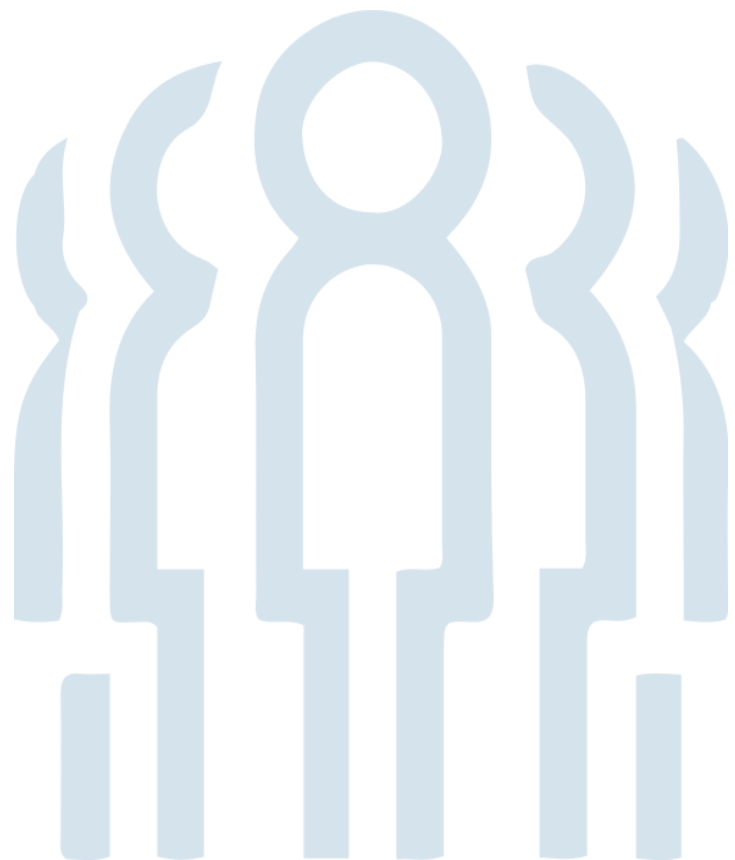


Netflix TextMining

2 조 (오경민, 이대한, 임성훈, 조혜진)

CONTENTS



Chapter 01 [감성 분석] 리뷰 데이터로 감성 예측하기

Chapter 02 Netflix 뉴스 데이터 텍스트의 감성분석하기

Chapter 03 Netflix 뉴스 텍스트에서 토픽 분석하기

Chapter

01

[감성 분석] 리뷰 데이터로 감성 예측하기

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 목표설정

- 예시 데이터에 텍스트 마이닝의 감정 분석 기술을 사용하여 다양한 감성 분석 모델을 구축한 뒤 최적의 모델을 설정 후 새로운 데이터에 대한 감성을 분석

■ 핵심 개념 이해

■ 텍스트 마이닝

- 비정형의 텍스트 데이터로부터 패턴을 찾아내어 의미 있는 정보를 추출하는 분석 과정 또는 기법
- 데이터 마이닝과 자연어 처리, 정보 검색 등의 분야가 결합된 분석 기법을 사용
- 텍스트 마이닝의 프로세스
텍스트 전처리 → 특성 벡터화 → 머신러닝 모델 구축 및 학습/평가 프로세스 수행
» 텍스트 전처리에는 토큰화, 불용어 제거, 표제어 추출, 형태소 분석 등의 작업이 포함

■ 특성 벡터화와 특성 추출

- 머신러닝 알고리즘으로 분석하기 위해서는 텍스트를 구성하는 단어 기반의 특성 추출을 하고 이를 숫자형 값인 벡터 값으로 표현해야 함
- 특성 벡터화의 대표적인 방법으로 BoW와 Word2ve가 있음
- BOW: 문서가 가지고 있는 모든 단어에 대해 순서는 무시한 채 빈도만 고려하여 단어가 얼마나 자주 등장하는지로 특성 벡터를 만드는 방법
카운트 기반 벡터화와 TF-IDF 기반 벡터화 방식이 있음

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 핵심 개념 이해

▪ 카운트 기반 벡터화

- 단어 피처에 숫자형 값을 할당할 때 각 문서에서 해당 단어가 등장하는 횟수(단어 빈도)를 부여하는 벡터화 방식
- 문서별 단어의 빈도를 정리하여 문서 단어 행렬(DTM)을 구성하는 데 단어 출현 빈도가 높을수록 중요한 단어로 다루어짐
- 문서 d 에 등장한 단어 t 의 횟수는 $tf(t,d)$ 로 표현
- 카운트 기반 벡터화는 사이킷런의 CountVectorizer 모듈에서 제공

	그래서	데이터	분석	...	이다	한다
doc#1	13	20	16	...	65	71
doc#2	11	15	32	...	69	81

그림 13-1 카운트 기반 벡터화의 DTM 예: $tf(\text{"데이터"}, \text{doc\#1}) = 20$

▪ TF-IDF 기반 벡터화

- 특정 문서에 많이 나타나는 단어는 해당 문서의 단어 벡터에 가중치를 높임
- 모든 문서에 많이 나타나는 단어는 범용적으로 사용하는 단어로 취급하여 가중치를 낮추는 방식

- d 에 등장한 단어 t 의 TF-IDF $tf-idf(t, d) = tf(t, d) \times idf(t, d)$

- (역문서 빈도) $idf(t, d) = \log \frac{n_d}{1 + df(d, t)}$

- n^d : 전체 문서의 개수

- $df(d, t)$ 는 단어 t 가 포함된 문서 d 의 개수

	그래서	데이터	분석	...	이다	한다
doc#1	0.12	0.52	0.42	...	0.19	0.20
doc#2	0.13	0.48	0.67	...	0.18	0.22

그림 13-2 TF-IDF 기반 벡터화의 DTM 예: $tf-idf(\text{"데이터"}, \text{doc\#1}) = 0.52$

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 핵심 개념 이해

▪ 감성 분석(오피니언 마이닝)

- 텍스트에서 사용자의 주관적인 의견이나 감성, 태도를 분석하는 텍스트 마이닝의 핵심 분석 기법 중 하나
- 텍스트에서 감성을 나타내는 단어를 기반으로 긍정 또는 부정의 감성을 결정
- 감성 사전 기반의 감성 분석은 감성 단어에 대한 사전을 가진 상태에서 단어를 검색하여 점수를 계산
- 최근에는 머신러닝 기반의 감성 분석이 늘어나고 있음

▪ 토픽 모델링

- 문서를 구성하는 키워드를 기반으로 토픽(주제)을 추출하고 그 토픽을 기준으로 문서를 분류(클러스터링) 및 분석하는 기법
- 문서에서 다루는 토픽을 도출하여 동향을 파악하고 새로운 문서의 토픽을 예측하는 분석에 사용

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 핵심 개념 이해

▪ LDA

- 디리클레 분포를 이용하여 주어진 문서에 잠재되어 있는 토픽을 추론하는 확률 모델 알고리즘을 사용
- 하나의 문서는 여러 토픽으로 구성되어 있고, 문서의 토픽 분포에 따라서 단어의 분포가 결정된다고 가정
- 토픽의 개수 k : 토픽 분석의 성능을 결정짓는 중요한 요소이자 사용자가 지정해야 하는 하이퍼 매개변수

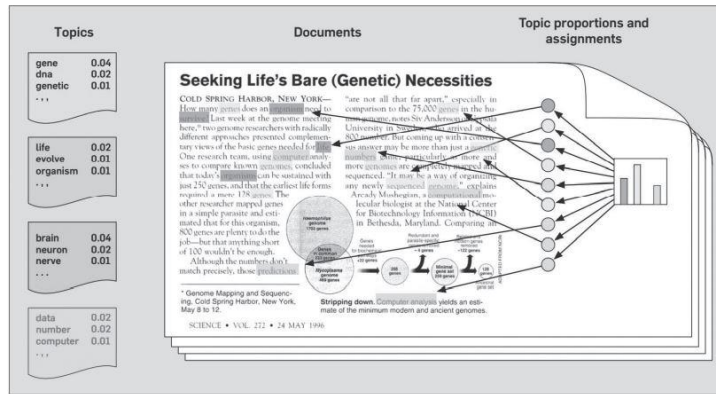


그림 13-3 문서에 LDA를 적용한 토픽 도출 예시(출처: 브리티시컬럼비아대학교)

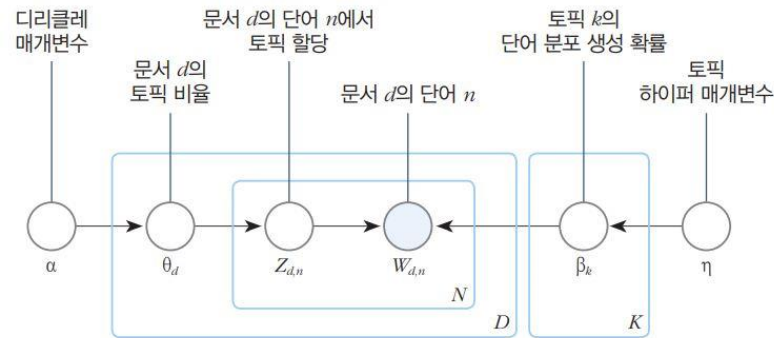


그림 13-4 LDA 그래픽 모델

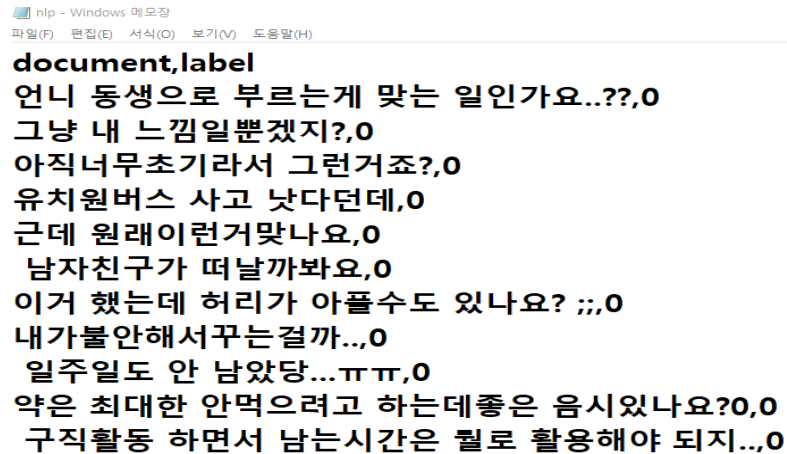
▪ pyLDAvis

- LDA를 이용한 토픽 모델링 분석 결과를 시각화하는 라이브러리
- 유사성에 따라 토픽 간 거리 지도와 선택한 토픽에서 관련성 높은 단어 30개를 바 차트로 시각화하여 보여줌.

01. [감성 분석] 리뷰 데이터로 감성 예측하기

▪ 데이터 수집

- 파일을 열어서 내용을 확인



```
nlp - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

document,label
언니 동생으로 부르는게 맞는 일인가요??,0
그냥 내 느낌일뿐겠지?,0
아직너무초기라서 그런거죠?,0
유치원버스 사고 났다던데,0
근데 원래이런거맞나요,0
남자친구가 떠날까봐요,0
이거 했는데 허리가 아플수도 있나요? ;;,0
내가불안해서꾸는걸까...,0
일주일도 안 남았당...ㅠㅠ,0
약은 최대한 안먹으려고 하는데좋은 음식있나요?0,0
구직활동 하면서 남는시간은 월로 활용해야 되지..,0
```

- nlp.csv 파일 : 감정 분석 모델을 만들기 위한 예시 데이터

- 총 14134개의 데이터로 2개의 document, label 컬럼을 가지며 0 : 7067 , 1 : 7067로 동일한 비율의 label을 가진다.

