목차

학습목표

- 텍스트 마이닝 기법 중에서 감성 분석과 토픽 분석을 이해
- 로지스틱 회귀를 이용한 감성 분류 모델을 학습하고 분석을 수행
- 토픽 모델링 기법을 이해하고 토픽 분석을 학습.

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 분석 미리보기

영화 리뷰 데이터로 감성 예측하기	
목표	영화 리뷰 데이터에 감성 분석 모델을 구축하여 새로운 데이터의 감성을 분석한다.
핵심 개념	텍스트 마이닝, 특성 벡터화, BoW, TF-IDF, DTM, 감성 분석, 토픽 모델링, LDA, pyLDAvis
데이터 수집	영화 리뷰 데이터: https://github.com/e9t/nsmc에서 다운로드

데이터 준비 및 탐색

훈련용 / 평가용 데이터
- 파일 불러오기: pd.read_csv()
- 정보 확인: nsmc_train_df.info() / nsmc_test_df.info()
- 결측 데이터 제거: _train_df['document'].notnull() / nsmc_test_df['document'].notnull()
- 레이블 확인: nsmc_train_df['label'].value_counts() / nsmc_test_df['label'].value_counts()
- 한글 외의 문자 제거: lambda x: re.sub(r'[^ ㄱ-ㅣ 가-힣]+', "", x)

분석 모델 구축

1. 특성 벡터화
- 형태소 기반 토큰화: Okt()
- TF-IDF 기반 벡터 생성: TfidfVectorizer()

2. 감성 분석 모델 구축
- 로지스틱 회귀 기반 분석 모델 생성
- 최적 하이퍼 매개변수 도출: GridSearchCV()
- 최적 하이퍼 매개변수 모델의 훈련: SA_lr_best

분석 모델 평가

1. 평가 데이터를 이용한 모델 정확도 확인
- 평가용 데이터의 피터 벡터화: tfidf.transform()
- 감성 예측: SA_lr_best.predict()
- 정확도: accuracy_score()

2. 새로운 텍스트에 대한 감성 예측
- 텍스트 입력: input()
- 텍스트 전처리: 한글 이외의 문자 제거
- 텍스트의 특성 벡터화: tfidf.transform()
- 감성 예측값 출력

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 목표설정

- 영화 리뷰 데이터에 텍스트 마이닝의 감성 분석 기술을 사용하여 감성 분석 모델을 구축한 뒤 새로운 데이터에 대한 감성을 분석
- 토픽 모델링을 수행하여 관련 토픽도 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 + 토픽 모델링] 영화 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 d에 등장한 단어 t의 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("데이터", 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

- d에 등장한 단어 t의 TF-IDF $tf\text{-}idf(t, d) = tf(t,d) \times idf(t,d)$
- (역문서 빈도)idf(t,d) $idf(t, d) = \log \frac{n_d}{1+df(d, t)}$
- 는 : 전체 n^d서의 개수
- df(d,t)는 단어 t가 포함된 문서 d의 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("데이터", doc#1) = 0.52

01. [감성 분석 + 토픽 모델링] 영화 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 + 토픽 모델링] 영화 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

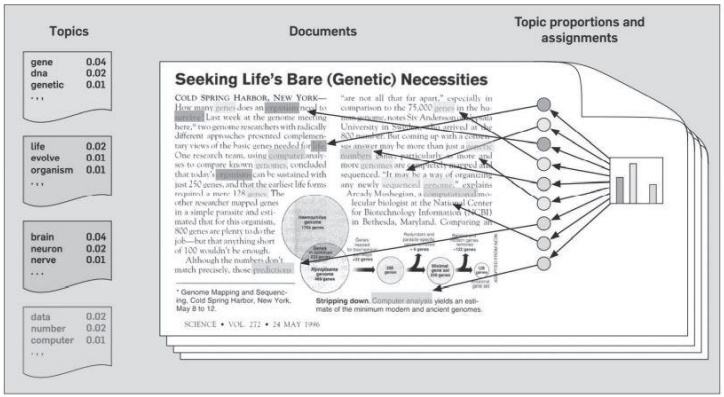


그림 13-3 문서에 LDA를 적용한 토픽 도출 예시(출처: 브리티시컬럼비아대학교)

