결과 보고서

# 코로나 이펙트

20151006 강다훈
20152321 송준권

# 목차



해당 프로젝트 결과물은 이곳에서 확인 가능합니다.

# 1. 프로젝트 개요

## I. 프로젝트 소개

### A. 프로젝트 목표

- 코로나 바이러스 감염증 관련 뉴스 기사와 댓글을 정형화하고 이를 시계열 분석하여 생활, 문화의 변화를 분석하여 사용자가 보기 쉽게 가시화하는 것을 목표로 함.

### B. 기획 배경

- 현재 대한민국은 코로나 바이러스 감염증(COVID-19)으로 인한 팬데믹 상태에 빠져있음. 이로 인해 사람들이 살아가는 방식 또한 변화됨.

### C. 기획 의도

- 코로나 바이러스의 확산을 막기 위한 사람들의 철저한 개인위생, 초, 중, 고, 대학교의 대면 수업을 대신한 온라인 수업, 사회적 거리 두기로 인한 인터넷 쇼핑 사용량의 증가 등 여러 가지 방면의 긍정적인 변화를 보여줌. 하지만 가정폭력의 증가, 시장경제의 침체 등 부정적인 변화도 있었음.

- 코로나 바이러스 이전과 이후 생활, 문화 방면에서의 차이점을 토대로 빈도 분석하여 코로나로 인한 문화적 영향력을 알아봄. 이를 알아보기 위해서 여러 가지 방법이 존재하는데, 이 중에 우리는 인터넷 뉴스를 기반으로 여론을 조사함.

- 텍스트 마이닝을 활용하여 비 정형화된 뉴스 기사를 다량의 데이터로 가공함. 다양한 데이터 분석 방법을 이용하여 코로나로 변화된 생활, 문화를 분석.

### D. 분석 환경

개발 언어 : 파이썬(Python)

개발 도구

- Google Colab(Colaboratory)
    - 구글 클라우드 기반의 Jupyter notebook 개발환경.

- Jupyter notebook
    - 문서의 포맷으로 웹 브라우저에서 파이썬 코드를 작성하고 실행할 수 있는 소프트웨어.
- IPykernel, IPython
    - 인터프리터 환경에서 파이썬을 조금 더 유연하게 작업할 수 있도록 사용되는 command shell과 이것을 프론트에서 사용할 수 있도록 지원해주는 코어 기능.

## II. 프로젝트 구성도

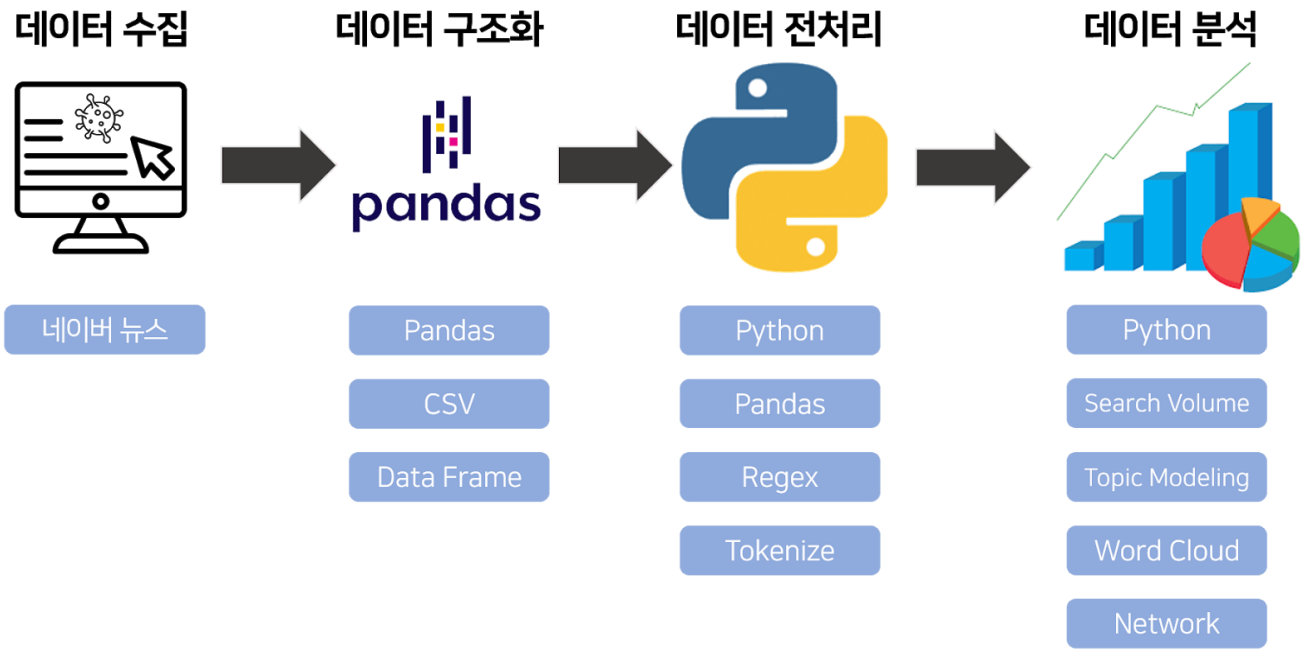
1. 네이버 뉴스에서 '코로나'가 포함된 뉴스 파싱

2. 파싱한 데이터를 기사 제목, 작성 날짜, 네이버에서 분류된 카테고리, 기사 본문, 네이버에서 분류된 언론사 고유번호, 기사 링크로 나누어 DataFrame에 저장

3. DataFrame에 저장된 파일들을 클렌징하여 필요없는 데이터 삭제 및 일반명사, 고유명사(NNP,NNG)만을 사용하기 위해 토큰화

4. 검색량 분석,LDA 토픽 모델링, WordCloud, 네트워크 시각화 등 여러가지 방식으로 분석 및 시각화

5. 분석한 데이터를 이용하여 결론 도출




## III. 프로젝트 업무 분장

| 성명 | 담당업무 |
|---|---|
| 송준권 | 데이터 수집 / 정제 / 분석 / 시각화, 문서 작성 |
| 강다훈 | 데이터 정제 / 분석 / 시각화, 문서 및 발표 자료 제작 |

# 2. 추진보고서

## 프로젝트 추진 보고서(팀실적 보고용) - 1주차

작성자 : 송준권 작성일자 : 2020.09.06.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 주제 선정 | 진도(%) | 5% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 추진 내용 | 금주 저희는 주제 선정을 진행하였습니다.<br>여러 가지 아이디어가 나왔습니다.<br><br>■ 첫 번째, 코로나 관련 주제입니다. 코로나 확진자 수가 계속해서 감소하고 있으나, 몇몇 사건들로 인해 급증하는 모습을 보여줬습니다. 코로나 관련 뉴스를 통해서 키워드를 추출해 코로나 이전과 이후 생활, 문화 방면에서의 차이점을 토대로 빈도 분석을 하여 코로나로 인한 문화적 영향력을 알아보는 것입니다.<br><br>■ 다음으로 생각한 것은 꾸준히 쟁점이 되는 집값 / 부동산 정책 관련 주제입니다. 꾸준히 서울 및 수도권의 집값이 상승하였고, 정부에서는 계속해서 부동산 대책을 내고 있습니다. 뉴스를 통하여 부동산 정책과 집값 변동, 그리고 이에 대한 여론을 시계열 분석하여 각 부동산 정책의 영향력과 발전 방향을 모색하는 것입니다.<br><br>● 부동산 관련 주제도 좋다고 생각 하였으나, 여론에 대해 감성 분석을 수행하였을 때, 정치적 의견, 맞춤법 문제 등 감성 사전 구축에 있어 장애요소가 많아 시간 내에 수행하기 어려울 것으로 판단했습니다.<br><br>● 코로나 관련 이슈는 현재 가장 집중되고, 가장 중요한 문제이며 현재 사람들이 어떤 방식으로 변화되어 살아가고 있는지 분석해보고 싶어 해당 주제로 선정하였습니다. | | |
| 문제점 및 대책 | ■ 분석 데이터 선정 및 분석 방법에 관련하여 정보가 부족하여 개인 조사를 진행하여 취합하기로 하였습니다. | | |
| 향후 추진 계획 | ■ 차주 주제를 조금 더 구체화하여 '시스템 정의서' 작성 및 사용될 기술을 조사할 예정입니다. | | |

# 프로젝트 추진 보고서(팀실적 보고용) - 2주차

작성자 : 강다훈 작성일자 : 2020.09.10.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 시스템 정의서 작성 | 진도(%) | 10% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 추진 내용 | 금주에는 주제를 구체화하기 위해 시스템정의서를 함께 작성하였습니다.<br>주제 선정의 이유와 필요성에 대해 깊게 고민하였습니다. 또한, 대략적인 분석의 흐름을 그림을 그려 다시 확인했습니다.<br><br>사용될 언어와 기술을 선정하였습니다. R과 Python 중 Python을 선택하였는데, Python이 익숙하고 러닝 커브가 낮을 것으로 생각하여 선정했습니다.<br>파이썬에서 해당 목표를 위해 사용될 패키지는 다음과 같습니다.<br><br>크롤링: BeautifulSoup, Selenium<br>NLP: KoNLPy, NLTK<br>데이터 분석: pandas<br>데이터 시각화: matplot | | |
| 문제점 및 대책 | ■ 데이터 분석을 처음 해보기 때문에, 세부적인 사전 지식과 패키지 사용 요령이 부족하여, 각자 개인 공부를 통해 익숙해지기로 하였습니다.<br>■ 분석 방법에는 다양한 것들이 있는데, 이에 대해 조금 더 알아보기로 하였습니다. | | |
| 향후 추진 계획 | ■ 추진계획서에서, 개요와 추진 일정 등 구체적인 진행 계획을 정하기로 하였습니다.<br>■ 각자 개인 공부를 통해 기술에 대해 꾸준히 학습하기로 하였습니다 | | |

# 프로젝트 추진 보고서(팀실적 보고용) - 3주차

작성자 : 강다훈 작성일자 : 2020.09.17.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 크롤러 구현 | 진도(%) | 20% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 추진 내용 | 금주에는 네이버의 뉴스들을 크롤링 하는 코드를 작성하였습니다.<br>코드는 구글 Colaboratory를 사용하여 코딩하였습니다.<br>크롤링을 하기 위한 패키지인 selenium은 멀티프로세싱이 불가능하여 BeautifulSoup4를 사용하였습니다.<br><br>기사 크롤링할 때 언론사들의 홈페이지 포맷이 각각 다르기 때문에 '네이버 뉴스'의 포맷을 선택하였고, 뉴스의 정보중 제목, 링크, 날짜, 본문, 카테고리를 가져오는 것으로 선택하였습니다.<br><br>기사들을 검색할 때 네이버 뉴스는 4000개 이상의 기사가 검색되면 이후의 기사는 더 자세한 키워드를 입력하라고 나오고 더 이상의 기사를 제공하지 않기 때문에 2020.1.1. ~ 2020.1.31. 같이 기간을 지정하여 사용자에게 기간을 입력받아 정해진 기간에 쓰인 기사들을 수집하였습니다. 또한 1~31일이 한 엑셀에 저장되는 것이 아니라 며칠씩 묶어서 저장할지 입력받아 구분하기 편리하게 하였습니다.<br><br>이렇게 크롤링을 해도 4000건 이상의 기사가 나올 수 있기 때문에 언론사별로 구분하여 한 언론사가 기간 내에 쓴 기사들을 가져오도록 했습니다. | | |
| 문제점 및 대책 | ■ 크롤링을 구현하는 과정에서 아직은 부족함이 많이 느껴져 각자 개인 공부를 통해 익숙해지기로 하였습니다.<br><br>■ 크롤링 부분이 완벽하지 않기에 보완해볼 방법을 생각하고 있습니다. | | |
| 향후 추진 계획 | ■ 크롤링을 완벽히 구현, 저장하고, 이후의 단계인 텍스트 마이닝 부분을 구현해보기로 했습니다.<br><br>■ 크롤링을 하여 많은 정보를 수집하고 부족한 부분을 수정할 예정입니다. | | |

# 프로젝트 추진 보고서(팀실적 보고용) - 4주차

작성자 : 송준권 작성일자 : 2020.09.26.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 전처리 과정 | 진도(%) | 30% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 추진 내용 | 금주에는 크롤링된 뉴스 텍스트를 전처리하여 다시 저장하는 과정을 수행했습니다.<br>■ 전처리는 모두 정규 표현식을 사용하여 처리했습니다. 정규 표현식은 패턴과 일치하는 텍스트를 찾고 그것을 가공하는 것에 매우 유용합니다.<br>■ 먼저 뉴스 텍스트 일부를 확인하며 자주 사용되는, 그래서 분석 결과에 좋지 않은 영향을 줄 수 있는 문자열을 찾았습니다.<br>■ 해당 문자열 들을 모아 정으로 정규 표현식으로 사용할 수 있도록 패턴을 추출했습니다.<br>■ 추출된 패턴을 기반으로 텍스트에서 해당 문자열을 제거하고 저장하였습니다.<br>■ 처리는 구글 코랩을 이용하여 처리하였고, 구글 드라이브에 데이터들을 저장하였습니다. | | |
| 문제점 및 대책 | ■ 전처리 단계에서 거른다고 걸렀으나, 향후 분석 과정에서 추가로 식별되는 불필요한 단어들이 있을 수 있습니다. 분석 중, 식별 시에 바로 제거하고 다시 분석을 수행할 예정입니다.<br>■ 이제는 여러 분석 알고리즘을 이용하여 분석을 조금씩 수행해야 하는데, 다양한 분석 방법 중 어떤 방법이 결과를 도출해 내는 데에 유용할지 몰라 다양한 시도가 필요할 것 같습니다. | | |
| 향후 추진 계획 | ■ 차주 키워드 분석을 이용하여 먼저 코로나 관련 뉴스에서 가장 많이 사용된 키워드를 알아볼 예정입니다. | | |

## 프로젝트 추진 보고서(팀실적 보고용) - 5주차

작성자 : 강다훈 작성일자 : 2020.10.02.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 전처리 및 자연어 처리 | 진도(%) | 40% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 추진 내용 | 금주에는 전주에 수행했던 전처리 부분이 부족하여 수정하였고,<br>자연어 처리를 수행을 위해 형태소분석기를 선정 및 테스트를 하였습니다.<br>자연어 처리 중 키워드 분석을 위해 N-gram, TF-IDF를 통해 실습을 수행하였습니다.<br><br>■ 전주에 전처리를 한 뒤 단어 토큰화를 실행하였더니 생각보다 필요 없는 데이터가 많이 추출되어 전처리를 꼼꼼하게 실행하였습니다.<br><br>■ 전처리된 데이터들을 형태소분석기를 사용하여 가치 있는 명사들을 추출하려고 하였습니다.<br><br>■ 형태소 분석기에는 MeCab, Khaiii, Kiwi 등 몇 가지의 후보들이 있었지만, MeCab을 통한 데이터의 질이 가장 좋았고 속도 또한 월등하여 MeCab을 선택하였습니다.<br><br>■ 형태소 분석기를 사용한 이후 키워드들을 분석하여 빈도수 및 중요도를 체크하기 위하여 N-gram, TF-IDF를 사용하여 자연어 처리를 하기로 결정하였습니다. 또한 이를 적용하기 위한 공부와 실습을 해보기로 했습니다. | | |
| 문제점 및 대책 | ■ 전처리 단계에서 불필요한 단어들이 계속 추출되어 전처리 과정을 좀 더 꼼꼼히 하는 중입니다. 현재도 꾸준히 발견되어 전처리 과정에 좀 더 신경 써서 진행할 예정입니다.<br><br>■ 아직까지는 분석의 질이 좋지 않아 다른 분석방법(word2vec)이나 기사 내 특정 단어 언급 양을 측정하여 시각화하는 것 등 여러 가지 방법을 시도해볼 것입니다. | | |
| 향후 추진 계획 | ■ 차주에는 분석 알고리즘인 N-gram, TF-IDF를 완벽하게 구현하고 두 개의 방식만이 아닌 word2vec, 군집분석 등을 통하여 다양한 방법으로 구현해볼 예정입니다. | | |

## 프로젝트 추진 보고서(팀실적 보고용) - 6주차

작성자 : 송준권 작성일자 : 2020.10.10.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | Word2Vec, 시나리오 등 | 진도(%) | 45% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 추진 내용 | 금주에는 원하는 데이터가 나올 수 있도록 가상의 분석 시나리오를 작성하였고 일부 테스트하였습니다. 몇 가지 분석방법을 이용하여 데이터를 추출해 보았습니다.<br>■ 연관어 분석에서 가장 흔히 쓰이는 Word2Vec, Doc2Vec 분석 기법을 이용했습니다. 해당 기법을 이용해 벡터화된 문맥을 생성하고, 유사도를 통해 키워드를 수집했습니다.<br>■ Word2Vec, Doc2Vec 에서는 문서(뉴스)의 수가 많아 적은 차원의 벡터 사이즈를 가질 때 결과가 좋지 않았습니다. 차원의 개수를 점점 늘려 갔을 때 결과가 개선되는 것을 확인했습니다.<br>■ Word2Vec로 주제인 문화와 관련 있는 키워드를 수집 중입니다. 해당 키워드를 모두 시계열로 빈도수 분석하여 관심도를 확인할 예정입니다.<br>■ 결과를 예측하며 분석 시나리오를 작성하였습니다. 빈도수를 시계열 분석하여 해당 키워드에 대한 관심도를 확인하였고, 확진자 수 추이와 대조해 보았습니다. | | |
| 문제점 및 대책 | ■ 간단하게 시나리오를 작성하며 분석의 흐름을 정하는 것을 꾀했으나, 테스트 케이스로 일부 분석을 수행하였을 때, 결과가 예상과 조금 달라 정확도를 높이는 작업이 필요하다고 생각했습니다.<br>■ word2vec, doc2vec에서 vector_size, epochs, window 등 여러 수치를 조금씩 변경해나가며 정확도를 높이는 작업을 진행할 예정입니다. | | |
| 향후 추진 계획 | ■ 시나리오 작성에 조금 더 신경을 써, 원하는 결과를 정제할 수 있도록 정확도를 높일 수 있도록 다양한 분석 / 연구 결과를 찾아볼 예정입니다. | | |

## 프로젝트 추진 보고서(팀실적 보고용) - 7주차

작성자 : 강다훈 작성일자 : 2020.10.18.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 시나리오 선정 | 진도(%) | 50% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 추진 내용 | 금주에는 여러 가지 시나리오를 선정하였습니다.<br>선정된 시나리오는 총 7가지로 다음과 같습니다.<br><br>■ 전체 기사 개수를 시계열 분석하여 선 도표 그래프로 표현하는 시나리오<br>■ 네이버의 DataLab, 구글의 Trends를 사용하고, 직접 선정한 키워드를 사용하여 검색양의 차이를 비교하는 시나리오<br>■ K-Means 클러스터링을 수행하여 군집화된 목록 중 '문화' 관련 군집을 선정한 뒤 Word Cloud 방식으로 가시화하는 시나리오<br>■ 토큰화 및 불용어 제거된 단어들 중 명사만을 사용하여 Word2Vec를 이용하여 most_similar를 확인하고 가시화하는 시나리오<br>■ 이전에 선정했던 문화 관련 키워드를 활용하여 선도표로 그래프화 이후 사전조사한 이슈(ex. 집단감염)의 기간과 대조해보며 영향력을 확인해보는 시나리오<br>■ LDA 토픽 모델링을 사용해 가시화하는 시나리오<br>■ Doc2Vec를 사용해 가시화하는 시나리오<br><br>■ 위 2개의 시나리오는 개략적으로 생각해본 시나리오이고, 이후 차차 구체화할 예정입니다. | | |
| 문제점 및 대책 | ■ 시나리오 선정 부분에서 구체적이지 않은 부분이 있어 이를 구체화하고 원하는 결과를 얻기 위한 가시화 방법을 찾고있습니다. | | |
| 향후 추진 계획 | ■ 1,2,3번의 시나리오를 수행하고, 더 좋은 방식의 시나리오가 있으면 적용해볼 예정입니다. | | |

## 프로젝트 추진 보고서(팀실적 보고용) - 8주차

작성자 : 강다훈 작성일자 : 2020.10.25.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 시나리오 수행 | 진도(%) | 55% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 추진 내용 | 금주에는 여러 가지 시나리오 중 1,2,3번의 시나리오를 수행하고 가시화하였습니다.<br><br>■ 전체 기사 개수를 시계열 분석하여 선 도표 그래프로 표현하는 시나리오.<br>위의 시나리오는 네이버 기사를 15일(보름)씩 크롤링했던 데이터를 기반으로 특정 사건과 비교하여 특정 사건이 일어났을 때 기사 양의 변화(관심도)를 확인하기 위한 시나리오입니다.<br><br>■ 네이버의 DataLab, 구글의 Trends를 사용하고, 직접 선정한 키워드를 사용하여 검색 양의 차이를 비교하는 시나리오.<br>위의 시나리오는 네이버 및 구글에서 제공하는 검색량 데이터를 가지고 선 그래프로 시각화한 시나리오로써 코로나 이전과 이후의 문화적 변화를 확인하기 위한 시나리오입니다.<br><br>■ K-Means 클러스터링을 수행하여 군집화된 목록 중 '문화' 관련 군집을 선정한 뒤 Word Cloud 방식으로 가시화하는 시나리오<br>위 시나리오의 K-Means 클러스터링 수행 부분에 군집을 정하는 부분에 있어 적정 군집의 개수를 찾는 것과 군집을 구성하는 데이터들의 정확도를 높이기 위하여 코드를 계속 수정 중에 있습니다. | | |
| 문제점 및 대책 | ■ 시나리오 가시화 부분에서 날짜를 일별로 표현하면 그래프의 높낮이가 요동치는 부분을 발견하여 보름 및 주간으로 수정하였습니다.<br><br>■ K-Means 클러스터링 부분에서 실습을 해보았는데 군집의 개수가 너무 많고 군집화 또한 제대로 이루어지지 않은 것 같아 코드를 수정하면서 보완하고 있습니다. | | |
| 향후 추진 계획 | ■ 1,2번의 가시화 부분에서 필요한 정보를 추가하고 더 한눈에 들어올 수 있도록 가시화 할 예정입니다.<br><br>■ K-Means 클러스터링과 4,5번 시나리오를 수행하고 가시화할 예정입니다. | | |