01. [감성 분석] 리뷰 데이터로 감성 예측하기

- 데이터 준비 및 탐색

2. 훈련용 데이터 준비하기

1. 감성 분석을 위한 예시 데이터인 nlp.csv 파일을 로드

In [1]:	<pre>#warning 메시지 표시 안함 import warnings warnings.filterwarnings(action = 'ignore') import pandas as pd</pre>
In [2]:	<pre>df = pd.read_csv('data/nlp.csv') df.info()</pre>
Out[2]:	<pre><class 'pandas.core.frame.DataFrame'> RangeIndex: 14134 entries, 0 to 14133 Data columns (total 2 columns): # Column Non-Null Count Dtype --- - 0 document 14134 non-null object 1 label 14134 non-null int64 dtypes: int64(1), object(1)</pre>

In [2]: 예시 데이터 파일을 읽고 `pd.read_csv()` 데이터프레임 객체 `df`에 저장

01. [감성 분석] 리뷰 데이터로 감성 예측하기

▪ 데이터 준비

- 한글 외의 문자 제거하기

In [1]:	<pre>import re</pre>
In [2]:	<pre>df['document'] = df['document'].apply(lambda x : re.sub(r'[^ㄱ-ㅣ가-힣]+', " ", x)) df['document'].head()</pre>
Out[2]:	<pre>0 언니 동생으로 부르는게 맞는 일인가요 1 그냥 내 느낌일뿐겠지 2 아직너무초기라서 그런거죠 3 유치원버스 사고 났다던데 4 근데 원래이런거맞나요 Name: document, dtype: object</pre>

In [1]: 정규식을 사용하기 위해 re 모듈을 импорт

In [2]: ‘ㄱ’으로 시작하거나 ‘가’부터 ‘힣’까지의 문자를 제외한 나머지는 공백으로 치환

01. [감성 분석] 리뷰 데이터로 감성 예측하기

▪ 데이터 준비

- 훈련용 데이터, 테스트 데이터 분리 8 : 2

In [3]:

```
x_data = df['document']
y_data = df['label']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data,
                                                    test_size = 0.2, # test_size : 분리하는 비율
                                                    random_state = 0, # random_state : 데이터 고정
                                                    stratify = y_data)
# stratify : y데이터를 고르게 가져오는 거
```

In [3]: 감성분석을 위해 내용 데이터는 x_data에, 금부정 레이블 데이터는 y_data 객체에 넣고,
8 : 2의 비율로 x_train, x_test, y_train, y_test로 나누었다.

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

1. 분석 모델 구축

1. 형태소 단위로 토큰화한 한글 단어에 대해 TF-IDF 방식을 사용하여 벡터화 작업을 수행

In [4]:	<pre>from konlpy.tag import Okt okt = Okt()</pre>
In [5]:	<pre>def okt_tokenizer(text): tokens = okt.morphs(text) return tokens</pre>
In [6]:	<pre>from sklearn.feature_extraction.text import TfidfVectorizer tfidf = TfidfVectorizer(tokenizer = okt_tokenizer, ngram_range=(1,2), min_df = 3, max_df = 0.7) tfidf.fit(x_train) train_tfidf = tfidf.transform(x_train) test_tfidf = tfidf.transform(x_test)</pre>

In [4]: 형태소 분석에 사용할 konlpy 패키지의 Okt 클래스를 임포트하고 okt 객체를 생성

In [5]: 문장을 토큰화하기 위해 okt_tokenizer 함수를 정의하고 okt.morphs() 함수를 사용하여 형태소 단위로 토큰화 작업을 수행

In [6]: 사이킷런의 TfidfVectorizer를 이용하여 TF-IDF 벡터화에 사용할 tfidf 객체를 생성

토큰 생성기 `tokenizer`는 우리가 정의한 `okt_tokenizer()` 함수로 설정하고 토큰의 단어 크기 `ngram_range`는 1~2개 단어로 함
토큰은 출현 빈도가 최소 `min_df` 3번 이상이고 최대 `max_df` 90% 이하인 것만 사용
벡터화할 데이터 `x_train`에 대해 벡터 모델 `tfidf`의 내부 설정값을 조정 `fit()` 하고 벡터로 변환을 수행 `transform()`

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

2. 여러 감성 분류 모델 구축하기

- 1. 로지스틱 회귀 모델로 감성 분류 모델 구축하기

1. 머신러닝의 로지스틱 회귀 모델을 이용하여 긍정과 부정의 감성 이진 분류 모델을 구축

In [7]:	<pre>from sklearn.linear_model import LogisticRegression SA_lr = LogisticRegression(random_state = 0)</pre>
In [8]:	<pre>SA_lr.fit(train_tfidf, y_train)</pre>
Out[8]:	<pre>LogisticRegression(random_state = 0)</pre>

In [7]: 사이킷런의 LogisticRegression 클래스에 대해 객체 SA_lr을 생성

In [8]: nsmc_train_tfidf를 독립변수 X로 하고 label 컬럼을 종속 변수 Y로 하여 로지스틱 회귀 모델 SA_lr의 내부 설정 값을 조정 fit()

1. 로지스틱 회귀의 하이퍼 매개변수 C의 최적값을 구하기 위해 C 값을 다르게 한 여러 모형을 만들고 실행하여 각 성능을 비교 (GridSearchCV 클래스 사용)

In [9]:	<pre>from sklearn.model_selection import GridSearchCV params = {'C': [1, 3, 3.5, 4, 4.5, 5]} SA_lr_grid_cv = GridSearchCV(SA_lr, param_grid = params, cv = 3, scoring = 'accuracy', verbose = 1)</pre>
---------	---

In [9]: 하이퍼 매개변수 C에 대해 비교 검사를 할 6개 값 [1, 3, 3.5, 4, 4.5, 5]을 params로 하고, 교차 검증 cv를 3, 모형 비교 기준은 정확도로 설정 scoring='accuracy'하여 GridSearchCV 객체를 생성

01. [감성 분석] 리뷰 데이터로 감성 예측하기

- 분석 모델 구축

- 2. 로지스틱 회귀 모델로 감성 분류 모델 구축하기

- 2. 로지스틱 회귀의 하이퍼 매개변수 C의 최적값을 구하기 위해 C 값을 다르게 한 여러 모델을 만들고 실행하여 각 성능을 비교
(GridSearchCV 클래스 사용)

In [10]:	SA_lr_grid_cv.fit(train_tfidf, y_train)
Out[10]:	Fitting 3 folds for each of 6 candidates, totalling 18 fits GridSearchCV(cv = 3, estimator = LogisticRegression(random_state = 0), param_grid = {'C': [1, 3, 3.5, 4, 4.5, 5]}, scoring = 'accuracy', verbose = 1)
In [11]:	print(SA_lr_grid_cv.best_params_, round(SA_lr_grid_cv.best_score_, 4))
Out[11]:	{'C': 3} 0.8553
In [12]:	# 최적 매개변수의 best 모델 저장 SA_lr_best = SA_lr_grid_cv.best_estimator_

In [10]: GridSearchCV 객체에 nsmc_train_tfidf와 label 컬럼에 대해 설정값을 조정fit()

In [11]: GridSearchCV에 의해 찾은 최적의 C 매개변수best_params와 최고 점수best_score를 출력하여 확인

In [12]: 최적 매개변수가 설정된 모델best_estimator을 SA_lr_best 객체에 저장

01. [감성 분석] 리뷰 데이터로 감성 예측하기

- 분석 모델 평가

- 평가용 데이터를 이용하여 모델 정확도 확인하기

1. 평가용 데이터를 벡터화한 뒤 모델 정확도를 계산하여 출력

In [13]:	<pre>predict = SA_lr_best.predict(test_tfidf)</pre>
In [14]:	<pre>from sklearn.metrics import accuracy_score print('감성 분류 모델의 정확도 : ', round(accuracy_score(y_test, predict)) print('감성 분류 모델의 ROC :', roc_auc_score(y_test, predict))</pre>
Out[14]:	<pre>감성 분류 모델의 정확도 : 0.8478952953661125 감성 분류 모델의 ROC : 0.8478920230512588</pre>

In [13]: 감성 분류 모델 `SA_lr_best`에 `test_tfidf` 벡터를 사용하여 감성을 예측 `predict()`

In [14]: 평가용 데이터의 감성 결과값 `y_test`과 감성 예측값 `predict`을 기반으로

정확도를 계산 `accuracy_score()`, ROC_AUC 점수를 계산 `roc_auc_score()` 하여 출력

- 감성 분류 모델의 정확도가 84.7%
- 감성 분류 모델의 ROC는 84.7%

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

1. 분류 보고서(classification_report)를 이용하여 정밀도, 재현율 확인하기

In [15]:	<pre>pred_train_rbf = SA_lr_best.predict(train_tfidf) from sklearn.metrics import classification_report cfreport_train_rbf = classification_report(y_train, pred_train_rbf) print(cfreport_train_rbf)</pre>																														
Out[15]:	<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.84</td><td>0.86</td><td>0.85</td><td>1414</td></tr><tr><td>1</td><td>0.85</td><td>0.84</td><td>0.85</td><td>1413</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.85</td><td>2827</td></tr><tr><td>macro avg</td><td>0.85</td><td>0.85</td><td>0.85</td><td>2827</td></tr><tr><td>weighted avg</td><td>0.85</td><td>0.85</td><td>0.85</td><td>2827</td></tr></table>		precision	recall	f1-score	support	0	0.84	0.86	0.85	1414	1	0.85	0.84	0.85	1413	accuracy			0.85	2827	macro avg	0.85	0.85	0.85	2827	weighted avg	0.85	0.85	0.85	2827
	precision	recall	f1-score	support																											
0	0.84	0.86	0.85	1414																											
1	0.85	0.84	0.85	1413																											
accuracy			0.85	2827																											
macro avg	0.85	0.85	0.85	2827																											
weighted avg	0.85	0.85	0.85	2827																											

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

2. 여러 감성 분류 모델 구축하기

- 2. Naive_Bayes 모델로 감성 분류 모델 구축하기

1. 머신러닝의 Naive_Bayes 모델을 이용하여 긍정과 부정의 감성 이진 분류 모델을 구축

In [16]:	<pre>from sklearn.naive_bayes import MultinomialNB from sklearn.metrics import accuracy_score # 정확도 계산 naive = MultinomialNB()</pre>
In [17]	<pre>params = {'alpha': [0.5, 1.0, 1.5, 2.0]} naive_grid_cv = GridSearchCV(naive, # 모델 param_grid = params, # 파라미터 사전 cv = 3, scoring = 'accuracy', # 점수 계산 방식 verbose = 1)</pre>

In [16]: 사이킷런의 naive_bayes 패키지의 MultinomialNB 클래스에 대해 객체 naive를 생성

In [17]: 나이브 베이저의 하이퍼 매개변수 alpha의 최적값을 구하기 위해 GridSearchCV 클래스 사용.