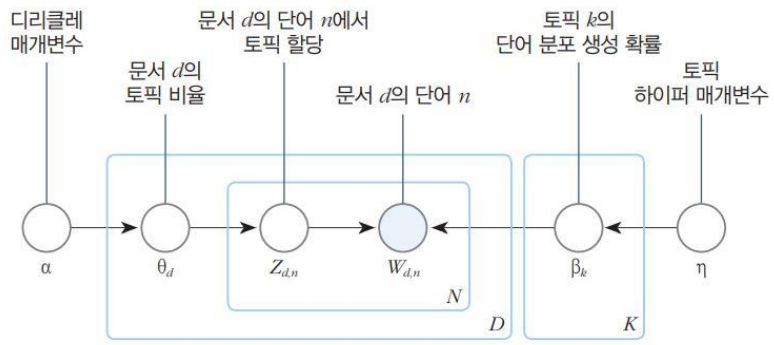


그림 13-4 LDA 그래픽 모델

▪ pyLDAvis

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

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 데이터 수집

1. ratings.txt, ratings_test.txt, ratings_ train.txt 파일을 차례대로 다운로드

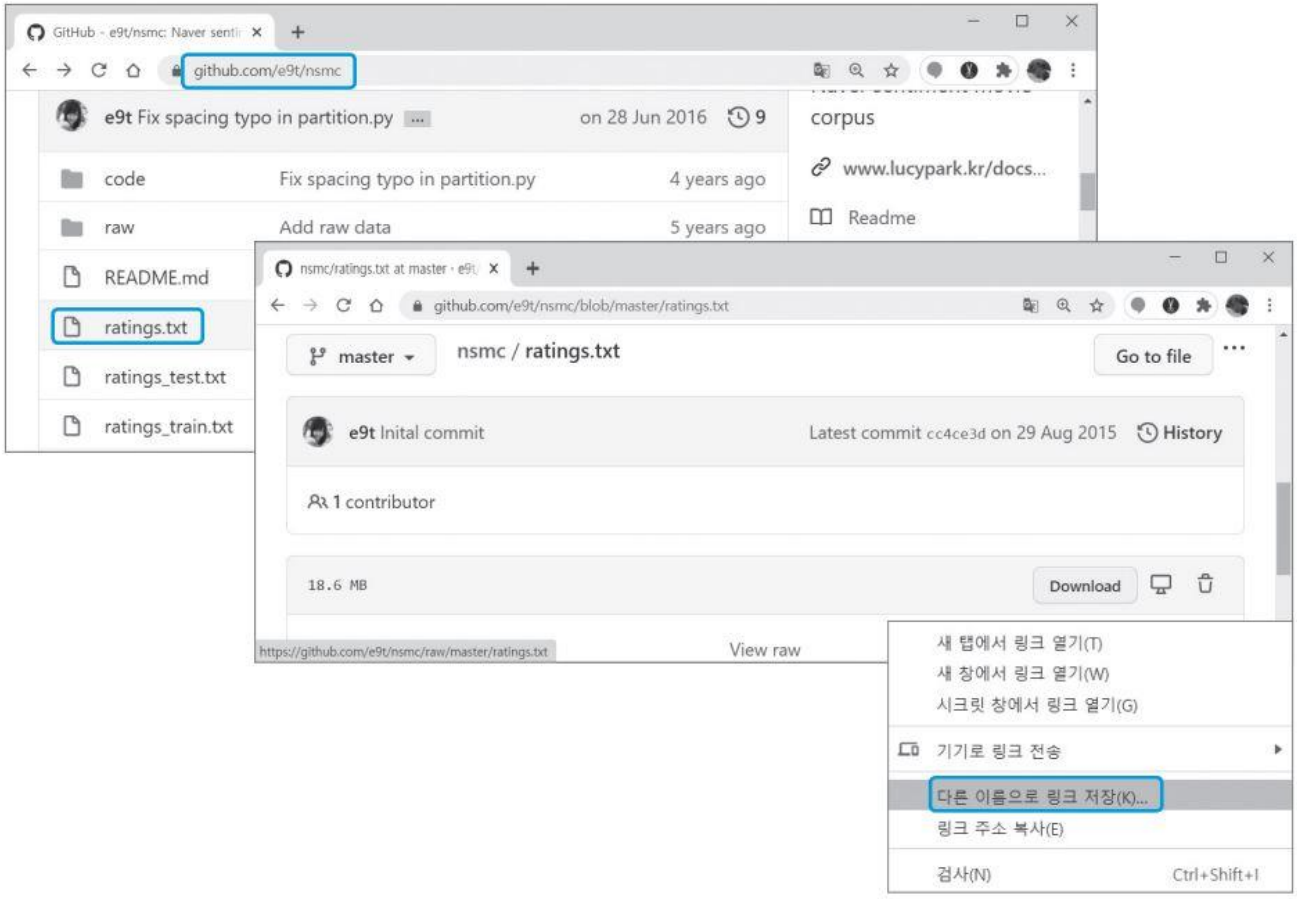


그림 13-5 깃허브에 공개된 데이터셋 다운로드

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 데이터 수집

2. 다운로드한 파일을 열어서 내용을 확인

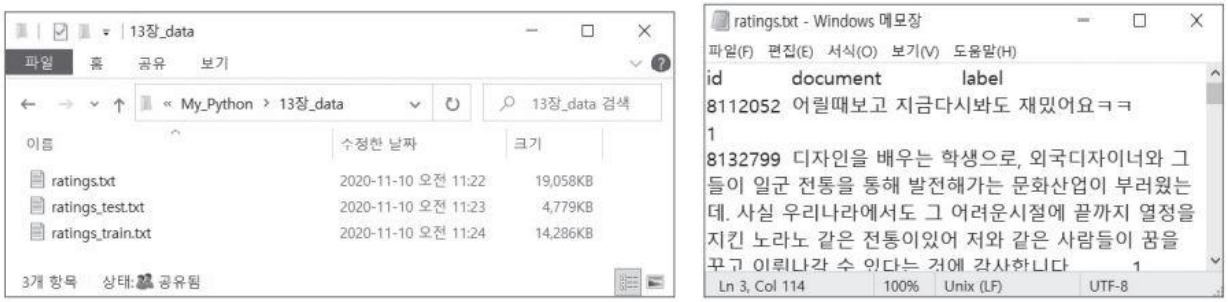


그림 13-6 다운로드한 파일 확인

- ratings.txt 파일: 네이버 영화 페이지에서 리뷰를 크롤링하여 수집한 200K 용량의 데이터 파일
- ratings_test.txt 파일: 50K를 평가용으로 분리한 파일
- ratings_train.txt 파일: 150K를 훈련용으로 준비한 것

- 파일 내용은 3개의 컬럼 (id, document, label)이 탭(₩t)으로 분리되어 있음
- label 컬럼은 감성 분류 클래스 값
- 1~10점의 평점 중에서 중립적인 평점인 5~8점은 제외하고 1~4점을 부정 감성 0으로, 9~10점을 긍정 감성 1로 표시

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 데이터 준비 및 탐색

1. pandas 버전 확인하기

1. 주피터 노트북에서 '감성분석'으로 노트북 페이지를 추가하고, pandas 버전을 확인

In [ ]:
```
import pandas as pd
pd.show_versions()
```

Out[ ]:
```
INSTALLED VERSIONS
------------------
commit           : 67a3d4241ab84919856b84fc9ebc9abcbe66c6b3
python           : 3.8.3.final.0
python-bits      : 64
OS               : Windows
OS-release       : 10
...

pandas           : 1.1.4
numpy            : 1.18.5
...
```

2. pandas 버전이 1.1.4 이상이 아니라면, --upgrade 명령으로 버전을 업그레이드

In [ ]:
```
!pip install --upgrade pandas
```

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 데이터 준비 및 탐색

2. 훈련용 데이터 준비하기

1. 다운로드한 파일 중에서 훈련용 데이터인 ratings_train.txt 파일을 주피터 노트북에서 로드

In [1]:

```
#warning 메시지 표시 안함
import warnings
warnings.filterwarnings(action = 'ignore')

import pandas as pd
```

In [2]:

```
nsmc_train_df = pd.read_csv('./13장_data/ratings_train.txt', encoding = 'utf8', sep = '₩t', engine = 'python')

nsmc_train_df.head()
```

Out[2]:

	id	document	label
0	9976970	아 더빙.. 진짜 짜증나네요 목소리	0
1	3819312	흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나	1
2	10265843	너무재밋었다그래서보는것을추천한다	0
3	9045019	교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정	0
4	6483659	사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 ...	1

In [2]: 훈련용 데이터 파일을 읽고pd.read_csv( ) 데이터프레임 객체nsmc_train_df에 저장

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 데이터 준비 및 탐색

2. 훈련용 데이터 준비하기

1. 다운로드한 파일 중에서 훈련용 데이터인 ratings_train.txt 파일을 주피터 노트북에서 로드

In [3]:	nsmc_train_df.**info**()
Out[3]:	<class 'pandas.core.frame.DataFrame'> RangeIndex: 150000 entries, 0 to 149999 Data columns (total 3 columns): id 150000 non-null int64 document 149995 non-null object label 150000 non-null int64 dtypes: int64(2), object(1) memory usage: 3.4+ MB

In [3]: 훈련용 데이터셋의 정보를 확인

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 데이터 준비 및 탐색

2. 훈련용 데이터 준비하기

2. 결측치 제거하기

In [4]:	nsmc_train_df = nsmc_train_df[nsmc_train_df['document'].notnull()]
In [5]:	nsmc_train_df.info()
Out[5]:	<class 'pandas.core.frame.DataFrame'> Int64Index: 149995 entries, 0 to 149999 Data columns (total 3 columns): id 149995 non-null int64 document 149995 non-null object label 149995 non-null int64 dtypes: int64(2), object(1) memory usage: 4.6+ MB
In [6]:	nsmc_train_df['label'].value_counts()
Out[6]:	0 75170 1 74825 Name: label, dtype: int64

In [4]: document 컬럼이 non-null인 샘플만 nsmc_train_df에 다시 저장

In [5]: 수정된 nsmc_train_df의 정보를 다시 확인

In [6]: 감성 분류 클래스의 구성을 확인

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 데이터 준비 및 탐색

2. 훈련용 데이터 준비하기

3. 한글 외의 문자 제거하기

In [7]:
```
import re
```

In [8]:
```
nsmc_train_df['document'] = nsmc_train_df['document']
        .apply(lambda x : re.sub(r'[^ ㄱ-ㅣ가-힣]+', " ", x))
nsmc_train_df.head()
```

Out[8]:

	id	document	label
0	9976970	아 더빙 진짜 짜증나네요 목소리	0
1	3819312	흠 포스터보고 초딩영화줄 오버연기조차 가볍지 않구나	1
2	10265843	너무재밋었다그래서보는것을추천한다	0
3	9045019	교도소 이야기구먼 솔직히 재미는 없다 평점 조정	0
4	6483659	사이몬페그의 익살스런 연기가 돋보였던 영화 스파이더맨에서 늙어보이기만 했던 커스틴 ...	1

In [7]: 정규식을 사용하기 위해 re 모듈을 임포트

In [8]: 'ㄱ'으로 시작하거나 '가'부터 '힣'까지의 문자를 제외한 나머지는 공백으로 치환

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 데이터 준비 및 탐색

2. 평가용 데이터 준비하기

1. 다운로드한 파일 중에서 평가용 데이터인 ratings_test.txt 파일을 로드

In [9]:

```
nsmc_test_df = pd.read_csv('./13장_data/ratings_test.txt', encoding = 'utf8', sep = '₩t', engine = 'python')
nsmc_test_df.head()
```

Out[9]:

	id	document	label
0	6270596	굳 ㅋ	1
1	9274899	GDNTOPCLASSINTHECLUB	0
2	8544678	뭐야 이 평점들은.... 나쁘진 않지만 10점 짜리는 더더욱 아니잖아	0
3	6825595	지루하지는 않은데 완전 막장임... 돈주고 보기에는....	0
4	6723715	3D만 아니었어도 별 다섯 개 줬을텐데.. 왜 3D로 나와서 제 심기를 불편하게 하죠??	0