## 프로젝트 추진 보고서(팀실적 보고용) - 9주차

작성자 : 송준권 작성일자 : 2020.10.31.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 시나리오 수행 | 진도(%) | 60% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 추진 내용 | 금주에는 시나리오 수행 중 사용할 문서를 분류하고, 선정하는 작업을 진행했습니다.<br><br>■ 대표적으로 토픽 모델링 알고리즘인 LDA와 클러스터링을 위한 K-Means를 이용하여 분류 후 수행 결과를 비교해 보았습니다.<br><br>■ 먼저 LDA Model을 이용하여 토픽 모델링 작업을 수행했습니다. 먼저, 토픽의 개수를 지정하기 위해 개수를 점진적으로 늘려 모델을 생성하였고, 이후 CoherenceModel을 통해 토픽의 개수를 선정하였습니다.<br><br>■ 혹여 토픽 모델링 작업에 결과가 좋지 않을 것을 염려하여 문서 분류에 많이 쓰는 K-Means 알고리즘을 통하여 분류를 수행하였습니다.<br><br>■ 문서의 양이 많고, 또 비지도 학습으로 수행하는 만큼 정확도를 높이기 위해 분류에 많은 시간을 할애하고 있습니다. | | |
| 문제점 및 대책 | ■ 비지도 학습이기 때문에 정확도 측정에 난항이 있으나, 다양한 방법으로 측정을 시도하고 있습니다.<br><br>■ 텍스트 마이닝의 경우 시각화의 방식이 정해져 있어 다양한 방법을 모색 중입니다. | | |
| 향후 추진 계획 | ■ 분류된 문서만을 활용하여 여러 시각화를 시도할 예정입니다. | | |

## 프로젝트 추진 보고서(팀실적 보고용) - 10주차

작성자 : 강다훈 작성일자 : 2020.11.15.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 시나리오 수행 | 진도(%) | 65% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 추진 내용 | 금주에는 lda Topic Modeling, Network Visualization (네트워크 시각화)을 수행하였습니다.<br><br>■ lda Topic Modeling을 수행할 때 토픽의 개수를 2~3개를 선정하여 가장 좋은 토픽을 가졌는지, 얼마나 기사를 잘 분류했는지를 확인했고, 72개의 토픽을 가진 lda 모델을 선정하였습니다.<br><br>■ 72개의 topic에 속해있는 문서들을 분류하여 일상생활의 변화와 관련된 토픽들을 선정하여 가시화에 사용할 문서들을 선정하였습니다. 이후 더 좋은 결과를 얻기 위하여 한번 분류된 문서들을 토대로 lda modeling을 한 번 더 수행하였습니다. 하지만 분류 정확도가 높아지지는 않아 한번 수행한 lda 모델을 사용하기로 하였습니다.<br><br>■ 다른 시각화 방법을 찾아본 결과 Network Visualization (네트워크 시각화)은 클러스터와 관련 항목 간의 관계를 쉽게 식별할 수 있기 때문에 이 방법을 사용하기로 하였습니다.<br><br>■ 현재 네트워크 시각화를 수행 중에 있으며 최적의 노드의 개수를 찾아 정확도를 높이기 위해 많은 시간을 투자하고 있습니다. | | |
| 문제점 및 대책 | ■ 시각화를 할 때 현재는 가시성이 좋지 않지만 가시성을 좋게 하고, 다양한 가시화 방법을 적용하기 위해 노력 중에 있습니다.<br><br>■ 네트워크 시각화 부분에서 시각화의 방법이 여러 가지이기 때문에 다양한 결과를 얻을 수 있어 여러 가지의 방법을 사용 중에 있습니다. | | |
| 향후 추진 계획 | ■ 네트워크 시각화를 완료하여 가시화하고, lda modeling또한 원하는 토픽을 가져와 가시화 할 예정입니다.<br>■ 문서화 작업을 시작하여 사용한 기술과 결과 등을 기술할 예정입니다. | | |

## 프로젝트 추진 보고서(팀실적 보고용) - 11주차

작성자 : 송준권 작성일자 : 2020.11.20.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 시나리오 수행 | 진도(%) | 70% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 내용 | 금주는 이전까지의 분석했던, 공부했던 내용을 바탕으로 조금 더 보기 좋게 하는 시각화 작업과 문서화 작업을 진행했습니다.<br>■ 차트를 그릴 때는 bokeh 패키지를 이용합니다. bokeh는 그래프를 이미지로만 만드는 것이 아니라 결과물을 js로 렌더링할 수 있습니다.<br>■ 작성했던 시나리오 순서대로 도표를 만들었습니다. 전체 뉴스 건수와 기타 검색 엔진의 제공 정보를 활용한 그래프를 그렸습니다.<br>■ 결과물로 만들 주피터 노트북 문서에 들어갈 텍스트도 작성 중입니다. 현재 주 내용이 수행한 코드나 결과를 보기 좋게 작업 중입니다.<br>■ 문서 작업은 공동 작업 공간인 구글 드라이브에서 작성 후 문서 파일로 저장할 예정입니다.<br>■ 금주는 사용했던 데이터와 시나리오를 기준으로 시스템 정의와 설계 부분을 작성 중입니다. | | |
| 문제점 및 대책 | ■ 일부 부분에서 시각화 결과물이 보기 좋지 않아 지속적인 수정 작업에 있습니다.<br>■ 소프트웨어 개발 프로젝트와 양식이 달라 문서 구성을 계속 수정 중에 있습니다. | | |
| 향후 추진 계획 | ■ 금주에 이어 제출하는 문서와 결과물의 문서화 작업을 계속 진행할 예정입니다. | | |

## 프로젝트 추진 보고서(팀실적 보고용) - 12주차

작성자 : 강다훈 작성일자 : 2020.11.27.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 시나리오 수행 | 진도(%) | 80% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 내용 | 금주는 전주에 했던 시각화 작업과 문서화 작업을 이어서 진행했습니다.<br>■ 전주에 피드백 받았던 내용을 수정하고, 시각화 부분에서 부족한 점을 보완중에 있습니다.<br>■ 결과물로 제출할 주피터 노트북 문서에 들어갈 텍스트도 작성 중입니다. 현재 주 내용의 수행한 코드와 결과를 보기 좋게 작업 중입니다.<br>■ 금주는 사용했던 데이터와 시나리오를 기준으로 결과 보고서 작성중에 있습니다 | | |
| 문제점 및 대책 | ■ 기존에 작성하던 보고서들을 결과보고서에 담아야 하기에 수정 및 보완하면서 작성할 예정입니다. | | |
| 향후 추진 계획 | ■ 금주에 이어 제출하는 문서와 결과물의 문서화 작업을 계속 진행할 예정입니다. | | |

## 프로젝트 추진 보고서(팀실적 보고용) - 13주차

작성자 : 강다훈 작성일자 : 2020.12.07.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 시나리오 수행 | 진도(%) | 80% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 내용 | 금주는 전주에 했던 문서화 작업을 이어서 진행했습니다.<br>■ 최종 결과물로 제출할 결과 보고서 파일과 결과물로 제출할 주피터 노트북 문서를 작성하는데 집중하고 있습니다.<br>■ 상호 간의 피드백을 통하여 더 나은 결과물을 뽑기 위해 노력 중에 있습니다 | | |
| 문제점 및 대책 | ■ 비전공자 및 일반인들도 잘 알아볼 수 있도록 설명을 추가하고 있고, 전공자들의 코드 이해를 위해 주석을 추가하고 있습니다. | | |
| 향후 추진 계획 | ■ 금주에 이어 제출하는 문서와 결과물의 문서화 작업을 계속 진행할 예정입니다. | | |

## 프로젝트 추진 보고서(팀실적 보고용) - 14주차

작성자 : 송준권 작성일자 : 2020.12.11.

| 프로젝트 명 | 코로나 이펙트 | | |
|---|---|---|---|
| 추진 활동 | 시나리오 수행 | 진도(%) | 100% |
| 팀 명 | 두잇 | 팀원 성명 | 강다훈, 송준권 |
| 내용 | 금주는 마무리하는 단계로 지금껏 생성하고, 수행했던 문서, 코드, 파일을 정리하고 다듬는 시간을 가졌습니다.<br>■ 최종 결과물인 노트북 파일을 완성하였습니다. 완성된 파일을 gist 에 업로드 하여 Jupyter Notebook Viewer에서 확인할 수 있도록 하였습니다.<br>■ 시스템 정의 / 설계서, 최종 보고서를 검토하며 마무리 하였습니다.<br>■ 동영상을 촬영하였습니다. 소프트웨어가 아닌 보고서 형식의 결과물이므로 과정을 설명하며 촬영했습니다. | | |
| 문제점 및 대책 | ■ 해당 분야에 대한 이해 없이 처음부터 무작정 분석을 진행하여 많은 벽을 느꼈습니다.<br>■ 데이터를 처리하는 것도 어려웠지만, 그것의 의미를 추출하는 것이 매우 어렵다고 느꼈습니다. | | |
| 향후 추진 계획 | | | |

# 3. 프로젝트 상세정보

## I. 사전데이터 수집 및 가공

### A. 기사 파싱

파싱 대상 : 네이버 뉴스

수집 기간 : 2020년 01월 01일 ~ 2020년 08월31일

수집 키워드 : “코로나”

대상 언론사 : 네이버 뉴스 포맷을 지원하는 종합지 및 방송/통신사 24곳

데이터 저장 형식 : csv 파일로 pandas의 DataFrame 자료구조 형태로 저장.

네이버에서 검색할 때 뉴스기사는 한번에 4000개까지밖에 보여지지 않음.
최대한 많은양의 기사를 얻기 위하여 2일 단위로 검색하게 하였고,
언론사 고유번호를 사용하여 특정 언론사에서 하루에 기사를 작성했을 시 특정 키워드가 포함된 기사의 수는 4000개가 안될것으로 판단하여 언론사 고유번호를 사용하여 최대한 많은 양의 기사를 얻을 수 있도록 함.

저장된 데이터 구조는 아래와 같음

| title | date | category | text | press | link |
|---|---|---|---|---|---|
| 뉴스 기사 제목 | 기사 작성 일자 | 네이버에서 분류한 카테고리 | 기사 본문 | 네이버에서 사용되는 언론사 고유번호 | 기사 링크 |

| | title | date | category | text | press | link |
|---|---|---|---|---|---|---|
| 0 | 공사대금 문제로 시공사 대표 폭행한 전직 프로야구 선수 체포 | 2020-01-01 14:36:45 | 사회 | ₩t₩t₩t(서울=연합뉴스) 권선미 기자 = 전직 프로야구 선수가 입주 예정이던 신... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... |
| 1 | 말레이시아 '음식점 흡연' 708명에 첫 벌금 부과 | 2020-01-03 11:32:14 | 세계 | ₩t₩t₩t1년 계도기간 끝나고 올 1월 1일부터 일제 단속(자카르타=연합뉴스) 성... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... |
| 2 | 미스트롯 전국투어 의정부 공연, 안전 문제로 취소 | 2020-01-05 16:46:19 | 생활 | ₩t₩t₩t공연기획사 "무대 설치 도중 구조물 일부 무너져"원본보기미스트롯 전국투어... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... |
| 3 | 국내서 '중국 원인불명 폐렴' 증상자 발생...36세 중국 여성(종합2보) | 2020-01-08 18:42:46 | 사회 | ₩t₩t₩t지난달 우한시 방문 후 입국...현재 격리치료 중-건강상태 양호질본, 폐렴 ... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... |
| 4 | 국내서 '중국 원인불명 폐렴' 증상자 발생...36세 중국 여성(종합) | 2020-01-08 18:01:20 | 사회 | ₩t₩t₩t지난달 우한시 방문 중국인 여성...현재 건강상태 양호질본, 폐렴 유발 원인... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... |


[그림 1. 종합지 및 방송/통신사 언론사 리스트]

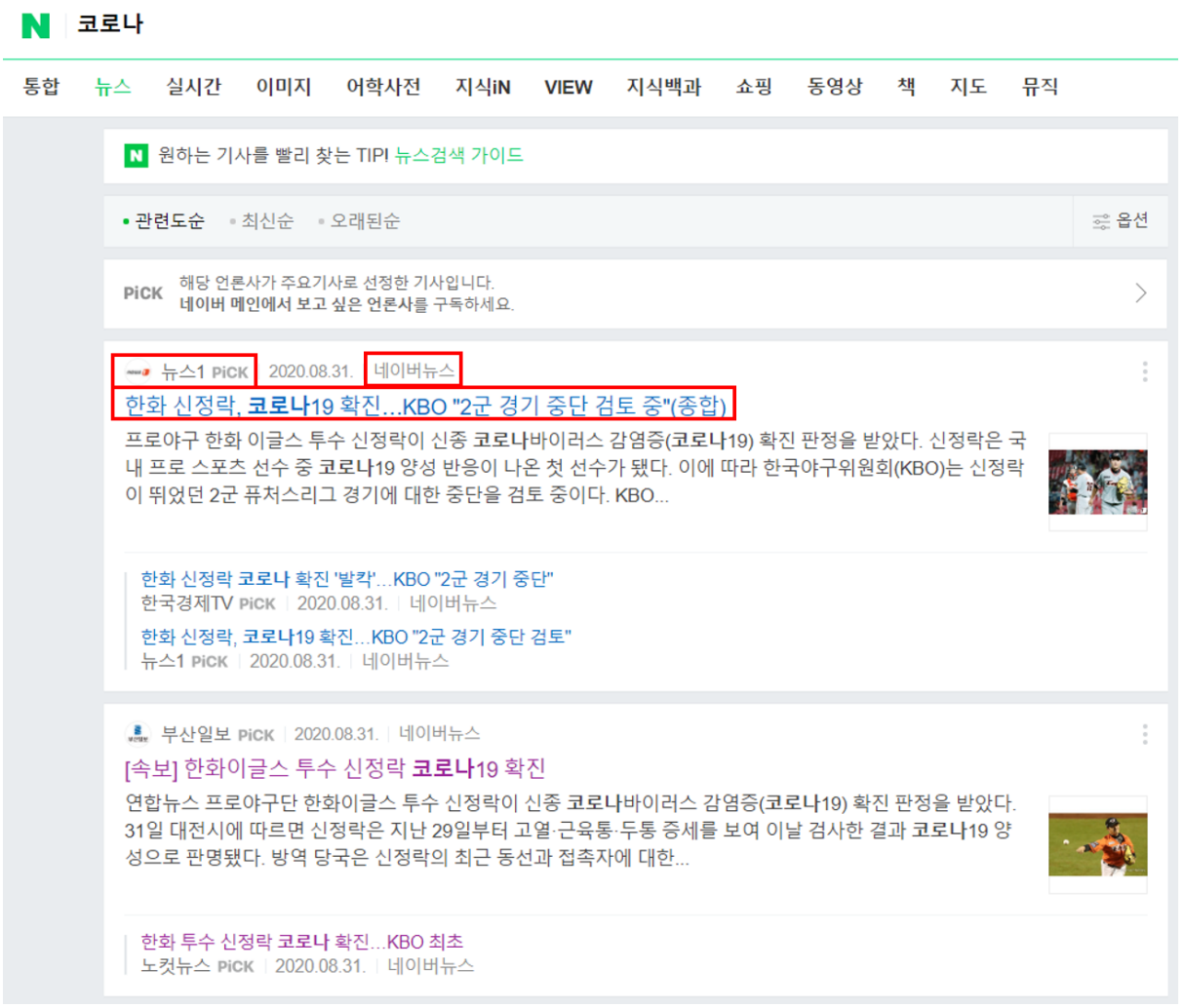
Crawl.py (깃허브에 올려둔 뉴스 파싱코드)
아래 코드는 깃허브에 저장해두고 본 코드에서 불러와서 실행함.

```
# coding: utf-8


# 필요한 패키지 설치
import requests
from bs4 import BeautifulSoup as bs4
from multiprocessing import Pool
import csv
import datetime
import time
```

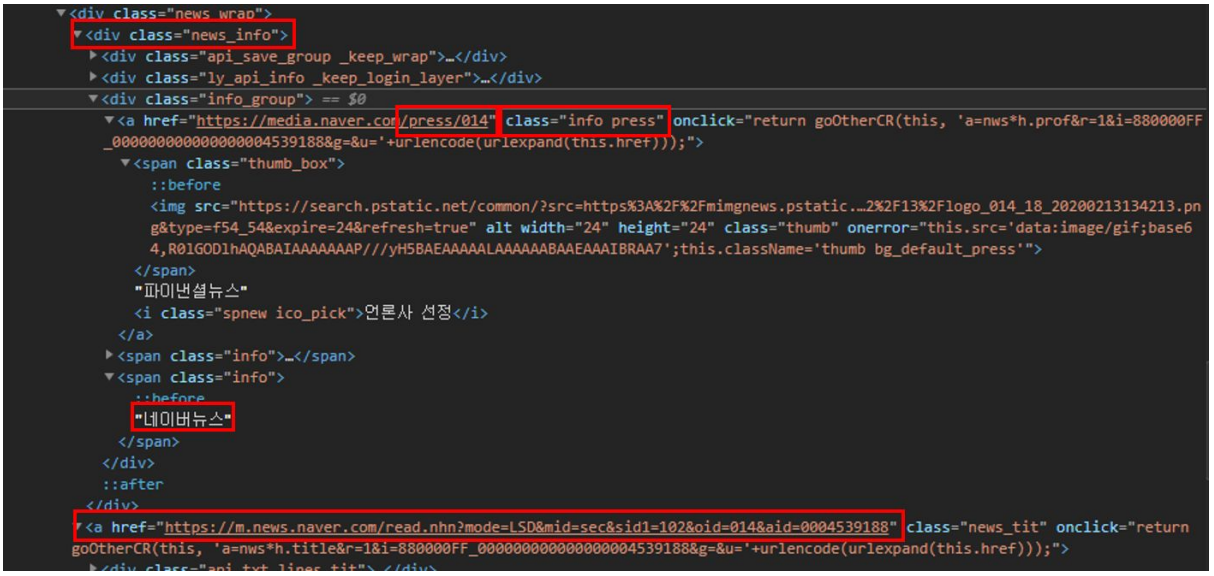
```
import os
import sys

# request 보낼 시에 사용할 헤더 생성.

headers = {
  "user-agent": "Mozilla/5.0 (iPhone; CPU iPhone OS 13_1_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.1 Mobile/15E148 Safari/604.1",
  "authority": "news.naver.com",
  "cache-control": "max-age=0",
  "upgrade-insecure-requests": "1",
  "dnt": "1",
  "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8",
  "accept-encoding": "gzip, deflate, br",
  "accept-language": "ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7,la;q=0.6,da;q=0.5",
}


# 프로그램 실행 도중 중단을 대비하여 완료된 파일 저장.
# done.txt에 있는 파일은 다시 동일 내용 실행 시에도 중복으로 수행되지 않음.

DONEFILE = "done.txt"

if os.path.isfile(DONEFILE):
  with open(DONEFILE, "r", encoding="utf-8") as done:
    done_list = done.readlines()
  done_list = [_t.replace("\n", "") for _t in done_list]
else:
  done_list = []
```

```
# 뉴스 검색 결과 한페이지 정보 수집
# '네이버뉴스' 일 경우 csv 파일에 기록

def crawl_whole_search_page(link):
```

```
  while True:
    ## 차단 확인 ( 짧은시간에 지속적으로 요청하면 검색이 안되도록 잠시 차단함)
    tmp = requests.get(link, headers=headers)
    if "200" not in str(tmp):
      print("load err, wait 30sec")
      time.sleep(30)
    else:
      break
  soup = bs4(tmp.text, "html.parser")
  new_div_list = soup.select("div.group_news > ul  div.news_wrap")
  tmp = []
  for _news in new_div_list:
    _data = crawl_news(_news)
    if _data is not None:
      tmp.append(_data)
  return tmp


# 뉴스 정보 반환
# '네이버뉴스' 일 경우만사용
def crawl_news(_news):
  if "네이버뉴스" in str(_news.select_one("div.news_info")):
    title = _news.select_one("div.api_txt_lines").text
    link = _news.select_one("a.news_tit")["href"]
    press = _news.select_one("a.info.press")["href"]
    press = press.split("/")[-1]
    date, text, category = crawl_inpage(link)
    return [title, date, category, text, press, link]
  else:
    return None

# 연예 카테고리가 DOM 구조가 다른 경우를 위해 애러 처리함.
# 카테고리가 지정되지 않은 경우 “연예 “ 카테고리로 지정
def get_news_category(soup):
  try:
    return soup.select_one("em.media_end_categorize_item").text
  except:
    return "연예"
```




[그림2. 네이버의 검색창]




[그림3. 네이버 HTML코드에 언론사번호 및 기사제목,
기사의 형식등이 포함되어 있음. ]

생성된 검색어 목록을 이용해 키워드, 검색 범주, 언론사로 crawl_news(_news)에서 뉴스 기사를 수집함.

```
#뉴스 기사 페이지를 파싱함.

def crawl_inpage(_link):
  date = ""
  text = ""
  category = ""
  # DOM 구조가 다를 경우를 대비해 예외처리
  try:
    tmp = requests.get(_link, headers=headers)
    soup = bs4(tmp.text, "html.parser")
   # 기사의 날짜 본문 카테고리를 가져옴
    date = soup.select_one("span._ARTICLE_DATE_TIME")["data-date-time"]
    text = soup.select_one("div#dic_area").text
    text = text.replace("\n", "")
    category = get_news_category(soup)
  except AttributeError:
    for _i in [date, text, category]:
      if _i == "":
        _i = "None"
  finally:
    return date, text, category
```

기사에 접속하여 기사의 작성날짜, 본문 텍스트 및 카테고리를 수집하였음.
본문은 다음 그림과 같음


[그림4. 네이버뉴스 형식의 기사에 접속한 모습]

다음의 text를 가져오기 위해서 HTML 코드를 분석한 결과는 다음과 같음.