하이퍼 매개변수 alpha에 대해 비교 검사를 할 4개 값[0.5, 1.0, 1.5, 2.0]을 params로 하고, 교차 검증cv를 3, 모형 비교 기준은 정확도로 설정scoring='accuracy'하여 GridSearchCV 객체를 생성

01. [감성 분석] 리뷰 데이터로 감성 예측하기

- 분석 모델 구축

2. Naive_Bayes 모델로 감성 분류 모델 구축하기

2. Naive_Bayes의 하이퍼 매개변수 alpha의 최적값을 구하기 위해
alpha 값을 다르게 한 여러 모형을 만들고 실행하여 각 성능을 비교
(GridSearchCV 클래스 사용)

In [18]:	<code>naive_grid_cv.fit(train_tfidf, y_train)</code>
In [19]:	<code>print(naive_grid_cv.best_params_, round(naive_grid_cv.best_score_, 4))</code>
Out[19]:	{'alpha': 0.5} 0.8526
In [20]:	<code># 최적 매개변수의 best 모델 저장 naive_best = naive_grid_cv.best_estimator_</code>

In [18]: GridSearchCV 객체에 train_tfidf와 y_train에 대해 설정값을 조정fit()

In [19]: GridSearchCV에 의해 찾은 최적의 alpha 매개변수best_params와 최고 점수best_score를 출력하여 확인

In [20]: 최적 매개변수가 설정된 모형best_estimator을 naive_best 객체에 저장

01. [감성 분석] 리뷰 데이터로 감성 예측하기

- 분석 모델 평가

- 평가용 데이터를 이용하여 모델 정확도 확인하기

1. 평가용 데이터를 벡터화한 뒤 모델 정확도를 계산하여 출력

In [21]:	<code>predict = naive_best.predict(test_tfidf)</code>
In [22]:	<code>from sklearn.metrics import accuracy_score</code> <code>print('감성 분류 모델의 정확도 : ', round(accuracy_score(y_test, predict))</code> <code>print('감성 분류 모델의 ROC :', roc_auc_score(y_test, predict))</code>
Out[22]:	감성 분류 모델의 정확도 : 0.8556773965334277 감성 분류 모델의 ROC : 0.8556721231722808

In [21]: 감성 분류 모델 `naive_best`에 `test_tfidf` 벡터를 사용하여 감성을 예측 `predict()`

In [22]: 평가용 데이터의 감성 결과값 `y_test`과 감성 예측값 `predict`을 기반으로
정확도를 계산 `accuracy_score()`, ROC_AUC 점수를 계산 `roc_auc_score()` 하여 출력

- 감성 분류 모델의 정확도가 85.5%
- 감성 분류 모델의 ROC는 85.5%

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

1. 분류 보고서(classification_report)를 이용하여 정밀도, 재현율 확인하기

In [23]	<pre>pred_test_rbf = naive_best.predict(test_tfidf) from sklearn.metrics import classification_report cfreport_test_rbf = classification_report(y_test, pred_test_rbf) print(cfreport_test_rbf)</pre>				
out [24]		precision	recall	f1-score	support
	0	0.84	0.88	0.86	2121
	1	0.87	0.83	0.85	2120
	accuracy			0.85	4241
	macro avg	0.85	0.85	0.85	4241
	weighted avg	0.85	0.85	0.85	4241

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

2. 여러 감성 분류 모델 구축하기

- 3. LGBM 모델로 감성 분류 모델 구축하기

1. 머신러닝의 LGBM 모델을 이용하여 긍정과 부정의 감성 이진 분류 모델을 구축
2. LGBM의 최적의 모델을 구축하기 위해 하이퍼 매개변수들의 최적값을 구해주는 GridSearchCV 클래스 사용

In [25]:	<pre>from lightgbm import LGBMClassifier lgb = LGBMClassifier(random_state = 123, n_jobs = -1)</pre>
In [26]:	<pre>from sklearn.model_selection import GridSearchCV grid_param = {'max_depth': [1, 3, 5, 7], 'n_estimators': [100, 300, 500], 'learning_rate': [0.05, 0.1, 1, 10], 'subsample': [0.5, 1]} lgb_grid = GridSearchCV(estimator = lgb, param_grid = grid_param, cv = 3, scoring = 'accuracy') fit_param = {'early_stopping_rounds': 50, 'eval_metric': 'error', 'eval_set': [[test_tfidf, y_test]]}</pre>

In [25]: LGBM 클래스에 대해 객체 lgb를 생성

In [26]: 다양한 하이퍼 매개변수를 설정하여 GridSearchCV 객체를 생성

LGBM모델의 fit()에 대한 하이퍼 파라미터를 사전 형식의 fit_param객체 생성

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

2. LGBM 모델로 감성 분류 모델 구축하기

2. LGBM의 최적의 모델을 구축하기 위해 하이퍼 매개변수들의 최적값을 구해주는 GridSearchCV 클래스 사용

In [26]:	<pre>lgb_grid.fit(train_tfidf, y_train, **fit_param) print(lgb_grid.best_params_, round(lgb_grid.best_score_, 4))</pre>
Out[26]:	<pre>{'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 500, 'subsample': 0.5} 0.7654</pre>
In [27]:	<pre>lgb_best = lgb_grid.best_estimator_</pre>

In [26]: GridSearchCV에 의해 찾은 최적의 매개변수들을 확인

사전 형식의 fit의 하이퍼파라미터를 적용하기 위해 unpacking(사전 형식은 **)하여 돌려줌.

In [27]: 최적 매개변수가 설정된 모형 `best_estimator`을 `lgb_best` 객체에 저장

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

1. 평가용 데이터를 이용하여 모델 정확도 확인하기

1. 평가용 데이터를 벡터화한 뒤 모델 정확도를 계산하여 출력

In [24]:	<pre>lgb_best = lgb_grid.best_estimator_ predict = lgb_best.predict(test_tfidf)</pre>
In [25]:	<pre>from sklearn.metrics import accuracy_score print(accuracy_score(y_test, predict)) print(roc_auc_score(y_test, predict))</pre>
Out[25]:	<pre>0.7948355146798727 0.7948017049202647</pre>

In [24]: 감성 분류 모델 `estimator` 에 `test_tfidf` 벡터를 사용하여 감성을 예측 `predict()`

In [25]: 평가용 데이터의 감성 결과값 `y_test` 과 감성 예측값 `predict` 을 기반으로

정확도를 계산 `accuracy_score()`, ROC_AUC 점수를 계산 `roc_auc_score()` 하여 출력

- 감성 분류 모델의 정확도가 79.4%
- 감성 분류 모델의 AUC가 79.4%

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

1. 분류 보고서(classification_report)를 이용하여 정밀도, 재현율 확인하기

In [26]:	<pre>pred_test_rbf = estimator.predict(test_tfidf) from sklearn.metrics import classification_report cfreport_test_rbf = classification_report(y_test, pred_test_rbf) print(cfreport_test_rbf)</pre>
Out[26]:	<pre> precision recall f1-score support 0 0.75 0.89 0.81 1414 1 0.86 0.70 0.77 1413 accuracy 0.79 2827 macro avg 0.81 0.79 0.79 2827 weighted avg 0.81 0.79 0.79 2827</pre>

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

2. 여러 감성 분류 모델 구축하기

- 4. SVM 모델로 감성 분류 모델 구축하기

1. 머신러닝의 SVM 모델을 이용하여 긍정과 부정의 감성 이진 분류 모델을 구축
2. SVM의 하이퍼 매개변수 C와 gamma의 최적값을 구하기 위해 C 값을 다르게 한 여러 모형을 만들고 실행하여 각 성능을 비교 (GridSearchCV 클래스 사용)

In [27]:	<pre>from sklearn.svm import SVC model_rbf = SVC(kernel='rbf', random_state = 123)</pre>
In [28]:	<pre>param_grid = [{'C' : [0.01, 0.1, 1, 3, 7], 'gamma' : [0.01, 0.05, 0.1, 1, 3]}] from sklearn.model_selection import GridSearchCV grid_search = GridSearchCV(model_rbf, param_grid, cv = 3)</pre>

In [27]: SVM 클래스에 대해 객체 model_rbf 를 생성

In [28]: 하이퍼 매개변수 C에 대해 비교 검사를 할 4개 값[0.01, 0.1, 1, 3]과 gamma에 대해 비교 검사를 할 4개의 값 [0.01, 0.1, 1, 3]을 params로 하고, 교차 검증cv을 3, 모형 비교 기준은 정확도로 설정scoring='accuracy'하여 GridSearchCV 객체를 생성

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

2. 감성 분류 모델 구축하기

2. SVM의 하이퍼 매개변수 C와 gamma의 최적값을 구하기 위해 C 값을 다르게 한 여러 모형을 만들고 실행하여 각 성능을 비교 (GridSearchCV 클래스 사용)

In [29]:	<pre>grid_search.fit(train_tfidf, y_train) print(grid_search.best_params_) print(grid_search.best_score_)</pre>
Out[29]:	<pre>{'C': 7, 'gamma': 0.05} 0.843813566817016</pre>
In [30]:	<pre>svm_best = grid_search.best_estimator_</pre>

In [29]: GridSearchCV에 의해 찾은 최적의 C, gamma 매개변수 `best_params`와 최고 점수 `best_score`를 출력하여 확인

In [30]: 최적 매개변수가 설정된 모형 `best_estimator`을 `svm_best` 객체에 저장

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

1. 평가용 데이터를 이용하여 모델 정확도 확인하기

1. 평가용 데이터를 벡터화한 뒤 모델 정확도를 계산하여 출력

In [31]:	<pre>predict = svm_best.predict(test_tfidf)</pre>
In [32]:	<pre>from sklearn.metrics import accuracy_score, roc_auc_score print(accuracy_score(y_test, predict)) print(roc_auc_score(y_test, predict))</pre>
Out[32]:	<pre>0.8539087371772197 0.8538998349334478</pre>

In [31]: 감성 분류 모델 `svm_best`에 `test_tfidf` 벡터를 사용하여 감성을 예측 `predict()`

In [32]: 평가용 데이터의 감성 결과값 `y_test`과 감성 예측값 `test_predict`을 기반으로 정확도와 AUC를 계산하여 출력

- 감성 분류 모델의 정확도가 85.3%
- 감성 분류 모델의 AUC가 85.3%

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

1. 분류 보고서(classification_report)를 이용하여 정밀도, 재현율 확인하기

In [33]:	<pre>pred_test_rbf = svm_best.predict(test_tfidf) from sklearn.metrics import classification_report cfreport_test_rbf = classification_report(y_test, pred_test_rbf) print(cfreport_test_rbf)</pre>
Out[33]:	<pre> precision recall f1-score support 0 0.84 0.88 0.86 1414 1 0.87 0.83 0.85 1413 accuracy 0.85 macro avg 0.85 0.85 0.85 weighted avg 0.85 0.85 0.85</pre>

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

2. 여러 감성 분류 모델 구축하기

- 5. RandomForest 모델로 감성 분류 모델 구축하기

1. 머신러닝의 RandomForestClassifier 모델을 이용하여 긍정과 부정의 감성 이진 분류 모델을 구축

```
In [34]: from sklearn.ensemble import RandomForestClassifier  
RC = RandomForestClassifier(random_state=123)
```

2. RandomForest의 최적의 모델을 구축하기 위해 하이퍼 매개변수들의 최적값을 구해주는 GridSearchCV 클래스 사용

```
In [35]: parameters = {'max_depth' : [1, 2, 3], 'min_samples_split' : [2, 3]}  
  
grid_search_RC= GridSearchCV(RC,  
                             param_grid=parameters,  
                             cv=3)
```

In [17]: RandomForestClassifier 클래스에 대해 객체 RC를 생성

In [19]: 하이퍼 매개변수 max_depth에 대해 비교 검사를 할 3개 값 [1,2,3]과 min_samples_split에 대해 비교 검사를 할 2개의 값 [2,3]을 params로 하고, 교차 검증 cv를 3, 모형 비교 기준은 정확도로 설정 scoring='accuracy'하여 GridSearchCV 객체를 생성

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

2. 감성 분류 모델 구축하기

2. RandomForest의 최적의 모델을 구축하기 위해 하이퍼 매개변수들의 최적값을 구해주는 GridSearchCV 클래스 사용

In [36]:	grid_search_RC.fit(train_tfidf, y_train)
In [37]:	print(grid_search_RC.best_params_) print(grid_search_RC.best_score_)
Out[37]:	{'max_depth': 3, 'min_samples_split': 2} 0.7226496860351994
In [38]:	<i># 최적 매개변수의 best 모델 저장</i> rf_best = grid_search_RC.best_estimator_

In [36]: GridSearchCV 객체에 train_tfidf와 y_train에 대해 설정값을 조정(fit)

In [37]: GridSearchCV에 의해 찾은 최적의 Cmax_depth min_samples 매개변수best_params와 최고 점수best_score를 출력하여 확인

In [38]: 최적 매개변수가 설정된 모형best_estimator을 rf_best 객체에 저장

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

1. 분류 보고서(classification_report)를 이용하여 정밀도, 재현율 확인하기

In [39]:	<pre>pred_test_RC = rf_best.predict(test_tfidf) from sklearn.metrics import classification_report cfreport_test_RC = classification_report(y_test, pred_test_RC) print(cfreport_test_RC)</pre>
Out[39]:	<pre> precision recall f1-score support 0 0.77 0.69 0.73 1414 1 0.72 0.79 0.75 1413 accuracy 0.74 macro avg 0.74 weighted avg 0.74</pre>

