:20])
```

Out :

['소년', '은', '개다', '소녀', '보다', '윤', '초시', '증손녀', '딸', '이라는', '걸', '알', '수', '.', '소녀', '는', '개울', '에다', '손', '잠그다']

• 한글 텍스트 전처리

> 스톱 워드 제거 함수

```
def Stopwords(words, Stopwords = None):
    filtered_words = []
    for word in words:
        if word not in stopwords:
            filtered_words.append(word)
    return filtered_words
filtered_words = Stopwords(word_tokens, stopwords)
print(filtered_words[:20])
```

Out :

['소년', '은', '개다', '소녀', '보다', '윤', '초시', '증손녀', '딸', '이라는', '걸', '알', '수', '.', '소녀', '는', '개울', '에다', '손', '잠그다']

• 한글 피처 벡터화

> 카운트 벡터화

```
from sklearn.feature_extraction.text import CountVectorizer


cnt_vect = CountVectorizer(tokenizer=okt_tokenizer, stop_words=stopwords)
cnt_vect.fit(filtered_words)
words_cnt_vect = cnt_vect.transform(filtered_words)
words_cnt_vect
```

Out : <2276x834 sparse matrix of type '<class 'numpy.int64'>'
with 2308 stored elements in Compressed Sparse Row format>

• 텍스트 분류 실습

> 국민청원 게시판 데이터

```
import pandas as pd

df = pd.read_csv('./petition.csv')
df.head()
```

Out :

	article_id	start	end	answered	votes	category	title	content
0	21	2017-08-19	2017-11-17	0	9	안전/환경	스텔라 데이지호에 대한 제안입니다.	스텔라 데이지호에 대한 제안입니다.\n3월31일 스텔라 데이지호가 침몰하고 5달째가...
1	22	2017-08-19	2017-11-17	0	17	기타	비리제보처를 만들어주세요.	현 정부에 국민들이 가장 원하는 것은 부패척결입니다. 우리 사회에 각종 비리들이 ...
2	23	2017-08-19	2017-09-03	0	0	미래	제2의 개성공단	만일 하시는 대통령님 및 각 부처 장관님,주무관님들 안녕하세요!!\n전남 목포에서 ...
3	24	2017-08-19	2017-08-26	0	53	일자리	공공기관 무조건적인 정규직전환을 반대합니다.	현정부에서 정규직 일자리를 늘리는 것에 찬성합니다. 그런데 공공기관 비정규직들은 인...
4	25	2017-08-19	2017-09-03	0	0	미래	제2의 개성공단	만일 하시는 대통령님 및 각 부처 장관님,주무관님들 안녕하세요!!\n전남 목포에서 ...

• 텍스트 분류 실습

> 국민청원 게시판 데이터

```
from sklearn.preprocessing import LabelEncoder

df_data = df[['content', 'category']]

lb_enc = LabelEncoder()
df_data['category'] = lb_enc.fit_transform(df_data['category'])
df_data.head()
```

Out :

	content	category
0	스텔라 데이지호에 대한 제안입니다.\n3월31일 스텔라 데이지호가 침몰하고 5달째가...	9
1	현 정부에 국민들이 가장 원하는 것은 부패척결입니다. 우리 사회에 각종 비리들이 ...	2
2	만일 하시는 대통령님 및 각 부처 장관님,주무관님들 안녕하세요!!\n전남 목포에서 ...	5
3	현정부에서 정규직 일자리를 늘리는 것에 찬성합니다. 그런데 공공기관 비정규직들은 인...	13
4	만일 하시는 대통령님 및 각 부처 장관님,주무관님들 안녕하세요!!\n전남 목포에서 ...	5

• 텍스트 분류 실습

> 국민청원 게시판 데이터

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =train_test_split(df_data['content'], df_data['category'],
                                                   test_size = 0.2, random_state = 0)
print(f'학습 데이터 수:{len(X_train)}, 평가 데이터 수:{len(X_test)}')
```