[그림5.기사의 입력시간, 본문, 카테고리의 태그]

```
# 파싱할 날짜, 키워드, 기간, 언론사가 포함된 딕셔너리 파일을 기준으로 파싱 수행.
def start_crawl(_dict):
  global done_list
  baseurl =
"https://m.search.naver.com/search.naver?where=m_news&query={key
word}&sm=mtb_tnw&sort=0&photo=0&field=0&pd=3&ds={org_start}&de=
{org_end}&docid=&related=0&mynews=1&office_type=1&office_section_c
ode=1&news_office_checked=1{press}&nso=so%3Ar%2Cp%3Afrom{start}t
o{end}"
  keyword = _dict["keyword"]  # 검색창에 입력할 키워드
  org_start = _dict["start"] # 검색 시작 날짜
  org_end = _dict["end"] # 검색 끝나는 날짜
  press = _dict["pressnum"] # 언론사 고유번호
  start = org_start.replace(".", "")
  end = org_end.replace(".", "")

  fname = "./{}_{}_{}_{}.csv".format(keyword, press, start[4:], end[4:])
  n_url = baseurl.format(
    keyword=keyword,
    org_start=org_start,
    org_end=org_end,
    press=press,
```

```
    start=start,
    end=end,
)

if fname in done_list:
    return

# 파일에 저장
with open(fname, "w", newline="", encoding="utf-8") as f:
    file = csv.writer(f, delimiter=",", quotechar='"', quoting=csv.QUOTE_ALL)
    # header
    file.writerow(["title", "date", "category", "text", "press", "link"])
    tmp = ""
    for page in range(267):
        url = n_url + "&start=" + str(page * 15 + 1)
        page_rows = crawl_whole_search_page(url)
        # 중복기록최소화를 위해 이전과 동일한 기사인 경우 저장하지 않음
        if tmp != page_rows:
            file.writerows(page_rows)
            tmp = page_rows
print("file: '{}' done".format(fname))

# 파싱 내용 저장 완료후 done.txt에 추가
with open(DONEFILE, "a", encoding="utf-8") as done:
    done.write(fname + "\n")
```

네이버에 검색 URL 구조는 다음과 같음.

**"https://m.search.naver.com/search.naver?where=m_news&query={keyword}&sm=mtb_tnw&sort=0&photo=0&field=0&pd=3&ds={org_start}&de={org_end}&docid=&related=0&mynews=1&office_type=1&office_section_code=1&news_office_checked=1{press}&nso=so%3Ar%2Cp%3Afrom{start}to{end}"**

이중 우리가 필요한부분인 키워드, 검색시작날짜와 끝나는 날짜를 입력하면 뉴스 기사를 수집함. 최대한 많은 데이터를 수집할 수 있도록 setSearchList() 에서 검색 목록을 쪼개어 생성함.

```
# 파싱할 목록 생성
def setSearchList():
# 언론사 고유번호 리스트
```

```
press_num_list = [
    32,
    5,
    20,
    21,
    81,
    22,
    23,
    25,
    28,
    469,
    437,
    56,
    214,
    57,
    374,
    55,
    448,
    52,
    421,
    3,
    1,
    422,
    449,
    215,
]

# command line 지원
if len(sys.argv) == 5:
    KEYWORD = sys.argv[1]
    STARTDATE = sys.argv[2]
    ENDDATE = sys.argv[3]
    DAYS = int(sys.argv[4])
else:
    KEYWORD = input("키워드 입력 : ")
    STARTDATE = input("시작 날짜 입력 예) 20200131:  ")
    ENDDATE = input("종료 날짜 입력 예) 20200131:  ")
    # 몇일 단위로 쪼개어 검색할지.
    DAYS = int(input("몇일치씩 다운로드 할까요? : "))

start_date = datetime.date(
```

```
    int(STARTDATE[0:4]), int(STARTDATE[4:6]), int(STARTDATE[6:])
  )
  end_date = datetime.date(int(ENDDATE[0:4]), int(ENDDATE[4:6]),
int(ENDDATE[6:]))

  arr = []
  date = start_date
  while date <= end_date:
    start = str(date).replace("-", ".")
    date += datetime.timedelta(days=DAYS - 1)
    if date > end_date:
      date = end_date
    end = str(date).replace("-", ".")
    date += datetime.timedelta(days=1)
    for press in press_num_list:
      press = "%03d" % press
      dic = {"keyword": KEYWORD, "start": start, "end": end, "pressnum":
press}
      arr.append(dic)
  return arr

# main 메서드
# 빠른 처리를 위해 multiporecessing 패키지 사용
if __name__ == "__main__":
  search_list = setSearchList()
  pool = Pool(processes=8)
  pool.map(start_crawl, search_list)
```

본격적인 코딩은 구글의 Colaboratory를 이용.

[conclusion.ipynb]

```
root = '/content/drive/Shared drives/Corona Effect/case8/'

# 구글 드라이브에 접근하여 사용할 수 있도록 마운트

import os, sys
from google.colab import drive

drive.mount('/content/drive', force_remount=True)
os.symlink(root, '/content/corona')

root = '/content/corona/'

%cd {root}
```

```
import pandas as pd
import re, os

# 보름 단위로 수집하기 위해서 배열 생성
dateset = [('0101','0115'), ('0116','0131'), ('0201','0215'), ('0216', '0229'),
('0301','0315'), ('0316','0331'),
('0401','0415'), ('0416','0430'), ('0501','0515'), ('0516', '0531'), ('0601','0615'),
('0616','0630'),
('0701','0715'), ('0716','0731'), ('0801','0815'), ('0816','0831')]

# 폴더에 있는 파일 목록을 가져오는 메서드
def get_filelist(path, extension):
 flist = os.listdir(path)
 result = []
 if path.endswith('/') == False:
   path += '/'
 for file in flist:
   if file.endswith(f'.{extension}'):
     result.append(path+file)
 return sorted(result)
```

```
# 네이버 뉴스를 파싱하는 메서드 선언
def crawl_news(l: tuple):
  os.makedirs(f'{l[0]}-{l[1]}', exist_ok=True)
  os.chidir(f'{l[0]}-{l[1]}')

# 미리 작성한 파싱 코드를 깃에서 불러와 실행(위에서 설명한 Crawl.py)
  !git clone https://github.com/elppaaa/naver-news-crawler.git
  !mv naver-news-crawler/crawl.py ./
  !rm -r naver-news-crawler

  start = '2020' + l[0]
  end = '2020' + l[1]
  %time %run crawl.py 코로나 {start} {end} 2
  !rm crawl.py


# 언론사 / 일 단위로 수집된 뉴스 기사를 보름 단위로  합치는 메서드
def merge_news(d: str):
  df = pd.DataFrame()
  flist = get_filelist('../newscsvraw/' + d, 'csv')

  for file in flist:
    tmp = pd.read_csv(file)
    df = pd.concat([df, tmp])
  df.to_csv(f'{d}.csv' ,index=False)

# 뉴스 크롤링 수행
for v in dateset:
  os.chdir(root +'newscsvraw')
  crawl_news(v)

# 크롤링한 뉴스 보름 단위로 저장
for l in dateset:
  name = f'{l[0]}-{l[1]}'
  os.chdir(root + 'newscsv')
  merge_news(name)
```

```
# 크롤링된 뉴스를 확인
pd.read_csv(get_filelist(root + 'newscsv', 'csv')[0]).head()
```

| | title | date | category | text | press | link |
|---|---|---|---|---|---|---|
| 0 | 공사대금 문제로 시공사 대표 폭행한 전직 프로야구 선수 체포 | 2020-01-01 14:36:45 | 사회 | ₩t₩t₩t(서울=연합뉴스) 권선미 기자 = 전직 프로야구 선수가 입주 예정이던 신... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... |
| 1 | 말레이시아 '음식점 흡연' 708명에 첫 벌금 부과 | 2020-01-03 11:32:14 | 세계 | ₩t₩t₩t1년 계도기간 끝나고 올 1월 1일부터 일제 단속(자카르타=연합뉴스) 성... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... |
| 2 | 미스트롯 전국투어 의정부 공연, 안전 문제로 취소 | 2020-01-05 16:46:19 | 생활 | ₩t₩t₩t공연기획사 "무대 설치 도중 구조물 일부 무너져"원본보기미스트롯 전국투어... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... |
| 3 | 국내서 '중국 원인불명 폐렴' 증상자 발생...36세 중국 여성(종합2보) | 2020-01-08 18:42:46 | 사회 | ₩t₩t₩t지난달 우한시 방문 후 입국...현재 격리치료 중·건강상태 양호질본, 폐렴 ... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... |
| 4 | 국내서 '중국 원인불명 폐렴' 증상자 발생...36세 중국 여성(종합) | 2020-01-08 18:01:20 | 사회 | ₩t₩t₩t지난달 우한시 방문 중국인 여성...현재 건강상태 양호질본, 폐렴 유발 원인... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... |

## B. 전처리(정제 & 정규화)

모든 데이터 분석 프로젝트에서 데이터 전처리는 반드시 거쳐야 하는 과정.
분석 결과/인사이트와 모델 성능에 직접적인 영향을 미치는 과정이기 때문에
중요하게 다루어짐.
우리가 파싱한 데이터셋은 바로 분석이 불가능할 정도로 지저분함.
분석이 가능한 상태로 만들기 위해 전처리를 실행함.

뉴스 기사의 본문에는 기자의 이름, 이메일, 언론사 이름, 광고 등 분석할 때 필요 없는 데이터들이 섞여있는 경우가 있어 정규표현식을 사용해 수행하고 중복 또는 비어있는(Nan) 기사들을 제거하여 저장.

네이버 뉴스가 제공하는 카테고리 6종[정치, 경제, 사회, IT, 생활, 세계]중
사람들의 생활, 문화의 변화를 분석하기 위해 필요한 4종[경제, 사회, 생활, 세계]의
카테고리가 적절하다고 판단하여 사용

저장된 데이터는 이전과 동일하나, 기사 본문(text) 필드만 전처리 후 저장.

| title | date | category | text | press | link |
|---|---|---|---|---|---|
| 뉴스 기사 제목 | 기사 작성 일자 | 네이버에서 분류한 카테고리 | 기사 본문 | 네이버에서 사용되는 언론사 고유번호 | 기사 링크 |

```
import pandas as pd
import re, os

# 보름단위를 데이터셋에 저장
dateset = [('0101','0115'), ('0116','0131'), ('0201','0215'), ('0216', '0229'),
('0301','0315'), ('0316','0331'),
('0401','0415'), ('0416','0430'), ('0501','0515'), ('0516', '0531'), ('0601','0615'),
('0616','0630'),
('0701','0715'), ('0716','0731'), ('0801','0815'), ('0816','0831')]
```

```
제거할 문자열 목록을 정규표현식으로 생성함
regexlist = []

# 다음과 같은 리스트 제거
# 이메일의 형식
regexlist.append( re.compile(r'\w+@[\w\.]+', flags=re.U))

# 제보하기 같은 맨 마지막 광고
regexlist.append( re.compile(r'◀[^▶]+▶', flags=re.U))
regexlist.append(re.compile(r'▶.+$', flags=re.U))

# 뉴시스
regexlist.append(re.compile(r'공감언론 뉴시스[^\>]*', flags=re.U))
# 화살표
regexlist.append(re.compile(r'☞.+$', flags=re.U))
# Copyright
regexlist.append(re.compile(r'Copyright.+$', flags=re.U))

# # /뉴스1 © News1
regexlist.append(re.compile(r'\/\w*연합뉴스', flags=re.U))


# 대괄호 제거
regexlist.append(re.compile(r'\[[^\]]*\]', flags=re.U))
# 대괄호랑 비슷한 특수문자 대괄호 [ ]
regexlist.append(re.compile(r' [ [^ ] ]* ] ', flags=re.U))


# 소괄호 제거
regexlist.append(re.compile(r'\([^\)]*\)', flags=re.U))
# < > 제거
regexlist.append(re.compile(r'<[^\>]*>', flags=re.U))
# 【 】
regexlist.append(re.compile(r'【[^】]*】', flags=re.U))


#인터넷 주소
regexlist.append(re.compile(r'http[^\s\(\[=]+', flags=re.U))

# 기자, 특파원
regexlist.append(re.compile('\w{2,3}\s{0,2}(기자|특파원)', flags=re.U))
```

```
# 정규표현식이 필요하지 않은 단순 문자열
replacelist = [
 '페이스북 tuney\.kr/LeYN1',
 '트위터 @yonhap_graphics',
 '현지시각',
 '원본보기',
 '© 뉴스1'
]

#  각 기사 본문을 정규표현식을 이용하여 불필요한 문자열을 제거함.
def textrunner(row):
 val = row['text']
 row['text'] = regexrunner(val)
 return row

# 위의 컴파일된 re 객체를 이용하여 불필요한 텍스트 제거
def regexrunner(text):
 for r in regexlist:
   text = r.sub('', str(text))
 for r in replacelist:
   text = text.replace(r, '')
 return text
```

```
#  클렌징 작업
filelist = get_filelist(root + 'newscsv', 'csv')
os.makedirs(root+'cleansing', exist_ok=True)
os.chdir(root+'cleansing')
for file in filelist:
   df = pd.read_csv(open(file, 'r'))
   file = file.split('/')[-1]
   df = df.drop_duplicates().dropna()  # NaN, 중복 제거
   df= df[df.category.isin(['경제',  '사회',  '생활',  '세계'])]  # 카테고리 선정
   df = df.sort_values(by=['date'], axis=0) # 시간 순서대로 정렬.
   df.apply(textrunner, axis=1).drop('index', axis=1).to_csv(file, index=False)
```

```
# 클렌징된 여러개의 기사를 하나의 파일로 합침

f = root+'cleansing/cleansing_full.csv'
os.chdir(root + 'cleansing')

for file in get_filelist(root+'cleansing', 'csv'):
 if not os.path.exists(f):
   df = pd.read_csv(open(file, 'r'))
   df['date'] = pd.todatetime(df['date'])
   df.sort_values(by=['date'], axis=0)
   df.to_csv(f, mode='w', index=False, encoding='utf-8')
 else:
   df = pd.read_csv(open(file, 'r'))
   df['date'] = pd.todatetime(df['date'])
   df.sort_values(by=['date'], axis=0)
   df.to_csv(f, mode='a', index=False, header=False, encoding='utf-8')
```

## C. 토큰화

토큰화란 주어진 말뭉치(corpus)에서 토큰(token)이라 불리는 단위로 나누는 작업을 말함.  토큰이란 문법적으로 더 이상 나눌 수 없는 언어요소.
텍스트 토큰화란 말뭉치로부터 토큰을 분리하는 작업을 뜻함.

한국어에는 '어절'이라는 띄어쓰기 단위가 있는데 어절 토큰화와 단어 토큰화가 같지가 않음. 영어와 다르게 한국어는 독립적인 단어로만 구성되있지 않기에 형태소라는 개념을 인지하고 사용해야함.

전처리된 데이터를 활용하여 기사 본문과 제목을 형태소분석기중 MeCab을 이용해 용도에 따라 사용하기 위해 모든 품사인 경우와 명사만을 이용하는 경우로 나누어 저장함.  이때, 명사는 일반명사, 고유명사(NNP,NNG), 외국어(SL)만을 사용.

파이썬에서는 변수에 담겨있는 데이터를 구조 그대로 저장할 수 있는 pickle 이라는 라이브러리를 사용.

경로 : tokenzied/

파일 확장자 : .pkl

구조 : 토큰화된 문서 배열로 이루어진 2차원 배열

```
# KoNLPy, Mecab dic 설치
!set -x \
&& pip install -q konlpy \
&& curl -s
https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh | bash -x > /dev/null
```

```
import pandas as pd
import re, os
import pickle
from konlpy.tag import Mecab
```

```
# 특수문자를 제거하고 단어만 사용하기 위한 정규표현식
r = re.compile(r'[^ ㄱ-ㅣ 가-힣|0-9|a-zA-Z]+', flags=re.U | re.M)

mecab = Mecab()

# “ . ” “ , ”을 공백으로 대치
def clean_text(text):
  text = text.replace(".", " ").strip()
  text = text.replace("·", " ").strip()
  text = r.sub('', string=text)
  return text

# NNP, NNG, SL만 분리
def get_nouns(sentence):
 return [morphs[0] for morphs in mecab.pos(sentence) if morphs[1] in
['NNP', 'NNG',  'SL'] and len(morphs[0]) > 1]


def tokenize(df):
  processed_data = []
  for sent in df['text']:
    sentence = clean_text(str(sent).replace('\n', '').strip())
    processed_data.append(get_nouns(sentence))
  return processed_data
```

```
# 폴더 생성 및 이동
os.makedirs(root + 'tokenized', exist_ok=True)
os.chdir(root + 'tokenized')

# 파일저장
for i, rows in
enumerate(pd.read_csv(root+'cleansing/cleansing_full.csv',
chunksize=50000, iterator=True)):
 with open('%02d.pkl' % i, 'wb') as f:
  pickle.dump(tokenize(rows), f)
```

토큰화된 파일을 읽어 정상적으로 토큰화가 되었는지 확인

```
with open(root + 'tokenized/01.pkl', 'rb') as f:
 print(pickle.load(f)[:2])
```

OUT: [['미혹', '현명', '선택', '리어', '오셀로', '한국', '정치', '채찍', '로알드', '고조부', '셀베르그', '목사', '성령', '강림절', '노르웨이', '웨스트민스터', '가의', '예배당', '사람', '셀베르그', '목사', '성령', '강림', '설교', '공포', '교인', '이상', '목숨', '사람', '떼죽음', '지옥', '사람', '성경', '제단', '유리창', '유리창', '탈출', '셀베르그', '목사', '사람', '당시', '참사', '대서특필', '유럽', '신문', '목사', '판단력', '칭송', '위기', '방법', '생각', '무리', '진취', '시대', '영웅', '독립', '인간', '생각', '사람', '훗날', '노르웨이', '국회의원', '활동', '공공', '기관', '건물', '건축법', '제정', '도움', '셀베르그', '목사', '천재', '이야기', '로알드', '고조부', '고조부', '고조부', '위기관리', '능력', '현실', '능력', 'DNA', '어린이', '마음', '아이', '얘기', '세상', '기여', '리어', '오셀로', '맥베스', '리어', '사랑', '막내딸', '코델리아', '코델리아', '아버님', '자식', '의무', '사랑', '이상', '이하', '마음', '현명', '감언이설', '막내딸', '코델리아', '권력', '재산', '결과', '광야', '미치광이', '프랑스', '왕비', '막내딸', '도움', '막판', '권력', '코델리아', '감옥', '살해', '리어', '슬픔', '가슴', '장군', '오셀로', '마음씨', '데스', '데모', '니아', '행복', '결혼', '부하', '기수', '이아고', '거짓말', '아내', '살해', '사람', '마음', '사정', '농락', '괴물', '의심', '조금', '핏속', '작용', '유황광', '억측', '괴물', '의처증', '오셀로', '이아고', '농락', '이아고', '자신', '자결', '스코틀랜드', '맹장', '맥베스', '마녀', '망언', '자신', '백성', '죽음', '예언', '던컨', '살해', '자신', '장군', '쿠오', '자손', '망언', '죄책감', '부인', '맥베스', '여자', '맥베스', '맥베스', '영원', '불패', '망언', '의지', '더프', '해당', '리어', '오셀로', '자신', '백성', '고통', '죽음', '맥베스', '분수', '마녀', '욕망', '파멸', '사람', '공자', '가르침', '비극', '도쿄', '선언', '삼성', '반도체', '삼성그룹', '창업', '이병철', '회장', '일본', '도쿄', '고밀도', '집적회로', '규모', '투자', '발표', '삼성', '대한민국', '일본', '미국', '세계', '도쿄', '선언', '당시', '삼성', '가전제품', '고밀', '집적회로', '반도체', '산업', '선도', '인텔', '도쿄', '선언', '과대망상증', '환자', '다음', '기흥', '반도체', '공장', '기공식', '참석', '임직원', '불안감', '삼성그룹', '절반', '이상', '우려', '회장', '삼성', '모험', '결과', '기대', '이상', '삼성전자', '매출액', '한국', 'GDP', '당기', '순이익', '반도체', '경기', '부진', '매출액', '당기', '순이익', '예상', '한국', '경제', '클럽', '기관', '차임', '이병철', '회장', '삼성', '그때', '반도체', '사업', '삼성', '반도체', '성공', '리더', '결단', '기업', '국가', '발전', '결정', '역할', '국민', '국민', '외면', '정치', '대한민국', '국회', '식물', '국회', '오명', '물리', '동물', '국회', '지양', '국회', '선진', '화법', '이제', '민생', '대화', '타협', '정치', '본질', '이익', '꼼수', '집착', '정치', '정치', '정치', '국민', '생명', '재산', '안전', '걱정', '책무', '지도자', '리어', '오셀로', '맥베스', '좌우', '이병철', '회장', '앞날', '변화', '투자', '준비', '코앞', '이익', '사람', '적극', '설득', '방향', '셀베르그', '목사', '위기', '상황', '생각', '타개', '방안', '목숨', '황금', '돼지해', '쥐해', '자년', '국회의원', '선거', '이후', '정치', '정치가', '과제', '실천', '이익', '추구', '정치', '식물', '국회', '정치', '국회의원', '정치가', '선출', '나라', '주인', '유권자'], ['전직', '프로', '야구', '선수', '입주', '예정', '신축', '빌라', '공사', '대금', '문제', '시공사', '대표', '말다툼', '폭력', '행사', '혐의', '경찰', '체포', '경찰', '서울', '서초', '경찰서', '전직', '프로', '야구', '선수', '특수', '폭행', '혐의', '입건', '지난해', '오전', '서울', '서초구', '서초동', '건축', '시공사', '사무실', '회사', '대표', '혐의', '경찰', '골프채', '주장', '신고', '출동', '경찰', '현행범', '체포', '경찰', '조사', '신축', '빌라', '입주', '시공사', '당초', '공사비', '내역', '추가', '공사비', '요구', '진술', '병원', '입원', '귀가', '경찰', '관계자', '상대', '정확', '범행', '동기', '조사', '한편', '초반', '지방', '구단', '소속', '프로', '야구', '선수', '생활']]

# II.분석 시나리오

## I. 전체 기사 시계열 분석

**목표**
크롤링한 전체 뉴스 기사를 1주 단위로 취합하여 기사 건수의 변화 추이를 시계열 방식으로 시각화하는 시나리오

**기능**
코로나로 인한 사건, 사고의 발생으로 주목이 많이 될수록 기사 건수는 증가한다고 판단하여 뉴스 기사 건수를 통해 그 시기의 사건 및 사고를 대조함.
또한 국가 통계 포털에서 제공하는 일 단위 확진자 수 데이터를 활용하여 함께 비교하였음.