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

3. 여러 모델 성능 평가 시각화

1. 여러 감성 분류 모델 성능 평가 확인 후 하나의 모델 선택

In [40]:	<pre>models = [SA_lr_best, naive_best, lgb_best, svm_best, rf_best] model_df = pd.DataFrame(columns = ['Logistic', 'naive_bayes', 'LGBM', 'SVM', 'RandomForest'], index = ['train', 'test', 'roc']) for idx, model in enumerate(models): train = model.score(train_tfidf, y_train) test = model.score(test_tfidf, y_test) predict = model.predict(test_tfidf) roc = roc_auc_score(y_test, predict) model_df.iloc[:, idx] = [train, test, roc] print(model_df)</pre>																								
Out[40]:	<table><tr><th></th><th>Logistic</th><th>naive_bayes</th><th>LGBM</th><th>SVM</th><th>RandomForest</th></tr><tr><td>train</td><td>0.948704</td><td>0.918723</td><td>0.827540</td><td>0.929955</td><td>0.746706</td></tr><tr><td>test</td><td>0.847895</td><td>0.855677</td><td>0.794836</td><td>0.853909</td><td>0.739300</td></tr><tr><td>roc</td><td>0.847892</td><td>0.855672</td><td>0.794802</td><td>0.853900</td><td>0.739318</td></tr></table>		Logistic	naive_bayes	LGBM	SVM	RandomForest	train	0.948704	0.918723	0.827540	0.929955	0.746706	test	0.847895	0.855677	0.794836	0.853909	0.739300	roc	0.847892	0.855672	0.794802	0.853900	0.739318
	Logistic	naive_bayes	LGBM	SVM	RandomForest																				
train	0.948704	0.918723	0.827540	0.929955	0.746706																				
test	0.847895	0.855677	0.794836	0.853909	0.739300																				
roc	0.847892	0.855672	0.794802	0.853900	0.739318																				

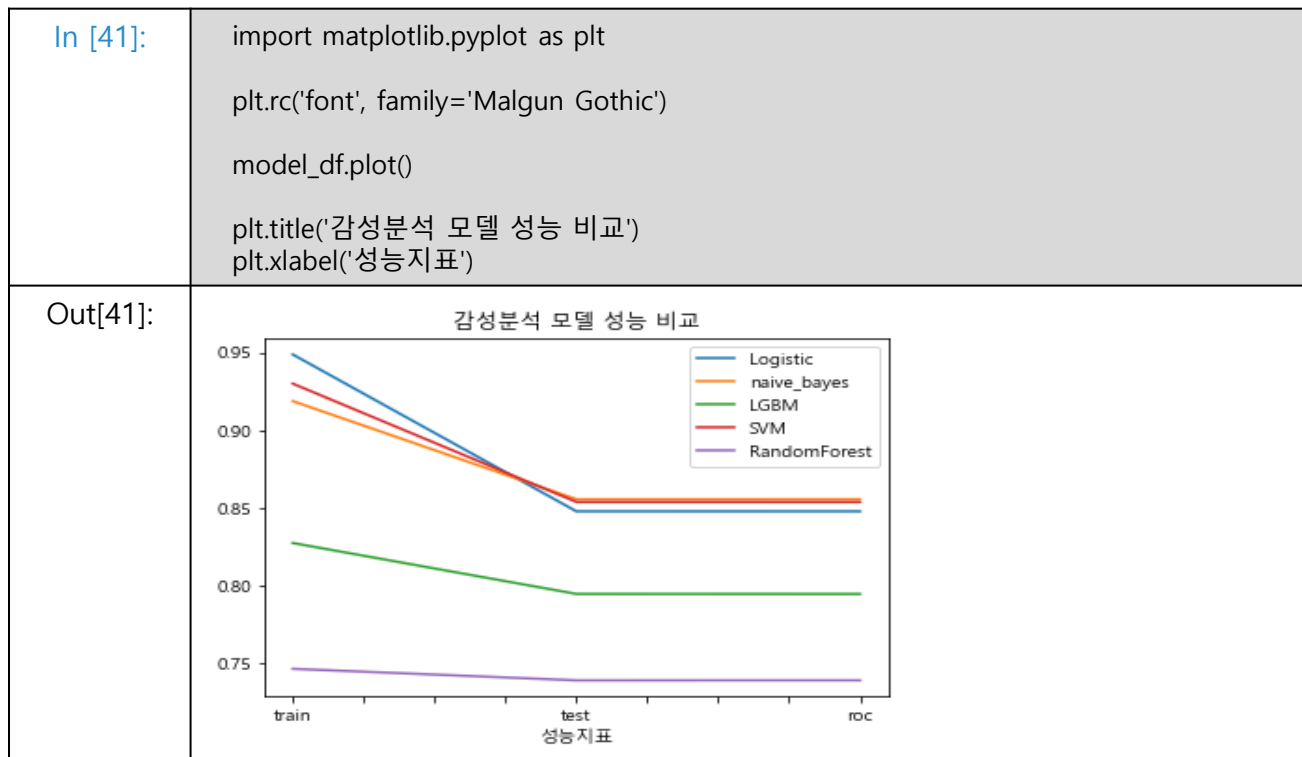
In [40]: Logistic, naive_bayes, LGBM, SVM, RandomForest모델의 train데이터 정확도, test데이터 정확도, roc_auc_score의 점수를 각각 데이터 프레임에 넣기

01. [감성 분석] 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

3. 여러 모델 성능 평가 시각화

1. 여러 감성 분류 모델 성능 평가 확인 후 하나의 모델 선택



• 감정 결과

- test 정확도와, roc 점수를 비교했을 때 성능이 가장 우수한 SVM 모델을 사용하여 네이버 기사를 분류.

Chapter

02

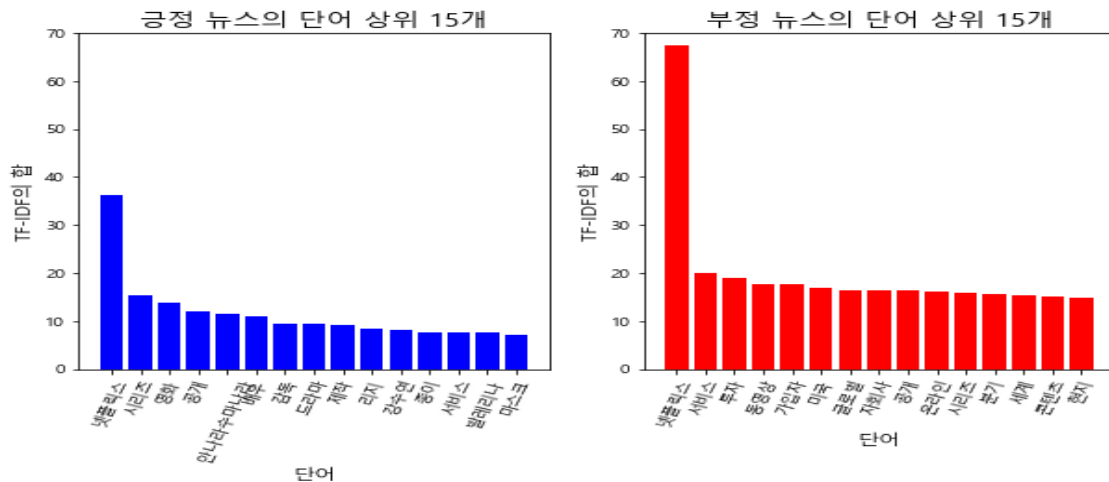
Netflix 뉴스 데이터 텍스트의 감성 분석하기

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 분석 미리보기

코로나 뉴스 텍스트의 감성 분석하기	
목표	네이버 뉴스의 Netflix 관련 텍스트에 대해 감성 분석을 수행한다.
핵심개념	텍스트 마이닝, 특성 벡터화, BoW, TF-IDF, DTM, 감성분석, 토픽모델링, LDA
데이터 수집	네이버 뉴스의 json 파일 : 네이버 API로 크롤링하여 저장(예제 소스로 제공)
데이터 준비 및 탐색	1. 파일 불러오기 : <code>pd.read_json()</code> 2. 분석할 컬럼 추출 3. 한글 외의 문자 제거
감성 분석	title/description 컬럼에 대한 감성 분석 - TF-IDF 기반 특성 벡터화 : <code>tfidf.transform()</code> - 분석 모델을 이용한 감성분석 : <code>svm_best.predict()</code> - 분석 결과를 데이터 프레임에 저장
결과 확인 및 시각화	

감성 분석 결과 시각화



02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 목표설정

- 목표: 감성 분류 모델을 이용하여 네이버 뉴스에서 크롤링 한 '넷플릭스' 관련 텍스트에 대해 감성을 분석

■ 데이터수집

- '네이버 API를 이용한 크롤링'으로 네이버 뉴스를 크롤링하여 텍스트 데이터를 수집
- 최근 1,000개의 뉴스가 크롤링 되어 저장된 json 파일을 생성
- 예제소스로 제공하는 '넷플릭스_naver_news.json' 파일을 이용해도 됨

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 데이터 준비 및 탐색

1. 분석할 컬럼을 추출하여 데이터프레임 구성하기

1. My_Python 폴더에 data폴더를 만든 뒤 크롤링한 파일을 옮기고 앞에서 만든 '감성분석' 페이지에 이어서 실행
2. 뉴스의 내용이 들어 있는 description 컬럼과 제목이 들어 있는 title 컬럼을 추출

In [42]:	<pre>import json file_name = '넷플릭스_naver_news' df = pd.read_json(file_name,'.json', orient = 'str') data_df = df[['title', 'description']]</pre>
In [43]:	<pre>print(data_df.head())</pre>
Out[43]:	<pre> title description 0 [단독]넷플릭스 '사냥개들', 김새론 추가 촬영 취소...제작 지연 불가피 이어 넷플릭스 오리지널 시리즈 '사냥개들'의 추가 촬영분을 찍지 않는 ... 1 널달새 사용자 11% 증발... 넷플릭스 웨이브 등 OTT 초비상 세계 1위 넷플릭스가 지난 1분기 11년 만에 첫 유료 가입자 감소 사... 2 美 바이든 오자 '무임승차' 넷플릭스, 韓 투자 '생색' [OTT온에어] 넷플 릭스가 자회사 스캔라인VFX 코리아를 통해 1억달러(약 1천260억... 3 "한국 덕 그렇게 보면서" 넷플릭스 '쥐꼬리' 투자, 정부가 '들러리' "넷 플릭스와 한국 창작 생태계의 깊은 파트너십과 우정은 마치 '간부' ... 4 김새론, 넷플릭스 '사냥개들' 남은 촬영 취소...&quot;하차는 아냐&... 음주운 전 사고로 물의를 일으킨 배우 김새론이 넷플릭스 시리즈 '사냥개들...</pre>

In [42]: pandas의 내장함수 `pd.read_json(orient = 'str')` 활용하여 분석할 데이터 파일 로드 후 df 객체에 저장
description 컬럼과 title 컬럼을 따로 추출하여 data_df 객체에 저장