Out : 학습 데이터 수:316437, 평가 데이터 수:79110

• 텍스트 분류 실습

> 국민청원 게시판 데이터

```
from sklearn.feature_extraction.text import CountVectorizer


cnt_vect = CountVectorizer(tokenizer = okt_tokenizer, stop_words = stopwords)
cnt_vect.fit(X_train)
X_train_cnt_vect = cnt_vect.transform(X_train)
X_test_cnt_vect = cnt_vect.transform(X_test)
```

• 텍스트 분류 실습

> 국민청원 게시판 데이터

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train_cnt_vect , y_train)
pred = lr_clf.predict(X_test_cnt_vect)
print('예측 정확도는 {0:.3f}'.format(accuracy_score(y_test,pred)))
```

Out : 예측 정확도는 0.617

• 감성 분석

> 문서의 주관적인 감성/의견/감정/기분 등을 파악하기 위한 방법

> 지도 학습과 비지도 학습 둘다 가능하다.

- 지도 학습 : 리뷰 데이터 등 텍스트와 평가 점수가 같이 있어 긍정(1), 부정(0) 등으로 구분한 데이터로 분석
- 비지도 학습 : Lexicon이라는 감성 어휘 사전을 이용하여 텍스트의 긍정적, 부정적 감성 여부를 판단

• 감성 분석

> IMDB 영화 평가 분석

```
import pandas as pd

review_df = pd.read_csv('./labeledTrainData.tsv', header=0, sep="₩t", quoting=3)
review_df.head(3)
```

Out :

	id	sentiment	review
0	"5814_8"	1	"With all this stuff going down at the moment ...
1	"2381_9"	1	"\"The Classic War of the Worlds\" by Timothy ...
2	"7759_3"	0	"The film starts with a manager (Nicholas Bell...

• 감성 분석

> IMDB 영화 평가 분석

```
import re

review_df['review'] = review_df['review'].str.replace('<br />',' ')
review_df['review'] = review_df['review'].apply( lambda x : re.sub("[^a-zA-Z]", " ", x) )
review_df.head(3)
```

Out :

	id	sentiment	review
0	"5814_8"	1	With all this stuff going down at the moment ...
1	"2381_9"	1	The Classic War of the Worlds by Timothy ...
2	"7759_3"	0	The film starts with a manager Nicholas Bell...

• 감성 분석

> IMDB 영화 평가 분석

```
from sklearn.model_selection import train_test_split

class_df = review_df['sentiment']
feature_df = review_df.drop(['id','sentiment'], axis=1)

X_train, X_test, y_train, y_test= train_test_split(feature_df, class_df,
                                                   test_size=0.3, random_state=156)

print(X_train.shape, X_test.shape)
```

Out : (17500, 1) (7500, 1)

• 감성 분석

> IMDB 영화 평가 분석

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_auc_score

pipeline = Pipeline([
    ('cnt_vect', CountVectorizer(stop_words='english', ngram_range=(1,2) )),
    ('lr_clf', LogisticRegression(solver='liblinear', C=10))
])
```

• 감성 분석

> IMDB 영화 평가 분석

```
pipeline.fit(X_train['review'], y_train)
pred = pipeline.predict(X_test['review'])
pred_probs = pipeline.predict_proba(X_test['review'])[:,1]

print(f'예측 정확도는 {accuracy_score(y_test ,pred):.4f}')
print(f'ROC-AUC는 {roc_auc_score(y_test, pred_probs):.4f}')
```

Out : 예측 정확도는 0.8861
ROC-AUC는 0.9503

• 감성 분석

> IMDB 영화 평가 분석

```
from sklearn.feature_extraction.text import TfidfVectorizer
pipeline = Pipeline([
    ('tfidf_vect', TfidfVectorizer(stop_words='english', ngram_range=(1,2) )),
    ('lr_clf', LogisticRegression(solver='liblinear', C=10))])
pipeline.fit(X_train['review'], y_train)
pred = pipeline.predict(X_test['review'])
pred_probs = pipeline.predict_proba(X_test['review'])[:,1]
print(f'예측 정확도는 {accuracy_score(y_test ,pred):.4f}')
print(f'ROC-AUC는 {roc_auc_score(y_test, pred_probs):.4f}')
```