**사용된 라이브러리**
bokeh를 사용하여 반응형 그래프를 사용
- 그래프에 hover 효과를 부여하여 마우스를 이용하여 날짜와 기사 개수를 확인.

```
# 시각화 툴 및 데이터를 불러오기 위한 선언
from bokeh.plotting import output_notebook, figure, show
from bokeh.palettes import viridis, Spectral, Paired
from bokeh.models import ColumnDataSource, DatetimeTickFormatter,
HoverTool, Legend, Range1d, LinearAxis
import datetime
import pandas as pd


output_notebook()
```

```
# 국가 통계 포털의 일 확진자 데이터 불러오기
corona_data = pd.read_csv(root + 'corona_count_modify.csv')
corona_data['date'] = pd.to_datetime(corona_data['date'])
```

```
# 국가 통계 포털의 일 확진자 데이터 확인
corona_data.head()
```

| | date | 신규 확진 | 누적 확진 | 격리 증감 | 누적 격리 | 신규 격리해제 | 누적 격리 해제 | 신규 사망 | 누적 사망 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2020-02-04 | 18 | 18 | - | - | 0 | 0 | 0 | 0 |
| 1 | 2020-02-05 | 1 | 19 | - | - | 1 | 1 | 0 | 0 |
| 2 | 2020-02-06 | 4 | 23 | - | - | 0 | 1 | 0 | 0 |
| 3 | 2020-02-07 | 1 | 24 | - | - | 1 | 2 | 0 | 0 |
| 4 | 2020-02-08 | 0 | 24 | - | - | 0 | 2 | 0 | 0 |

```
# 일일 확진자 데이터를 1주일(1week)단위로 합치고, 데이터 확인
corona_data = corona_data.groupby(pd.Grouper(key='date', freq='1w'))
corona_data = corona_data.sum()['신규 확진']

pd.DataFrame(corona_data).head()
```

| date | 신규 확진 |
|---|---|
| 2020-02-09 | 27 |
| 2020-02-16 | 2 |
| 2020-02-23 | 573 |
| 2020-03-01 | 3134 |
| 2020-03-08 | 3398 |

```
# 클렌징된 뉴스를 불러옴.
df = pd.read_csv(root+'cleansing/cleansing_full.csv')
df['date'] = pd.to_datetime(df['date'])
```

```
# 데이터를 1주 단위로 그룹을 묶고 개수를 셈
wcount = df.groupby(pd.Grouper(key='date', freq='1w')).count()
```

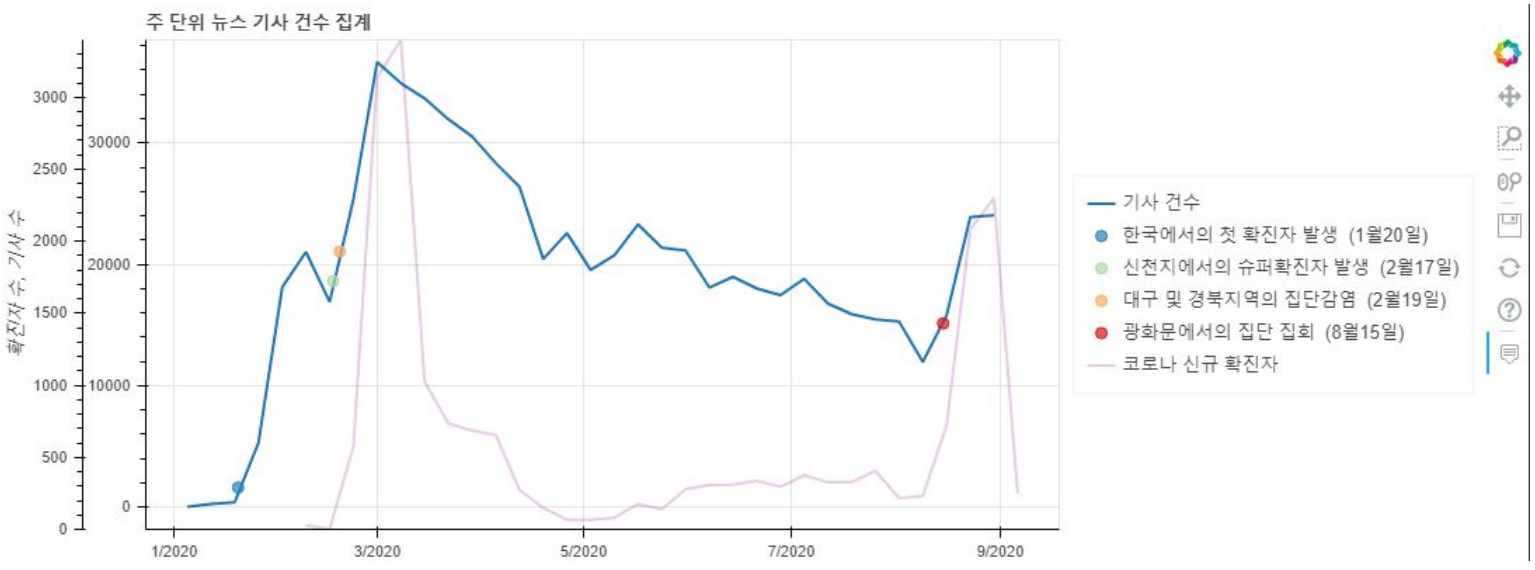
```
# 재정렬
wcount = wcount.reset_index()
```

```
# 코로나 관련하여 큰 사건사고들이 들어갈 날짜를 나열
event = (pd.to_datetime(['2020-01-20', '2020-02-17', '2020-02-19','2020-08-15']),
        [1617,18621, 21061, 15134])
```

```
# plot 생성 옵션 설정
plot = figure(title="주 단위 뉴스 기사 건수 집계", width=1400, height=400,
x_axis_type='datetime', sizing_mode='stretch_width')

# 드래그 방지
plot.toolbar.active_drag = None

# Hover tool 추가
plot.add_tools(HoverTool(
  tooltips = [
    ("date", "$x{%F}"),
    ("count", "$y{D}")
  ],
  formatters={
     '$x': 'datetime'
  })
)

# 뉴스 기사 건수 line plot
p1 = plot.line(x=wcount['date'], y=wcount['title'], line_width=2)

# 코로나 관련 사건 사고들을 표기
p2 = plot.circle(x=event[0][0], y=event[1][0], size=8, fill_alpha=0.7,
color=Spectral[4][0])
p3 = plot.circle(x=event[0][1], y=event[1][1], size=8, fill_alpha=0.7,
color=Spectral[4][1])
p4 = plot.circle(x=event[0][2], y=event[1][2], size=8, fill_alpha=0.7,
color=Spectral[4][2])
p5 = plot.circle(x=event[0][3], y=event[1][3], size=8, fill_alpha=0.7,
color=Spectral[4][3])

# 코로나 신규 확진자 수 line plot
p6 = plot.line(x = corona_data.index, y = corona_data, y_range_name =
'corona', line_alpha=0.2, line_width=2, color="purple")
```

```
# 그래프에 설명 할 범례 표기
plot.add_layout(
  Legend(items=[
            ('기사 건수',[p1]),
            ('한국에서의 첫 확진자 발생\n (1월20일)',[p2]),
            ('신천지에서의 슈퍼확진자 발생\n (2월17일)',[p3]),
            ('대구 및 경북지역의 집단감염\n (2월19일)', [p4]),
            ('광화문에서의 집단 집회\n (8월15일)', [p5]),
            ('코로나 신규 확진자', [p6])
], location="center"), "right")

# 그래프가 2개로 표현되기에 Y축에 기사건수 및 확진자수 표기
plot.extra_y_ranges = {"corona": Range1d(start=0,
end=max(corona_data.values))}
plot.add_layout(LinearAxis(y_range_name="corona", axis_label="확진자 수,
기사 수"), 'left')

show(plot)
```

국가 통계 포털에서 제공하는 값은 2020-02-04 이후의 확진자 정보만 제공되어 확인할 수 있었음. 2~3월 사이 신천지 대구교회 관련 집단 감염 사건 당시 뉴스 기사 수도 급증하는 것을 볼 수 있음. 이후에는 확진자가 감소세를 보이다가 8월 수도권 중심의 2차 대유행 시기에 확진자가 급증하면서 뉴스의 건수도 급증하는 것을 볼 수 있음. 결국 코로나 신규 확진자 수가 증가하면 기사의 수도 증가하며, 관심 또한 높아진다고 볼 수 있음.



## II. 검색엔진 트렌드 분석

**목표**

구글의 'trends' 과 네이버 'DataLab' 의 검색량을 수집하여 변화 추이를 코로나 사태 이전부터 최근까지의 검색량 변화를 시계열 방식으로 시각화하는 시나리오

**기능**

검색량은 사람들의 관심의 지표라고 생각하여 두 개의 검색엔진을 통하여 코로나 이전과 이후의 검색량을 통하여 사회적 거리두기 등 지침을 통한 통제가 이루어졌을 때 사람들의 관심사 및 변화된 생활에 대하여 파악할 수 있음.
데이터는 주 단위이고, 검색어 별로 검색량의 절대적인 차이가 있기 때문에 각 키워드의 검색량은 상대적 수치임.
키워드는 "국내여행", "해외여행", "온라인 쇼핑", "OTT 서비스", "홈짐" 총 5가지이고이 분석 또한 코로나 신규 확진자 수와 비교 하였음.

```
# 시각화 툴 및 데이터를 불러오기위한 선언
from bokeh.plotting import output_notebook, figure, show
from bokeh.palettes import viridis, Spectral, Paired
from bokeh.models import ColumnDataSource, DatetimeTickFormatter,
HoverTool, Legend, Range1d, LinearAxis
import datetime
import pandas as pd


output_notebook()
```

```
# 네이버의 "Datalab"의 검색량 데이터 로드

naver = pd.DataFrame(pd.read_excel(root + 'naver.xlsx'))
naver = naver.set_index(['date'])
naver.head()
```

| date | abroad | domestic | shopping | ott | health |
|---|---|---|---|---|---|
| 2019-10-07 | 36.31309 | 49.02165 | 39.97107 | 29.13172 | 36.78625 |
| 2019-10-14 | 39.51153 | 52.79557 | 42.60166 | 27.71416 | 38.08508 |
| 2019-10-21 | 42.69417 | 55.52359 | 42.31487 | 30.19902 | 39.23932 |
| 2019-10-28 | 42.30951 | 56.81352 | 43.33785 | 29.84043 | 36.86712 |
| 2019-11-04 | 42.88913 | 58.16882 | 45.41473 | 31.10288 | 36.68578 |

```
# 구글의 "Trends"의 검색량 데이터 로드

google = pd.DataFrame(pd.read_excel(root + 'google.xlsx'))
google = google.set_index(['date'])
google.head()
```

| date | ott | domestic | abroad | shopping | health |
|---|---|---|---|---|---|
| 2019-10-06 | 33 | 43 | 71 | 51 | 16 |
| 2019-10-13 | 33 | 26 | 66 | 51 | 27 |
| 2019-10-20 | 35 | 42 | 52 | 53 | 43 |
| 2019-10-27 | 32 | 87 | 67 | 53 | 11 |
| 2019-11-03 | 36 | 43 | 75 | 54 | 22 |

```
# 국가 통계 포털의 일 확진자 데이터 로드
corona_data = pd.read_csv(root + 'corona_count_modify.csv')
corona_data['date'] = pd.to_datetime(corona_data['date'])
```

```
# 국가 통계 포털의 일 확진자 데이터 확인
corona_data.head()
```

| | date | 신규 확진 | 누적 확진 | 격리 증감 | 누적 격리 | 신규 격리해제 | 누적 격리 해제 | 신규 사망 | 누적 사망 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2020-02-04 | 18 | 18 | - | - | 0 | 0 | 0 | 0 |
| 1 | 2020-02-05 | 1 | 19 | - | - | 1 | 1 | 0 | 0 |
| 2 | 2020-02-06 | 4 | 23 | - | - | 0 | 1 | 0 | 0 |
| 3 | 2020-02-07 | 1 | 24 | - | - | 1 | 2 | 0 | 0 |
| 4 | 2020-02-08 | 0 | 24 | - | - | 0 | 2 | 0 | 0 |

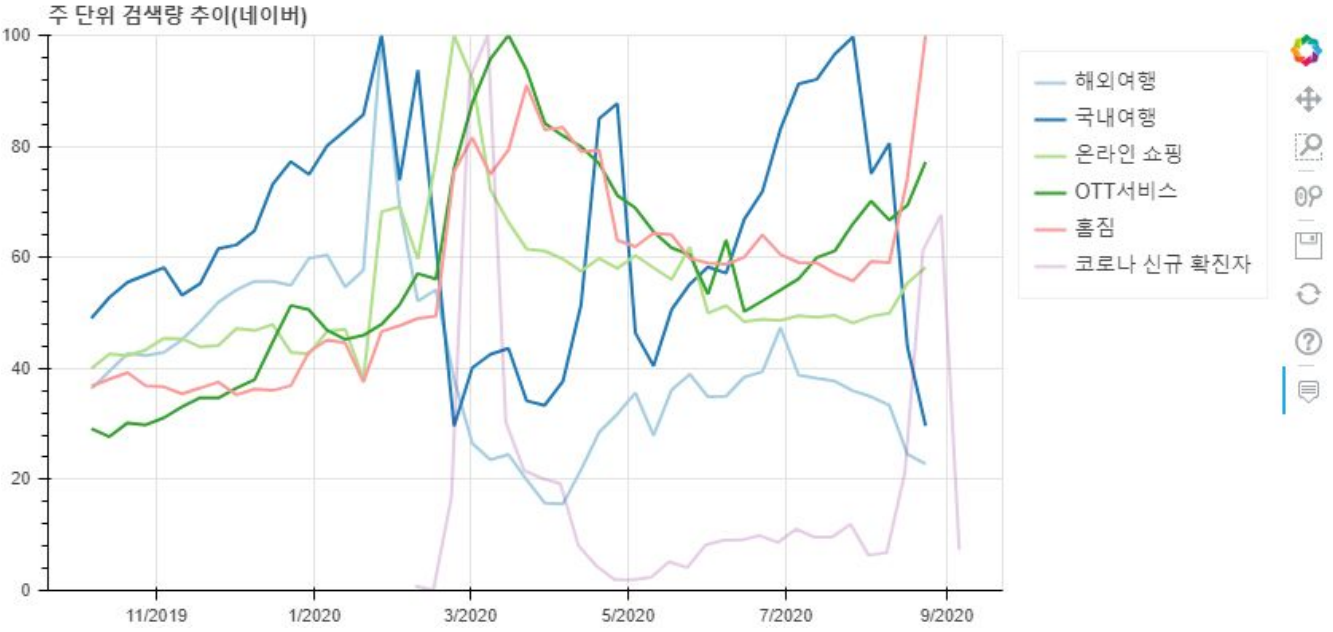
```
# 일일 확진자 데이터를 1주일(1week)단위로 합치고, 데이터 확인
corona_data = corona_data.groupby(pd.Grouper(key='date', freq='1w'))
corona_data = corona_data.sum()['신규 확진']

pd.DataFrame(corona_data).head()
```

| date | 신규 확진 |
|---|---|
| 2020-02-09 | 27 |
| 2020-02-16 | 2 |
| 2020-02-23 | 573 |
| 2020-03-01 | 3134 |
| 2020-03-08 | 3398 |

```
# plot 설정
plot = figure(title="주 단위 검색량 추이(네이버)", width=1400, height=400,
x_axis_type='datetime', sizing_mode='stretch_width', y_range =
Range1d(start=0, end=100))
plot.toolbar.active_drag = None

# 마우스 hover 시에 표시할 데이터 설정
plot.add_tools(HoverTool(
  tooltips = [
    ("날짜", "$x{%F}"),
    ("해외여행", "@abroad"),
    ("국내여행", "@domestic"),
    ("온라인 쇼핑", "@shopping"),
    ("OTT서비스", "@ott"),
    ("홈짐", "@health"),
  ],
  formatters={
     '$x': 'datetime'
  },
  )
)

# 컬러 설정
color = Paired[len(naver.columns)]

# 사용할 데이터셋 설정
source = ColumnDataSource(dict(
  x = naver.index,
  abroad = naver['abroad'],
  domestic = naver['domestic'],
  shopping = naver['shopping'],
  ott = naver['ott'],
  health = naver['health'],
))

# 키워드들을 그래프에 표기
p1 = plot.line(x = 'x', y = 'abroad', line_width=2, color = color[0],
source=source)
p2 = plot.line(x = 'x', y = 'domestic', line_width=2, color = color[1],
source=source)
```

```
p3 = plot.line(x = 'x', y = 'shopping', line_width=2, color = color[2],
source=source)
p4 = plot.line(x = 'x', y = 'ott', line_width=2, color = color[3], source=source)
p5 = plot.line(x = 'x', y = 'health', line_width=2, color = color[4], source=source)

# 코로나 신규 확진자 표기
p6 = plot.line(x = corona_data.index, y = corona_data, y_range_name =
'corona', line_alpha=0.2, line_width=2, color="purple")

# 그래프에 설명 할 범례 표기
legend = Legend(items=[
        ('해외여행',[p1]),
        ('국내여행', [p2]),
        ('온라인 쇼핑', [p3]),
        ('OTT서비스', [p4]),
        ('홈짐', [p5]),
        ('코로나 신규 확진자', [p6])
], location="top_left")

# 범례 클릭시 숨기기, 차트 오른쪽에 범례 표시
legend.click_policy = 'hide'
plot.add_layout(legend, "right")
plot.extra_y_ranges = {"corona": Range1d(start=0,
end=max(corona_data.values))}
show(plot)
```