In [43]: data_df 객체의 내용을 출력하여 확인

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 감성 분석

1. 감성 분석 수행 후 결과값을 데이터프레임에 저장하기

1. 감성분석 수행 전에 텍스트 전처리하기

In [44]:	<pre>import re data_df['title'] = data_df['title'].apply(lambda x : re.sub(r'[^ㄱ- 가-힝]+', " ", x)) data_df['description'] = data_df['description'].apply(lambda x : re.sub(r'[^ㄱ- 가-힝]+', " ", x)) data_df.head()</pre>																		
Out[44]:	<table><tr><th></th><th>title</th><th>description</th></tr><tr><td>0</td><td>단독 넷플릭스 사냥개들 김새론 추가 촬영 취소 제작 지연 불가피 이어 넷플릭스 오리</td><td>지널 시리즈 사냥개들 의 추가 촬영분을 찍지 않는 것으로 알...</td></tr><tr><td>1</td><td>넉달새 사용자 증발 넷플릭스 웨이브 등 소비상 세계 위 넷플릭스 가 지난 분</td><td>기 년 만에 첫 유료 가입자 감소 사태를 겪으며...</td></tr><tr><td>2</td><td>바이든 오자 무임승차 넷플릭스 투자 생색 온에어 넷플릭스 가 자회사 스캔라인</td><td>코리아를 통해 억달러 약 천 억원 국내 투자 ...</td></tr><tr><td>3</td><td>한국 덕 그렇게 보면서 넷플릭스 쥐꼬리 투자 정부가 들러리 넷플릭스 와 한국 창작</td><td>생태계의 깊은 파트너십과 우정은 마치 간부 같다 딴...</td></tr><tr><td>4</td><td>김새론 넷플릭스 사냥개들 남은 촬영 취소 하차는 아냐 공식입장 음주운전 사고로 물</td><td>의를 일으킨 배우 김새론이 넷플릭스 시리즈 사냥개들 의 남은...</td></tr></table>		title	description	0	단독 넷플릭스 사냥개들 김새론 추가 촬영 취소 제작 지연 불가피 이어 넷플릭스 오리	지널 시리즈 사냥개들 의 추가 촬영분을 찍지 않는 것으로 알...	1	넉달새 사용자 증발 넷플릭스 웨이브 등 소비상 세계 위 넷플릭스 가 지난 분	기 년 만에 첫 유료 가입자 감소 사태를 겪으며...	2	바이든 오자 무임승차 넷플릭스 투자 생색 온에어 넷플릭스 가 자회사 스캔라인	코리아를 통해 억달러 약 천 억원 국내 투자 ...	3	한국 덕 그렇게 보면서 넷플릭스 쥐꼬리 투자 정부가 들러리 넷플릭스 와 한국 창작	생태계의 깊은 파트너십과 우정은 마치 간부 같다 딴...	4	김새론 넷플릭스 사냥개들 남은 촬영 취소 하차는 아냐 공식입장 음주운전 사고로 물	의를 일으킨 배우 김새론이 넷플릭스 시리즈 사냥개들 의 남은...
	title	description																	
0	단독 넷플릭스 사냥개들 김새론 추가 촬영 취소 제작 지연 불가피 이어 넷플릭스 오리	지널 시리즈 사냥개들 의 추가 촬영분을 찍지 않는 것으로 알...																	
1	넉달새 사용자 증발 넷플릭스 웨이브 등 소비상 세계 위 넷플릭스 가 지난 분	기 년 만에 첫 유료 가입자 감소 사태를 겪으며...																	
2	바이든 오자 무임승차 넷플릭스 투자 생색 온에어 넷플릭스 가 자회사 스캔라인	코리아를 통해 억달러 약 천 억원 국내 투자 ...																	
3	한국 덕 그렇게 보면서 넷플릭스 쥐꼬리 투자 정부가 들러리 넷플릭스 와 한국 창작	생태계의 깊은 파트너십과 우정은 마치 간부 같다 딴...																	
4	김새론 넷플릭스 사냥개들 남은 촬영 취소 하차는 아냐 공식입장 음주운전 사고로 물	의를 일으킨 배우 김새론이 넷플릭스 시리즈 사냥개들 의 남은...																	

In [44]: 기사 제목이나, 본문에 태그 </d>나 "가 포함되어 있어

'ㄱ'으로 시작하거나 '가'부터 '힝'까지의 문자를 제외한 나머지는 공백으로 치환하여 텍스트 전처리 실행

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 감성 분석

1. 감성 분석 수행 후 결과값을 데이터프레임에 저장하기

1. title 컬럼에 대한 감성 분석을 수행

```
In [45]: #1) 분석할 데이터의 피처 벡터화 ---<< title >> 분석
data_title_tfidf = tfidf.transform(data_df['title'])

#2) 최적 매개변수 학습 모델에 적용하여 감성 분석
data_title_predict = svm_best.predict(data_title_tfidf)

#3) 감성 분석 결과값을 데이터프레임에 저장
data_df['title_label'] = data_title_predict
```

1. description 컬럼에 대해서도 같은 작업을 하여 감성 분석을 수행

```
In [46]: #1) 분석할 데이터의 피처 벡터화 ---<< description >> 분석
data_description_tfidf = tfidf.transform(data_df['description'])

#2) 최적 매개변수 학습 모델에 적용하여 감성 분석
data_description_predict = svm_best.predict(data_description_tfidf)

#3) 감성 분석 결과값을 데이터프레임에 저장
data_df['description_label'] = data_description_predict
```

1. 분석 결과 데이터프레임을 CSV 파일로 저장

```
In [47]: data_df.to_csv('data/'+file_name+'.csv', encoding = 'euc-kr')
```

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

1. 감성 분석 결과 확인하기

1. 감성 분석 결과를 확인

In [48]:	data_df.head()
In [49]:	print(data_df['title_label'].value_counts())
Out[49]:	0 609 1 391 Name: title_label, dtype: int64
In [50]:	print(data_df['description_label'].value_counts())
Out[50]:	0 648 1 352 Name: description_label, dtype: int64

• 감정 결과

- 데이터프레임 내용과 부정 감성 및 긍정 감성의 개수를 비교해보면 title 분석 결과와 description 분석 결과에 차이가 있음
 1. 단어를 기준으로 분석하기 때문에 단어의 개수가 부족하면 정확도가 떨어짐
 2. 우리가 구축한 감성 분류 모델(SVM)의 정확도가 85.3%이기 때문에 틀린 결과가 존재할 수도 있음.
 3. 분류 모델의 학습 데이터로 사용했던 리뷰의 구성 단어와 분석 데이터인 뉴스를 구성하는 단어의 차이로 인한 오차도 존재.

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

1. 감성 분석 결과 확인하기

2. 감성 분석 결과를 분리 저장하기

뉴스 본문에 대한 감성 분석을 기준으로 긍정 감성 데이터와 부정 감성 데이터를 분리 후 비교 분석

In [51]:	<pre>columns_name = ['title', 'title_label', 'description', 'description_label'] NEG_data_df = pd.DataFrame(columns = columns_name) POS_data_df = pd.DataFrame(columns = columns_name) for i, data in data_df.iterrows(): title = data["title"] description = data["description"] t_label = data["title_label"] d_label = data["description_label"] if d_label == 0: # 부정 감성 샘플만 추출 NEG_data_df = NEG_data_df.append(pd.DataFrame([[title, t_label, description, d_label]], columns = columns_name), ignore_index = True) else : # 긍정 감성 샘플만 추출 POS_data_df = POS_data_df.append(pd.DataFrame([[title, t_label, description, d_label]], columns=columns_name),ignore_index=True) # 파일에 저장 NEG_data_df.to_csv('./13장_data/'+file_name+'_NES.csv', encoding = 'euc-kr') POS_data_df.to_csv('./13장_data/'+file_name+'_POS.csv', encoding = 'euc-kr')</pre>
In [52]:	<pre>len(NEG_data_df), len(POS_data_df)</pre>
Out[52]:	<pre>(648, 352)</pre>

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

1. 명사 단어 추출하기 - 먼저, 긍정 감성 뉴스에서 형태소 분석을 하여 명사를 추출

In [53]:	POS_description = POS_data_df['description']
In [54]:	<pre>POS_description_noun_tk = [] for d in POS_description: POS_description_noun_tk.append(oka.nouns(d)) # 명사 형태소만 추출</pre>
In [55]:	<pre>print(POS_description_noun_tk) # 작업 확인용 출력</pre>
Out[55]:	<pre>[['뉴미디어', '트렌드', '라이브', '스트리밍', '넷플릭스', '판', '스', '우파', '국내', '출시', '방향', '김 양원', '오늘', '첫', '소식', '넷플릭스', '대규모', '구조조정', '식이', '지난', '분기', '넷플릭스', '처음', '유료'], ['김새론', '사진', '골드', '메달리스트', '이데일리', '스타', '김가영', '기자', '넷플릭스', '사 냥개', '진', '음주운전', '물의', '배우', '김새론', '편집', '논의', '중이', '일', '넷플릭스', '관계자', '이 데일리', '김새론', '씨'], ['토종', '웨이브', '티빙', '왓챠', '등', '글로벌', '독자', '위', '넷플릭스', '도', '지난달', '한국', '독자', '빅데이터', '분석', '솔루션', '모바일', '인덱스', '넷플릭스', '웨이브', '티빙', '쿠팡', '플레이', '디즈니', '플러스', '시즌'], ['서울', '전광판', '넷플릭스', '광고', '송출되', '모습', ' 사진', '연합뉴스', '지난해', '오징어', '게임', '....</pre>

In [54]: 형태소 토큰화를 하여 명사 토큰oka.nouns()만 추출 후 리스트를 구성

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

1. 명사 단어 추출하기 - 먼저, 긍정 감성 뉴스에서 형태소 분석을 하여 명사를 추출

In [56]:	<pre>POS_description_noun_join = [] for d in POS_description_noun_tk: d2 = [w for w in d if len(w) > 1] # 길이가 1보다 큰 토큰만 추출 POS_description_noun_join.append(" ".join(d2)) # 토큰 연결하여 리스트 구성</pre>
In [57]:	<pre>print(POS_description_noun_join) # 작업 확인용 출력</pre>
Out[57]:	<pre>['뉴미디어 트렌드 라이브 스트리밍 넷플릭스 우파 국내 출시 방향 김양원 오늘 소식 넷플릭스 대규모 구조조정 식이 지난 분기 넷플릭스 처음 유료', '김새론 사진 골드 메 달리스트 이데일리 스타 김가영 기자 넷플릭스 사냥개 음주운전 물의 배우 김새론 편 집 논의 중이 넷플릭스 관계자 이데일리 김새론', '토종 웨이브 티빙 왓챠 글로벌 독자 넷플릭스 지난달 한국 독자 빅데이터 분석 솔루션 모바일 인덱스 넷플릭스 웨이브 티 빙 쿠팡 플레이 디즈니 플러스 시즌', '서울 전광판 넷플릭스 광고 송출되 모습 사진 연합뉴스 지난해 오징어 게임 대박 온라인 동영상 서비스 넷플릭스 하락 상치 독자 수가 하락 본사 인력 감축',...</pre>

In [57]: 토큰의 길이가 1인 것은 제외 후 연결join()하여 리스트를 구성

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

2. 부정 감성 뉴스에도 같은 작업 수행

In [58]:	<pre>NEG_description = NEG_data_df['description'] NEG_description_noun_tk = [] NEG_description_noun_join = [] for d in NEG_description: NEG_description_noun_tk.append(oka.nouns(d)) #명사 형태소만 추출 for d in NEG_description_noun_tk: d2 = [w for w in d if len(w) > 1] #길이가 1보다 큰 토큰만 추출 NEG_description_noun_join.append(" ".join(d2)) # 토큰 연결하여 리스트 구성 print(NEG_description_noun_join)</pre>
Out[58]:	<p>['넷플릭스 리지 시리즈 사냥개 추가 촬영 로터 취재 결과 김새론 넷플릭스 리지 시리즈 제작 콘 텐츠 사냥개 사실 하차 수순', '세계 넷플릭스 지난 분기 유료 가입자 감소 사태 증시 업계 사용 자 감소 빅데이터 분석 플랫폼 모바일 인덱스 한국 시장 업체 넷플릭스', '넷플릭스 자회사 스캔 라인 코리아 통해 달러 억원 국내 투자 계획 관련 지난 산업 통상 자원부 바이든 미국 대통령 방한 계기 넷플릭스 자회사 스캔 라인 코리아', '넷플릭스 한국 창작 생태계 파트너 우정은 마치 가필드 넷플릭스 대외 정책 부문 부사 지난해 고문 넷플릭스 한국 특수 효과 제작 시설 달러 억 원', '음주운전 사고 물의 배우 김새론 넷플릭스 시리즈 사냥개 촬영 참여 넷플릭스 하차 입장 넷플릭스 사냥개 관계자 연예 제작 진과', '넷플릭스 브로드밴드 사용 논쟁 양상 국내 유럽 넷플 릭스 사용 요구 전세계 문제 번지 최근 넷플릭스 자신 콘텐츠', '세계 최대 온라인 동영상 서비 스 기업 넷플릭스 주가 약세...']</p>