Out : 예측 정확도는 0.8936
ROC-AUC는 0.9598

• 토픽 모델링

> LDA(Latent Dirichlet Analysis)

- 확률 기반으로 토픽이 어떤 비율로 구성되어 있는지 분석

구분	내용
리뷰 1	여긴 삼겹살이랑 냉면 맛집으로 추천
리뷰 2	이집 미남 사장님 친절하시고 최고
리뷰 3	사장님 친절하시고 삼겹살이랑 된장찌개 맛집이라 추천

토픽 구분	내용
토픽 A (음식 Topic)	**삼겹살 25%, 냉면 12.5%, 맛집 25%, 추천 25%, 된장찌개 12.5%,** 미남 0%, 사장님 0%, 친절 0%, 최고 0%
토픽 B (사장님 Topic)	삼겹살 0%, 냉면 0%, 맛집 0%, 추천 0%, 된장찌개 0%, **미남 16.5%, 사장님 33%, 친절 33%, 최고 16.5%**

• 토픽 모델링

> LDA 모델링 프로세스

1. 토픽 개수, 단어 개수 결정 : 사용자가 직접 정함
2. 단어의 토픽 할당 : 확률 분포를 기반으로 문서별 토픽 할당
3. 모든 단에 대해 토픽 재할당(반복)

문서1	삼겹살	냉면	맛집	추천
토픽	A	A	A	A
문서2	미남	사장님	친절	최고
토픽	B	B	B	B

문서3	사장님	친절	삼겹살	된장찌개	맛집	추천
토픽	B	B	A	???	A	A

A

• 토픽 모델링

> LDA(Latent Dirichlet Analysis)

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation

cats = ['rec.motorcycles', 'rec.sport.baseball', 'comp.graphics',
        'comp.windows.x', 'talk.politics.mideast',
        'soc.religion.christian', 'sci.electronics', 'sci.med'   ]

news_df= fetch_20newsgroups(subset='all', remove=('headers', 'footers', 'quotes'),
                                        categories=cats, random_state=0)
```

• 토픽 모델링

> LDA(Latent Dirichlet Analysis)

```
count_vect = CountVectorizer(max_df=0.95,
                             max_features=1000, min_df=2,
                             stop_words='english',
                             token_pattern = '[a-zA-Z]+',
                             ngram_range=(1,2))

feat_vect = count_vect.fit_transform(news_df.data)
print('CountVectorizer Shape:', feat_vect.shape)
```

Out : CountVectorizer Shape: (7862, 1000)

• 토픽 모델링

> LDA(Latent Dirichlet Analysis)

```
lda = LatentDirichletAllocation(n_components=8, random_state=0)
lda.fit(feat_vect)


print(lda.components_.shape)
print(lda.components_)
```

Out : (8, 1000)
[[4.39641624e+01 6.82089542e+01 2.06222844e-01 ... 1.55382838e-01
  2.84174833e+01 1.25024645e-01]
 [3.26389559e+01 8.26641485e+01 9.61820871e+01 ... 2.21062877e+01
  3.75472202e+00 1.11229866e+00]
 [1.25288452e-01 1.25089995e-01 8.43915268e+00 ... 1.25044249e-01
  1.25095795e-01 6.91762967e+00]
 ...
 [6.42644668e+00 9.50355854e+00 1.31246647e+02 ... 1.36592523e-01
  1.97260097e+01 8.24344804e+02]
 [6.34931733e+01 1.50853781e+02 1.25076392e-01 ... 9.04989065e+00
  1.18895683e+02 1.25019502e-01]
 [3.41017375e+00 1.46706347e+02 6.12373077e+01 ... 1.25043590e-01
  1.04626629e+00 1.25091960e-01]]

• 토픽 모델링

> LDA(Latent Dirichlet Analysis)