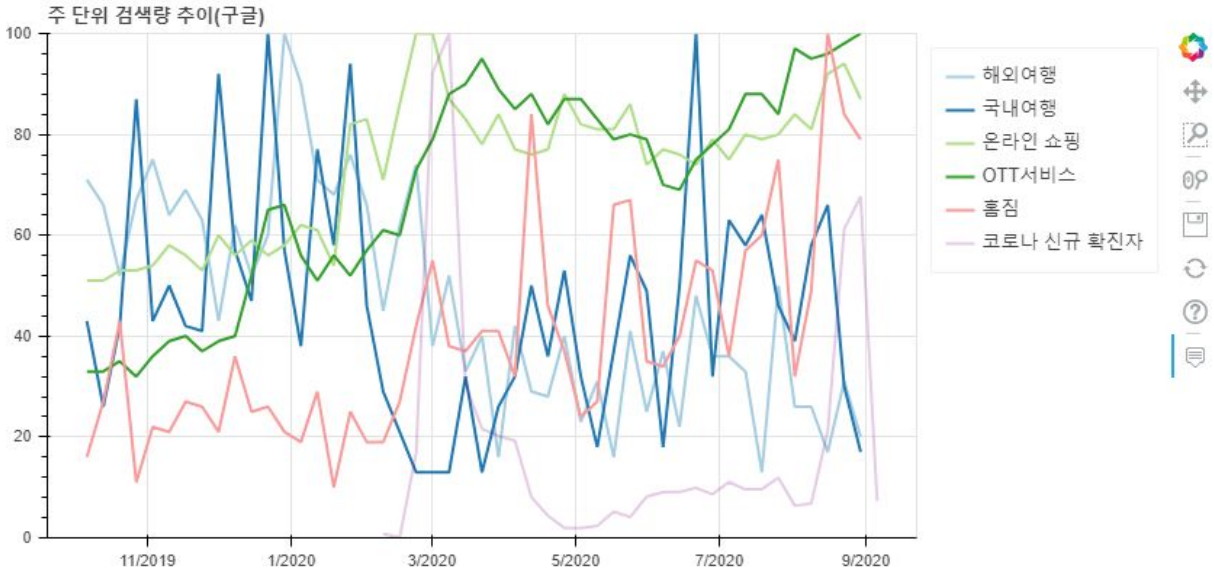
네이버의 트렌드 분석 그래프



```
# 기사의 타이틀 표기
plot = figure(title="주 단위 검색량 추이(구글)", width=1400, height=400,
x_axis_type='datetime', sizing_mode='stretch_width', y_range =
Range1d(start=0, end=100))
plot.toolbar.active_drag = None

# 마우스 hover 시에 표시할 데이터 설정
plot.add_tools(HoverTool(
  tooltips = [
    ("날짜", "$x{%F}"),
    ("국내여행", "@domestic"),
    ("해외여행", "@abroad"),
    ("온라인 쇼핑", "@shopping"),
    ("OTT서비스", "@ott"),
    ("홈짐", "@health"),
  ],
  formatters={
     '$x': 'datetime'
  },
  )
)

# 컬러 설정
color = Paired[len(google.columns)]

# 데이터셋 설정
source = ColumnDataSource(dict(
  x = google.index,
  abroad = google['abroad'],
  domestic = google['domestic'],
  shopping = google['shopping'],
  ott = google['ott'],
  health = google['health'],
))

# 키워드들을 그래프에 표기
p1 = plot.line(x = 'x', y = 'abroad', line_width=2, color = color[0],
source=source)
p2 = plot.line(x = 'x', y = 'domestic', line_width=2, color = color[1],
source=source)
```

```
p3 = plot.line(x = 'x', y = 'shopping', line_width=2, color = color[2],
source=source)
p4 = plot.line(x = 'x', y = 'ott', line_width=2, color = color[3], source=source)
p5 = plot.line(x = 'x', y = 'health', line_width=2, color = color[4], source=source)

# 코로나 신규 확진자 표기
p6 = plot.line(x = corona_data.index, y = corona_data, y_range_name =
'corona', line_alpha=0.2, line_width=2, color="purple")

# 그래프에 설명 할 범례 표기
legend = Legend(items=[
        ('해외여행',[p1]),
        ('국내여행', [p2]),
        ('온라인 쇼핑', [p3]),
        ('OTT서비스', [p4]),
        ('홈짐', [p5]),
        ('코로나 신규 확진자', [p6])
], location="top_left")

legend.click_policy = 'hide'
plot.add_layout(legend, "right")
plot.extra_y_ranges = {"corona": Range1d(start=0,
end=max(corona_data.values))}

show(plot)
```

구글 트렌드 분석 그래프



네이버 및 구글의 트렌드 검색량을 분석했고 우리가 도출한 결과는 위와 같음.
구글은 네이버와 다르게 그래프가 요동치는것을 볼 수 있는데 이 이유는 찾지 못하였음.

전체적으로 여행 관련 검색어(해외여행, 국내여행)는 코로나 이전은 지속적으로 증가하는 경향을 보였고, 검색량의 최고점을 갱신한 1월 26일은 대한의사협회에서 중국 발 입국자들의 입국금지 조치 및 필요성을 권고한 날임. 그 영향을 받아서 인지 사람들의 관심도가 높아졌고, 이후에는 1월 말쯤부터 관심도가 급감하는 모습을 볼 수 있음.
4월 말 경 신규 확진자가 10명대로 많이 줄어든 모습을 보여주었음. 이로 인하여 여행을 가고 싶던 사람들은 해외여행을 갈 수 없으니 국내여행으로 눈을 많이 돌렸다고 판단됨. 또한 4월 30일부터 부처님 오신 날을 시작으로 긴 연휴가 있었고, 이에 대한 기대감 때문인지 국내여행에 대한 검색량은 증가하였고, 해외여행 또한 소폭 상승한 것을 볼 수 있음.
하지만 국내여행의 검색량이 증가하고 기간이 조금 지난 뒤 일일 코로나 신규 확진자 수는 조금씩 증가하였음.
또한 온라인쇼핑의 경우에는 코로나 범유행으로 인하여 비대면으로 물건들을 구입하는 문화와 마스크 및 생필품을 구비하려는 것으로 유추해 볼 수 있음.
OTT 서비스와 홈짐의 경우 사회적 거리두기의 영향으로 외부에서 주로 즐겼던 영화 및 문화활동을 집에서 즐기는 OTT 서비스에 대한 관심도가 증가하였고 헬스장 및 기타 체육시설들 또한 사회적 거리 두기로 인해 운영이 중지되거나 제한하는 경우가 있어 집에서 간편히 운동을 하려는 사람들 또한 많아졌다고 판단됨.

## III. LDA 토픽 모델링

뉴스 기사를 어떤 토픽으로 나누어야 할지, 어느정도의 세분류가 필요할지 가늠을 하지 못하고 라벨링이 된 정형 데이터가 아니므로 비지도 학습인 LDA 토픽 모델링을 선택하였음.

잠재 디리클레 할당(Latent Dirichlet Allocation, 이하 LDA)는 토픽 모델링에서 가장 대표적으로 사용되는 알고리즘. LDA를 이용하여 문서의 집합에 어떠한 토픽들이 존재하는지 유추가 가능함. LDA에서 문서들은 토픽들의 혼합으로 구성되어 있고, 토픽들은 확률 분포에 기반하여 단어들을 생성한다고 가정함. LDA를 활용하여 각 문서(뉴스 기사)들이 어떠한 토픽을 가지고 있는지 확인할 수 있음.

간단한 LDA 토픽 모델링의 과정을 말하자면, 토픽의 개수가 하이퍼파라미터이고, 문서 집합 내에서 몇 개의 토픽이 존재할지를 가정하면, 토픽의 개수에 맞추어 문서 분류를 수행함. 문서에 사용할 토픽의 혼합을 확률 분포에 기반하여 결정됨. 문서에 사용할 각 단어들을 정하고 토픽 분포에서 토픽 T를 확률적으로 선택함. 선택한 토픽 T에서 단어의 출현 확률 분포에 기반해 문서에 사용할 단어가 선택됨.

사용자는 알고리즘에게 토픽의 개수를 알려주고, 모든 단어들을 토픽의 개수 k개 중 하나의 토픽에 할당됨. 모든 문서의 모든 단어에 대해 k개 중 하나의 토픽으로 랜덤 부여함. 작업이 끝나면 문서는 토픽을 가지며, 토픽은 단어 분포를 가지는 상태임. 처음에는 랜덤으로 부여되는 것이지만, 모드 문서의 모든 단어에 대해 분배를 진행하고나면 목표했던 상태로 재 할당됨.

수행할 때에 각각의 문서가 어떤 토픽인지 라벨링을 해줄 필요는 없으나 토픽의 개수를 지정하여 분류를 수행함. 토픽의 개수도 적당한 개수를 알 수 없기 때문에 토픽의 개수가 2-80개인 모델을 생성한 후 Coherence Model을 이용하여 해당 모델의 coherecne, 일관성을 측정하였음.

결과를 평가하는 기준은 Coherence와 Perplexity 두가지임.

Perplexity는 생성된 확률 모델이 실제의 확률과 얼마나 유사한지를 측정함. LDA 또한 확률 기반의 모델임으로 Perplexity를 활용하여 오랫동안 측정되어 왔음.
깁스 샘플링 과정에서 반복 횟수가 증가할수록 Perplexity는 감소하는 경향을 보여주는데 어느 정도가 지나면 Perlexity는 증가와 감소를 반복하며 요동치는 모습을 보여줌. Perlexity의 값은 작으면 작을수록 해당 토픽이 실제 문헌 결과를 잘 반영한다고 판단함. 하여 학습이 잘 되었다고 평가할 수는 있지만, 사람이 보기에 해석하기 좋다고 볼 수는 없음.

Topic Coherence은 먼저, 결과로 나온 주제들에 대해 각각의 주제에서 상위 N개의 단어를 뽑음. 모델링이 잘 되어있을 수록 의미론적으로 유사한 단어가 많이 모여 있을 테니 이러한 유사도를 측정함.

**목표**

토픽 모델링중 하나인 LDA를 사용해 파싱된 문서의 토픽(주제)을 확인하고 해당 토픽에 속해있는 문서들을 확인하고 시각화하는 시나리오.

**기능**

우리가 분석하려는 생활, 문화 관련 토픽들을 확인하고 선택하여 이후의 시나리오에 사용하기 위함. 또한 특정 군집이 어떤 토픽으로 이루어져있는지 등 사용자가 직접 볼 수 있는 기능을 제공

**사용된 라이브러리**

gensim 라이브러리를 사용하여 LDA모델 생성, Coherence값 측정
pyLDAvis 라이브러리를 사용하여 LDA 모델을 시각화

```
# 시각화패키지인 pyLDAvis 설치
!pip install -q pyLDAvis

import os
import numpy as np
import gensim
import pickle
from gensim.models import TfidfModel
import pyLDAvis
from gensim.corpora import Dictionary
from gensim.models.ldamodel import LdaModel
from gensim.models.coherencemodel import CoherenceModel
from gensim.models.ldamulticore import LdaMulticore

from bokeh.plotting import output_notebook, figure, show
from bokeh.palettes import Spectral
```

```
# 폴더에 있는 파일 목록을 가져오는 메서드
def get_filelist(path, extension):
 flist = os.listdir(path)
 result = []
 if path.endswith('/') == False:
   path += '/'

 for file in flist:
   if file.endswith(f'.{extension}'):
```

```
    result.append(path+file)
 return sorted(result)


class TokenCorpus():
 def __init__(self, flist):
  self.flist = flist
 def __iter__(self):
  for file in self.flist:
   with open(file, 'rb') as f:
    for el in pickle.load(f):
     yield el

def fake_tokenizer(e):
 return e
```

아래는 추출한 토큰 (명사들로 이루어진)들을 이용하여 LdaModel을 생성함.

```
# 토큰화된 파일을 불러오는 메서드
flist = get_filelist(root + 'tokenized', 'pkl')

#명사들을 이용하여 Dictionary 생성
corpus = TokenCorpus(flist)

# Dictionary 객체와 doc2bow()메서드를 활용하여 BoW 데이터를 생성
dic = Dictionary(corpus)
bow = [dic.doc2bow(doc) for doc in corpus]
```

`BoW` 방식으로 데이터를 생성하여 LDA 모델링을 수행함.
이때 `gensim` 에서 제공하는 `LdaMulticore` 클래스를 활용하여 멀티 코어로 조금 더 빠른 속도로 작업을 수행하였음.

```
# dictionary, bow 일부 출력
print(f'dic: {dic}')
print(f'bow: {bow[:3]}')
```

**OUT:**

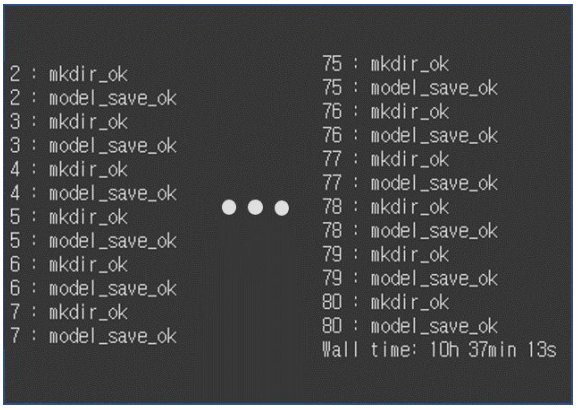
```
dic: Dictionary(187974 unique tokens: ['DNA', 'GDP', '가르침', '가슴', '가의']...)
bow: [[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1), (11, 1), (12,
1), (13, 1), (14, 2), (15, 1), (16, 1), (17, 1), (18, 1), (19, 1), (20, 1), (21, 1), (22, 4), (23, 1), (24,
1), (25, 1), (26, 1), (27, 1), (28, 1), (29, 1), (30, 1), (31, 2), (32, 1), (33, 1), (34, 3), (35, 5), (36,
3), (37, 2), (38, 1), (39, 1), (40, 1), (41, 2), (42, 1), (43, 1), (44, 1), (45, 1), (46, 1), (47, 1), (48,
1), (49, 2), (50, 2), (51, 2), (52, 1), (53, 1), (54, 2), (55, 2), (56, 1), (57, 2), (58, 1), (59, 1), (60,
1), (61, 1), (62, 1), (63, 2), (64, 4), (65, 1), (66, 1), (67, 1), (68, 1), (69, 2), (70, 1), (71, 6), (72,
2), (73, 3), (74, 1), (75, 3), (76, 1), (77, 3), (78, 2), (79, 7), (80, 1), (81, 1), (82, 6), (83, 2), (84,
1), (85, 1), (86, 1), (87, 1), (88, 1), (89, 1), (90, 6), (91, 1), (92, 1), (93, 1), (94, 1), (95, 1), (96,
2), (97, 1), (98, 1), (99, 1), (100, 1), (101, 1), (102, 1), (103, 1), (104, 1), (105, 1), (106, 8),
(107, 2), (108, 1), (109, 1), (110, 1), (111, 3), (112, 6), (113, 2), (114, 1), (115, 1), (116, 3),
(117, 1), (118, 1), (119, 1), (120, 3), (121, 1), (122, 1), (123, 1), (124, 1), (125, 1), (126, 1),
(127, 1), (128, 2), (129, 1), (130, 1), (131, 5), (132, 2), (133, 1), (134, 1), (135, 1), (136, 2),
(137, 1), (138, 1), (139, 1), (140, 1), (141, 1), (142, 1), (143, 1), (144, 1), (145, 1), (146, 1),
(147, 1), ...
```

조금 더 정확한 분류를 위해 적절한 토픽의 개수로 분류 되어야 함.
일관성(Coherence)를 측정하여 적절한 개수로 분류될 수 있도록 함.

Coherence 측정을 위해 모델을 먼저 생성.

```
# 현재 셀 이 완료되기까지의 시간을 표기
%%time

# 토픽의 갯수가 2~80개까지 모델을 생성
for num_topic in range(2, 81):
  model = LdaMulticore(bow, id2word=dic, chunksize=50000,
num_topics=num_topic, passes=5, workers=5)
  os.mkdir(f'{root}lda/lda{num_topic}')
  print(num_topic,':', 'mkdir_ok')
  model.save(f'{root}lda/lda{num_topic}/lda_{num_topic}.model')
  print(num_topic,':', 'model_save_ok')
```

2~80까지의 LDA모델을 저장.




2 - 80개의 모델을 모두 생성한 이후, 생성된 모델의 Coherence를 측정.
측정 또한 `gensim` 에서 제공하는 `CoherenceModel`을 사용

```
coherences = []

# 현재 셀 이 완료되기까지의 시간을 표기
%%time
# 생성된 모델을 통해 Coherence값을 측정(c_v방식)
for num_topic in range(2, 81):
  model_ = LdaModel.load
(f'{root}lda/lda{num_topic}/lda_{num_topic}.model')
  cm = CoherenceModel(model=model_, corpus=bow, coherence='c_v', texts =
corpus )
  coherence = cm.get_coherence()
  print(num_topic,": Coherence :",coherence)
  coherences.append(coherence)
```

**OUT:**

```
2 : Coherence : 0.48772722392274304
3 : Coherence : 0.5024692370031295
4 : Coherence : 0.5184390771841996
5 : Coherence : 0.49047271096697126
6 : Coherence : 0.5112172922743616
7 : Coherence : 0.5213304156680164
8 : Coherence : 0.517480594886506
9 : Coherence : 0.5189379027111818
10 : Coherence : 0.5180527972651984
11 : Coherence : 0.5625508468956504
12 : Coherence : 0.5355458991677943
13 : Coherence : 0.5734285472119004
14 : Coherence : 0.543958713899022
```

15 : Coherence : 0.5582416796221631
16 : Coherence : 0.5607023584749189
17 : Coherence : 0.5678667324535196
18 : Coherence : 0.5815529095676452
19 : Coherence : 0.585782700030285
20 : Coherence : 0.5757239224152753
21 : Coherence : 0.5848502237690997
22 : Coherence : 0.5563004303482265
23 : Coherence : 0.5929176505540723
24 : Coherence : 0.5700702733292392
25 : Coherence : 0.6151559248423709
26 : Coherence : 0.5986191886580192
27 : Coherence : 0.5790337774768051
28 : Coherence : 0.5798824318626895
29 : Coherence : 0.597321401794611
30 : Coherence : 0.5897087506927119
31 : Coherence : 0.584177442307314
32 : Coherence : 0.588540910226627
33 : Coherence : 0.5703267974285344
34 : Coherence : 0.6143133182199115
35 : Coherence : 0.5817312412020064
36 : Coherence : 0.5828608156995574
37 : Coherence : 0.5967198668390947
38 : Coherence : 0.6166175882481894
39 : Coherence : 0.5979523504256816
40 : Coherence : 0.5896274658509014
41 : Coherence : 0.6055486348709933
42 : Coherence : 0.5946703595061039
43 : Coherence : 0.5977955916086776
44 : Coherence : 0.6151154467938915
45 : Coherence : 0.597864385351971
46 : Coherence : 0.6042404534346403
47 : Coherence : 0.5849343834281301
48 : Coherence : 0.5849533737036006
49 : Coherence : 0.5929683469987458
50 : Coherence : 0.6026533846926153
51 : Coherence : 0.6108805025133145
52 : Coherence : 0.5887241010344252
53 : Coherence : 0.6022412311435085
54 : Coherence : 0.5997074562894262
55 : Coherence : 0.5856374160907954

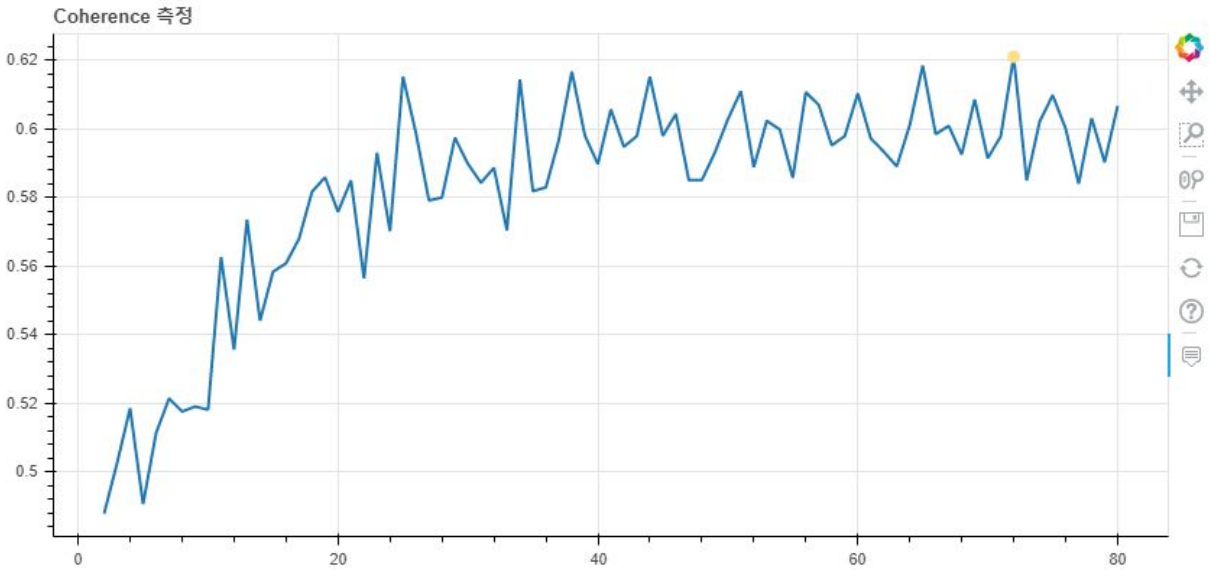
```
56 : Coherence : 0.6105731269252487
57 : Coherence : 0.6068501639398536
58 : Coherence : 0.5950757302951051
59 : Coherence : 0.597807829342405
60 : Coherence : 0.6102333416121613
61 : Coherence : 0.5971407000767899
62 : Coherence : 0.593250126401489
63 : Coherence : 0.5889637440564149
64 : Coherence : 0.6009754500832518
65 : Coherence : 0.6183164721151004
66 : Coherence : 0.5983560042727993
67 : Coherence : 0.6007957624635888
68 : Coherence : 0.5924335646722968
69 : Coherence : 0.6084779050519726
70 : Coherence : 0.5913562165470134
71 : Coherence : 0.5976876451367187
72 : Coherence : 0.6209473307101653
73 : Coherence : 0.584870529162155
74 : Coherence : 0.6020356617182847
75 : Coherence : 0.6097080934143727
76 : Coherence : 0.5999550917435844
77 : Coherence : 0.583917935421626
78 : Coherence : 0.6030155664924468
79 : Coherence : 0.5901761128708746
80 : Coherence : 0.6065975133160937
Wall time: 14h 29min 56s
```

```
output_notebook()

tooltips = [
        ("x, y", "$x{D}, $y")
]

# 그래프 제목 표기
plot = figure(title="Coherence 측정", width=1400, height=400,
sizing_mode='stretch_width',  tooltips=tooltips)
plot.toolbar.active_drag = None

# 2~80번까지 모델의 Coherence값을 그래프로 표현
plot.line(x = range(2, 81), y = coherences, line_width=2)
```

```
highest = sorted(coherences, reverse=True)[0]

#가장 높은 숫자 표시
plot.circle(x = coherences.index(highest) + 2, y = highest, size = 8, color = "#fee08b")

show(plot)
```

Coherence 값은 0 - 1 사이이며, Coherence 값이 낮아도 좋지 않지만, 값이 너무 높은것도 정상적이지 않을 수 있음.