In [10]:

```
nsmc_test_df.info()
```

Out[10]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 3 columns):
id              50000 non-null int64
document        49997 non-null object
label           50000 non-null int64
dtypes: int64(2), object(1)
memory usage: 1.1+ MB
```

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 데이터 준비 및 탐색

2. 평가용 데이터 준비하기

1. 다운로드한 파일 중에서 평가용 데이터인 ratings_test.txt 파일을 로드

In [11]:	**#document 칼럼이 Null인 샘플 제거** **nsmc_test_df = nsmc_test_df[nsmc_test_df['document'].notnull()]**
In [12]:	nsmc_test_df['label'].**value_counts()**
Out[12]:	1 25171 0 24826 Name: label, dtype: int64
In [13]:	nsmc_test_df['document'] = nsmc_test_df['document'] **.apply(lambda x : re.sub(r'[^ ㄱ-ㅣ가-힣]+', " ", x))**

01. [감성 분석 + 토픽 모델링] 영화 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 [14]:	from konlpy.tag import Okt okt = Okt()

In [15]:	def okt_tokenizer(text):
	  tokens = okt.morphs(text)
	  return tokens

In [16]:	from sklearn.feature_extraction.text import TfidfVectorizer tfidf = TfidfVectorizer(tokenizer = okt_tokenizer, ngram_range = (1, 2), min_df = 3, max_df = 0.9) tfidf.fit(nsmc_train_df['document']) nsmc_train_tfidf = tfidf.transform(nsmc_train_df['document'])

In [14]: 형태소 분석에 사용할 konlpy 패키지의 Okt 클래스를 임포트하고 okt 객체를 생성

In [15]: 문장을 토큰화하기 위해 okt_tokenizer 함수를 정의하고 okt.morphs() 함수를 사용하여 형태소 단위로 토큰화 작업을 수행

In [16]: 사이킷런의 TfidfVectorizer를 이용하여 TF-IDF 벡터화에 사용할 tfidf 객체를 생성

토큰 생성기tokenizer는 우리가 정의한 okt_tokenizer() 함수로 설정하고 토큰의 단어 크기ngram_range는 1~2개 단어로 함

토큰은 출현 빈도가 최소min_df 3번 이상이고 최대 max_df 90% 이하인 것만 사용

벡터화할 데이터nsmc_train_df['document']에 대해 벡터 모델 tfidf의 내부 설정값을 조정fit( )하고 벡터로 변환을 수행transform( )

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

2. 감성 분류 모델 구축하기

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

In [17]:	from sklearn.linear_model **import LogisticRegression** SA_lr = **LogisticRegression**(random_state = 0)
In [18]:	SA_lr.fit(nsmc_train_tfidf, nsmc_train_df['label'])
Out[18]:	LogisticRegression(random_state = 0)

In [17]: 사이킷런의 LogisticRegression 클래스에 대해 객체 SA_lr을 생성

In [18]: nsmc_train_tfidf를 독립변수 X로 하고 label 컬럼을 종속 변수 Y로 하여 로지스틱 회귀 모델SA_lr의 내부 설정값을 조정fit( )

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

In [19]:	**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)

In [19]: 하이퍼 매개변수 C에 대해 비교 검사를 할 6개 값[1, 3, 3.5, 4, 4.5, 5]을 params로 하고, 교차 검증cv을 3, 모형 비교 기준은 정확도로 설정scoring='accuracy'하여 GridSearchCV 객체를 생성

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 분석 모델 구축

2. 감성 분류 모델 구축하기

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

In [20]:	SA_lr_grid_cv.**fit**(nsmc_train_tfidf, nsmc_train_df['label'])
Out[20]:	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 [21]:	print(SA_lr_grid_cv.**best_params_**, round(SA_lr_grid_cv.**best_score_**, **4**))
Out[21]:	{'C': 3} 0.8553
In [22]:	*#최적 매개변수의 best 모델 저장* SA_lr_best = SA_lr_grid_cv.**best_estimator_**

In [20]: GridSearchCV 객체에 nsmc_train_tfidf와 label 컬럼에 대해 설정값을 조정fit( )

In [21]: GridSearchCV에 의해 찾은 최적의 C 매개변수best_params와 최고 점수best_score를 출력하여 확인

In [22]: 최적 매개변수가 설정된 모형best_estimator을 SA_lr_best 객체에 저장

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

1. 평가용 데이터를 이용하여 모델 정확도 확인하기

1. 평가용 데이터를 벡터화한 뒤 모델 정확도를 계산하여 출력

In [23]:	*#평가용 데이터의 피처 벡터화* nsmc_test_tfidf = tfidf.**transform**(nsmc_test_df['document'])
In [24]:	test_predict = SA_lr_best.**predict**(nsmc_test_tfidf)
In [25]:	**from** sklearn.metrics **import accuracy_score** print('감성 분석 정확도 : ', round(**accuracy_score**(nsmc_test_df['label'], test_predict), 3))
Out[25]:	감성 분석 정확도 : 0.857

In [23]: 평가용 데이터nsmc_test_df['document']에 In [16]에서 생성한 tfidf 객체를 적용하여 벡터 변환을 수행transform( )

In [24]: 감성 분류 모델SA_lr_best에 nsmc_test_tfidf 벡터를 사용하여 감성을 예측predict( )

In [25]: 평가용 데이터의 감성 결과값nsmc_test_df['label']과 감성 예측값test_predict을 기반으로 정확도를 계산accuracy_score( )하여 출력

- 감성 분류 모델의 정확도가 85.7%

01. [감성 분석 + 토픽 모델링] 영화 리뷰 데이터로 감성 예측하기

■ 분석 모델 평가

2. 새로운 텍스트로 감성 예측 확인하기

1. 감성 분류 모델에 새로운 텍스트를 직접 입력하여 감성 예측을 수행

In [26]:	st = input('감성 분석할 문장 입력 >> ')
Out[26]:	감성 분석할 문장 입력 >> 웃자 ^o^ 오늘은 좋은 날이 될 것 같은 예감100%! ^^*
In [27]:	*#0) 입력 텍스트에 대한 전처리 수행* st = re.**compile**(r'[ㄱ-ㅣ가-힣]+').findall(st) ; print(st) st = [" ".**join**(st)] ; print(st)
Out[27]:	'웃자', '오늘은', '좋은', '날이', '될', '것', '같은', '예감'] ['웃자 오늘은 좋은 날이 될 것 같은 예감']
In [28]:	#1) 입력 텍스트의 피처 벡터화 st_tfidf = tfidf.transform(st) #2) 최적 감성 분석 모델에 적용하여 감성 분석 평가 st_predict = SA_lr_best.predict(st_tfidf)

In [29]:	#3) 예측값 출력하기
	if(st_predict == 0):
	  print(st , "->> 부정 감성")
	else :
	  print(st , "->> 긍정 감성")

Out[29]:	['웃자 오늘은 좋은 날이 될 것 같은 예감'] ->> 긍정 감성

In [27]: 입력받은 텍스트st에 대해 In [14]에서 수행한 것과 같은 전처리 작업을 수행

In [28]: idf 객체로 벡터화transform( ) 후에 모델에 적용하여 감성 예측predict( )을 수행

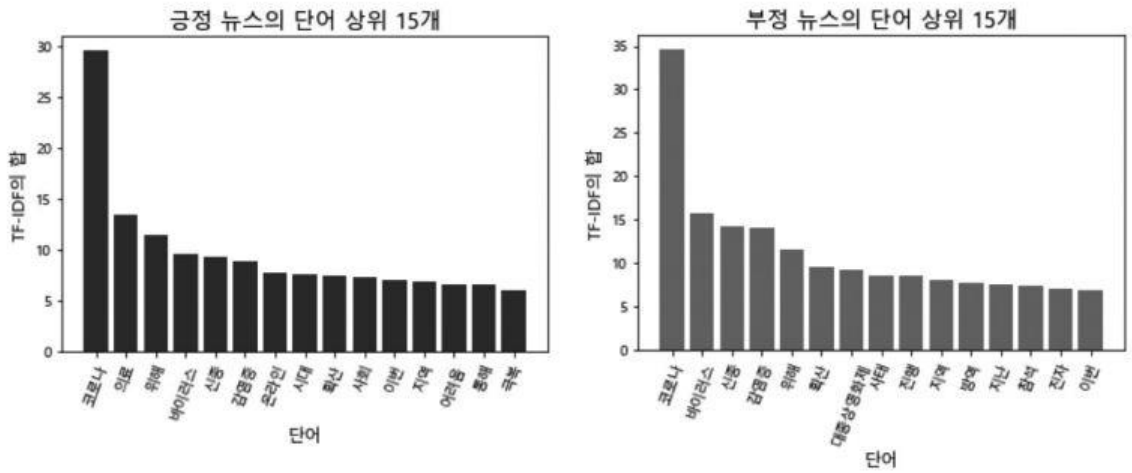
In [29]: 감성 예측 결과를 출력하여 확인

02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 분석 미리보기

코로나 뉴스 텍스트의 감성 분석하기	
목표	네이버 뉴스의 코로나 관련 텍스트에 대해 감성 분석을 수행한다.
핵심 개념	텍스트 마이닝, 특성 벡터화, BoW, TF-IDF, DTM, 감성 분석, 토픽 모델링, LDA, pyLDAvis
데이터 수집	네이버 뉴스의 json 파일: 네이버 API로 크롤링하여 저장(예제소스로 제공)
데이터 준비 및 탐색	1. 파일 불러오기: json.load() 2. 분석할 컬럼을 추출하여 데이터프레임 구성 3. 한글 외의 문자 제거
감성 분석	title / description 컬럼에 대한 감성 분석 • TF-IDF 기반 특성 벡터화: tfidf.transform() • 분석 모델을 이용한 감성 분석: SA_lr_best.predict() • 분석 결과를 데이터프레임에 저장
결과 확인 및 시각화	

감성 분석 결과 시각화



02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 목표설정

- 목표: 감성 분류 모델을 이용하여 네이버 뉴스에서 크롤링 한 '코로나' 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 '코로나_naver_news.json' 파일을 이용해도 됨

02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 데이터 준비 및 탐색

1. 분석할 컬럼을 추출하여 데이터프레임 구성하기

1. My_Python 폴더에 data 폴더를 만든 뒤 크롤링한 파일을 옮기고 앞에서 만든 '감성분석' 페이지에 이어서 실습

In [30]:	**import** json file_name = '코로나_naver_news' **with open**('./13장_data/'+ file_name+'.json', encoding = 'utf8') as j_f: data = **json.load**(j_f)
In [31]:	print(data)
Out[31]:	[{'cnt': 1, 'description': '코로나발 경제 위기 대응을 위해 돈 쓸 곳은 늘어났지만, 국세 수입은 줄어들면서 정부의 재정 마련에 대한 우려가 컸다. 이 때문에 한국개발원(KDI) 등 국책연구기관들은 증세를 화두로 꺼내들었지만, 정부 여당은 증세에... ', 'pDate': '2020-06-04 14:12:00', 'title': "결국 '증세론' 먼저 꺼내든 與..."증세없는 '기본소득' 불가능""}, {'cnt': 2, 'description': '▲ 지난 2일 창녕군보건소 앞에 설치한 선별진료소에서 검사자가 체온을 측정하고 있다.©(사진제공=창녕군청) 코로나 장기화 대비 비대면 선별진료 도입 경남 창녕군은 지난 2월 28일 도내 최초로 코로나19 선별진료소... ', 'pDate': '2020-06-04 14:12:00', 'ti ...

In [30]: 분석할 데이터 파일을 로드json.load( )하여 data 객체에 저장

In [31]: data 객체의 내용을 출력하여 확인

02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 데이터 준비 및 탐색

1. 분석할 컬럼을 추출하여 데이터프레임 구성하기

2. 뉴스의 내용이 들어 있는 description 컬럼과 제목이 들어 있는 title 컬럼을 추출하여 데이터프레임으로 구성

In [32]:	data_title = []
	data_description = []
	for item in data:
	  data_title.append(item['title'])
	  data_description.append(item['description'])

In [33]:	data_title
Out[33]:	["결국 '증세론' 먼저 꺼내든 與..."증세없는 '기본소득' 불가능"", '창녕군, '창녕형'비대면 선별진료소 운영', "DK모바일, 메인 홍보 모델로 AOA '설현' 선정", '김병민 "기본소득도 필요하면 논의 테이블에 올려야" [인터뷰]',
In [34]:	data_description
Out[34]:	['코로나발 경제 위기 대응을 위해 돈 쓸 곳은 늘어났지만, 국세 수입은줄어들면서 정부의 재정 마련에 대한 우려가 컸다. 이 때문에 한국개발원(KDI) 등 국책연구기관들은 증세를 화두로 꺼내들었지만, 정부 여당은 증세에... ', '▲ 지난 2일 창녕군보건소 앞에 설치한 선별진료소에서 검사자가 체온을 측정하고 있다. ©(사진제공=창녕군청) 코로나 장기화 대비 비대면 선별진료 도입경남 창녕군은 지난 2월 28일 도내 최초로 코로나19 선별진료소... ',
In [35]:	data_df = pd.DataFrame({'title':data_title, 'description':data_ description})

In [32]: 전체 데이터가 들어 있는 data 객체에서 뉴스 한 개에 해당하는 item의 title과 description을 각각 추출하여 리스트를 구성append( )하는 작업을 반복

In [33]: data 객체의 내용을 출력하여 확인

In [34]: 생성된 data_description 리스트를 확인

In [35]: 리스트를 데이터프레임 객체로 저장

02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 감성 분석

1. 감성 분석 수행 후 결과값을 데이터프레임에 저장하기

1. title 컬럼에 대한 감성 분석을 수행

In [38]:
```
#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 = SA_lr_best.predict(data_title_tfidf)