```
def display_topics(model, feature_names, no_top_words):
    for topic_index, topic in enumerate(model.components_):
        print('Topic #',topic_index)
        topic_word_indexes = topic.argsort()[::-1]
        top_indexes=topic_word_indexes[:no_top_words]
        feature_concat = ' '.join([feature_names[i] for i in top_indexes])
        print(feature_concat)

feature_names = count_vect.get_feature_names_out()
display_topics(lda, feature_names, 15)
```

• 토픽 모델링

> LDA(Latent Dirichlet Analysis)

Out :

```
Topic # 0
s god people t think does church believe say know question just christian like christians
Topic # 1
graphics edu mail information s software image available data d ftp send computer thanks e
Topic # 2
x x x dos file n c entry output s program windows n x build int char
Topic # 3
israel s jews turkish israeli jewish people government armenian new arab state university war turkey
Topic # 4
god s armenians people said armenian t jesus father children man son christ azerbaijan lord
Topic # 5
r x m edu s b t o z jpeg file p c h v
Topic # 6
t s don don t just like know m think good time year ve people didn
Topic # 7
use window t s using like problem does used work display need just want application
```

christian

computer
: graphics

computer
: windows.x

• 토픽 모델링

> 한글 텍스트를 통한 LDA 실습

```
import pandas as pd
df = pd.read_csv('./petition.csv')


cats = ['정치개혁', '인권/성평등', '안전/환경', '교통/건축/국토', '육아/교육']
df_cats = df[df['category'].isin(cats)]
df_samples = df_cats.sample(frac=0.05, random_state = 0)
df_samples['category'].value_counts()
```

Out :

```
category
정치개혁        3204
인권/성평등      1780
안전/환경        1462
교통/건축/국토     1408
육아/교육        1251
Name: count, dtype: int64
```

• 토픽 모델링

> 한글 텍스트를 통한 LDA 실습

```
import re
df_samples['content'] = df_samples['content'].apply(lambda x : re.sub('[^ ㄱ-ㅣ가-힣]+',
                                                                      '', x))
from konlpy.tag import Okt
okt = Okt()
def okt_tokenizer(text):
    tokens_ko = okt.morphs(text, stem = True)
    return tokens_ko

with open('./stopword.txt','r',encoding='utf-8') as f:
    word = f.read()
stopwords = word.split('₩n')
```

• 토픽 모델링

> 한글 텍스트를 통한 LDA 실습

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vect = TfidfVectorizer(max_df = 0.9,
                             max_features = 1000,
                             min_df = 2,
                             ngram_range = (1, 2),
                             tokenizer = okt_tokenizer,
                             stop_words = stopwords)
df_tfidf_vect = tfidf_vect.fit_transform(df_samples['content'])
df_tfidf_vect
```

• 토픽 모델링

> 한글 텍스트를 통한 LDA 실습

```
from sklearn.decomposition import LatentDirichletAllocation


lda = LatentDirichletAllocation(n_components = 5, random_state = 0)
lda.fit(df_tfidf_vect)
print(lda.components_.shape)
print(lda.components_)
```

Out :
```
(5, 1000)
[[13.44219722 28.78301737 83.66396296 ... 18.04596276 25.10402458
  37.19120501]
 [ 0.20031814  2.30313409 14.2507791  ... 10.71921902  3.06712658
   4.96343437]
 [ 0.27582174  1.76417409  7.87763301 ...  1.58944024  5.58179836
   5.56486215]
 [24.45821272  8.91192152 21.56846874 ...  7.90590962  2.90571889
   3.10954766]
 [ 0.20028978  0.20173748  2.37864408 ...  3.58021781  0.20071726
   1.67322463]]
```

• 토픽 모델링

> 한글 텍스트를 통한 LDA 실습

```
feature_names = tfidf_vect.get_feature_names_out()
display_topics(lda, feature_names, 10)
```

Out :

```
Topic # 0
은 도 는 없다 적 보다 한 생각 아니다 사람
Topic # 1
국민 국회의원 청소년 법 은 폐지 국회 의원 나라 당
Topic # 2
이명박 출국금지 학생 학교 교사 금지 교육 해주다 유치원 청원
Topic # 3
은 처벌 는 한 사건 피해자 주택 부동산 적 도
Topic # 4
문재인 조두순 대통령 적폐 반대 청산 추다 정부 적폐 청산 국민
```