위 결과를 토대로 가장 응집도가 높은 모델을 채택하였음.
그래프를 확인해보면, 토픽의 개수가 72(Coherence: 0.62)일때 가장 높은 것을 확인할 수 있음.




추가로, 기사 내용을 직접 확인해서 확인한 결과 가장 적절하다고 판단되는 72개의 토픽을 가진 모델을 사용함.
아래는 `pyLDAvis` 패키지를 활용하여 LDA 모델을 시각화 한 결과임.

토픽의 번호는 분류를 수행할 때 임의로 부여되는 번호이며, 이것이 의미를 지니지는 않음. 토픽에 번호는 모델 생성 시마다 달라질 수 있음.

분류 시에 문서에 사용할 단어 수를 결정함. 이것은 `gensim` 에서 제공하는 `Dictionary` 클래스를 활용하여 단어 뭉치들로 Coherence를 측정하였음.

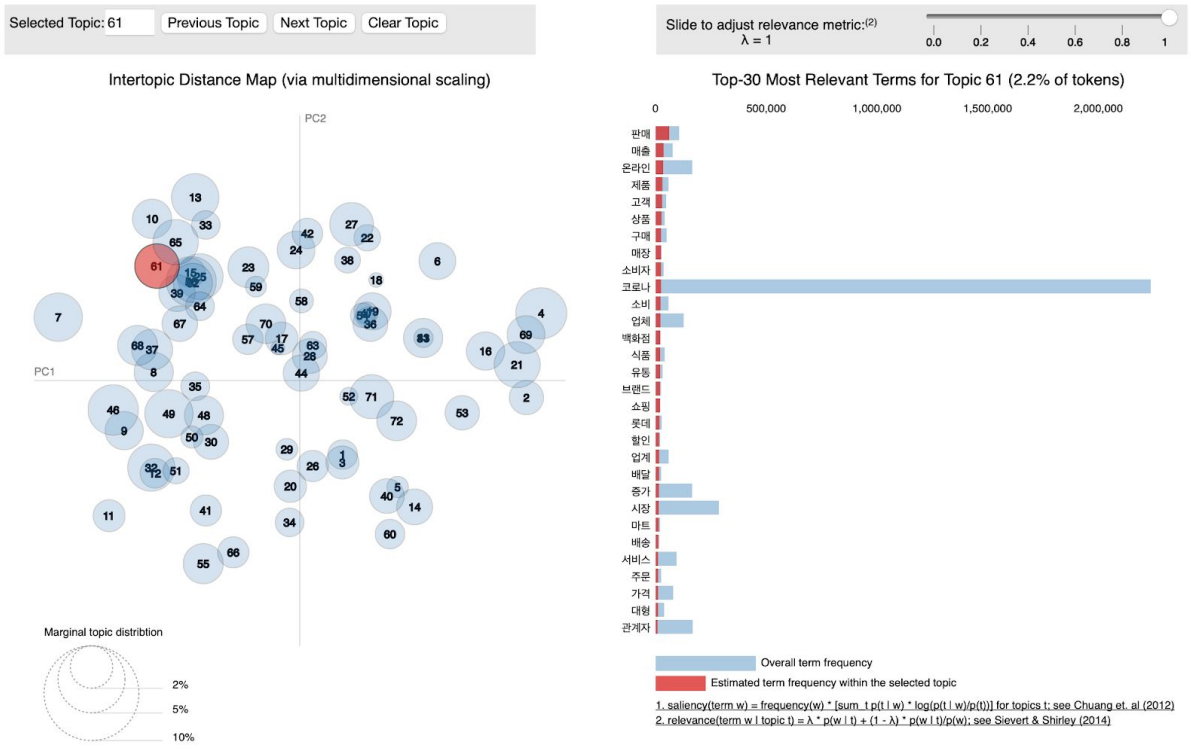
```
# 72개의 토픽을 가진 모델을 불러옴
model_72 = gensim.models.LdaModel.load (f'{root}lda/lda72/lda_72.model')

# pyLDAvis를 통해 시각화 수행
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(model_72, bow, dic)
pyLDAvis.display(vis)
```

***pyLDAvis*** 패키지를 활용하여 시각화 한 결과는 아래와 같음.
1 - 72 까지 72개의 토픽이 존재하고, 각 토픽은 토픽 간의 관련성을 기준으로 위치함.

시각화된 결과에서는 2차원으로 축소된 토픽의 벡터화 결과와, 각 토픽들의 키워드 두 가지 정보를 확인할 수 있음.

LDA 모델에서 토픽의 단어 개수의 차원을 가지고 있고, ***PyLDAvis***에서는 이것을 ***PCA***를 이용하여 2차원으로 축소한 후 시각화함. PCA는 토픽 간에 다른 패턴으로 등장하는 단어들을 중심으로 2차원 지도의 좌표를 학습하는 역할을 함.

오른쪽의 슬라이드를 이용하여 상관 관계를 조절할 수 있음. 슬라이드가 1에 가까울수록 해당 자주 출현한 단어, 0에 가까울수록 해당 토픽에서만 존재하는 단어를 의미함.




파일 다운로드 링크(다운로드 이후 확인해주세요) :
https://drive.google.com/file/d/1rq-hRYp2CY5qwR1XHgIVXb2opFqT16ss/view?usp=sharing

## IV. LDA 토픽 모델을 사용한 토픽 분류 및 시각화

LDA 토픽 모델링을 통하여 토픽의 개수를 2~80개로 테스트하여, LDAModel 토픽의 개수를 정하였음.

앞서 토픽 개수가 72개 일때 Coherence(일관성)스코어가 가장 높게나와 채택하였지만, 조금 더 정확하게 분석하기 위해 각 문서에서 가장 비중이 높은 토픽들을 기준삼아 72가지 문서를 분류 후 각 토픽을 확인하였음.

```
from gensim.models.ldamodel import LdaModel
from gensim.corpora import Dictionary
import pandas as pd
import numpy as np
import pickle
```

```
# LDA72번의 토픽을 확인(토픽별)
def format_topics_sentences(ldamodel, corpus, texts):
  df = pd.DataFrame()
# ldamodel[corpus] : lda_model에 corpus를 넣어 각 토픽당 비중 계산
  for i, row in enumerate(ldamodel[corpus]):
     row = sorted(row, key=lambda x: (x[1]), reverse=True)
     if row is not None:
       topic_num, prop_topic = row[0]
       df = df.append(pd.Series([int(topic_num), round(prop_topic,4)]),
ignore_index=True)
# Domonant_Topic  = 구성된 토픽
# Perc_Contribution = 문서에서 토픽의 비중
  df.columns = ['Dominant_Topic', 'Perc_Contribution']
  return(df)
```

```
# 폴더에 있는 파일 목록을 가져오는 메서드
def get_filelist(path, extension):
 flist = os.listdir(path)
 result = []
 if path.endswith('/') == False:
   path += '/'
```

```
 for file in flist:
   if file.endswith(f'.{extension}'):
     result.append(path+file)
 return sorted(result)


class TokenCorpus():
 def __init__(self, flist):
   self.flist = flist

 def __iter__(self):
   for file in self.flist:

     with open(file, 'rb') as f:
       for el in pickle.load(f):
         yield el

def fake_tokenizer(e):
 return e
```

```
# 토큰화된 파일 목록을 가져오는 메서드
flist = get_filelist(root + 'tokenized', 'pkl')

#명사들을 이용하여 Dictionary 생성
corpus = TokenCorpus(flist)

# Dictionary 객체와 doc2bow()메서드를 활용하여 BoW 데이터를 생성
dic = Dictionary(corpus)
bow = [dic.doc2bow(doc) for doc in corpus]
```

```
# LdaModeling을 통해 선정한 72번 모델을 가져옴
model = LdaModel.load(root + 'lda/lda72/lda_72.model')
```

```
# 토픽 분류 수행
df = format_topics_sentences(ldamodel=model, corpus=bow, texts=corpus)
```

```
# column 설정
df = df.reset_index()
df.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib']
```

```
# 위 결과를 Dominant_Topic_test.csv 로 저장.
df.to_csv(root + 'Dominant_Topic_test.csv', index=False)
```

```
df = pd.read_csv(root + 'Dominant_Topic_test.csv')
```

```
# { 토픽: [문서번호] } 로 딕셔너리 생성
topic_dict = dict()

for row in df.iterrows():
 v = int(row[1]['Dominant_Topic'])
 i = int(row[0])
 if  v in topic_dict:
   topic_dict[v].append(i)
 else:
   topic_dict[v] = [i]
```

```
# lda72 폴더에 분류된 파일 저장

os.makedirs(root + 'lda72', exist_ok=True)
os.chdir(root + 'lda72')

# 생성한 딕셔너리를 기준으로 분류 수행
# 뉴스 기사를 토픽별로 csv 파일에 저장함
for key in topic_dict.keys():
 tmp = pd.DataFrame()
 for i, rows in enumerate(pd.read_csv(root + 'cleansing/cleansing_full.csv',
chunksize=50000, iterator=True)):
   tmp = pd.concat([tmp, rows[rows.index.isin(topic_dict[key])]  ])
 tmp.to_csv("%02d.csv" % key)
```

```
sorted(os.listdir(root + 'lda72'))[:10]
```

**OUT:**

```
['00.csv',
 '01.csv',
 '02.csv',
 '03.csv',
 '04.csv',
 '05.csv',
 '06.csv',
 '07.csv',
 '08.csv',
 '09.csv']
```

72개의 토픽으로 토픽 모델링을 수행했을 때  각 모델의 대표적인 키워드 확인

```
# 각 토픽별 키워드
# 딕셔너리와 model으로 불러오기
corpus = TokenCorpus(flist)
dic = Dictionary(corpus)

model = LdaModel.load(root + 'lda/lda72/lda_72.model')

# 각 토픽 대표 단어를 출력
for i in range(72):
 print(f"{i}:\t{[dic[t[0]] for t in model.get_topic_terms(i, topn=20)]}")
```

**OUT:**

```
0:      ['일본', '도쿄', '코로나', '긴급', '사태', '정부', '감염', '신문', '크루즈', '감염자',
'NHK', '선언', '보도', '확인', '신종', '프린세스', '바이러스', '감염증', '다이아몬드',
'이날']
1:      ['환자', '확진', '치료', '코로나', '경남', '완치', '퇴원', '입원', '판정', '병원', '격리',
'추가', '사망자', '남성', '이날', '국내', '발생', '여성', '바이러스', '신종']
2:      ['코로나', '환자', '질환', '교수', '병원', '감염', '건강', '사망', '치료', '이상', '경우',
'증상', '결과', '위험', '어린이', '중증', '의학', '대남', '필요', '의료']
3:      ['확진', '판정', '코로나', '검사', '방문', '오후', '조사', '거주', '접촉자', '남성',
'동선', '여성', '경기도', '이날', '보건소', '격리', '오전', '양성', '감염증', '신종']
4:      ['홍콩', '시장', '중국', '베이징', '코로나', '무증상', '감염자', '교도소', '확진',
'모닝', '당국', '발생', '보안법', '사우스', '본토', '차이나', '포스트', '국가', '위생', '건강']
5:      ['직원', '센터', '코로나', '근무', '서울', '폐쇄', '물류', '확진', '재택근무',
'바이러스', '감염증', '신종', '선별', '쿠팡', '진료소', '업무', '방역', '오전', '건물', '출근']
6:      ['기업', '사업', '산업', '투자', '그룹', '회장', '기술', '경영', 'SK', '혁신', '대표',
'글로벌', '회사', '부회장', '분야', '삼성', '개발', '전략', '성장', '미래']
7:      ['금융', '시장', '투자', '부동산', '금리', '자산', '주식', '가격', '상승', '주택', '펀드',
```

'거래', '규제', '채권', '은행', '자금', '기준', '투자자', '매입', '서울']
8: ['사람', '세계', '한국', '시대', '동물', '리치', '작가', '인간', '게임', '변화', '코로나', '사회', '반려', '시작', '영화', '이야기', '자신', '세대', '역사', '작품']
9: ['행사', '문화', '진행', '축제', '개최', '관광', '전시', '코로나', '온라인', '올해', '체험', '예정', '프로그램', '참여', '선정', '이번', '박물관', '공원', '주제', '국립']
10: ['북한', '미국', '장관', '합의', '협상', '무역', '관계', '위원장', '갈등', '한국', '제재', '폼페이', '양국', '문제', '국무', '외교', '관련', '대통령', '안보', '국가']
11: ['대통령', '선거', '투표', '후보', '민주당', '의원', '트럼프', '대선', '총선', '정치', '바이든', '공화', '부통령', '유권자', '지지', '국회의원', '상원', '오바마', '연설', '코로나']
12: ['지원', '코로나', '지역', '사회', '기부', '전달', '사업', '극복', '복지', '활동', '참여', '어려움', '센터', '계층', '캠페인', '취약', '장애', '제공', '물품', '아동']
13: ['확진', '코로나', '사망자', '누적', '집계', '이탈리아', '브라질', '하루', '세계', '증가', '전날', '기준', '이날', '봉쇄', '바이러스', '확산', '감염자', '신규', '감염증', '미국']
14: ['공연', '예술', '영화', '문화', '코로나', '관객', '극장', '음악', '무대', '콘서트', '온라인', '배우', '뮤지컬', '예정', '국립', '작품', '서울', '진행', '개봉', '연극']
15: ['검사', '격리', '판정', '음성', '코로나', '진단', '결과', '양성', '확진', '검체', '실시', '진행', '대상', '채취', '확인', '선별', '진료소', '당국', '증상', '해제']
16: ['서울', '서울시', '집회', '광화문', '시민', '코로나', '종로구', '단체', '시장', '금지', '이날', '오후', '시위', '박원순', '오전', '정부', '회견', '행정명령', '확산', '명령']
17: ['경북', '포항', '지역', '경산', '안동', '포항시', '남구', '코로나', '경산시', '바이러스', '감염증', '신종', '청도', '호국', '칠곡군', '의성', '지진', '북구', '대구경북', '경주']
18: ['교회', '신천지', '예배', '신도', '교인', '사랑', '목사', '코로나', '종교', '명단', '참석', '방역', '조사', '관련', '확진', '시설', '확산', '경기도', '미사', '모임']
19: ['영국', '유럽', '프랑스', '코로나', '독일', '러시아', '정부', '이탈리아', '조치', '봉쇄', '스페인', 'EU', '확산', '국가', '국경', '총리', '통신', '런던', '바이러스', '현지']
20: ['확진', '감염', '집단', '발생', '판정', '추가', '관련', '조사', '당국', '접촉', '확인', '방문', '경로', '코로나', '접촉자', '지역', '방역', '요양', '사례', '환자']
21: ['충북', '생활', '충남', '시설', '대전', '센터', '청주', '천안', '지역', '세종', '입소', '공무원', '코로나', '연수원', '교민', '아산', '임시', '주민', '청주시', '격리']
22: ['학교', '학생', '등교', '수업', '개학', '교육청', '학년', '교육', '교사', '교육부', '초등', '고등학교', '원격', '학부모', '유치원', '연기', '온라인', '학습', '교실', '코로나']
23: ['거리', '두기', '사회', '시설', '운영', '생활', '코로나', '단계', '이용', '도서관', '재개', '조치', '지침', '확산', '방역', '실내', '중단', '정부', '체육', '연장']
24: ['지급', '지원', '지원금', '고용', '재난', '긴급', '신청', '소득', '코로나', '가구', '정부', '보험', '대상', '노동자', '근로자', '기준', '급여', '경우', '이상', '생계']
25: ['한국', '코로나', '입국', '국가', '금지', '정부', '베트남', '조치', '싱가포르', '현지', '여행', '인도네시아', '외국인', '이란', '제한', '태국', '한국인', '필리핀', '대만', '이스라엘']
26: ['방역', '코로나', '시설', '소독', '예방', '확산', '점검', '수칙', '안전', '관리', '이용', '실시', '방지', '마스크', '출입', '착용', '설치', '방문', '감염', '운영']

27: ['코로나', '확산', '취소', '신종', '연기', '바이러스', '단계', '감염증', '결정', '예정', '상황', '우려', '중단', '연휴', '사태', '경보', '행사', '일정', '여행', '관광객']
28: ['치료제', '코로나', '치료', '환자', '효과', '데시', '비르', '사용', '승인', '클로로퀸', '임상', '결과', '의약품', '하이드', '약물', '임상시험', '투여', '미국', '약품', '식품']
29: ['영상', '여성', '방송', '차별', '사진', '코로나', '보도', '자신', '공개', '뉴스', '기사', '언론', '사람', '페이스북', '유튜브', '사건', 'SNS', '남성', '내용', '논란']
30: ['병원', '의료', '환자', '의료진', '코로나', '치료', '진료', '간호사', '입원', '병상', '의사', '확진', '병동', '음압', '신종', '격리', '이송', '감염증', '부족', '바이러스']
31: ['상황', '문제', '생각', '사람', '정도', '정부', '필요', '국민', '부분', '경우', '말씀', '교수', '얘기', '사회', '이야기', '나라', '중요', '가능', '입장', '지적']
32: ['도시', '공사', '건설', '아파트', '주택', '분양', '사업', '가구', '단지', '개발', '부지', '청약', '예정', '조성', '계획', '입주', '국토', '파크', '규모', '공간']
33: ['미국', '뉴욕', '코로나', '주지사', '뉴욕주', '워싱턴', 'CNN', '타임스', '보도', '미군', '캘리포니아', '사람', '센터', '자택', 'AP', '확산', '미국인', '캘리포니아주', 'CDC', '카운티']
34: ['검찰', '법원', '재판', '소송', '혐의', '사건', '수사', '구속', '청구', '서울', '법무부', '변호사', '이만희', '의혹', '진행', '지법', '기소', '손해', '법정', '결정']
35: ['입국', '인천', '공항', '격리', '외국인', '해외', '귀국', '코로나', '검역', '도착', '국자', '국제공항', '체류', '국내', '터미널', '조치', '출국', '특별', '교민', '중국인']
36: ['수출', '감소', '증가', '산업', '코로나', '대비', '지난해', '전년', '기업', '영향', '생산', '올해', '수입', '이후', '해외', '지난달', '제조업', '무역', '국내', '소비']
37: ['제주', '호텔', '여행', '버스', '제주도', '택시', '가격', '코로나', '리자', '관광', '관광객', '이용', '운행', '승객', '격리', '기사', '도민', '예약', '숙박', '대중교통']
38: ['서비스', '정보', '제보', 'YTN', '제공', '온라인', '시스템', '데이터', '디지털', '활용', '기술', '채널', 'TV', '코로나', '검색', '도입', '이용', '스마트', '라인', '플랫']
39: ['바이러스', '코로나', '신종', '감염증', '감염', '확산', '예방', '연구', '전파', '가능', '사람', '질병', '메르스', '유행', '발생', '감염병', '동물', '사스', '전염병', '결과']
40: ['세계', '코로나', 'WHO', '국제', '보건', '기구', '대응', '국가', '한국', '화상', '사무총장', '회의', '각국', '협력', '강조', '나라', '바이러스', '사회', '유엔', '여수']
41: ['코로나', '농가', '농협', '지역', '농산물', '강원', '강원도', '감염증', '농업', '바이러스', '신종', '공원', '해수욕장', '농촌', '상인', '개장', '일손', '오후', '확산', '장병']
42: ['광주', '전남', '광주시', '지역', '코로나', '북구', '서구', '동구', '목포', '탁구장', '이용섭', '확진', '남도', '나주', '호남', '오전', '확산', '남대', '바이러스', '오후']
43: ['경찰', '조사', '위반', '신고', '정보', '수사', '코로나', '혐의', '관련', '행위', '고발', '사실', '확인', '처벌', '전화', '조치', '허위', '경찰청', '불법', '적발']
44: ['순천', '아프리카', 'CJ', '그룹', '게이츠', '참전', '재단', '소프트', '용사', '태안', '순천시', '보훈', '우주', '파주', '남양주시', '통운', '광양', '창업자', '파주시', '마이크']
45: ['경제', '코로나', '위기', '정부', '전망', '정책', '성장', '경기', '기업', '재정', '일자리', '올해', '고용', '금융', '규모', '회복', '상황', '사태', '충격', '추경']
46: ['클럽', '서울', '영업', '강남구', '주점', '명령', '업소', '유흥', '집합', '음식점', '강남', '휴업', '금지', '식당', '유흥업소', '코로나', '서초구', '구청장', '방문', '서울시']