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

2. description 컬럼에 대해서도 같은 작업을 하여 감성 분석을 수행

In [39]:
```
#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 = SA_lr_best.predict(data_description_tfidf)

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

3. 분석 결과 데이터프레임을 CSV 파일로 저장

In [40]:
```
data_df.to_csv('./13장_data/'+file_name+'.csv', encoding = 'euc-kr')
```

02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

1. 감성 분석 결과 확인하기

1. 감성 분석 결과를 확인

In [40]:	data_df.head()
In [41]:	print(data_df['title_label'].value_counts())
Out[41]:	0 485 1 315 Name: title_label, dtype: int64
In [42]:	print(data_df['description_label'].value_counts())
Out[42]:	0 430 1 370 Name: description_label, dtype: int64

- 감정 결과
  - 데이터프레임 내용과 부정 감성 및 긍정 감성의 개수를 비교해보면 title 분석 결과와 description 분석 결과에 차이가 있음
  - 단어를 기준으로 분석하기 때문에 단어 의 개수가 부족하면 정확도가 떨어짐
  - 우리가 구축한 감성 분류 모델의 정확도가 85.7%였으니 틀린 결과도 있을 것
  - 분류 모델의 학습 데이터로 사용했던 영화 리뷰의 구성 단어와 분석 데이터인 뉴스를 구성하는 단어의 차이로 인한 오차도 있을 것

02. [감성 분석 + 바 차트] 코로나 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 [43]:

```
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')
```

In [44]:

```
len(NEG_data_df), len(POS_data_df)
```

Out[44]:

```
(430, 370)
```

02. [감성 분석 + 바 차트] 코로나 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 [45]:	POS_description = POS_data_df['description']
In [46]:	POS_description_noun_tk = [] **for** d **in** POS_description: POS_description_noun_tk.**append**(**okt.nouns**(d)) *#명사 형태소만 추출*
In [47]:	print(POS_description_noun_tk) *#작업 확인용 출력*
Out[47]:	[['변화', '핵심', '중', '우리', '사회', '신종', '코로나', '바이러스', '감염증', '코로나', '의', '위기', '마주', '언제', '끝', '날', '또', '앞', '미래', '국가', '국민', '어려움', '해결', '저희'], ['한편', '코로나', '로', '해외', '식', '재료', '사재기', '국민', '먹거리', '안정', '생산', '것', '포스트', '코로나', '의', '과제', '부각', '농민', '기본소득', '도입', '통해', '안정', '생산', '기반', '확충', '것'], ['최근', '갤러리', '현대', '창업', '주년', '기념', '전', '이', '작품', '전시', '코로나', '의', '영향', '마스크', '착용', '관람객', '미술', '트렌드', '한국', '미술', '시장', '글', '정태희', '서울', '옥션', '스페셜리스트', '세계', '경기', '침체', '코로나', '여 ...

In [46]: 형태소 토큰화를 하여 명사 토큰okt.nouns( )만 추출 후 리스트를 구성

02. [감성 분석 + 바 차트] 코로나 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 [48]:	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*

In [49]:	print(POS_description_noun_join) *#작업 확인용 출력*
Out[49]:	['변화 핵심 우리 사회 신종 코로나 바이러스 감염증 코로나 위기 마주 언제 미래 국가 국민 어려움 해결 저희', '한편 코로나 해외 재료 사재기 국민 먹거리 안정 생산 포스트 코로나 과제 부각 농민 기본소득 도입 통해 안정 생산 기반 확충', '최근 갤러리 현대 창업 주년 기념 작품 전시 코로나 영향 마스크 착용 관람객 미술 트렌드 한국 미술 시장 정태희 서울 옥션 스페셜리스트 세계 경기 침체 코로나 여파', '코로나 감안 면접 비대 역량 검사 도입 국내 최대 치킨 프랜차이즈 제너시스 비비큐 회장 채용 관계자 한국 대표 책임감 브랜드 로서 코로나 침체 채용 분위기', ...

In [48]: 토큰의 길이가 1인 것은 제외 후 연결join( )하여 리스트를 구성

02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

2. 부정 감성 뉴스에도 같은 작업 수행

In [50]:

```
NEG_description = NEG_data_df['description']

NEG_description_noun_tk = []
NEG_description_noun_join = []

for d in NEG_description:
    NEG_description_noun_tk.append(okt.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
```

02. [감성 분석 + 바 차트] 코로나 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 [51]:	POS_tfidf = TfidfVectorizer(tokenizer = okt_tokenizer, min_df = 2) POS_dtm = POS_tfidf.fit_transform(POS_description_noun_join)
In [52]:	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)
In [53]:	POS_words #작업 확인용 출력
Out[53]:	[('코로나', 29.58865817494428), ('의료', 13.461836434575911), ('위해', 11.469142512665115), ('바이러스', 9.626197092990505), ('신종', 9.241355937335499), …

In [51]: TfidfVectorizer 객체를 생성하고 POS_description_noun_join에 대해 TF-IDF 값을 구하여 DTM을 구성

In [52]: DTM의 단어get_feature_names( )마다 컬럼의 합getcol(idx).sum( )을 구하여 단어별 TFIDF 값의 합을 구하고 내림차순으로 정렬

02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

4. 부정 감성 뉴스인 NEG_description_noun_join에 대해서도 같은 작업을 수행

In [54]:	NEG_tfidf = **TfidfVectorizer**(tokenizer = **okt_tokenizer**, min_df = 2) NEG_dtm = NEG_tfidf.**fit_transform**(NEG_description_noun_join)
In [55]:	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)
In [56]:	NEG_words #작업 확인용 출력
Out[56]:	[('코로나', 34.56440043805242), ('바이러스', 15.755970871602138), ('신종', 14.29640382643984), ('감염증', 14.034156174169826), ('위해', 11.557221696267492), ('확산', 9.566148700015386), ...

02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

5. DTM 기반 단어 사전의 상위 단어로 바 차트 그리기

In [57]:
```
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm

fm.get_fontconfig_fonts()
font_location = 'C:/Windows/Fonts/malgun.ttf'
font_name = fm.FontProperties(fname =
font_location).get_name()
matplotlib.rc('font', family = font_name)

max = 15  #바 차트에 나타낼 단어의 수
```

In [57]: 바 차트를 그리기 위해 matplotLib 패키지를 임포트하고 한글을 표시하기 위해 한글 폰트를 설정
바 차트에 나타낼 단어 개수를 max에 설정

02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

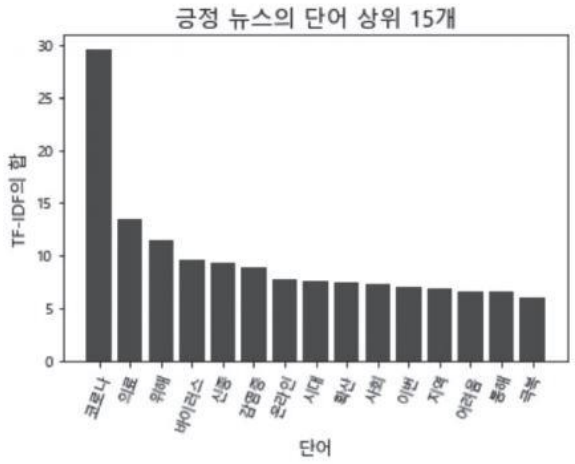
5. 긍정 뉴스와 부정 뉴스에 많이 나타난 단어를 바 차트로 나타냄

In [58]:

```
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.show()
```

Out[58]:



02. [감성 분석 + 바 차트] 코로나 뉴스 텍스트의 감성 분석하기

■ 결과 확인 및 시각화

2. 결과 시각화하기

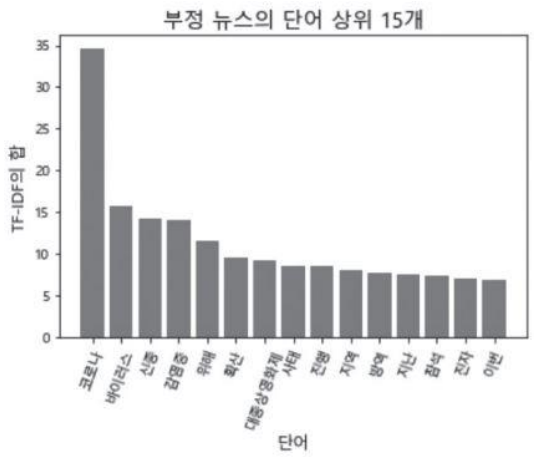
5. 긍정 뉴스와 부정 뉴스에 많이 나타난 단어를 바 차트로 나타냄

In [59]:

```
plt.bar(range(max), [i[1] for i in NEG_words[:max]], color = "blue")
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 POS_words[:max]], rotation = 70)