정치개혁

정치개혁 + 육아/교육

정치개혁 + 교통/건축/국토

정치개혁

• 토픽 모델링

> 한글 텍스트를 통한 LDA 실습

```
# 하나의 주제에서 주요 주제 뽑기
cats = ['육아/교육']

df_cats = df[df['category'].isin(cats)]
df_samples = df_cats.sample(frac=0.5, random_state = 0)

df_samples['content'] = df_samples['content'].apply(lambda x : re.sub('[^ ㄱ-ㅣ가-힣]+',
                                                                      '', x))
```

• 토픽 모델링

> 한글 텍스트를 통한 LDA 실습

```
tfidf_vect = TfidfVectorizer(max_df = 0.9,
                             max_features = 1000,
                             min_df = 2,
                             ngram_range = (1, 2),
                             tokenizer = okt_tokenizer,
                             stop_words = stopwords)
df_tfidf_vect = tfidf_vect.fit_transform(df_samples['content'])
df_tfidf_vect
```

Out :

```
<12768x1000 sparse matrix of type '<class 'numpy.float64'>'
        with 736161 stored elements in Compressed Sparse Row format>
```

• 토픽 모델링

> 한글 텍스트를 통한 LDA 실습

```
lda = LatentDirichletAllocation(n_components = 5, random_state = 0)
lda.fit(df_tfidf_vect)

feature_names = tfidf_vect.get_feature_names_out()
display_topics(lda, feature_names, 15)
```

Out :

```
Topic # 0
학생 은 는 학교 교육 수능 도 대학 수 시험 공부 생각 영어 적 한
Topic # 1
어린이집 도 은 는 없다 보육 받다 교사 키우다 보다 엄마 부모 한 돌보다 지원
Topic # 2
폐지 청소년 법 소년법 보호 보호 법 청소년 보호 해주다 처벌 부산 범죄 청원 동의 반대 사건
Topic # 3
교사 유치원 교육 은 학교 는 적 사립 한 도 수 하고 비리 사립 유치원 없다
Topic # 4
처벌 은 도 는 교복 사건 해주다 없다 보다 생각 아니다 피해자 학생 너무 법
```

• 토픽 모델링

> 한글 텍스트를 통한 LDA 실습

학생 은 는 학교 교육 수능 도 대학 수 시험 공부 생각 영어 적 한
> 학생, 학교 교육, 수능, 시험 공부, 영어
어린이집 도 은 는 없다 보육 받다 교사 키우다 보다 엄마 부모 한 돌보다 지원
> 어린이집 보육 교사 없다, 엄마 한부모 돌보다 지원 없다
폐지 청소년 법 소년법 보호 보호 법 청소년 보호 해주다 처벌 부산 범죄 청원 동의 반대 사건
> 청소년법, 소년법 폐지, 청소년 처벌 청원 동의, 반대
교사 유치원 교육 은 학교 는 적 사립 한 도 수 하고 비리 사립 유치원 없다
> 사립 학교 교사 비리, 유치원 없다
처벌 은 도 는 교복 사건 해주다 없다 보다 생각 아니다 피해자 학생 너무 법
> 피해자 생각, 학생 처벌

• 문서 군집화

> KMeans 군집화

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd

df = pd.read_csv('./petition.csv')

cats = ['안전/환경', '교통/건축/국토', '육아/교육', '일자리']
df_cats = df[df['category'].isin(cats)]
df_samples = df_cats.sample(frac=0.1, random_state = 0)
df_samples.shape
```

Out : (10758, 8)

• 문서 군집화

> KMeans 군집화