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

3. TF-IDF 기반 DTM 구성하기 - 긍정 감성 뉴스에 대한 DTM을 구성

문서에 나타난 단어의 TF-IDF를 구하는 작업은 문서 단위로 토큰이 연결되어 있는 POS_description_noun_join을 사용

In [59]:	<pre>POS_tfidf = TfidfVectorizer(tokenizer = okt_tokenizer, min_df = 2) POS_dtm = POS_tfidf.fit_transform(POS_description_noun_join)</pre>
In [60]:	<pre>POS_vocab = dict() for idx, word in enumerate(POS_tfidf.get_feature_names()): POS_vocab[word] = POS_dtm.getcol(idx).sum() POS_words = sorted(POS_vocab.items(), key = lambda x: x[1], reverse = True)</pre>
In [61]:	<pre>POS_words #작업 확인용 출력</pre>
Out[61]:	<pre>[('넷플릭스', 38.5481006912417), ('시리즈', 15.99828671413098), ('영화', 14.072278266336536), ('공개', 13.585390349693917), ('안나라수마나라', 12.340212389292661), ...]</pre>

In [59]: TfidfVectorizer 객체를 생성하고 POS_description_noun_join에 대해 TF-IDF 값을 구하여 DTM을 구성

In [60]: DTM의 단어 `get_feature_names()` 마다 컬럼의 합 `getcol(idx).sum()` 을 구하여 단어별 TFIDF 값의 합을 구하고 내림차순으로 정렬

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

4. 부정 감성 뉴스인 NEG_description_noun_join에 대해서도 같은 작업을 수행

In [61]:	<pre>NEG_tfidf = TfidfVectorizer(tokenizer = okt_tokenizer, min_df = 2) NEG_dtm = NEG_tfidf.fit_transform(NEG_description_noun_join)</pre>
In [62]:	<pre>NEG_vocab = dict() for idx, word in enumerate(NEG_tfidf.get_feature_names()): NEG_vocab[word] = NEG_dtm.getcol(idx).sum() NEG_words = sorted(NEG_vocab.items(), key = lambda x: x[1], reverse = True)</pre>
In [63]:	<pre>NEG_words #작업 확인용 출력</pre>
Out[63]:	<pre>[('넷플릭스', 68.07051490368431), ('서비스', 20.07046307312205), ('투자', 19.93888415305736), ('동영상', 17.869130785824858), ('자회사', 17.654560662391688), ('미국', 17.165277454792395), ('가입자', 16.94620910915019), ...]</pre>

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

5. DTM 기반 단어 사전의 상위 단어로 바 차트 그리기

In [64]:	<pre>import matplotlib.pyplot as plt plt.rc('font', family='Malgun Gothic') # 한글 폰트 지정 max = 15 #바 차트에 나타낼 단어의 수</pre>
In [65]:	<pre>plt.figure(figsize = (15, 5)) plt.subplot(1, 2, 1) plt.bar(range(max), [i[1] for i in POS_words[:max]], color="blue") plt.title("긍정 뉴스의 단어 상위 %d개" %max, fontsize=15) plt.xlabel("단어", fontsize=12) plt.ylabel("TF-IDF의 합", fontsize=12) plt.xticks(range(max), [i[0] for i in POS_words[:max]], rotation=70) plt.ylim([0, 70]) # y축 설정 plt.subplot(1, 2, 2) plt.bar(range(max), [i[1] for i in NEG_words[:max]], color="red") plt.title("부정 뉴스의 단어 상위 %d개" %max, fontsize=15) plt.xlabel("단어", fontsize=12) plt.ylabel("TF-IDF의 합", fontsize=12) plt.xticks(range(max), [i[0] for i in NEG_words[:max]], rotation=70) plt.ylim([0, 70]) # y축 설정</pre>

In [64]: 바 차트를 그리기 위해 matplotlib 패키지 임포트하고 한글을 표시하기 위해 한글 폰트를 설정

바 차트에 나타낼 단어 개수를 max에 설정

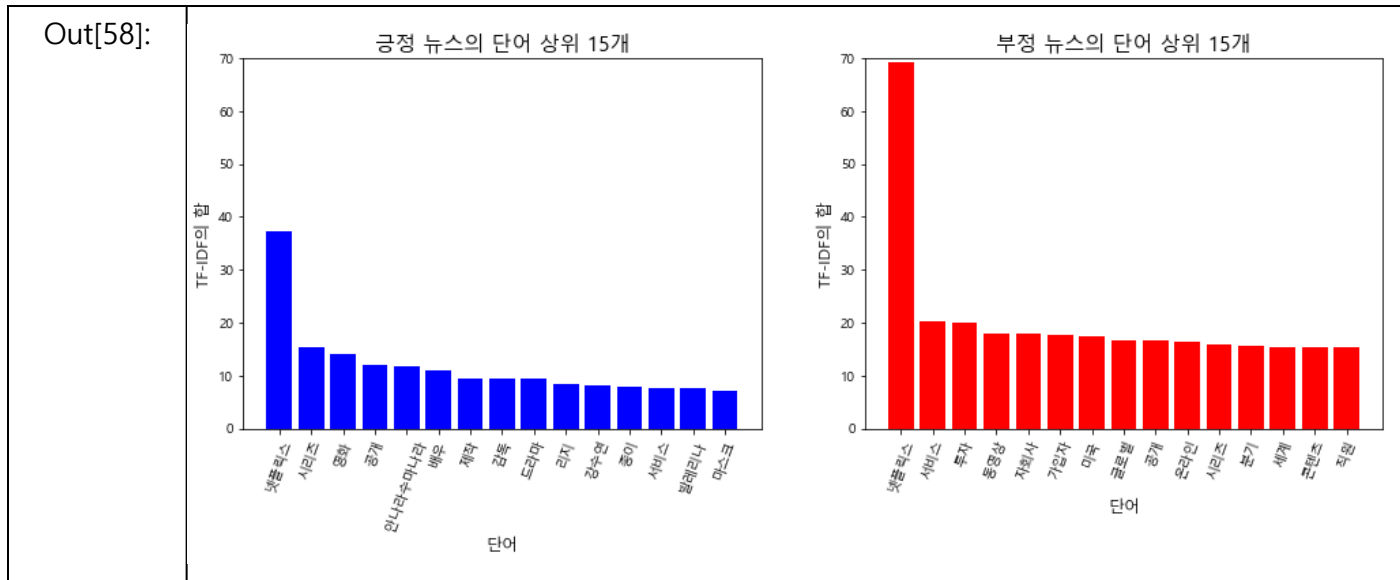
In [65]: 긍정은 오른쪽에 파란색으로, 부정은 왼쪽에 빨간색으로 막대 그래프 생성. (두 그래프를 병렬적으로 생성.)

02. [감성 분석, 바 차트] Netflix 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

5. 긍정 뉴스와 부정 뉴스에 많이 나타난 단어를 바 차트로 나타냄



• 감성 결과

- 부정 뉴스와 긍정 뉴스의 비율이 약 6.5 : 3.5로 부정 뉴스가 약 2배 정도 더 많음.
- 긍정 뉴스에서는 넷플릭스가 제공하는 시리즈별 영화, 드라마 혹은 최근 오픈한 영상들이 좋다는 의견이 대다수 존재.
- 반면에 부정 뉴스에는 서비스, 투자가 별로고 신선하지 못한 콘텐츠가 있다. 즉 옛날에 비해 비교적 다양하지 못하다는 의견이 존재함.

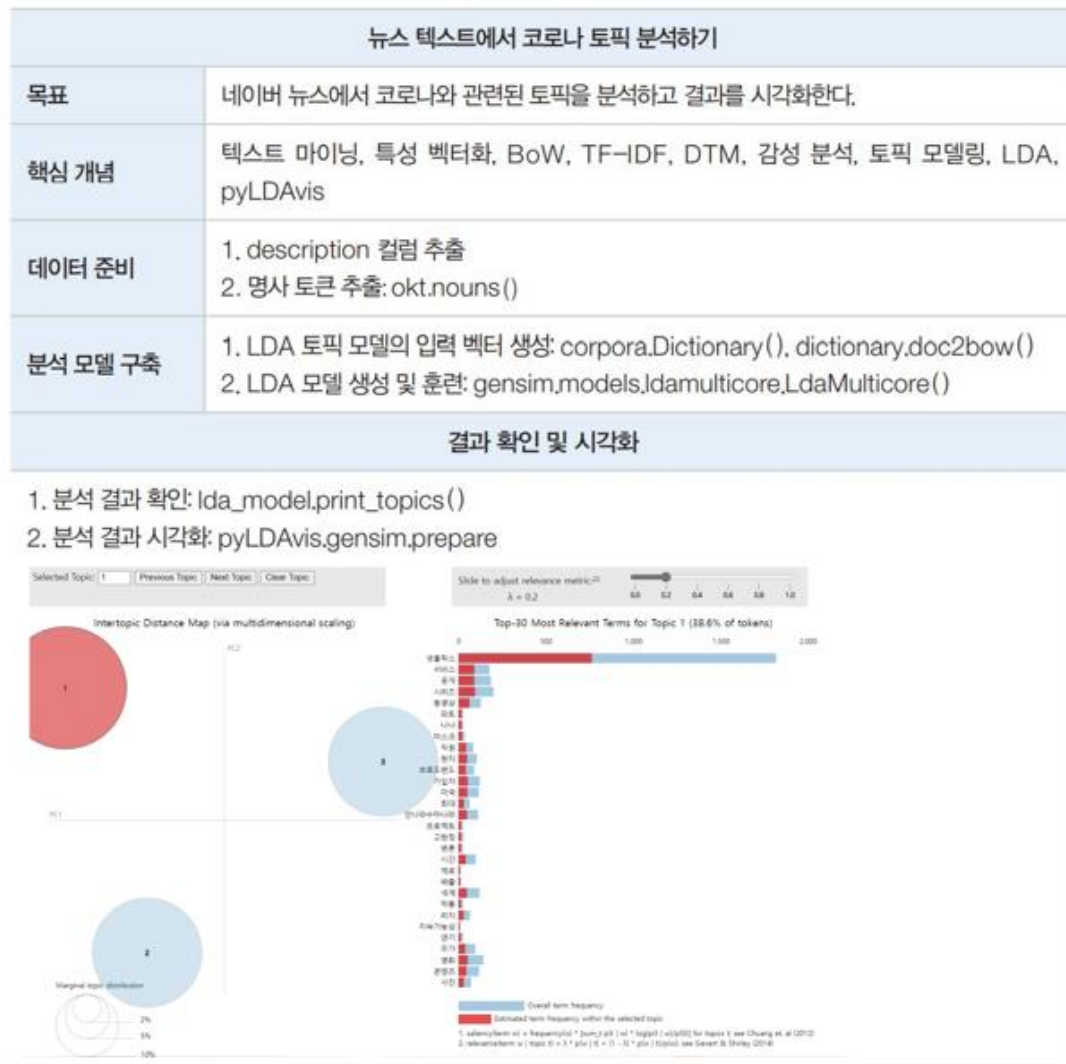
Chapter

03

Netflix 뉴스 텍스트에서 토픽 분석하기

03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 분석 미리보기



03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 목표설정

- 목표: 네이버 뉴스에서 '넷플릭스'와 관련된 어떤 토픽이 있는지 분석
머신러닝 기반의 LDA 토픽 모델을 사용

■ 데이터 준비

- 토픽 분석에 사용할 데이터는 앞에서 크롤링한 네이버 뉴스의 전체 description을 사용
- 토픽 모델은 단어별 확률 분포를 분석하므로 명사를 추출한 단어(토큰) 상태의 리스트를 준비