plt.show()
```

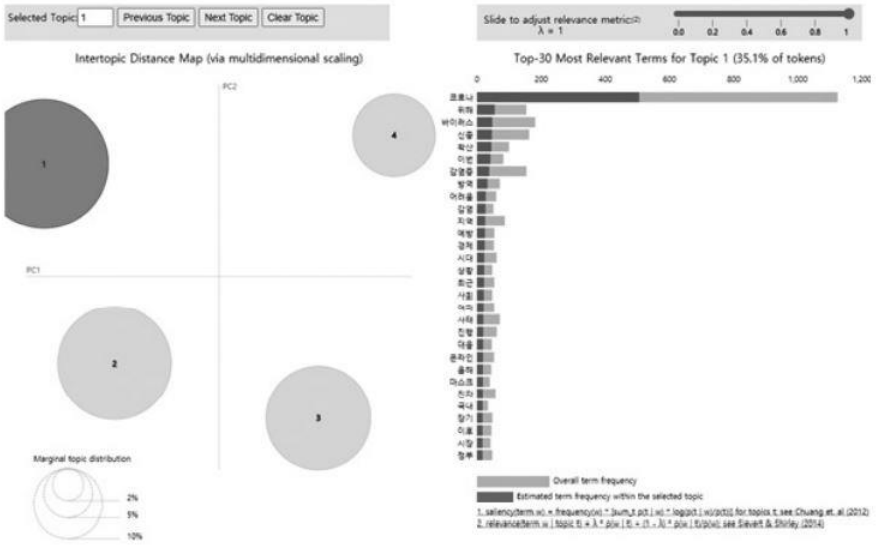
Out[59]:



03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 토픽 분석하기

■ 분석 미리보기

뉴스 텍스트에서 코로나 토픽 분석하기	
목표	네이버 뉴스에서 코로나와 관련된 토픽을 분석하고 결과를 시각화한다.
핵심 개념	텍스트 마이닝, 특성 벡터화, BoW, TF-IDF, DTM, 감성 분석, 토픽 모델링, LDA, pyLDAvis
데이터 준비	1. description 컬럼 추출 2. 명사 토큰 추출: okt.nouns()
분석 모델 구축	1. LDA 토픽 모델의 입력 벡터 생성: corpora.Dictionary(), dictionary.doc2bow() 2. LDA 모델 생성 및 훈련: gensim.models.ldamulticore.LdaMulticore()
결과 확인 및 시각화	
1. 분석 결과 확인: lda_model.print_topics() 2. 분석 결과 시각화: pyLDAvis.gensim.prepare	



03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 토픽 분석하기

■ 목표설정

- 목표: 네이버 뉴스에서 '코로나'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 [60]:	description = data_df['description']
In [61]:	description_noun_tk = [] **for** d **in** description: description_noun_tk.**append**(**okt.nouns**(d)) *#명사 형태소만 추출*

In [62]:	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)

In [63]:	print(description_noun_tk2)
Out[63]:	[['코로나', '경제', '위기', '대응', '위해', '국세', '수입', '정부', '재정', ' 마련', '대한', '우려', '때문', '한국', '개발', '국책', '연구기관', '증세', ' 화두', '정부', '여당', '증세'], ['지난', '창녕군', '보건소', '설치', '진료', ' 검사', '체온', '측정', '사진', '제공', '창녕군', '코로나', '장기', '대비', ' 비대', '진료', '도입', '경남', '창녕군', '지난', '도내', '최초', '코로나', '진 료'], ['한편', '설현', '최근', '코로나', '바이러스', '시리즈', '세계', '유행', '다큐멘터리', '내레이션', '처음', '도전', '호평', '드라마', '출연', '검토',

03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 토픽 분석하기

■ 분석 모델 구축

1. 토픽 분석을 위한 LDA 모델 구축하기

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

In [ ]:	**!pip install gensim**

2. 패키지를 설치한 후에 필요한 모듈을 임포트

In [64]:	**import** gensim **import** gensim.corpora as corpora

3. LDA 토픽 모델의 입력 벡터 생성하기

In [65]:	dictionary = corpora.**Dictionary**(description_noun_tk2)
In [66]:	print(dictionary[1]) *#작업 확인용 출력*
Out[66]:	경제
In [67]:	corpus = [dictionary.**doc2bow**(word) **for** word **in** description_noun_tk2]
In [68]:	print(corpus)
Out[68]:	[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1), (14, 1), (15, 2), (16, 2), (17, 1), (18, 1), (19, 1)], [(17, 2), (20, 1), (21, 1), (22, 1), (23, 1), (24, 1), (25, 1), (26, 1), (27, 1), (28, 1), (29, 1), (30, 1), (31, 2), (32, 3), (33, 3), (34, 1), (35, 1), (36, 1)], [(5, 1), (17, 1), (37, 1), (38, 1), (39, 1), (40, 1), (41, 1), (42, 1), ...

In [65]: description_noun_tk2에 포함된 단어에 대해 사전을 구성corpora.Dictionary( )

In [66]: 구성된 사전의 내용을 확인하기 위해 1번 단어dictionary[1]를 출력

In [67]: 단어 사전dictionary의 단어에 대해 BoW를 구하여doc2bow( ), 단어 뭉치corpus를 구성

In [68]: 단어 뭉치corpus를 출력하여 (word_id, work_count)의 BoW 구성을 확인

03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 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_model을 생성