```
from sklearn.preprocessing import LabelEncoder

lb_enc = LabelEncoder()
lb_enc.fit(df_samples['category'])
df_samples['category']=lb_enc.transform(df_samples['category'])

df_samples['content'] = df_samples['content'].apply(lambda x : re.sub('[^ ㄱ-ㅣ가-힣]+',
                                                                      '', x))
```

• 문서 군집화

> KMeans 군집화

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vect = TfidfVectorizer(max_df = 0.85,
                             min_df = 2,
                             ngram_range = (1, 2),
                             tokenizer = okt_tokenizer,
                             stop_words = stopwords)
df_tfidf_vect = tfidf_vect.fit_transform(df_samples['content'])
df_tfidf_vect
```

Out :
```
<10758x169450 sparse matrix of type '<class 'numpy.float64'>'
        with 1592015 stored elements in Compressed Sparse Row format>
```

• 문서 군집화

> KMeans 군집화

```
from sklearn.cluster import KMeans

km_cluster = KMeans(n_clusters = 4, max_iter = 10000, random_state = 0)
km_cluster.fit(df_tfidf_vect)
cluster_label = km_cluster.labels_
clust_df = df_samples[['content', 'category']]
clust_df['cluster'] = cluster_label
clust_df.head(3)
```

Out :

	content	category	cluster
323169	지난 사학분쟁조정위원회사분위는 제차 회의에서 영광학원대구대 정상화 관련하여 임시이...	2	1
273077	경제사정이어렵습니다서민소상공인자영업자비정규직이들에게추석때경제사면음주교통벌점사면을청원...	0	1
145850	청소년들의 음주와 흡연으로 인해 많은 사건 사고가 일어나고 있습니다 나이에 맞지 않...	1	1

• 문서 군집화

> KMeans 군집화

```
clust_df.groupby(['category', 'cluster'])['content'].count()
```

Out :

```
category  cluster
0         1          2464
          2           295
1         0            73
          1          2531
          2           404
          3            95
2         0             9
          1          1048
          2          1415
          3           112
3         0             1
          1          1052
          2          1257
          3             2
Name: content, dtype: int64
```

• 문서 군집화

> KMeans 군집화

```
clust_df[clust_df['cluster'] == 1]
```

Out :

	content	category	cluster
323169	지난 사학분쟁조정위원회사분위는 제차 회의에서 영광학원대구대 정상화 관련하여 임시이...	2	1
273077	경제사정이어렵습니다서민소상공인자영업자비정규직이들에게추석때경제사면음주교통벌점사면을청원...	0	1
145850	청소년들의 음주와 흡연으로 인해 많은 사건 사고가 일어나고 있습니다 나이에 맞지 않...	1	1
102333	저희 남편도 덤프트럭으로 건설현장에서 일을하고 있습니다한달일해서 번돈으로 덤프할부도...	0	1
180393	현재의 여경은 도무지 믿을수없는 여경들로 이루어져 있다고 여겨집니다여성가족부가 뭐라...	1	1
...	...	...	...
309768	음주운전을 한 당사자는 살인을 동조한 것입니다	0	1
1503	공공부문의 파견직용역업체에서 종사하고있는 원청업체기준에선 비정규직 소속된 용역업체에...	3	1
231467	여느 대 직장입니다 촛불집회도 바라만 봤었지 요즘처럼 반대집회에 나가고 싶은 마음이...	1	1
192292	올해월입주를 앞두고있습니다하지만 지금시점 입주를망설이고있습니다그이유는 분양당시 허위...	0	1
179012	제 아들의 일입니다 아들은 살 입니다 아들이 외할아버지로 부터 학대를 당했습니다아이...	2	1

• 문서 군집화

> KMeans 군집화

```
def get_cluster_details(cluster_model, cluster_data, feature_names, clusters_num, top_n_features=10):
    cluster_details = {}

    centroid_feature_ordered_ind = cluster_model.cluster_centers_.argsort()[:,::-1]
    for cluster_num in range(clusters_num):
        cluster_details[cluster_num] = {}
        cluster_details[cluster_num]['cluster'] = cluster_num

        top_feature_indexes = centroid_feature_ordered_ind[cluster_num, :top_n_features]
        top_features = [ feature_names[ind] for ind in top_feature_indexes ]
        top_feature_values = cluster_model.cluster_centers_[cluster_num, top_feature_indexes].tolist()

        cluster_details[cluster_num]['top_features'] = top_features
        cluster_details[cluster_num]['top_features_value'] = top_feature_values

    return cluster_details
```