In [66]:	<code>description = data_df['description']</code>
In [67]:	<code>description_noun_tk = [] for d in description: description_noun_tk.append(oka.nouns(d)) #명사 형태소만 추출</code>
In [68]:	<code>description_noun_tk2 = [] for d in description_noun_tk: item = [i for i in d if len(i) > 1] #토큰 길이가 1보다 큰 것만 추출 description_noun_tk2.append(item)</code>
In [69]:	<code>print(description_noun_tk2)</code>
Out[69]:	<code>['넷플릭스', '리지', '시리즈', '사냥개', '추가', '촬영', '로터', '취재', '결과', '김새론', '넷플릭스', '리지', '시리즈', '제작', '콘텐츠', '사냥개', '사실', '하차', '수순'], ['세계', '넷플릭스', '지난', '분 기', '유료', '가입자', '감소', '사태', '증시', '업계', '사용자', '감소', '빅데이터', '분석', '플랫폼', '모바일', '인덱스', '한국', '시장', '업체', '넷플릭스'], ['넷플릭스', '자회사', '스캔', '라인', '코리 아', '통해', '달러', '억원', '국내', '투자', '계획', '관련', '지난', '산업', '통상', '자원부', '바이든', '미국', '대통령', '방한', '계기', '넷플릭스', '자회사', '스캔', '라인', '코리아'],...</code>

03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 분석 모델 구축

1. 토픽 분석을 위한 LDA 모델 구축하기

1. gensim은 추가로 설치해야 하는 패키지이므로 최초 한번은 다음과 같이 !pip install을 이용해 설치

In [70]:	!pip install gensim
----------	----------------------------

1. 패키지를 설치한 후에 필요한 모듈을 임포트

In [71]:	import gensim import gensim.corpora as corpora
----------	---

1. LDA 토픽 모델의 입력 벡터 생성하기

In [72]:	dictionary = corpora. Dictionary (description_noun_tk2)
In [73]:	print(dictionary[1]) <i>#작업 확인용 출력</i>
Out[73]:	김새론
In [74]:	corpus = [dictionary. doc2bow (word) for word in description_noun_tk2]
In [75]:	print(corpus)
Out[75]:	[[(0, 1), (1, 1), (2, 2), (3, 1), (4, 2), (5, 2), (6, 1), (7, 1), (8, 2), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1), (14, 1)], [(2, 2), (15, 1), (16, 2), (17, 1), (18, 1), (19, 1), (20, 1), (21, 1), (22, 1), (23, 1), (24, 1), (25, 1), (26, 1), (27, 1), (28, 1), (29, 1), (30, 1), (31, 1), (32, 1)], [(2, 2), (30, 1), (33, 1), (34, 1), (35, 1), (36, 1), (37, 1), (38, 1)], ...]

In [72]: description_noun_tk2에 포함된 단어에 대해 사전을 구성 `corpora.Dictionary()`

In [73]: 구성된 사전의 내용을 확인하기 위해 1번 단어 `dictionary[1]`를 출력

In [74]: 단어 사전 `dictionary`의 단어에 대해 BoW를 구하여 `doc2bow()`, 단어 뭉치 `corpus`를 구성

In [75]: 단어 뭉치 `corpus`를 출력하여 (word_id, work_count)의 BoW 구성을 확인

03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 분석 모델 구축

1. 토픽 분석을 위한 LDA 모델 구축하기

4. LDA 토픽 모델의 생성 및 훈련하기

토픽의 개수를 4로 설정

gensim 패키지의 LDA 모듈을 이용하여 토픽 모델 객체인 `lda_model`을 생성

In [76]:	<code>k = 3 # 토픽의 개수 설정</code>
In [77]:	<code>lda_model = gensim.models.ldamulticore.LdaMulticore(corpus, iterations = 12, num_topics = k, id2word = dictionary, passes = 1, workers = 10)</code>

In [77]: `lda_model`의 파라미터 설명

- `iterations` (int, 선택 사항) – 코퍼스의 주제 분포를 추론 할 때 코퍼스를 통한 최대 반복 횟수.
- `num_topics` : 분류할 총 주제의 수 (= 토픽의 개수)
- `id2word` : 사전에 만들어둔 정수 인코딩 딕셔너리 (앞에 코드에서 생성한 `dictionary` 객체)
- `passes` : 모델을 학습시키는 코퍼스의 빈도
- `workers` (int, 선택 사항) – 병렬화에 사용할 작업자 프로세스 수입니다.

None 인 경우 사용 가능한 모든 코어 (`workers = cpu_count()-1`로 추정 됨)이 사용.

03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 결과 확인 및 시각화

1. 분석 결과 확인하기

1. 토픽 모델 객체에 저장되어 있는 토픽 분석 결과를 `lda_model.print_topics()` 함수를 사용하여 출력

In [78]:	<code>print(lda_model.print_topics(num_topics = k, num_words = 15))</code>
Out[78]:	<pre>[(0, '0.067*넷플릭스', 0.009*공개', 0.008*글로벌', 0.007*시리즈', 0.007*가입자', 0.007*투자', 0.007*서비스', 0.006*감소', 0.006*한국', 0.006*최근', 0.006*주가', 0.006*현지', 0.006*온라인', 0.005*직원', 0.005*동영상'), (1, '0.104*넷플릭스', 0.014*시리즈', 0.010*공개', 0.010*안나라수마나라', 0.009*서비스', 0.007*온라인', 0.007*콘텐츠', 0.007*영화', 0.007*글로벌', 0.006*가입자', 0.006*동영상', 0.006*세계', 0.005*투자', 0.005*분기', 0.005*배우'), (2, '0.082*넷플릭스', 0.009*한국', 0.007*분기', 0.007*영화', 0.007*미국', 0.006*브로드밴드', 0.006*가입자', 0.006*서비스', 0.006*투자', 0.006*현지', 0.006*국내', 0.006*콘텐츠', 0.006*자회사', 0.006*주가', 0.006*시리즈'), (3, '0.089*넷플릭스', 0.011*서비스', 0.010*공개', 0.010*시리즈', 0.009*동영상', 0.008*투자', 0.008*영화', 0.008*세계', 0.008*미국', 0.007*온라인', 0.006*글로벌', 0.006*스트리밍', 0.006*한국', 0.006*최근', 0.006*시간')]</pre>

• 분석 결과

- `num_words = 15`에 따라 토픽을 구성하는 주요 단어 15개가 토픽에 대한 영향력 비율과 함께 출력된 것을 확인.
- 네이버 뉴스를 크롤링할 때 검색어로 '넷플릭스'를 사용했기 때문에 모든 토픽에서 '넷플릭스' 단어가 압도적으로 많이 나옴.

03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 결과 확인 및 시각화

1. 분석 결과 확인하기

2. 분석 결과에 대한 정리와 분석을 실시

주요 단어를 고려하여 각 토픽 내용을 설명하는 레이블을 결정

토픽번호	주요단어(15개)	토픽 레이블
0	0.090* "넷플릭스" + 0.009* "시리즈" + 0.008*"공개" + 0.007*"제작" + 0.007* "한국" + 0.007* "글로벌" + 0.007* "서비스" + 0.006*"안나라수 마나라" + 0.006*"영화" + 0.006* "투자" + 0.006* "지난" + 0.006* "세계 " + 0.006* "온라인" + 0.006*"콘텐츠" + 0.005* "배우"	콘텐츠(영화, 안나라수마나라), 공개, 제작
1	'0.076* "넷플릭스" + 0.009* "투자" + 0.008* "자회사" + 0.008* "영화" + 0.008* "시리즈" + 0.007* "서비스" + 0.007* "분기" + 0.007*"감소" + 0.007* "최근" + 0.007* "글로벌" + 0.007* "동영상" + 0.007* "공개" + 0.006* "온라인" + 0.006* "국내" + 0.006*"가입자" '),	가입자, 감소
2	'0.096* "넷플릭스" + 0.012* "시리즈" + 0.012* "공개" + 0.011* "서비스" + 0.008* "동영상" + 0.007* "가입자" + 0.007* "영화" + 0.007*"미국" + 0.006*"현지" + 0.006* "안나라수마나라" + 0.006* "세계" + 0.006* " 글로벌" + 0.006* "온라인" + 0.006* "한국" + 0.006* "콘텐츠" '])]	현지, 미국

03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 결과 확인 및 시각화

2. 분석 결과 시각화하기

1. LDA 토픽 분석의 결과를 시각화하기 위해 pyLDAvis 패키지의 pyLDAvis.gensim.prepare() 함수를 사용
토픽 분석 결과를 가지고 있는 lda_model 객체와 단어 뭉치, 단어 사전을 매개변수로 사용
pyLDAvis은 추가 설치해야 하는 패키지이므로 최초 한번은 !pip install을 이용해 설치

In [79]:	!pip install pyLDAvis
In [80]:	<pre>import pyLDAvis.gensim lda_vis = pyLDAvis.gensim.prepare(lda_model, corpus, dictionary) pyLDAvis.display(lda_vis) # 파일로 저장 pyLDAvis.save_html(lda_vis, 'data/'+file_name+"3_vis.html")</pre>

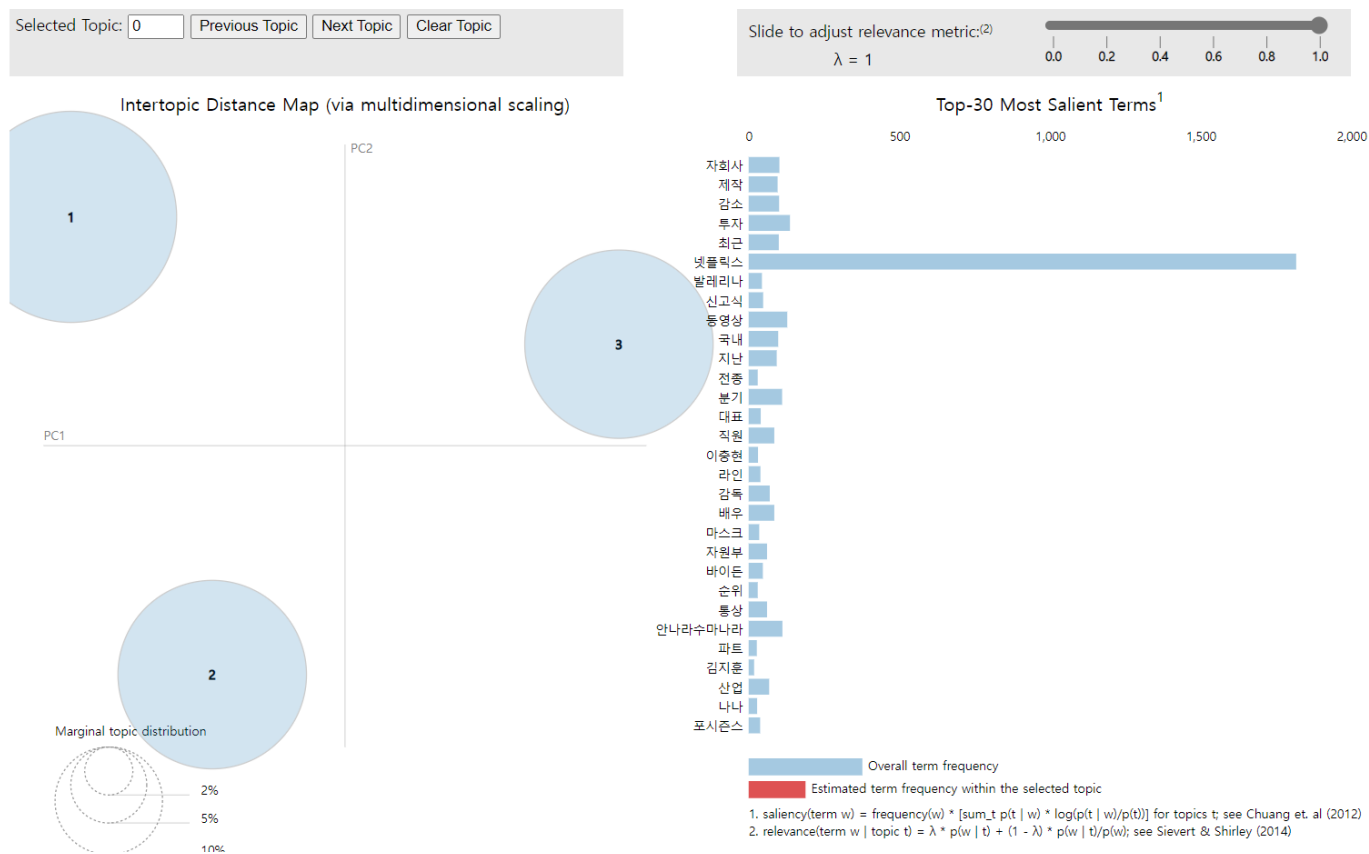
- In [80]: pyLDAvis를 파일로 저장.
pyLDAvis는 웹 브라우저 창에 표시되므로 저장 파일 형식도 html로 설정.