In [69]:	**k** = 4 #토픽의 개수 설정
In [70]:	lda_model = gensim.models.ldamulticore.LdaMulticore(corpus, iterations = 12, num_topics = k, id2word = dictionary, passes = 1, workers = 10)

03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 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 [71]:	print(lda_model.**print_topics**(num_topics = **k, num_words = 15**))
Out[71]:	[(0, '0.045*"코로나" + 0.012*"바이러스" + 0.010*"신종" + 0.010*"감염증" + 0.009*"위해" + 0.005*"지역" + 0.004*"방역" + 0.004*"확산" + 0.004*"생활" + 0.004*"어려움" + 0.004*"사태" + 0.004*"통해" + 0.004*"장기" + 0.004*"의료" + 0.003*"서울"'), (1, '0.043*"코로나" + 0.013*"감염증" + 0.011*"신종" + 0.009*"위해" + 0.008*"사태" + 0.008*"바이러스" + 0.004*"경제" + 0.004*"확산" + 0.004*"지원" + 0.004*"진행" + 0.004*"지역" + 0.004*"의료" + 0.003*" 운영" + 0.003*"예정" + 0.003*"통해"'), (2, '0.081*"코로나" + 0.009*"위해" + 0.008*"바이러스" + 0.007*"신종" + 0.007*"확산" + 0.007*"이번" + 0.006*"감염 증" + 0.005*"방역" + 0.004*"어려움" + 0.004*"감염" + 0.004*"지역" + 0.004*" 예방" + 0.004*"경제" + 0.004*"시대" + 0.004*"상황"'), (3, '0.067*"코로나" + 0.013*"바이러스" + 0.009*"신종" + 0.008*"감염증" + 0.008*"위해" + 0.007*"의 료" + 0.006*"지역" + 0.006*"확산" + 0.005*"지난" + 0.005*"진행" + 0.005*"시대" + 0.005*"서울" + 0.004*"이번" + 0.004*"포스트" + 0.004*"대종상영화제"')]

num_words = 15에 따라 토픽을 구성하는 주요 단어 15개가 토픽에 대한 영향력 비율 과 함께 출력된 것을 확인 가능

네이버 뉴스를 크롤링할 때 검색어로 '코로나'를 사용했기 때문에 모든 토픽에서 '코로나' 단어가 압도적으로 많이 나옴

03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 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

표 13-1 토픽 분석 결과(lda_model.print_topics)

토픽 번호	주요 단어(15개)	토픽 레이블
0	0.045*"코로나", 0.012*"바이러스", 0.010*"신종", 0.010*"감염증", 0.009*"위해", 0.005*"**지역**", 0.004*"방역", 0.004*"**확산**", 0.004*"생활", 0.004*"어려움", 0.004*"**사태**", 0.004*"통해", 0.004*"장기", 0.004*"의료", 0.003*"서울"	지역 확산 사태
1	0.043*"코로나", 0.013*"감염증", 0.011*"신종", 0.009*"위해", 0.008*"사태", 0.008*"바이러스", 0.004*"**경제**", 0.004*"확산", 0.004*"**지원**", 0.004*"진행", 0.004*"지역", 0.004*"의료", 0.003*"**운영**", 0.003*"예정", 0.003*"통해"	경제 영향
2	0.081*"코로나", 0.009*"위해", 0.008*"바이러스", 0.007*"신종", 0.007*"확산", 0.007*"이번", 0.006*"감염증", 0.005*"방역", 0.004*"어려움", 0.004*"감염", 0.004*"지역", 0.004*"**예방**", 0.004*"경제", 0.004*"**시대**", 0.004*"상황"	코로나 예방
3	0.067*"코로나", 0.013*"바이러스", 0.009*"신종", 0.008*"감염증", 0.008*"위해", 0.007*"의료", 0.006*"**지역**", 0.006*"**확산**", 0.005*"지난", 0.005*"진행", 0.005*"시대", 0.005*"서울", 0.004*"이번", 0.004*"포스트", 0.004*"**대종상영화제**"	지역 확산과 대종상영화제

03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 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 [ ]:	**!pip install pyLDAvis**
In [72]:	*#한글 UnicodeEncodeError 방지를 위해 기본 인코딩을 "utf-8"로 설정* **import** os **os.environ["PYTHONIOENCODING"] = "utf-8"** **import** pyLDAvis.gensim lda_vis = **pyLDAvis.gensim.prepare**(lda_model, corpus, dictionary)