47: ['백신', '개발', '코로나', '연구', '항체', '진단', '바이러스', '키트', '치료제', '임상', '바이오', '시험', '결과', '면역', '혈장', '물질', '연구소', '기술', '진행', '미국']
48: ['분기', '지수', '코로나', '시장', '기록', '대비', '증시', '하락', '실적', '상승', '영업', '포인트', '전망', '이익', '매출', '주가', '증가', '미국', '감소', '코스피']
49: ['가격', '유가', '원유', '국제', '석유', '수요', '하락', '감산', '공매도', '에너지', '코로나', '사우디', '선물', '배럴', '거래소', '하루', '거래', '정유', '미국', '폭락']
50: ['총리', '아베', '조사', '올림픽', '응답', '코로나', '도쿄', '응답자', '연기', '설문', '여론', '일본', '정부', '결과', '신조', '사태', '개최', '평가', '위원회', '국민']
51: ['인도', '아이', '아동', '부모', '시신', '코로나', '가족', '엄마', '어린이', '아기', '출산', '발견', '육아', '장례식장', '유족', '여성', '사망', '상태', '장례', '뉴델리']
52: ['증상', '환자', '발열', '의심', '확인', '기침', '격리', '상자', '호흡기', '감시', '조사', '감염', '방문', '바이러스', '유증', '경우', '사람', '코로나', '확진', '상태']
53: ['대구', '대구시', '코로나', '시민', '지역', '상담', '시장', '심리', '정신', '감염증', '뉴스', '바이러스', '신종', '스트레스', '권영진', '오전', '대구경북', '시청', '우울', '수성구']
54: ['미국', '트럼프', '대통령', '코로나', '백악관', '도널드', '바이러스', '이날', '경제', '보도', '관련', '연방', '대응', '브리핑', '워싱턴', '감염증', '행정부', '통신', '소장', '파우']
55: ['지원', '정부', '대응', '기관', '회의', '대책', '추진', '계획', '코로나', '방안', '장관', '지역', '관리', '공공', '마련', '확대', '인력', '상황', '체계', '관련']
56: ['코로나', '감사', '마음', '가족', '드라이브', '스루', '응원', '의료진', '메시지', '덕분', '극복', '국민', '시간', '챌린지', '친구', '편지', '아이', '할머니', '아들', '사람']
57: ['부산', '부산시', '선박', '해양', '선원', '항만', '러시아', '기온', '코로나', '해수부', '바다', '해양수산부', '입항', '해운', '미세먼지', '지역', '해운대', '수리', '조선', '해경']
58: ['학원', '대회', '경기', '선수', '코로나', '스포츠', '프로', '훈련', '경주', '축구', '결혼식', '리그', '관중', '야구', '예정', '취소', '연기', '골프', '강사', '경기장']
59: ['우한', '중국', '후베이', '코로나', '바이러스', '호주', '폐렴', '신종', '환자', '발원지', '당국', '봉쇄', '확산', '사람', '감염', '발생', '확진', '플로리다주', '발표', '발병']
60: ['판매', '매출', '온라인', '제품', '고객', '상품', '구매', '매장', '소비자', '코로나', '소비', '업체', '백화점', '식품', '유통', '브랜드', '쇼핑', '롯데', '할인', '업계']
61: ['대학', '시험', '학생', '교육', '온라인', '수업', '코로나', '채용', '강의', '진행', '평가', '대학교', '학교', '취업', '학기', '면접', '교수', '예정', '등록금', '전형']
62: ['주민', '차량', '사고', '지역', '피해', '도로', '마을', '발생', '경찰', '도시', '인근', '현장', '화재', '소방', '자전거', '봉쇄', '교통', '건물', '시간', '주변']
63: ['항공', '운항', '노선', '항공사', '코로나', '중단', '아시아나항공', '국제선', '공항', '대한항공', '항공기', '항공업', '여객', '화물', '업계', '사태', '제주항공', '직원', '이스타항공', '상황']
64: ['지원', '대출', '코로나', '기업', '자금', '은행', '금융', '공인', '소상', '중소기업', '상공', '피해', '임대료', '대상', '신용', '보증', '신청', '감면', '어려움', '지역']
65: ['중국', '코로나', '주석', '정부', '국가', '시진핑', '베이징', '책임', '인민', '사태', '바이러스', '방역', '미국', '주장', '상황', '매체', '대만', '신종', '공산당', '보도']
66: ['공장', '생산', '자동차', '울산', '업체', '가동', '코로나', '현대', 'LG', '현대차',

'삼성전자', '판매', '부품', '중단', '국내', '업계', '기아차', '시장', '공급', '쌍용']
67: ['위원회', '위원장', '의회', '시장', '코로나', '의원', '국회', '회의', '정책', '사회', '위원', '시민', '민주', '오전', '극복', '오후', '대표', '논의', '노조', '이날']
68: ['확진', '서울', '이태원', '코로나', '발생', '클럽', '관련', '지역', '기준', '신규', '경기', '인천', '이날', '추가', '누적', '해외', '전날', '국내', '환자', '방역']
69: ['마스크', '착용', '사용', '사람', '소독', '코로나', '시간', '약국', '공기', '택배', '구매', '경우', '공급', '가능', '에어컨', '필터', '배달', '이상', '감염', '시민']
70: ['방역', '본부', '중앙', '대책', '브리핑', '코로나', '감염', '재난', '안전', '발생', '당국', '정례', '사회', '상황', '정부', '사례', '환자', '총괄', '전파', '집단']
71: ['확진', '신규', '수도', '지역', '코로나', '하루', '발생', '감염', '확산', '증가', '이후', '환자', '최근', '상황', '기록', '사회', '이상', '전국', '누적', '연속']

위에서 확인한 토픽별 주요 키워드와, 분류된 기사 csv 파일을 확인하여 사용할 토픽을 선정하였음.

**15번 토픽**

공연 예술 및 해외 활동 감소에 관한 내용.

**23번 토픽**

초/중/고교 등교 연기에 관한 내용.

**24번 토픽**

사회적 거리 두기에 따른 시설 폐쇄 및 휴관에 관한 내용.

**25번 토픽**

긴급 재난 지원금 지급에 관련된 내용.

**28번 토픽**

졸업식 연기 및 행사 취소 및 소비 촉진 정책 연기와 관련된 내용.

**61번 토픽**

온라인 쇼핑 매출 증가에 관련된 내용

**62번 토픽**

온라인 수업 온라인 강의 등록금 인하 관련 내용.

**64번 토픽**

항공 노선 감축 및 운영 중단 관련 내용.

**65번 토픽**

대출, 기업 / 소상공인 지원, 임대료 관련 내용.

**67번 토픽**

기업의 부품 공급 차질, 공장 중단 관련 내용.

## V. LDA 토픽 모델링을 통해 분류된 키워드 WordCloud

**목표**

LdaModeling에서 선정한 '생활, 문화' 토픽의 문서를 사용하여 WordCloud 라이브러리를 사용하여 가시화 하는 시나리오

**기능**

'생활, 문화' 토픽의 문서의 키워드를 직관적으로 파악할 수 있도록 가시화하는 기능. 많은 양의 문서를 사용하므로 WordCloud를 이용하여 데이터의 특징을 알아보고자 사용

```
# 토큰화를 위한 KoNLPy, Mecab 설치
!set -x \
&& pip install -q konlpy \
&& curl -s
https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh |
bash -x > /dev/null

# 글꼴 설치
!apt-get install -qy fonts-nanum*
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
font_path = '/usr/share/fonts/truetype/nanum/NanumGothic.ttf'
```

```
import numpy as np
import pandas as pd
import re, os
import pickle
import matplotlib.pyplot as plt
from konlpy.tag import Mecab
```

```
from PIL import Image
from collections import Counter
from wordcloud import WordCloud
import matplotlib as mpl
import matplotlib.font_manager as fm

# 폰트 설정
font_path = '/usr/share/fonts/truetype/nanum/NanumGothic.ttf'
font_name = fm.FontProperties(fname=font_path, size=10).get_name()
plt.rc('font', family=font_name)
mpl.rc('font', family=font_name)
```

```
# 사용할 토픽 번호
# topic_use = [15, 23, 24, 25, 28, 61, 62, 64, 65, 67]
topic_use = [14, 22, 23, 24, 27, 60, 61, 63, 64, 66]
```

```
# 토픽 설명 - 범례 및 툴팁에서 사용함
description = [
 '15. 공연 예술 및 해외 활동 감소에 관한 내용.',
 '23. 초/중/고교 등교 연기에 관한 내용.',
 '24. 사회적 거리 두기에 따른 시설 폐쇄 및 휴관에 관한 내용.',
 '25. 긴급 재난 지원금 지급에 관련된 내용.',
 '28. 졸업식 연기 및 행사 취소 및 소비 촉진 정책 연기와 관련된 내용.',
 '61. 온라인 쇼핑 매출 증가에 관련된 내용.',
 '62. 온라인 수업 온라인 강의 등록금 인하 관련 내용.',
 '64. 항공 노선 감축 및 운영 중단 관련 내용.',
 '65. 대출, 기업 / 소상공인 지원, 임대료 관련 내용.',
 '67. 기업의 부품 공급 차질, 공장 중단 관련 내용.'
]
```

```
# 특수문자를 제거하고 단어만 사용하기 위한 정규표현식
r = re.compile(r'[^ ㄱ-ㅣ가-힣|0-9|a-zA-Z]+', flags=re.U | re.M)
mecab = Mecab()

# 공백 제거
def clean_text(text):
  text = text.replace(".", " ").strip()
  text = text.replace("·", " ").strip()
  text = r.sub('', string=text)
  return text

# NNP, NNG, SL만 분리
def get_nouns(sentence):
 return [morphs[0] for morphs in mecab.pos(sentence) if morphs[1] in ['NNP',
'NNG',  'SL'] and len(morphs[0]) > 1]

# tokenizer 실행 함수
def tokenize(df):
  processed_data = []
  # for sent in tqdm(df['text']):
  for sent in df['text']:
    sentence = clean_text(str(sent).replace('\n', '').strip())
    processed_data.append(get_nouns(sentence))
  return processed_data
```

```
# lda로 분류된 결과를 토큰화 하여 아래 폴더에 저장
os.makedirs(root + 'lda72_tokenized_all', exist_ok=True)
os.chdir(root + 'lda72_tokenized_all')
```

```
# 토픽별로 분류했던 뉴스기사 목록을 각각 토큰화 수행.
# for topic in topic_use:
for topic in range(72):
 num = '{:02}'.format(topic)
 rows = pd.read_csv(f'{root}lda72/{num}.csv')
 with open(f'{topic}.pkl', 'wb') as f:
  pickle.dump(tokenize(rows), f)
```

'코로나', '확진', '바이러스'와 같은 단어들은 어느 토픽에서는 높은 빈도로 출현하기 때문에 토픽의 특징이 살아나지 못해 제외하였음.
토큰화된 전체 목록에서 빈도수로 상위 50개의 단어를 불용어로 지정함.

```
# 전체 기사 토큰화 목록 불러오기
flist = get_filelist(root + 'tokenized', 'pkl')

# Counter를 이용하여 전체 언급 빈도 계산
counter_all = Counter()

for file in flist:
 pkl = pickle.load(open(file, 'rb'))
 counter_all.update(flatten(pkl))
```

```
# 전체 단어들 중 언급 순위 50개 단어 사용.
stopwords = counter_all.most_common(n=50)
stopwords = [t[0] for t in stopwords]

print(stopwords)
```

**OUT:**

['코로나', '확진', '바이러스', '신종', '지역', '환자', '감염', '중국', '감염증', '방역', '확산', '지원', '정부', '검사', '서울', '발생', '상황', '미국', '관련', '병원', '마스크', '판정', '이날', '격리', '사회', '시장', '경제', '조사', '추가', '확인', '기업', '이후', '가능', '한국', '경우', '조치', '진행', '대구', '시설', '사람', '교회', '국내', '센터', '이번', '오후', '당국', '사태', '대책', '세계', '예정']

```
# 폰트 옵션.
mpl.rcParams['axes.unicode_minus'] = False
# print([(f.name, f.fname) for f in fm.fontManager.ttflist if 'Nanum' in f.name])

fontprop = fm.FontProperties(fname=font_path, size=18)
```

```
# subplot을 생성하기 위한 준비.
fig, axs = plt.subplots(5, 2, figsize=(20,50), facecolor='w', edgecolor='k')
fig.subplots_adjust(hspace = .05, wspace = .01)
axs = axs.ravel()

for i, topic in enumerate(topic_use):
 # 각 토픽을 불러와 Counter로 출현 빈도 셈.
 pkl = pickle.load(open(f'{root}lda72_tokenized_all/{topic}.pkl', 'rb'))
 counter = Counter(flatten(pkl))

 # 불용어 제거
 for f in stopwords:
   try:
     counter.pop(f)
   except:
     continue
 # 워드 클라우드 생성.
 wc = WordCloud(font_path=font_path, width=1000, height=1000,
background_color="white", prefer_horizontal=1.0, max_font_size=300)
```

```
wc.generate_from_frequencies(counter)


# subplot에 그리기
axs[i].axis("off")
axs[i].set_title(description[i], fontproperties=fontprop)
axs[i].imshow(wc, interpolation="bilinear")
```

불용어를 처리하기 이전에는 대부분 '코로나', '바이러스' 등 압도적인 빈도수의 키워드들로 인해 모든 토픽이 가득차 토픽 간 차이가 불 분명 하였으나, 어느정도 불용어를 제거한 이후

워드 클라우드를 생성했을 때에는 단어가 적절히 표현되어 각 토픽의 주제를 확인할 수 있었음.

## VI. LDA 토픽 모델링을 통해 분류된 토픽 파이 차트

```
from math import pi
import pandas as pd

from bokeh.io import output_file, show, output_notebook
from bokeh.palettes import Category20c
from bokeh.models import Legend, HoverTool, Wedge
from bokeh.plotting import figure
from bokeh.transform import cumsum
```

```
# 토픽 설명 - 범례 및 툴팁에서 사용함
description = [
 '15. 공연 예술 및 해외 활동 감소에 관한 내용.',
 '23. 초/중/고교 등교 연기에 관한 내용.',
 '24. 사회적 거리 두기에 따른 시설 폐쇄 및 휴관에 관한 내용.',
 '25. 긴급 재난 지원금 지급에 관련된 내용.',
 '28. 졸업식 연기 및 행사 취소 및 소비 촉진 정책 연기와 관련된 내용.',
 '61. 온라인 쇼핑 매출 증가에 관련된 내용.',
 '62. 온라인 수업 온라인 강의 등록금 인하 관련 내용.',
 '64. 항공 노선 감축 및 운영 중단 관련 내용.',
 '65. 대출, 기업 / 소상공인 지원, 임대료 관련 내용.',
 '67. 기업의 부품 공급 차질, 공장 중단 관련 내용.'
]
```

토픽 분류를 담을 dataframe을 생성하여, 데이터 사용하였음.

```
# 사용할 토픽 선언
topic_use = [14, 22, 23, 24, 27, 60, 61, 63, 64, 66]
```

```
# 사용할 토픽들로 이루어진 dataframe 생성
# 각 토픽을 불러와 topic column을 추가하고 하나의 dataframe으로 합침.
topic_df = pd.DataFrame()
for num in topic_use:
 tmp_df = pd.read_csv(root+'lda72/%02d.csv' % num)
 c = tmp_df.columns.values
 c[0] = 'Document_No'
 tmp_df.columns = c
 tmp_df = tmp_df.set_index('Document_No')
 tmp_df['topic'] = num
 topic_df = pd.concat([topic_df, tmp_df])

topic_df.head()
```

**OUT:**

| Document_No | title | date | category | text | press | link | topic |
|---|---|---|---|---|---|---|---|
| 12 | '미스트롯' 의정부 공연 취소 "무대 구조물 무너져" 사과 | 2020-01-05 16:00:51 | 생활 | '내일은 미스트롯' 전국투어 시즌2 청춘 공연 포스터 2020.01.05. ... | 3 | https://m.news.naver.com/read.nhn?mode=LSD&mid... | 14 |
| 13 | 미스트롯 전국투어 의정부 공연, 안전 문제로 취소 | 2020-01-05 16:46:19 | 생활 | ~~WtWtWt~~공연기획사 "무대 설치 도중 구조물 일부 무너져"미스트롯 전국투어 콘서트... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... | 14 |
| 21 | 익산시, 중앙동 도시재생·문화예술공간 조성 협약 | 2020-01-06 16:14:21 | 사회 | =전북 익산시는 중앙동 문화예술공간 조성을 통한 도시재생 뉴딜사업의 성공 추진을... | 3 | https://m.news.naver.com/read.nhn?mode=LSD&mid... | 14 |
| 22 | 익산시, SM엔터 등과 '아이돌 양성소' 만든다...협약 체결 | 2020-01-06 16:34:17 | 사회 | 옛 하노바호텔에 아카데미, 댄스 연습실 등 조성6일 전북 익산시청 상황실에서 중앙동... | 421 | https://m.news.naver.com/read.nhn?mode=LSD&mid... | 14 |
| 271 | 영종 청소년가요제·월드뮤직경연대회, 2020년 첫 예선 성황리 개최 | 2020-01-12 20:05:00 | 경제 | ~~WtWtWt~~ 2020 제1회 영종 월드뮤직경연대회 예선광경/사진=공연&문화허브M터... | 8 | https://m.news.naver.com/read.nhn?mode=LSD&mid... | 14 |

```
# 토픽을 기준으로 묶어 개수를 셈.
t = topic_df.groupby('topic').count()

# 출현 빈도를 dataframe으로 표현
value = pd.DataFrame(t.T.loc['title'])

value.head()
```

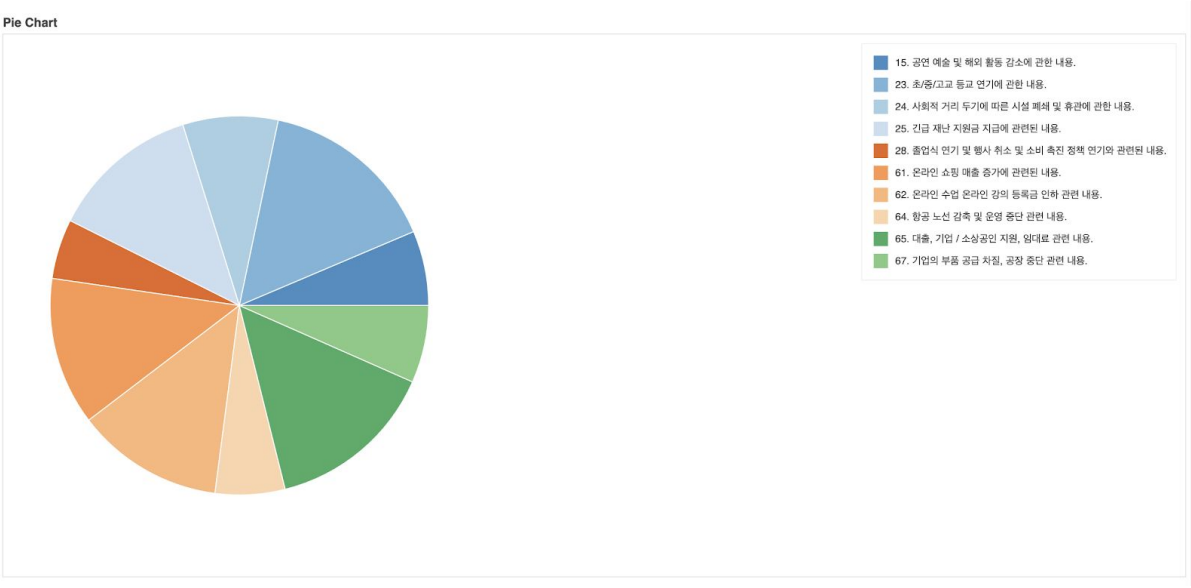
**OUT:**

| topic | title |
|---|---|
| 14 | 7784 |
| 22 | 18706 |
| 23 | 9930 |
| 24 | 15601 |
| 27 | 6227 |

```
output_notebook()



# angle: 파이를 그릴때 사용될 넓이값
# color: 각 토픽의 컬러 지정
# percentage: 각 토픽의 비율
# description: 범례

value['angle'] = value['title'] / value['title'].sum() * 2 * pi
value['color'] = Category20c[len(value)]
value['percentage'] = value['title'] / value['title'].sum()
value['description'] = description
```

```
# plot 선언, 크기 지정
p = figure(title="Pie Chart", toolbar_location=None, x_range=(-0.5, 2.0),
sizing_mode='stretch_width')

# 툴팁 설정
hover = HoverTool(
  tooltips = [
       ('topic', '@description'),
      ('value', '@percentage{%F}'),
  ])
p.add_tools(hover)

# 차트 추가
p1 = p.wedge(x=0, y=1, radius = 0.4, start_angle=cumsum('angle',
include_zero=True),
          end_angle=cumsum('angle'), legend_field='description',
line_color='white', fill_color='color', hover_alpha = 1.0, fill_alpha = 0.9,
source=value)
```

```
# 축 숨김, grid 숨김
p.axis.axis_label = None
p.axis.visible = False
p.grid.grid_line_color = None
p.legend.label_text_font_size = '8pt'
show(p)
```




