**Python 설치**

1. 아나콘다 검색-필수 라이브러리 설치 해줌(Numpy, sk-learn,pandas 등)   
   - 구글에서 “anaconda download” 입력  
   - 첫번째 검색 결과 클릭
2. 자신의 운영 환경에 맞는지 확인 후 다운로드  
   - Python 3.8 • 64-Bit Graphical Installer • 477 MB
3. 설치  
   - install for 옵션에서 Just Me설치  
   - Advanced 옵션에서 모두 선택  
   - VS Code 설치는 Skip
4. 아나콘다 프로픔트 실행 확인

**Jupyter notebook 실행**

1. 아나콘다 프롬프트 실행
2. 탐색기 활용 실행

**Python 기초**

1. 기초  
   - 값 : 3,4,5 등  
   - 문자열 : 외 따움표 표시 ‘문자열’  
   - 사칙연산 : +,-,%,\*  
   - 변수 : A = ‘비정형 분석’  
   - 리스트 : Text = [‘사과’, ‘바나나’]  
   - 사전 : Dic = { ‘사과’ : 1, ‘바나나’ : 2 }  
     
     
   - 함수( 함수는 값을 처리하는 일종의 명령이다. 함수 이름 뒤에는 괄호를 붙여 값을 넘겨준다. 함수는 넘겨 받은 값을 처리하고 그 결과를 다시 돌려준다. ) : len(A)  
   - 메소드( 함수와 비슷 하지만 값의 종류에 따라 다른 방식으로 작동 하는 함수들이다. ‘값.메소드’ 형태로 표현 ) : A.split() 공백 기준으로 분리  
   - 모듈( 비슷한 기능을 모아 놓은 것을 모듈이라고 함. 여러가지 모듈을 불러와서 사용 가능 ) :   
   import math 수학 계산 관련 모듈 -> math.sqrt(4)  
   import math as m -> m.squrt(9)  
   from math import cos -> cos(0)  
   from math import \* -> cos(0)

**Word Cloud 그리기**

텍스트 분석 해보면 가장 먼저 떠오르고 한번씩 해보는 빈도 분석의 일종임.

1. WordCloud 설치  
   - pip install로는 window 지원하지 않아 다온로드 후 설치   
   - Unofficial Windows Binaries for Python Extension Packages  
   - wordcloud 1.8.1 cp38 cp38 win\_amd64.whl   
   - pip install wordcloud 1.8.1 cp38 cp38 win\_amd64.죄  
   - 맥, 리눅스 pip install wordcloud
2. 데이터 스크래핑  
   - GET은 서버 측에 존재하는 데이터를 가져오기 위한 요청 방법이다. 파이썬의 requests 라이브러리로 GET 요청을 보내보자.   
   - **pip install requests** 먼저 라이브러리 설치  
   - 주피터 노트북에서는 !pip install requests

- 모듈 호출 해보자   
- **import requests**  
- 저작권이 만료된 텍스트들을 모아놓은 구텐베르크 프로젝트(http://gutenberg.org)에서 그림 동화를 가져와보자  
 **url = 'http://www.gutenberg.org/files/2591/2591-0.txt'**

**res = requests.get(url)  
- res.status\_code**  
- 2XX: 성공. 보통 200을 사용한다  
지금 까지 스크래핑을 통해 데이터를 수집 해 보았다.

1. 단어 추출  
   - res.text[:100] -> 단어 백개 까지 확인 해봄  
   - import re -> 문장 정규화 모듈 호출

* words = re.findall(r'[a-zA-Z]+', res.text)   
  -> 알파벳만 남기고 삭제(특수문자, 줄바꿈 등)

- words[:10] -> 단어 10개 확인

4. 단어 세기  
word\_count = {} # 사전을 만든다

for word in words: # 모든 단어에 대해서

word = word.lower() # 단어를 소문자로 바꾼다

if word in word\_count: # 사전에 단어가 있으면

word\_count[word] += 1 # 단어의 개수를 1 증가 시킨다

else: # 없으면

word\_count[word] = 1 # 단어의 개수를 1로 한다

ring이라는 단어가 몇 번 등장했는지 확인해보자.

word\_count['ring']

**빈도순 정렬**

import pandas as pd

df = pd.DataFrame(list(word\_count.items()), columns=['word', 'n'])

확인

df.sort\_values('n', ascending=False).head()

**단어 구름 그리기**

jupyter notebook에서는 그래프를 그리면 새 창으로 뜬다. 그래프를 노트북과 함께 저장하려면 아래 매직 명령으로 설정을 바꿔준다

%matplotlib inline

단어 구름을 그리기 위해서는 matplotlib와 wordcloud 패키지들을 불러오면 된다.

import matplotlib.pyplot as plt

from wordcloud import WordCloud

워드 클라우드는 딕셔너리 형태로 데이터를 받는다. 만약 단어(words)와 빈도(count)가 각각 리스트 형태로 있다면 짝지어 딕셔너리를 만들어준다.

word\_count = dict(zip(df.word, df.n))

생성된 딕셔너리인 word\_count를 확인해보자.

list(word\_count.items())[:5]

단어와 해당 단어의 빈도수가 잘 짝지어진 것을 확인할 수 있다.

