URL : <https://colab.research.google.com/drive/1g94d9IKhc0VS3pxxJdQKsBhK_KqMvzhS>

import pandas as pd # 판다스 불러오기

df = pd.read\_csv("Naver\_review.csv") # 데이터프레임 정의

df

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.decomposition import LatentDirichletAllocation

# 데이터 로드

file\_path = '/content/Naver\_review.csv'

data = pd.read\_csv(file\_path)

# 데이터 100개만 사용

data = data.head(100)

# 텍스트 전처리 함수 (이전과 동일)

import re

def preprocess\_text(text):

    text = re.sub(r'[^\w\s]', '', str(text))  # 특수문자 제거, NaN 처리 추가

    text = re.sub(r'\d+', '', text)       # 숫자 제거

    text = text.lower()                    # 소문자 변환

    return text

# 텍스트 데이터 전처리

data['Cleaned\_Review'] = data['Review'].apply(preprocess\_text)

# TF-IDF 벡터화 (max\_features는 필요에 따라 조절)

vectorizer = TfidfVectorizer(max\_features=1000, stop\_words='english') # max\_features를 추가해 단어 수를 제한

X = vectorizer.fit\_transform(data['Cleaned\_Review'])

# LDA 모델 설정 및 학습 (n\_components는 토픽 수)

lda = LatentDirichletAllocation(n\_components=5, random\_state=42) # 토픽 수를 5로 설정

lda.fit(X)

# 결과 확인

print("TF-IDF 행렬 크기:", X.shape)

print("LDA 모델 완료")

# 토픽별 주요 단어 확인 함수 (선택 사항)

def display\_topics(model, feature\_names, no\_top\_words):

    for topic\_idx, topic in enumerate(model.components\_):

        print(f"Topic {topic\_idx}:")

        print(", ".join([feature\_names[i]

                        for i in topic.argsort()[:-no\_top\_words - 1:-1]]))

no\_top\_words = 10 # 각 토픽에서 상위 10개 단어 출력

display\_topics(lda, vectorizer.get\_feature\_names\_out(), no\_top\_words)

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.decomposition import LatentDirichletAllocation

from wordcloud import WordCloud

import matplotlib.pyplot as plt

import numpy as np

# 폰트 다운로드 및 설정

!sudo apt-get install -y fonts-nanum

!sudo fc-cache -fv

!rm ~/.cache/matplotlib -rf

plt.rc('font', family='NanumBarunGothic')

# 데이터 로드

file\_path = '/content/Naver\_review.csv'

data = pd.read\_csv(file\_path)

# 데이터 100개만 사용

data = data.head(100)

# 텍스트 전처리 함수 (이전과 동일)

import re

def preprocess\_text(text):

    text = re.sub(r'[^\w\s]', '', str(text))  # 특수문자 제거, NaN 처리 추가

    text = re.sub(r'\d+', '', text)       # 숫자 제거

    text = text.lower()                    # 소문자 변환

    return text

# 텍스트 데이터 전처리

data['Cleaned\_Review'] = data['Review'].apply(preprocess\_text)

# TF-IDF 벡터화 (max\_features는 필요에 따라 조절)

vectorizer = TfidfVectorizer(max\_features=1000, stop\_words='english') # max\_features를 추가해 단어 수를 제한

X = vectorizer.fit\_transform(data['Cleaned\_Review'])

# LDA 모델 설정 및 학습 (n\_components는 토픽 수)

lda = LatentDirichletAllocation(n\_components=5, random\_state=42) # 토픽 수를 5로 설정

lda.fit(X)

# 단어 사전 생성

feature\_names = vectorizer.get\_feature\_names\_out()

# 워드클라우드 생성 함수

def plot\_wordcloud(topic, ax):

    # 토픽의 단어 중요도 가져오기

    word\_importances = topic

    # 단어와 중요도 쌍 만들기

    word\_importance\_pairs = dict(zip(feature\_names, word\_importances))

    # 워드클라우드 생성

    wordcloud = WordCloud(width=800, height=400,

                          background\_color='white', font\_path='/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf').generate\_from\_frequencies(word\_importance\_pairs)

    # 플롯에 워드클라우드 표시

    ax.imshow(wordcloud, interpolation='bilinear')

    ax.axis('off')

# 각 토픽에 대한 워드클라우드 생성 및 표시

fig, axes = plt.subplots(1, 5, figsize=(20, 4)) # 1행 5열 서브플롯 생성

for i, topic in enumerate(lda.components\_):

    plot\_wordcloud(topic, axes[i])

    axes[i].set\_title(f"Topic {i+1}") # 각 서브플롯에 토픽 제목 설정

plt.tight\_layout()

plt.show()

# Topic 1 : 아기 종합용품 리뷰들 (악플 포함)

# Topic 2 : 카페 및 가게 종합용품 리뷰들 (악플 포함)

# Topic 3 : 아기 유모차 용품 리뷰들 (악플 포함)

# Topic 4 : 애견 식품 및 놀이 용품 리뷰들 (악플 포함)

# Topic 5 : 보온 및 발열 기능 옷/바지 용품 리뷰들 (악플 포함)

# 필요한 라이브러리 불러오기

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import f1\_score, classification\_report

from sklearn.preprocessing import LabelEncoder

import re

# 데이터 로드

file\_path = '/content/Naver\_review.csv'

data = pd.read\_csv(file\_path)

# 데이터 100개만 사용

data = data.head(100)

# 텍스트 전처리 함수 (이전과 동일)

def preprocess\_text(text):

    text = re.sub(r'[^\w\s]', '', str(text))  # 특수문자 제거, NaN 처리 추가

    text = re.sub(r'\d+', '', text)       # 숫자 제거

    text = text.lower()                    # 소문자 변환

    return text

# 텍스트 데이터 전처리

data['Cleaned\_Review'] = data['Review'].apply(preprocess\_text)

# 레이블 인코딩

label\_encoder = LabelEncoder()

data['Encoded\_Label'] = label\_encoder.fit\_transform(data['Label'])

# 데이터 분리

X = data['Cleaned\_Review']

y = data['Encoded\_Label']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# TF-IDF 벡터화

vectorizer = TfidfVectorizer(max\_features=1000, stop\_words='english')

X\_train\_tfidf = vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = vectorizer.transform(X\_test)

# 로지스틱 회귀 모델 학습

model = LogisticRegression(random\_state=42, solver='liblinear') # solver 추가

model.fit(X\_train\_tfidf, y\_train)

# 모델 예측

y\_pred = model.predict(X\_test\_tfidf)

# F1 스코어 평가 및 출력

f1 = f1\_score(y\_test, y\_pred, average='weighted')

print("-------------------- F1 스코어 평가 결과 --------------------")

print(f"가중 평균 F1 스코어: {f1:.4f}") # 소수점 네 자리까지 출력

print("----------------------------------------------------------\n")

# 분류 보고서 출력

print("-------------------- 분류 보고서 --------------------")

print(classification\_report(y\_test, y\_pred))

print("----------------------------------------------------------")