In [73]:	pyLDAvis.**display**(lda_vis)

03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 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 토픽이 포함되어 있거나 많이 겹쳐져 있다면 토픽의 개수 k 값을 다르게 하여 LDA 모델을 다시 실행

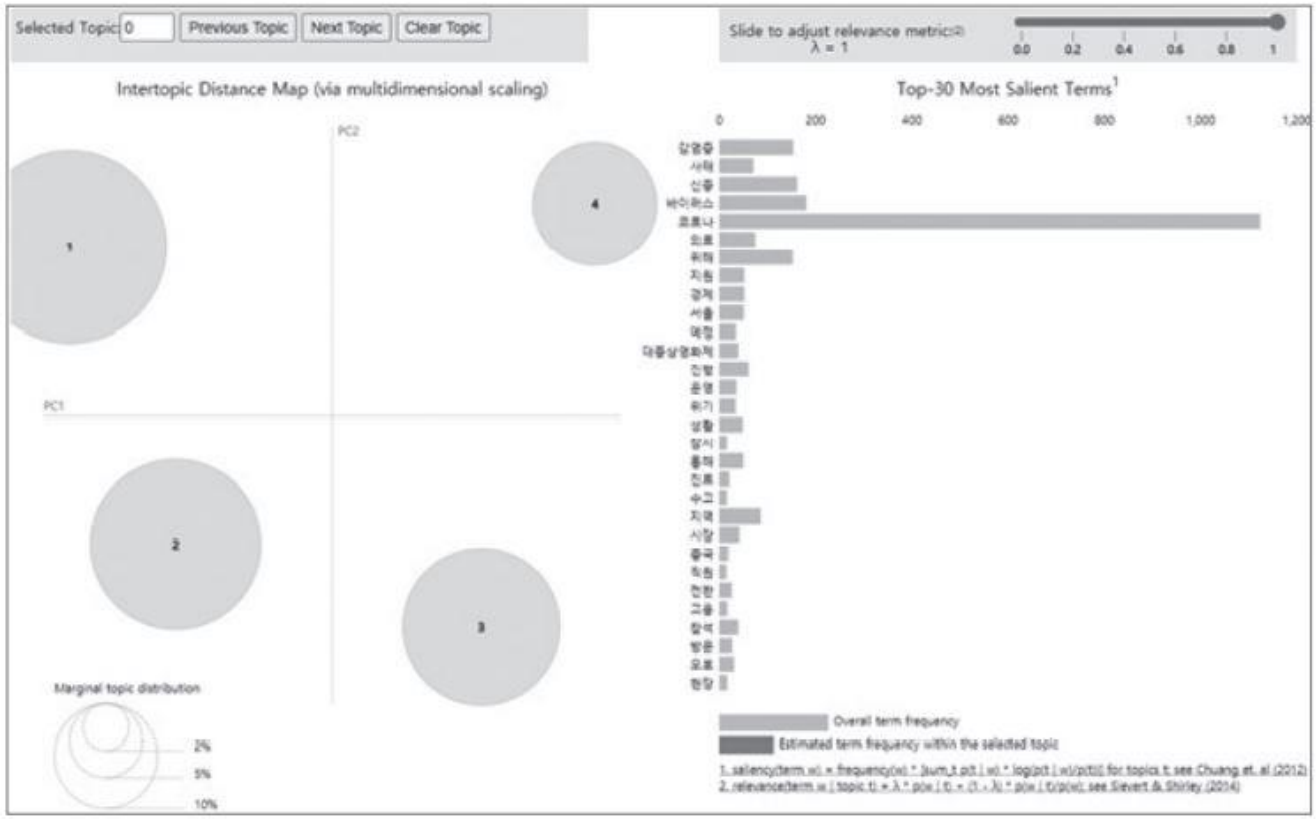


그림 13-7 LDA 토픽 분석 결과를 시각화한 pyLAvis

03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 토픽 분석하기

■ 결과 확인 및 시각화

2. 분석 결과 시각화하기

2. 왼쪽 영역의 토픽 간 거리 지도 영역에서 토픽 버블을 클릭해서 선택하면 오른쪽 영역에 토 픽에 대한 토큰 비율과 상위 단어 30개의 바 차트가 나타남

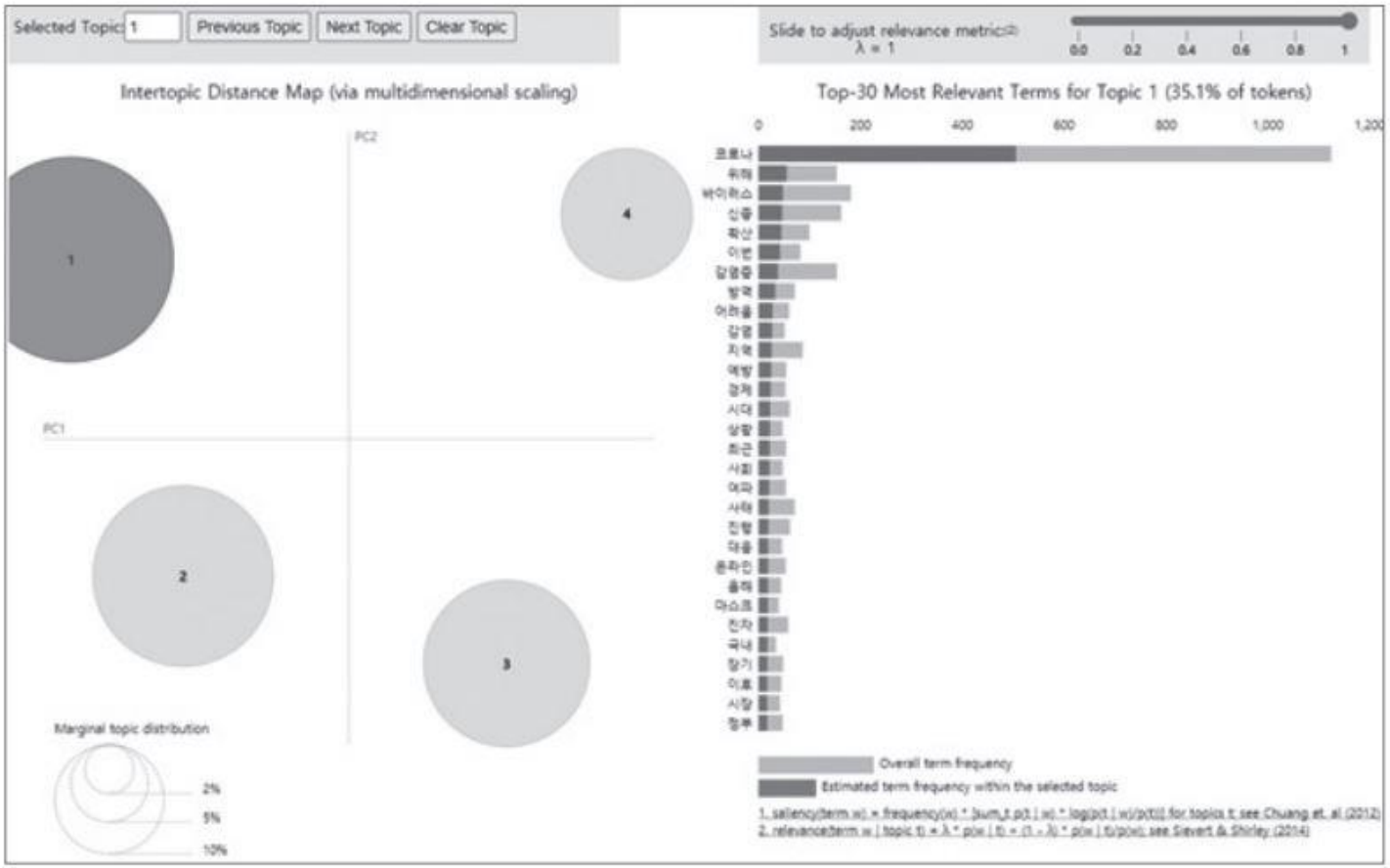


그림 13-8 pyLAvis의 토픽 간 거리 지도 영역

03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 토픽 분석하기

■ 결과 확인 및 시각화

2. 분석 결과 시각화하기

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

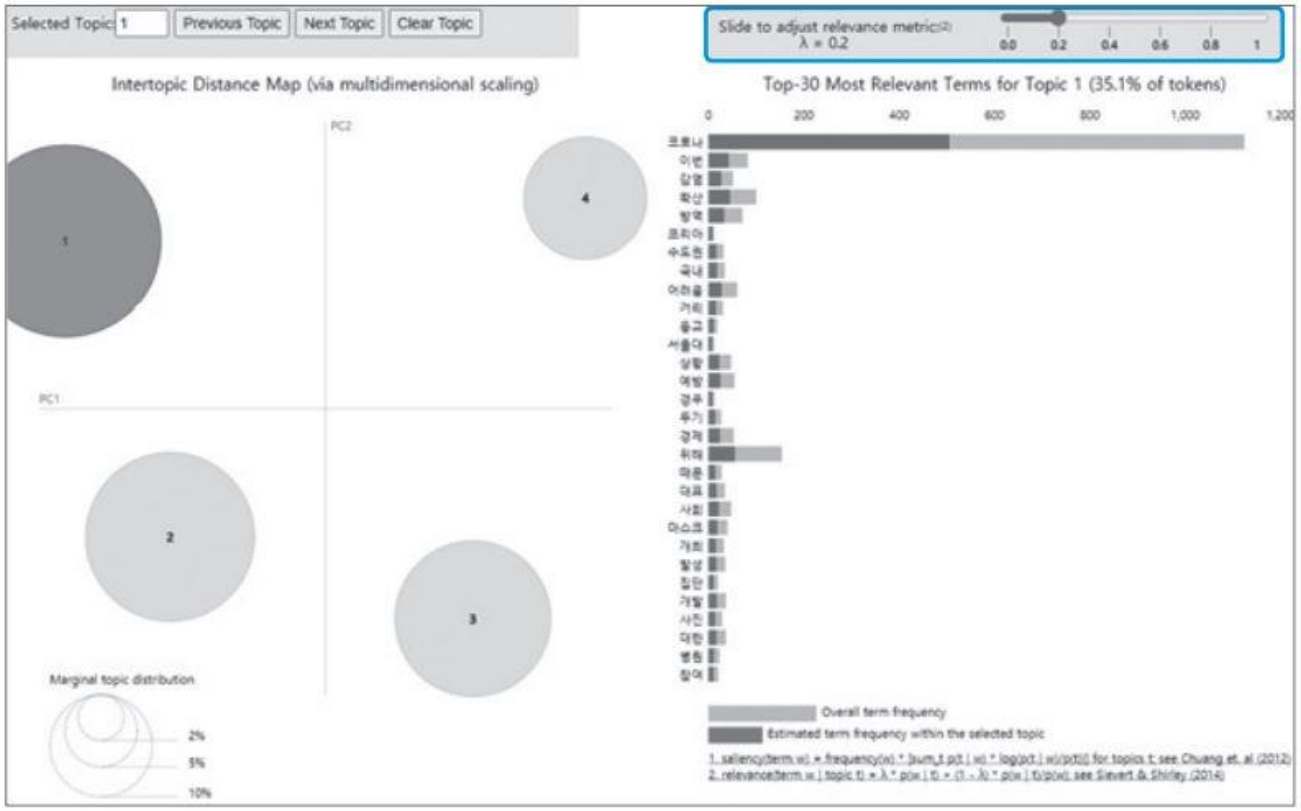


그림 13-9 pyLAvis의 관련성 메트릭 조정 슬라이드

03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 토픽 분석하기

■ 결과 확인 및 시각화

2. 분석 결과 시각화하기

2. 오른쪽 영역에 있는 단어 위로 마우스를 이동하면 단어의 토픽 영향력에 따라 토픽 버블의 크기가 조정되어 변함

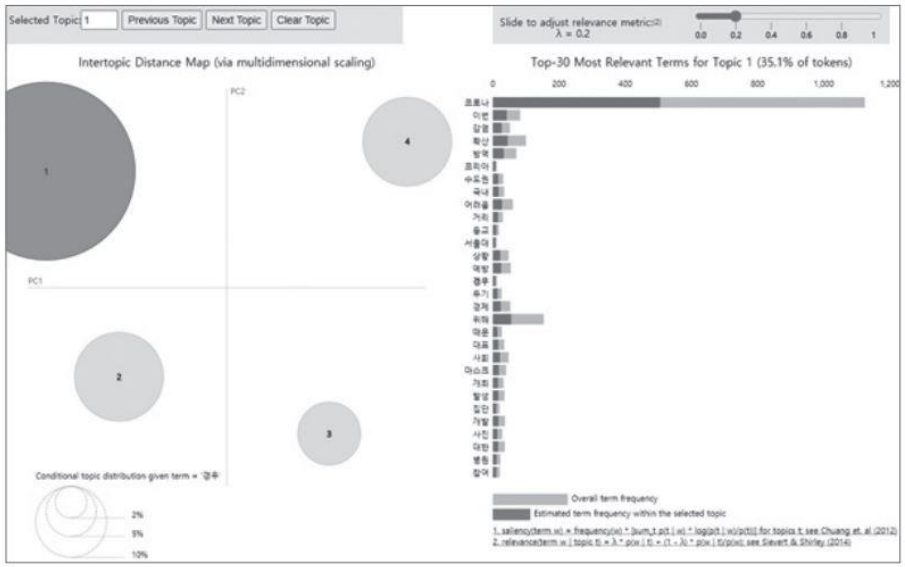


그림 13-10 pyLAvis의 토픽 버블

3. pyLDAvis를 파일로 저장하자. pyLDAvis는 웹 브라우저 창에 표시되므로 저장 파일 형식도 html로 설정

In [74]:	pyLDAvis.**save_html**(lda_vis, './13장_data/'+file_name+"_vis.html")

03. [토픽 분석 + LDA 토픽 모델] 뉴스 텍스트에서 코로나 토픽 분석하기

■ 결과 확인 및 시각화

2. 분석 결과 시각화하기

- 두 결과를 비교하여 종합적으로 분석

표 13-2 토픽 분석 결과 종합(lda_model.print_topics + pyLDAvis)

lda_model.print_topics()에 의한 결과			pyLDAvis에 의한 결과			최종 토픽 레이블
토픽 번호	주요 단어(15개)	토픽 레이블	토픽 번호	토큰 분포 비율	토픽 특화 단어(15개) (λ = 0.2)	
0	0.045*"코로나", 0.012*"바이러스", 0.010*"신종", 0.010*"감염증", 0.009*"위해", 0.005*"지역", 0.004*"방역", 0.004*"확산", 0.004*"생활", 0.004*"어려움", 0.004*"사태", 0.004*"통해", 0.004*"장기", 0.004*"의료", 0.003*"서울"	지역 확산 사태	3	23.2%	**전환**, 바이러스, 시청, 스웨덴, 감염증, 안정, 관련, 긴급, 신종, 생활, 측은, 기능, 창업, 게임, 본부	코로나 사태 전환
1	0.043*"코로나", 0.013*"감염증", 0.011*"신종", 0.009*"위해", 0.008*"사태", 0.008*"바이러스", 0.004*"경제", 0.004*"확산", 0.004*"지원", 0.004*"진행", 0.004*"지역", 0.004*"의료", 0.003*"운영", 0.003*"예정", 0.003*"통해"	경제 영향	4	14.4%	사태, 감영증, 산재, 보험, **필라테스**, 화학, 직원, 신종, 송가, 수행, **아마존**, **식료품**, **배달**, **명상**, 고용	경제 및 생활의 변화
2	0.081*"코로나", 0.009*"위해", 0.008*"바이러스", 0.007*"신종", 0.007*"확산", 0.007*"이번", 0.006*"감염증", 0.005*"방역", 0.004*"어려움", 0.004*"감염", 0.004*"지역", 0.004*"예방", 0.004*"경제", 0.004*"시대", 0.004*"상황"	코로나 예방	1	35.1%	코로나, 이번, 감염, 확산, **방역**, **코리아**, 수도권, 국내, 어려움, 거리, 등교, 서울대, 상황, **예방**, 경우	코리아 방역
3	0.067*"코로나", 0.013*"바이러스", 0.009*"신종", 0.008*"감염증", 0.008*"위해", 0.007*"의료", 0.006*"지역", 0.006*"확산", 0.005*"지난", 0.005*"진행", 0.005*"시대", 0.005*"서울", 0.004*"이번", 0.004*"포스트", 0.004*"대종상영화제"	지역 확산과 대종상 영화제	2	27.3%	코로나, 의료, **대종상영화제**, 잠시, 바이러스, **참석**, 거머, 서울, 거래, 비대, **기생충**, 포스트, **봉준호**, 법률, 지난	지역 확산과 대종상 영화제