• 문서 군집화

> KMeans 군집화

```
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('==============================')

clust_centers = km_cluster.cluster_centers_
feature_names = tfidf_vect.get_feature_names_out()
cluster_details = get_cluster_details(cluster_model=km_cluster,
                                      cluster_data=clust_df,
                                      feature_names=feature_names,
                                      clusters_num=4,
                                      top_n_features=10 )
print_cluster_details(cluster_details)
```

• 문서 군집화

> KMeans 군집화

Out :

```
####### Cluster 0
Top features: ['조두순 추다', '조두순', '반대', '추다', '추다 반대', '추다 소', '소 반대',
'소', '반대 추다', '추다 제발']
==========================================
####### Cluster 1
Top features: ['은', '도', '는', '없다', '한', '사람', '국민', '보다', '적', '하고']
==========================================
####### Cluster 2
Top features: ['학생', '교사', '학교', '교육', '은', '는', '도', '어린이집', '수', '유치원']
==========================================
####### Cluster 3
Top features: ['청소년', '폐지', '법 폐지', '법', '청소년 보호', '폐지 해주다', '보호 법',
'보호', '소년법', '청소년 법']
==========================================
```

• 유사도 분석

> 유사도란?

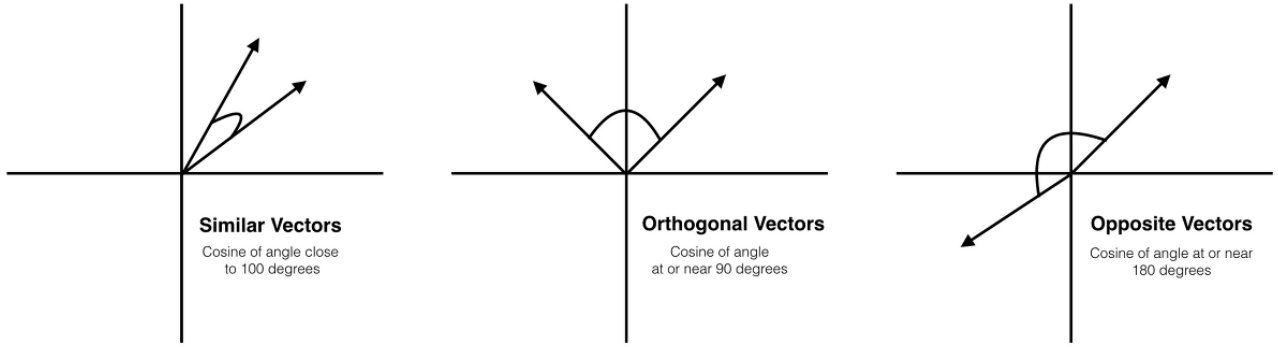
- 객체 간의 물리적 거리를 나타내는 정도

> 유사도 측정 방법

- 유클리디안 거리 측정
- 코사인 유사도
- 피어슨 유사도
- 맨해튼 거리

• 유사도 분석

> 코사인 유사도



- 두 벡터의 내적 값은 벡터 크기에 코사인 각도를 곱한 값

$$A \cdot B = \|A\|\|B\| \cos\theta$$

> 코사인 유사도 측정식

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

• 유사도 분석

> 코사인 유사도 실습

```
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
```

• 유사도 분석

> 코사인 유사도 실습

```
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)
```

Out : (3, 18)

• 유사도 분석

> 코사인 유사도 실습

```
feature_vect_array = feature_vect_simple.toarray()

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

similarity_simple = cos_similarity(vect1, vect2 )
print('문장1,문장2 코사인유사도:{0:.3f}'.format(similarity_simple))
```