WordCloud 함수에 배경색을 바꿀 수 있는 옵션인 backgroud\_color는 'white'로 지정하고, 넓이는 400, 높이는 200 이라는 옵션을 넘겨주면서 틀을 생성한다.

wc = WordCloud(background\_color='white', width=400, height=200)

이제 워드 클라우드 틀을 생성하였으니 단어들을 넘겨준다. .fit\_words()에 단어와 빈도수로 이루어진 딕셔너리를 넘겨준다.

cloud = wc.fit\_words(word\_count)

cloud 변수의 그림을 노트북에 붙여 넣으려면 아래와 같이 한다.

cloud.to\_image()

**한글 처리**

한글의 경우 워드 클라우드에서 옵션으로 font\_path를 지정해야한다. 굴림체로 지정할경우 다음과 같이 폰트 위치를 지정해준다.

kor\_wc = WordCloud(background\_color='white', width=400, height=200,

font\_path='gulim.ttf')

### 다음 뉴스 크롤링 해보기

#### 검색결과 가져오기

다음(http://daum.net)에서 "인공지능"으로 검색을 한다.

검색 결과에서 '뉴스', '최신순'을 클릭한다.

이제 해당 검색결과의 URL을 requests로 가져온다.

**import** **requests**

res = requests.get('http://search.daum.net/search?w=news&cluster=n&q=**%E**C%9D%B8**%E**A%B3%B5**%E**C%A7**%80%**EB%8A%A5&sort=recency')

#### 기사 링크 모으기

다음 뉴스 검색 결과에서 기사 제목을 클릭하면 언론사 홈페이지로 이동한다. 언론사마다 홈페이지 디자인이 다르기 때문에 자동으로 기사 수집이 어렵다. 대신 주요 언론사의 기사는 기사 제목 옆에 "다음 뉴스"를 클릭하면 다음 내에서 확인할 수 있다.

**어크리뷰트 중 href -> URL 링크 주소로 지정한다.**

**import** **lxml.html**

root = lxml.html.fromstring(res.text)

urls = []

**for** link **in** root.cssselect('a.f\_nb'):

urls.append(link.attrib['href']) #URL 수집

#### 기사 본문 수집하기

다음 뉴스의 기사 페이지에서 본문은 다음과 같이 수집한다.

articles = []

**for** u **in** urls:

**if** **not** u.startswith('http'): #시작이 http로 시작하는지 체크

**continue**

res = requests.get(u)

root = lxml.html.fromstring(res.text) # URL의 텍스트 결과 값을 넘김

body = root.cssselect('.article\_view').pop() # pop를 통하여 마지막 불필요 부분 삭제

content = body.text\_content()

articles.append(content)

pandas를 이용해서 엑셀 파일로 저장한다.

**import** **pandas** **as** **pd**

df = pd.DataFrame({'article': articles})

df.to\_excel('기사.xlsx')

## **7.2. 네이버 영화 별점 스크랩**

네이버 영화에서 관객 리뷰와 별점 스크래핑한다.

먼저, 네이버 영화에서 분석하고 싶은 영화 페이지로 들어간다.

네티즌 별점을 누르면 네티즌들이 남긴 별점과 평을 볼 수 있다.

페이지 번호를 우클릭하고 주소 복사를 하면 아래와 같은 페이지 주소를 복사할 수 있다.

https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=150689&type=after&isActualPointWriteExecute=false&isMileageSubscriptionAlready=false&isMileageSubscriptionReject=false&page=1

주소의 맨 끝 부분을 보면 페이지를 나타낸다는 것을 알 수 있다.

Python에서 다음과 같이 주소를 만든다.

url = 'https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=150689&type=after&isActualPointWriteExecute=false&isMileageSubscriptionAlready=false&isMileageSubscriptionReject=false&page={}'

이제 다음과 같이 리뷰를 수집한다.

**import** **requests**

**import** **lxml.html**

*# 리뷰와 별점을 모을 빈 리스트를 만든다*

reviews = []

scores = []

**for** page **in** range(1, 30): *# 1~29페이지까지 반복*

res = requests.get(url.format(page)) *# 각 페이지에 접속한다*

root = lxml.html.fromstring(res.text) *# html을 처리한다*

*# 리뷰를 가져와 reviews에 추가한다*

**for** review **in** root.cssselect('.score\_reple p'):

reviews.append(review.text\_content())

*# 별점을 가져와 scores에 추가한다*

**for** score **in** root.cssselect('.score\_result .star\_score em'):

scores.append(score.text\_content())

수집한 리뷰를 데이터 프레임으로 만든다.

**import** **pandas**

df = pandas.DataFrame({'score': scores, 'review': reviews})

데이터 프레임을 CSV 파일로 저장한다.

df.to\_csv('movie\_review.csv', encoding='utf8', index=False)

review\_start = pandas.read\_csv('movie\_review.csv',encoding='utf8')

review\_start['review']=df['review'].str.replace(pat=r'[^ A-Za-z0-9가-힣+]',repl= r'', regex=True)

review\_start.head()