03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 결과 확인 및 시각화

2. 분석 결과 시각화하기

2. 왼쪽 영역에는 토픽 간 거리 지도가 있고, 오른쪽 영역에는 토픽에서 관련성 높은 30개 단어에 대한 바 차트가 있음
왼쪽 영역에 보이는 분포에서 원들이 많이 겹쳐져 있다면 토픽의 유사도가 높다는 의미이기 때문에
토픽의 개수 k 값을 다르게 하여 LDA 모델을 다시 실행
(최종 토픽 레이블을 만들기 위해 유사도가 적은 그룹을 선택 예정)

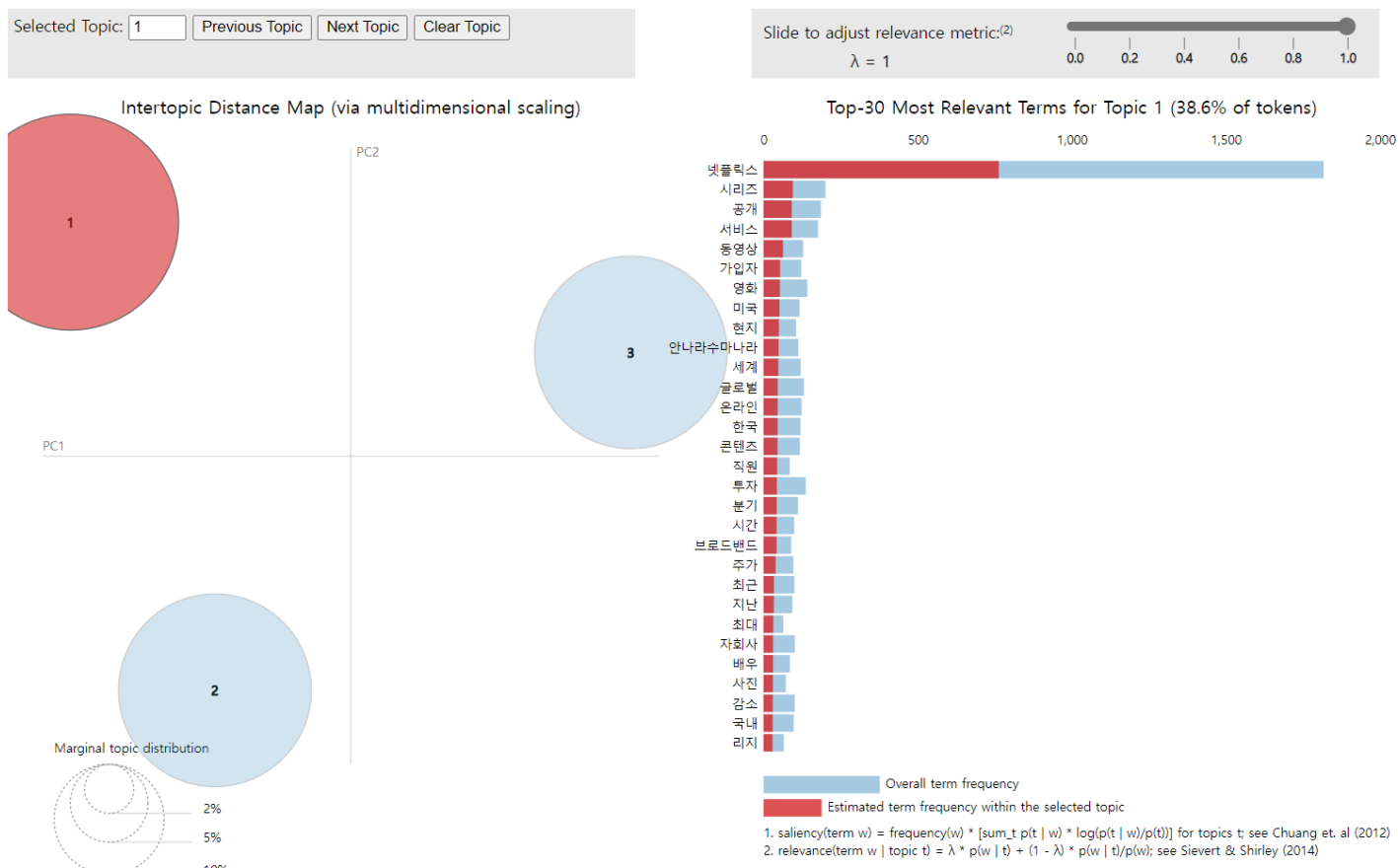


03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 결과 확인 및 시각화

2. 분석 결과 시각화하기

1. 왼쪽 영역의 토픽 간 거리 지도 영역에서 토픽 버블을 클릭해서 선택하면 오른쪽 영역에 토픽에 대한 토큰 비율이 상위 단어 30개의 바 차트에 붉은 색으로 나타남
2. 오른쪽 영역에 있는 단어 위로 마우스를 이동하면 단어의 토픽 영향력에 따라 토픽 버블의 크기가 조정되어 변함

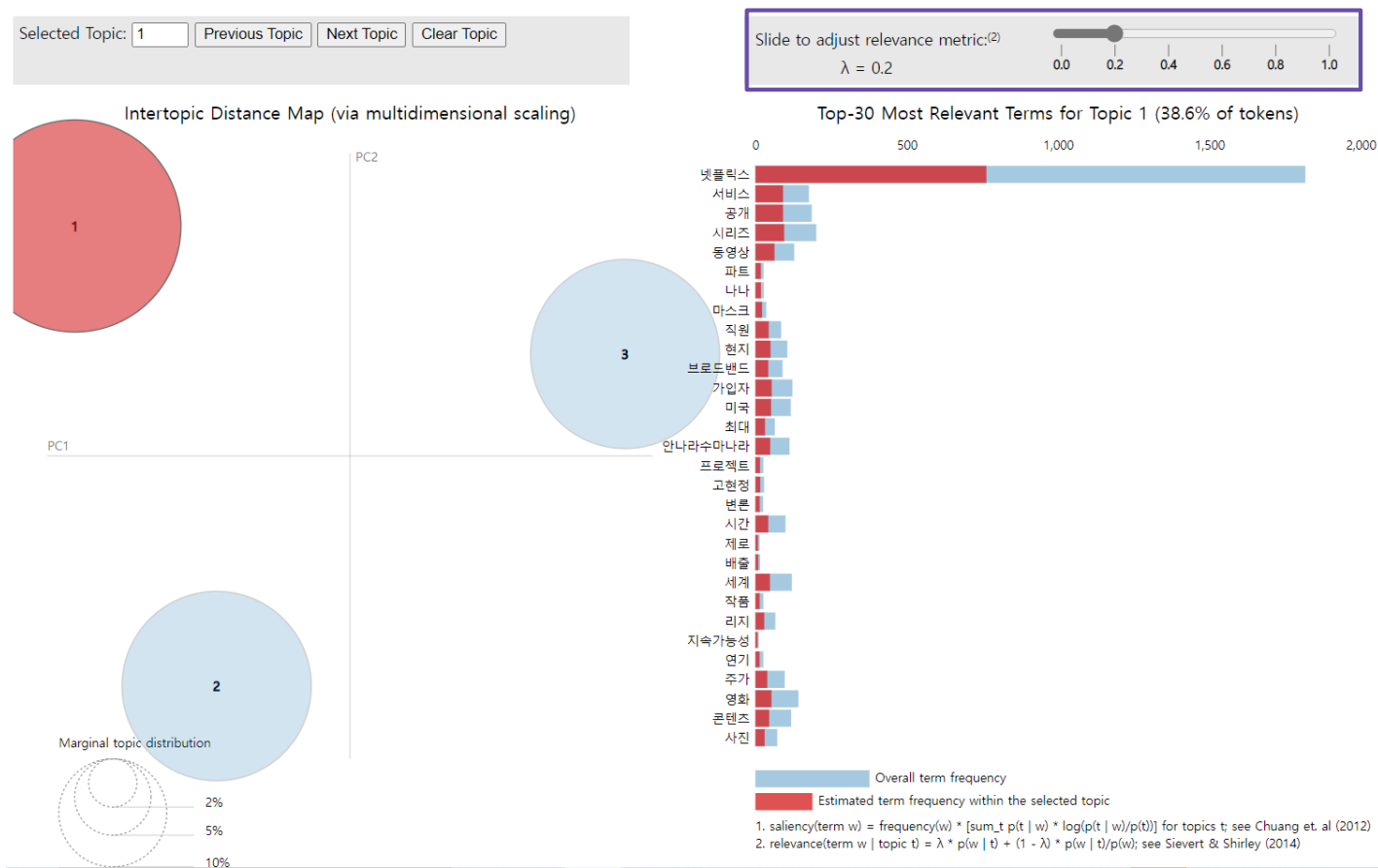


03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 결과 확인 및 시각화

2. 분석 결과 시각화하기

2. 오른쪽 영역 상단에 있는 관련성 메트릭스 조정 슬라이드를 움직이면 현재 선택한 토픽에 특화되어 많이 출현하는 단어를 확인할 수 있음



03. [토픽 분석, LDA 토픽 모델] Netflix 뉴스 텍스트에서 토픽 분석하기

■ 결과 확인 및 시각화

2. 분석 결과 시각화하기

- 두 결과를 비교하여 종합적으로 분석

lda_model.print_topics()에 의한 결과			pyLDavis에 의한 결과			최종 토픽 레이블
토픽 번호	주요단어(15개)	토픽 레이블	토픽 번호	토크 분포 비율	토픽 특화 단어(15개) ($\lambda = 0.2$)	
0	0.090*" 넷플릭스 " + 0.009*" 시리즈 " + 0.008*" 공개" " + 0.007*" 제작" " + 0.007*" 한국 " + 0.007*" 글로벌 " + 0.007*" 서비스 " + 0.006*" 안나라수마나라" " + 0.006*" 영화" " + 0.006*" 투자 " + 0.006*" 지난 " + 0.006*" 세계 " + 0.006*" 온라인 " + 0.006*" 콘텐츠" " + 0.005*" 배우 "	콘텐츠 (영화, 안나라수마나라), 공개, 제작	2	30.7%	넷플릭스, 제작 , 발레리나, 지난, 배우 , 감독, 안나라수마나라 , 한국, 전종, 김지훈, 순위, 동경제, 계정, 예정 , 애니메이션	넷플릭스 콘텐츠 제작, 공개 예정
1	'0.076*" 넷플릭스 " + 0.009*" 투자 " + 0.008*" 자회사 " + 0.008*" 영화 " + 0.008*" 시리즈 " + 0.007*" 서비스 " + 0.007*" 분기 " + 0.007*" 감소" " + 0.007*" 최근 " + 0.007*" 글로벌 " + 0.007*" 동영상 " + 0.007*" 공개 " + 0.006*" 온라인 " + 0.006*" 국내 " + 0.006*" 가입자" "),	가입자, 감소	3	30.7%	자회사, 넷플릭스, 최근, 감소 , 투자 , 신고식, 분기 , 대표, 라인, 통상, 바이든, 국내, 산업, 억원, 포시즌스	넷플릭스 가입자 감소
2	'0.096*" 넷플릭스 " + 0.012*" 시리즈 " + 0.012*" 공개 " + 0.011*" 서비스 " + 0.008*" 동영상 " + 0.007*" 가입자 " + 0.007*" 영화 " + 0.007*" 미국" " + 0.006*" 현지" " + 0.006*" 안나라수마나라 " + 0.006*" 세계 " + 0.006*" 글로벌 " + 0.006*" 온라인 " + 0.006*" 한국 " + 0.006*" 콘텐츠 "')]	현지, 미국	1	38.6%	넷플릭스, 서비스, 공개, 시리즈, 동영상, 파트, 나나, 마스크, 직원, 현지 , 브로드밴드 , 가입자, 미국, 최대, 안나라수마나라	넷플릭스, SK 브로드밴드 현지(국내) 망 사용료

감사합니다.