Out : 문장1,문장2 코사인유사도:0.402

• 유사도 분석

> 코사인 유사도 실습

```
vect1 = np.array(feature_vect_array[0]).reshape(-1,)
vect3 = np.array(feature_vect_array[2]).reshape(-1,)
similarity_simple = cos_similarity(vect1, vect3 )
print('문장1,문장3 코사인유사도:{0:.3f}'.format(similarity_simple))

vect2 = np.array(feature_vect_array[1]).reshape(-1,)
vect3 = np.array(feature_vect_array[2]).reshape(-1,)
similarity_simple = cos_similarity(vect2, vect3 )
print('문장2,문장3 코사인유사도:{0:.3f}'.format(similarity_simple))
```

Out : 문장1,문장3 코사인유사도:0.404
문장2,문장3 코사인유사도:0.456

• 유사도 분석

> 코사인 유사도 실습

```
from sklearn.metrics.pairwise import cosine_similarity


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

Out :

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

• 유사도 분석

> 실제 문서의 유사도 측정

```
clust2_df = clust_df[clust_df['cluster'] == 2]
clust2_df.head()
```

Out :

	content	category	cluster
349821	제주도에사는 개월 개월 두아이를 둔 엄마입니다두아이를 혼진 키우고 있어 작은애가 태어...	2	2
102333	저희 남편도 덤프트럭으로 건설현장에서 일을하고 있습니다한달일해서 번 돈으로 덤프할부도...	0	2
183884	노동 의욕이 있고 노동 능력이 있으나 마땅한 노동 기회가 주어지지 않아 일자리를 갖...	3	2
123536	어머니께서 이마트에서 일하십니다정확히 말하면 이마트 내 푸드점 하청입니다마트 안에서...	3	2
317640	대한민국 부모님들 사교육 때문에 금전적으로 힘드신 분들이 많을것입니다 학원을 안보내...	2	2

• 유사도 분석

> 실제 문서의 유사도 측정

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vect = TfidfVectorizer(max_df = 0.85,
                             min_df = 2,
                             ngram_range = (1, 2),
                             tokenizer = okt_tokenizer,
                             stop_words = stopwords)
df_tfidf_vect = tfidf_vect.fit_transform(clust2_df['content'])
df_tfidf_vect
```

• 유사도 분석

> 실제 문서의 유사도 측정

```
from sklearn.metrics.pairwise import cosine_similarity

similarity_pair = cosine_similarity(df_tfidf_vect[0], df_tfidf_vect)
print(similarity_pair)

clust2_df['similarity'] = similarity_pair.reshape(-1)
clust2_sorted = clust2_df.sort_values('similarity', ascending=False)
clust2_sorted.reset_index(inplace=True)
clust2_sorted['index'] = clust2_sorted['index'].astype(str)
clust2_sorted.head()
```

• 유사도 분석

> 실제 문서의 유사도 측정

Out :

```
[[1.         0.02604967 0.0129567  ... 0.02777904 0.01207334 0.03470284]]
```

	index	content	category	cluster	similarity
0	349821	제주도에사는 개월 개월 두아이를 둔 엄마입니다두아이를 혼진 키우고 있어작은애가 태어...	2	2	1.000000
1	322734	아직도 어린집비리 폭행사건은 연이어 발생하고 있습니다심각한 사회 문제로 자리를 잡았...	2	2	0.165662
2	173720	저는살아이를 키우는 부모입니다이번에 처음으로 어린이집에 보내게되었습니다저는 일을 안...	2	2	0.162593
3	250985	살 살 아이둘을 키우는 워킹맘입니다매일 어린이집 폭력문제 차량사고방금은 인분 만...	2	2	0.157038
4	292576	안녕하세요 저는 살 두아이에 엄마입니다현재 첫째 아이가 어린이집에 다니고 있지만하루...	2	2	0.148846

• 유사도 분석

> 실제 문서의 유사도 측정

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt


plt.figure(figsize=(20,10))
sns.barplot(x = clust2_sorted.iloc[1:11]['similarity'], y = clust2_sorted.iloc[1:11]['index'])
plt.xlim(0,0.2)
plt.title('clust2 similarity')
```

• 유사도 분석

> 실제 문서의 유사도 측정

Out :