## VII. LDA 토픽 모델링을 통해 분류된 토픽 시계열 분석

```
!pip install -q bokeh
from bokeh.plotting import output_notebook, figure, show
from bokeh.palettes import viridis, Category20c
from bokeh.models import ColumnDataSource, DatetimeTickFormatter,
Legend, LegendItem, HoverTool
from gensim.models.ldamodel import LdaModel
from gensim.corpora import Dictionary

import datetime
```

```
# 사용할 토픽 선언
topic_use = [14, 22, 23, 24, 27, 60, 61, 63, 64, 66]
```

```
# 토픽 설명 - 범례 및 툴팁에서 사용함
description = [
 '15. 공연 예술 및 해외 활동 감소에 관한 내용.',
 '23. 초/중/고교 등교 연기에 관한 내용.',
 '24. 사회적 거리 두기에 따른 시설 폐쇄 및 휴관에 관한 내용.',
 '25. 긴급 재난 지원금 지급에 관련된 내용.',
 '28. 졸업식 연기 및 행사 취소 및 소비 촉진 정책 연기와 관련된 내용.',
 '61. 온라인 쇼핑 매출 증가에 관련된 내용.',
 '62. 온라인 수업 온라인 강의 등록금 인하 관련 내용.',
 '64. 항공 노선 감축 및 운영 중단 관련 내용.',
 '65. 대출, 기업 / 소상공인 지원, 임대료 관련 내용.',
 '67. 기업의 부품 공급 차질, 공장 중단 관련 내용.'
]
```

```
topic_df = pd.DataFrame()
# 데이터 프레임에 토픽 column을 추가하여 토픽을 합침 - 이전과 동일
for num in topic_use:
 tmp_df = pd.read_csv(root+'lda72/%02d.csv' % num)
 c = tmp_df.columns.values
 c[0] = 'Document_No'
 tmp_df.columns = c
 tmp_df = tmp_df.set_index('Document_No')
 tmp_df['topic'] = num
 topic_df = pd.concat([topic_df, tmp_df])


topic_df.head()
```

| Document_No | title | date | category | text | press | link | topic |
|---|---|---|---|---|---|---|---|
| 12 | '미스트롯' 의정부 공연 취소 "무대 구조물 무너져" 사과 | 2020-01-05 16:00:51 | 생활 | '내일은 미스트롯' 전국투어 시즌2 청춘 공연 포스터 2020.01.05. ... | 3 | https://m.news.naver.com/read.nhn?mode=LSD&mid... | 14 |
| 13 | 미스트롯 전국투어 의정부 공연, 안전 문제로 취소 | 2020-01-05 16:46:19 | 생활 | ㅟㅟㅟ공연기획사 "무대 설치 도중 구조물 일부 무너져"미스트롯 전국투어 콘서트... | 1 | https://m.news.naver.com/read.nhn?mode=LSD&mid... | 14 |
| 21 | 익산시, 중앙동 도시재생·문화예술공간 조성 협약 | 2020-01-06 16:14:21 | 사회 | =전북 익산시는 중앙동 문화예술공간 조성을 통한 도시재생 뉴딜사업의 성공 추진을... | 3 | https://m.news.naver.com/read.nhn?mode=LSD&mid... | 14 |
| 22 | 익산시, SM엔터 등과 '아이돌 양성소' 만든다...협약 체결 | 2020-01-06 16:34:17 | 사회 | 옛 하노바호텔에 아카데미, 댄스 연습실 등 조성6일 전북 익산시청 상황실에서 중앙동... | 421 | https://m.news.naver.com/read.nhn?mode=LSD&mid... | 14 |
| 271 | 영종 청소년가요제·월드뮤직경연대회, 2020년 첫 예선 성황리 개최 | 2020-01-12 20:05:00 | 경제 | ㅟㅟㅟ 2020 제1회 영종 월드뮤직경연대회 예선광경/사진=공연&문화허브 M터... | 8 | https://m.news.naver.com/read.nhn?mode=LSD&mid... | 14 |

```
# 'date' column을 날짜 타입으로 변경
topic_df['date'] = pd.to_datetime(topic_df['date'])

# 기간, 토픽을 기준으로 그룹
t = topic_df.groupby([pd.Grouper(key='date', freq='W'), 'topic']).count()

t.head()
```

| topic<br>date | 14 | 22 | 23 | 24 | 27 | 60 | 61 | 63 | 64 | 66 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2020-01-05 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2020-01-12 | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 2020-01-19 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 2020-01-26 | 1.0 | 9.0 | 0.0 | 3.0 | 198.0 | 6.0 | 1.0 | 71.0 | 3.0 | 2.0 |
| 2020-02-02 | 106.0 | 355.0 | 13.0 | 51.0 | 1146.0 | 192.0 | 189.0 | 304.0 | 127.0 | 159.0 |

사용된 토픽은 아래와 같음.

```
description
```

**OUT:**

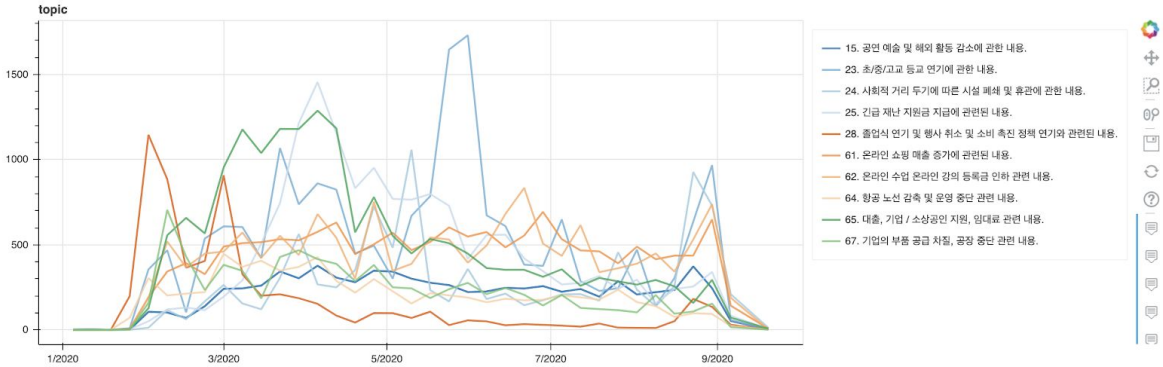
```
['15. 공연 예술 및 해외 활동 감소에 관한 내용.',
 '23. 초/중/고교 등교 연기에 관한 내용.',
 '24. 사회적 거리 두기에 따른 시설 폐쇄 및 휴관에 관한 내용.',
 '25. 긴급 재난 지원금 지급에 관련된 내용.',
 '28. 졸업식 연기 및 행사 취소 및 소비 촉진 정책 연기와 관련된 내용.',
 '61. 온라인 쇼핑 매출 증가에 관련된 내용.',
 '62. 온라인 수업 온라인 강의 등록금 인하 관련 내용.',
 '64. 항공 노선 감축 및 운영 중단 관련 내용.',
 '65. 대출, 기업 / 소상공인 지원, 임대료 관련 내용.',
 '67. 기업의 부품 공급 차질, 공장 중단 관련 내용.']
```

```
output_notebook()


# plot 생성, 드래그 방지
plot = figure(title='topic', plot_width=1000, plot_height=400,
x_axis_type='datetime', sizing_mode='stretch_width')
plot.toolbar.active_drag = None
```

```
# 색상 지정
colors = Category20c[len(t.columns)]


# 범례 저장 list
legend_list = []


# 토픽별로 lineplot을 그림
for i, (item, color) in enumerate(zip(t.iteritems(), colors)):
 c = plot.line(x = item[1].index, y = item[1].values, color=color, line_alpha = 0.8,
hover_line_width= 4, hover_alpha = 1.0, line_width = 2)
 legend_list.append([ description[i], [c] ])
```

```
 # Plotdp 툴팁 추가.
plot.add_tools(HoverTool(
    renderers = [c],
    tooltips = [
            ('date', '$x{%F}'),
            ('topic', description[i]),
            ('value', '$y{D}'),
            ],
    formatters={
      '$x': 'datetime'
  }))
```

```
# 범례 생성
legend = Legend(items=legend_list, level = 'annotation')
legend.click_policy = 'hide'

# plot 오른쪽에 범례 추가
plot.add_layout(legend, 'right')

# 범례 크기 조절
plot.legend.label_text_font_size = '11px'
show(plot)
```



## VIII. Word to Vector를 활용한 연관어 시각화

워드 활용해서 분석 진행함 gensim에서 제공되는 Word2Vec 클래스를 사용하였음. input 데이터는 선별한 토픽의 문서들로 이루어진 토큰화된 단어를 사용하였음. word2vec에서 단어의 동시 출현 빈도를 계산할 때 사용되는 알고리즘이 cbow와 skip-gram이 사용되고, 그중에서 연구 결과 정확도가 대체로 높다는 skip-gram을 이용하여 처리하였음.

처리 과정은 토큰화된 단어를 이용해 Word2Vec 모델을 생성하고, 생성된 모델을 시각화하기 위해 PCA로 2차원 벡터로 축소시킴. 그 뒤에 Bokeh를 이용하여 동적으로 그래프를 보여줌.

```
from gensim.models.word2vec import Word2Vec
import pickle
from sklearn.decomposition import PCA
import os
```

```
# 폴더에 있는 파일 목록을 가져오는 메서드
def get_filelist(path, extension):
 flist = os.listdir(path)
 result = []
 if path.endswith('/') == False:
   path += '/'
 for file in flist:
   if file.endswith(f'.{extension}'):
     result.append(path+file)
 return sorted(result)


# 여러 파일의 토큰을 제너레이터로 생성
class TokenCorpus():
```

```
 def __init__(self, flist):
   self.flist = flist

 def __iter__(self):
   for file in self.flist:
     with open(file, 'rb') as f:
       for el in pickle.load(f):
         yield el

# 2차원 배열 -> 1차원 배열
def flatten(l):
   flatList = []
   for elam in l:
       if type(elam) == list:
           for e in elam:
               flatList.append(e)
       else:
           flatList.append(elam)
   return flatList
```

```
# 사용할 토픽 선언
topic_use = [14, 22, 23, 24, 27, 60, 61, 63, 64, 66]

# 전체 토큰화된 것들 중에서 선별한 토픽만을 이용함.
flist = get_filelist(root + 'lda72_tokenized_all', 'pkl')
flist = [ flist[topic]  for topic in topic_use]

corpus = TokenCorpus(flist)
```

```
# 사용할 토픽 선언
topic_use = [14, 22, 23, 24, 27, 60, 61, 63, 64, 66]


corpus = TokenCorpus(flist)
```

```
# 전체 토큰화된 것들 중에서 선별한 토픽만을 이용함.
flist = get_filelist(root + 'lda72_tokenized_all', 'pkl')
flist = [ flist[topic]  for topic in topic_use]
```

```
# Word2Vec 모델 생성
# 200차원의 벡터로 skip-gram으로 학습하였음.
model = Word2Vec(corpus, size=200, iter=100,  window=5, min_count=2000,
sg=1)
model.init_sims(replace=True)
model.save(root+'w2v.model')
```

```
print(f'vector size: {model.vector_size}')
```

**OUT**:

vector size: 200

```
# 모든 단어의 벡터 값을 가져옴.
word_vectors = model.wv
vocabs = word_vectors.vocab.keys()
word_vectors_list = [word_vectors[v] for v in vocabs]
```

```
from bokeh.plotting import Figure, output_notebook, show
from bokeh.models import  ColumnDataSource, Select, CustomJS
```

```
from bokeh.palettes import viridis
from bokeh.layouts import layout, column

import pandas as pd
import numpy as np

output_notebook()
```

2차원으로 표현하기 위해 200차원의 워드 벡터를 PCA 알고리즘을 이용하여 2차원으로 축소하였음.

```
# PCA를 이용하여 2차원으로 축소함
pca = PCA(n_components = 2)
xys = pca.fit_transform(word_vectors_list)
```

```
# plot에서 x축, y축
xs = xys[:, 0]
ys = xys[:, 1]

print(f'x: {x[:5]}')
print(f'y: {y[:5]}')
```

```
# 사용할 데이터들을 dataframe으로 묶음.
df = pd.DataFrame()
df['x'] = xs
df['y'] = ys
df['annotate'] = list(vocabs)
df['color'] = 'grey'
df['alpha'] = 0.2
```

출현 빈도별로 크기를 다르게 하기 위해 Counter를 이용해 단어를 빈도수를 계산하였고, 이를 MinMaxScaler로 정규화 하였음.

```
# Counter를 이용하여 전체 언급 빈도 계산
counter = Counter()

for file in flist:
 pkl = pickle.load(open(file, 'rb'))
 counter.update(flatten(pkl))
```

```
d = pd.DataFrame(index=counter.keys(), data = counter.values(), columns =
['value'])
d = d.sort_index()
df = df.sort_values(by='annotate')
```

```
d = d[d.index.isin(df['annotate'])]

minmax = MinMaxScaler(feature_range=(5, 25))
value = minmax.fit_transform(d)
value = value.round(3)
```

```
d = pd.DataFrame(index=counter.keys(), data = counter.values(), columns =
['value'])
d = d.sort_index()
df = df.sort_values(by='annotate')

df['size'] = value

df.head()
```

**OUT:**

| | x | y | annotate | color | alpha | size |
|---|---|---|---|---|---|---|
| 713 | 0.055800 | 0.340743 | AFP | grey | 0.2 | 6.058 |
| 609 | 0.019441 | 0.376587 | AP | grey | 0.2 | 6.046 |
| 1227 | -0.004678 | 0.462070 | CNN | grey | 0.2 | 6.042 |
| 1439 | 0.213932 | 0.216895 | EU | grey | 0.2 | 6.012 |
| 253 | -0.095378 | 0.028138 | KBS | grey | 0.2 | 6.075 |

```
d = pd.DataFrame(index=counter.keys(), data = counter.values(), columns =
['value'])
d = d.sort_index()
df = df.sort_values(by='annotate')
```

Bokeh를 이용해 시각화를 수행하였음. 각 단어를 선택하면 해당 단어와 유사한 단어들이 보라색으로 표시됨.
bokeh는 plot들이 JS로 렌더링하기 때문에 파이썬 객체 word2vec 모델을 그대로 사용할 수 없음. 때문에 word2vec.wv.most_similar 결과를 모두 딕셔너리에 담아서 사용하였음.
word2vec 모델의 관점에서 해당 단어와 유사한 단어라는 뜻인데, 이는 코사인 유사도를 계산하여 유사도가 가장 높은 값을 사용함.
개수는 20으로 고정하였음.
메뉴의 단어들은 각 토픽의 대표적인 단어 10개를 사용하였음.

```
# 시각화에서 사용할 각 단어의 most_similar 값을 저장해둠.
word_similiar = dict()
for word in vocabs:
  word_similiar[word] = [w[0] for w in model.wv.most_similar(word, topn=20)]
```

```
# menu에서 사용할 단어들.
keyword_list = list(set(['공연', '예술', '영화', '문화', '코로나', '관객', '극장', '음악',
```

```
'무대', '콘서트', '학교', '학생', '등교', '수업', '개학', '교육청', '학년', '교육', '교사',
'교육부',  '거리', '두기', '사회', '시설', '운영', '생활', '코로나', '단계', '이용', '도서관',
'지급', '지원', '지원금', '고용', '재난', '긴급', '신청', '소득', '코로나', '가구',  '코로나',
'확산', '취소', '신종', '연기', '바이러스', '단계', '감염증', '결정', '예정','판매', '매출',
'온라인', '제품', '고객', '상품', '구매', '매장', '소비자', '코로나', '대학', '시험', '학생',
'교육', '온라인', '수업', '코로나', '채용', '강의', '진행',  '항공', '운항', '노선', '항공사',
'코로나', '중단', '아시아나항공', '국제선', '공항',  '지원', '대출', '코로나', '기업',
'자금', '은행', '금융', '공인', '소상', '중소기업', '공장', '생산', '자동차', '울산', '업체',
'가동', '코로나', '현대', 'LG', '현대차']))
```

```
# datatype을 bokeh에 맞게 변환함
source = ColumnDataSource(data=dict(
   x = df['x'],
   y = df['y'],
   color = df['color'],
   annotate = df['annotate'],
   alpha = df['alpha'],
   size = df['size']
))
```

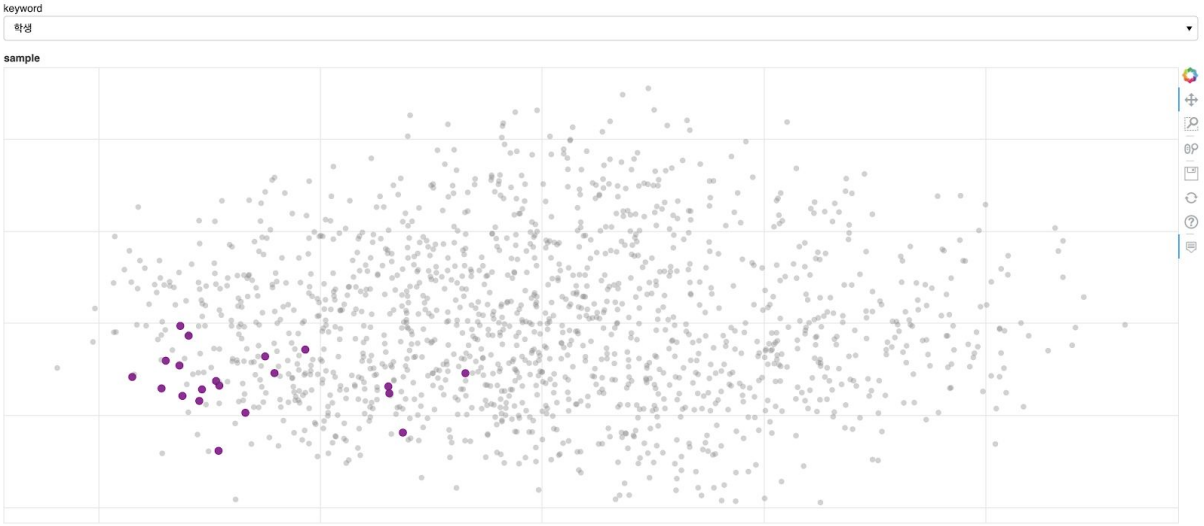
```
# 표시되는 원에 글자 툴팁
tooltips = [
        ('word', '@annotate')
]

# 그래프 그리기. 축 제거
p = Figure(title="sample", sizing_mode='stretch_width', tooltips = tooltips)
p.axis.visible = False
```

```
# 메뉴 선택시에 변경될 데이터 JS 콜백으로 구현
callback = CustomJS(args=dict(source=source, word_similiar=word_similiar),
                    code="""
 var data = source.data;
 var word_similiar = word_similiar;
 var value = cb_obj.value;
 var color = data['color'];
 var annotate = data['annotate'];
 var alpha = data['alpha'];
 var size = data['size'];
 for (var i = 0; i < size.length; i++) {
   if (word_similiar[value].includes(annotate[i])) {
     color[i] = 'purple';
     alpha[i] = 0.6;
   } else {
     color[i] = 'grey';
     alpha[i] = 0.2;
   }
 }
 console.log(source);
 source.change.emit();
 """ )
```

```
# 단어를 선택할 Select
menu = Select(title="keyword", options=sorted(keyword_list))
menu.js_on_change('value', callback)
```

```
# 위 source를 기반으로 그래프를 그림
p.circle(x='x', y='y', size = 'size', color='color', fill_alpha= 'alpha', line_width = 0,
```

```
source=source)
layout = column(menu, p)
layout.sizing_mode = 'stretch_width'


show(layout)
```




선정된 키워드와 유사한 키워드를 강조하여 2차원으로 표시하였으나, 시각화 결과가 주제와 밀접하게 관련되어있다고 보기는 어려웠음.
하지만, 기사에서 많이 언급되는 키워드가 벡터에서도, 위치에도 반영이 되기 때문에 계속해서 분석 또는 학습을 진행할 때에 도움이 될 것으로 보임.

# 4.결론

- 뉴스기사 건수 및 본문을 바탕으로 시기별로 코로나 관련하여 어떤 관심사가 높아졌는지 알 수 있었습니다.
- 토픽 모델링을 통해 다양한 코로나 관련 기사들을 볼 수 있었고, 이를 분류하여 원하는 토픽을 확인하였습니다.
- 사람들의 어떤 것들에 관심을 가지고, 또 관심이 어떻게 이동하는지 확인할 수 있었습니다.