**# # 패키지**

import numpy as np

import pandas as pd

import json

import datetime as dt

import matplotlib.pyplot as plt

import matplotlib.ticker as ticker

import matplotlib.cm as cm

import math

get\_ipython().run\_line\_magic('matplotlib', 'inline')

import seaborn as sns

import seaborn as sns

plt.style.use('seaborn-whitegrid')

import missingno

import re

import glob

import os

from scipy import stats

from scipy.integrate import trapz

import missingno as msno

import sys

import warnings

warnings.filterwarnings('ignore')

pd.set\_option('max\_columns', 10, 'max\_rows', 5, 'max\_colwidth', 10)

# 전처리

from nltk import sent\_tokenize

from nltk import word\_tokenize

**# # 데이터 읽기**

# # 텍스트 정규화

**# ## 텍스트 토큰화**

**# ### 문장 토큰화**

from nltk import sent\_tokenize

text\_sample = 'The Matrix is everywhere its all around us, here even in this room. You can see it out your window or on your television. You feel it when you go to work, or go to church or pay your taxes.'

sentences = sent\_tokenize(text=text\_sample)

print(type(sentences),len(sentences))

print(sentences)

**# ### 단어 토큰화**

from nltk import word\_tokenize

sentence = "The Matrix is everywhere its all around us, here even in this room."

words = word\_tokenize(sentence)

print(type(words), len(words))

print(words)

**# ### 여러 문장들에 대한 단어 토큰화**

from nltk import word\_tokenize, sent\_tokenize

#여러개의 문장으로 된 입력 데이터를 문장별로 단어 토큰화 만드는 함수 생성

def tokenize\_text(text):

# 문장별로 분리 토큰

sentences = sent\_tokenize(text)

# 분리된 문장별 단어 토큰화

word\_tokens = [word\_tokenize(sentence) for sentence in sentences]

return word\_tokens

#여러 문장들에 대해 문장별 단어 토큰화 수행.

word\_tokens = tokenize\_text(text\_sample)

print(type(word\_tokens),len(word\_tokens))

print(word\_tokens)

**# ### n-gram**

from nltk import ngrams

sentence = "The Matrix is everywhere its all around us, here even in this room."

words = word\_tokenize(sentence)

all\_ngrams = ngrams(words, 2)

ngrams = [ngram for ngram in all\_ngrams]

print(ngrams)

**# ## Stopwords 제거**

import nltk

stopwords = nltk.corpus.stopwords.words('english')

all\_tokens = []

# 위 예제의 3개의 문장별로 얻은 word\_tokens list 에 대해 stop word 제거 Loop

for sentence in word\_tokens:

filtered\_words=[]

# 개별 문장별로 tokenize된 sentence list에 대해 stop word 제거 Loop

for word in sentence:

#소문자로 모두 변환합니다.

word = word.lower()

# tokenize 된 개별 word가 stop words 들의 단어에 포함되지 않으면 word\_tokens에 추가

if word not in stopwords:

filtered\_words.append(word)

all\_tokens.append(filtered\_words)

print(all\_tokens)

# ### Stemming과 Lemmatization

from nltk.stem import LancasterStemmer

stemmer = LancasterStemmer()

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

print(stemmer.stem('amusing'),stemmer.stem('amuses'),stemmer.stem('amused'))

print(stemmer.stem('happier'),stemmer.stem('happiest'))

print(stemmer.stem('fancier'),stemmer.stem('fanciest'))

from nltk.stem import WordNetLemmatizer

lemma = WordNetLemmatizer()

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

print(lemma.lemmatize('happier','a'),lemma.lemmatize('happiest','a'))

print(lemma.lemmatize('fancier','a'),lemma.lemmatize('fanciest','a'))

**# ## BOW(Bag Of Words)**

# ### 사이킷런 CountVectorizer 테스트

text\_sample\_01 = 'The Matrix is everywhere its all around us, here even in this room. You can see it out your window or on your television. You feel it when you go to work, or go to church or pay your taxes.'

text\_sample\_02 = 'You take the blue pill and the story ends. You wake in your bed and you believe whatever you want to believe You take the red pill and you stay in Wonderland and I show you how deep the rabbit-hole goes.'

text=[]

text.append(text\_sample\_01); text.append(text\_sample\_02)

print(text,"\n", len(text))

**# ### CountVectorizer객체 생성 후 fit(), transform()으로 텍스트에 대한 feature vectorization 수행**

from sklearn.feature\_extraction.text import CountVectorizer

# Count Vectorization으로 feature extraction 변환 수행.

cnt\_vect = CountVectorizer()

cnt\_vect.fit(text)

ftr\_vect = cnt\_vect.transform(text)

# ### 피처 벡터화 후 데이터 유형 및 여러 속성 확인

print(type(ftr\_vect), ftr\_vect.shape)

print(ftr\_vect)

print(cnt\_vect.vocabulary\_)

cnt\_vect = CountVectorizer(max\_features=5, stop\_words='english')

cnt\_vect.fit(text)

ftr\_vect = cnt\_vect.transform(text)

print(type(ftr\_vect), ftr\_vect.shape)

print(cnt\_vect.vocabulary\_)

**# ### ngram\_range 확인**

cnt\_vect = CountVectorizer(ngram\_range=(1,3))

cnt\_vect.fit(text)

ftr\_vect = cnt\_vect.transform(text)

print(type(ftr\_vect), ftr\_vect.shape)

print(cnt\_vect.vocabulary\_)

**# # 문서 유사도 측정 방법 – 코사인 유사도**

**# ## 코사인 유사도 반환 함수 생성**

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

**# ## TF-IDF 벡터화 후 코사인 유사도 비교**

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)

print(type(feature\_vect\_simple))

# TFidfVectorizer로 transform()한 결과는 Sparse Matrix이므로 Dense Matrix로 변환.

feature\_vect\_dense = feature\_vect\_simple.todense()

#첫번째 문장과 두번째 문장의 feature vector 추출

vect1 = np.array(feature\_vect\_dense[0]).reshape(-1,)

vect2 = np.array(feature\_vect\_dense[1]).reshape(-1,)

#첫번째 문장과 두번째 문장의 feature vector로 두개 문장의 Cosine 유사도 추출

similarity\_simple = cos\_similarity(vect1, vect2 )

print('문장 1, 문장 2 Cosine 유사도: {0:.3f}'.format(similarity\_simple))

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))

**# ## 사이킷런의 cosine\_similarity()함수를 이용하여 비교**

from sklearn.metrics.pairwise import cosine\_similarity

similarity\_simple\_pair = cosine\_similarity(feature\_vect\_simple[0] , feature\_vect\_simple)

print(similarity\_simple\_pair)

from sklearn.metrics.pairwise import cosine\_similarity

similarity\_simple\_pair = cosine\_similarity(feature\_vect\_simple[0] , feature\_vect\_simple[1:])

print(similarity\_simple\_pair)

similarity\_simple\_pair = cosine\_similarity(feature\_vect\_simple , feature\_vect\_simple)

print(similarity\_simple\_pair)

print('shape:',similarity\_simple\_pair.shape)