# [ 웹크롤링 _ 나무위키 사이트 분석 및 시각화 ]

## <Step1. 크롤링> : 크롤링으로 웹 데이터 가져오기

[웹크롤링 라이브러리 사용하기]

- 파이썬에서는 BeautifulSoup과 requests라는 라이브러리로 웹 크롤러를 만들 수 있음
- requests는 특정 URL로부터 HTML 문서를 가져오는 작업을 수행
- 나무위키와 같은 페이지는 HTML 문서가 Javascript로 동적 로딩되는 경우가 있음
- requests 대신 셀레니움(selenium) 라이브러리를 이용해 크롬 브라우저로 동적 웹크롤링 수행
- selenium은 웹 브라우저를 자동으로 구동해주는 라이브러리
- selenium을 사용하기 위해 크롬 드라이버를 이용해 크롬 브라우저 자동으로 구동=> 크롬드라이버 필요

## [BeautifulSoup과 selenium을 이용한 웹 크롤링]

- anaconda prompt 혹은 Terminal에서 아래와 같은 패키지들을 설치
- (env_name) pip install selenium
- (env_name) pip install beautifulsoup4

## [크롬 브라우저 업데이트 및 크롬 드라이버 설치]

- 크롬 브라우저 설정에서 최신 버전으로 업데이트
- 크롬 드라이버 사이트에서 브라우저 버전에 맞는 드라이버 다운로드
  - https://chromedriver.chromium.org/downloads (https://chromedriver.chromium.org/downloads)
- chromedriver.exe 파일을 노트북 파일 경로에 이동

In [1]:

```
# -*- coding: utf-8 -*-

%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

## [ BeautifulSoup의 select() VS find_all() ]

- HTML의 특정 요소 선택
- select, select_one 의 경우 CSS 선택자를 이용하는 것처럼 사용 가능
- select의 경우 후손이나 자손 요소를 CSS 처럼 선택 가능
- 예) soup.select("dl > dt > a")
- find_all, find 의 경우 하나의 태그(name="table")나 하나의 클래스(class="tables")를 선택
- find의 경우 후손이나 자손 요소를 직접 선택할 수 없어 한번 더 변수에 담든지 루프 문을 이용해야 함
- 예) find_all(class="ah_roll"), find(name="table")

In [2]:

```
from selenium import webdriver
from bs4 import BeautifulSoup
import re # 정규식 표현을 위한 모듈


# 윈도우용 크롬 웹드라이버 실행 경로 (Windows) 지정
excutable_path = "chromedriver.exe"
driver = webdriver.Chrome(executable_path=excutable_path)

# 사이트의 html 구조에 기반하여 크롤링을 수행
source_url = "https://namu.wiki/RecentChanges" # 크롤링할 사이트 주소를 정의
driver.get(source_url)  # 크롬 드라이버를 통해 URL의 HTML 문서 가져옴

# 영진씨 방법
# from selenium.webdriver.common.by import By
# from selenium.webdriver.support.ui import WebDriverWait
# from selenium.webdriver.support import expected_conditions as EC
# element = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.CLASS_NAME, "app")))

# 명진씨 방법
import time
time.sleep(10)

req = driver.page_source
soup = BeautifulSoup(req, "html.parser") # BeautifulSoup의 soup 객체로 변환

#contents_table = soup.find(name="table")
#table_body = contents_table.find(name="tbody")
#table_rows = table_body.find_all(name="tr")
table_rows = soup.select("table tbody tr")
```

In [3]:

```
req
```

Out[3]:


```
'<html><head><link href="/skins/senkawa/6.4cd8c9c215a8ec226109.css" rel="styleshee
t"><link href="/skins/senkawa/3.6578e2190d7f10b0f082.css" rel="stylesheet"><script
async="" src="/cdn-cgi/bm/cv/669835187/api.js"></script><style type="text/css">.re
size-observer[data-v-b329ee4c]{position:absolute;top:0;left:0;z-index:-1;width:10
0%;height:100%;border:none;background-color:transparent;pointer-events:none;displa
y:block;overflow:hidden;opacity:0}.resize-observer[data-v-b329ee4c] object{displa
y:block;position:absolute;top:0;left:0;height:100%;width:100%;overflow:hidden;poin
ter-events:none;z-index:-1}</style><link rel="stylesheet" type="text/css" href="/s
kins/senkawa/10.a1e48eb86746983c527d.css"><script charset="utf-8" src="/skins/senk
awa/10.a1e48eb86746983c527d.js"></script><title>최근 변경내역 - 나무위키</title><l
ink data-n-head="1" rel="canonical" href="https://namu.wiki/RecentChanges"><link d
ata-n-head="1" rel="search" type="application/opensearchdescription+xml" title="나
무위키" href="/opensearch.xml"><link data-n-head="1" rel="icon" href="/favicon.sv
g" sizes="any" type="image/svg+xml"><link data-n-head="1" rel="icon" href="/favico
n-32.png" sizes="32x32" type="image/png"><link data-n-head="1" rel="icon" href="/f
avicon-192.png" sizes="192x192" type="image/png"><link data-n-head="1" rel="apple-
touch-icon" href="/img/apple_icon.png"><meta data-n-head="1" charset="utf-8"><meta
data-n-head="1" name="viewport" content="user-scalable=no, initial-scale=1.0, maxi
```

In [4]:

```
soup
```

Out[4]:

```
<html><head><link href="/skins/senkawa/6.4cd8c9c215a8ec226109.css" rel="styleshee
t"/><link href="/skins/senkawa/3.6578e2190d7f10b0f082.css" rel="stylesheet"/><scri
pt async="" src="/cdn-cgi/bm/cv/669835187/api.js"></script><style type="text/cs
s">.resize-observer[data-v-b329ee4c]{position:absolute;top:0;left:0;z-index:-1;wid
th:100%;height:100%;border:none;background-color:transparent;pointer-events:none;d
isplay:block;overflow:hidden;opacity:0}.resize-observer[data-v-b329ee4c] object{di
splay:block;position:absolute;top:0;left:0;height:100%;width:100%;overflow:hidden;
pointer-events:none;z-index:-1}</style><link href="/skins/senkawa/10.a1e48eb867469
83c527d.css" rel="stylesheet" type="text/css"/><script charset="utf-8" src="/skin
s/senkawa/10.a1e48eb86746983c527d.js"></script><title>최근 변경내역 - 나무위키</ti
tle><link data-n-head="1" href="https://namu.wiki/RecentChanges" rel="canonical"/>
<link data-n-head="1" href="/opensearch.xml" rel="search" title="나무위키" type="a
pplication/opensearchdescription+xml"/><link data-n-head="1" href="/favicon.svg" r
el="icon" sizes="any" type="image/svg+xml"/><link data-n-head="1" href="/favicon-3
2.png" rel="icon" sizes="32x32" type="image/png"/><link data-n-head="1" href="/fav
icon-192.png" rel="icon" sizes="192x192" type="image/png"/><link data-n-head="1" h
ref="/img/apple_icon.png" rel="apple-touch-icon"/><meta charset="utf-8" data-n-hea
d="1"/><meta content="user-scalable=no, initial-scale=1.0, maximum-scale=5.0, mini
```

In [5]:

```
# contents_table
```

In [6]:

```
# table_body
```

In [7]:

```
table_rows
```

Out[7]:

```
[<tr class="" data-v-17dde2e0=""><td data-v-17dde2e0=""><a data-v-17dde2e0="" href
="/w/%EB%AC%B4%ED%95%9C%EB%8F%84%EC%A0%84%20TV">무한도전 TV</a> <a data-v-17dde2e0
="" href="/history/%EB%AC%B4%ED%95%9C%EB%8F%84%EC%A0%84%20TV">[역사]</a> <a data-v
-17dde2e0="" href="/diff/%EB%AC%B4%ED%95%9C%EB%8F%84%EC%A0%84%20TV?rev=338&amp;old
rev=337">[비교]</a> <a data-v-17dde2e0="" href="/discuss/%EB%AC%B4%ED%95%9C%EB%8F%
84%EC%A0%84%20TV">[토론]</a> <span data-v-17dde2e0="">(<span class="SBT2YK7T" data
-v-17dde2e0="" data-v-6cbb5b59="">-156</span>)</span></td> <td data-v-17dde2e0="">
<div class="v-popover" data-v-17dde2e0="" data-v-1de7cf8c=""><div aria-describedby
="popover_bli7i68uhr" class="trigger" style="display: inline-block;"><a data-v-1de
7cf8c="">175.120.239.227</a> </div> </div> <!-- --></td> <td data-v-17dde2e0=""><t
ime data-v-17dde2e0="" datetime="2022-01-25T04:01:03.000Z">2022-01-25 13:01:03</ti
me></td></tr>,
 <tr class="" data-v-17dde2e0=""><td data-v-17dde2e0=""><a data-v-17dde2e0="" href
="/w/%EB%B6%84%EB%A5%98:%EB%B8%94%EB%9E%99%EC%8A%A4%ED%83%80%20-Theater%20Starless
-">분류:블랙스타 -Theater Starless-</a> <a data-v-17dde2e0="" href="/history/%EB%B
6%84%EB%A5%98:%EB%B8%94%EB%9E%99%EC%8A%A4%ED%83%80%20-Theater%20Starless-">[역사]
</a> <a data-v-17dde2e0="" href="/diff/%EB%B6%84%EB%A5%98:%EB%B8%94%EB%9E%99%EC%8
A%A4%ED%83%80%20-Theater%20Starless-?rev=7&amp;oldrev=6">[비교]</a> <a data-v-17dd
```

In [8]:

```
len(table_rows)
```

Out[8]:

```
111
```

## [페이지 링크주소 리스트 가져오기]

In [9]:

```
page_url_base = "https://namu.wiki" # 베이스 URL 정의
page_urls = [] # href 속성값을 담기 위한 빈 리스트 생성

for index in range(0, len(table_rows)):
    first_td = table_rows[index].find_all("td")[0]
    td_url = first_td.find_all("a")
    if len(td_url) > 0:
        # 특정 속성 선택시 attrs["속성명"] 또는 get("속성명") 사용
        # page_url = page_url_base + td_url[0].get("href")
        # attrs는 딕셔너리 형태로 속성명과 속성값을 불러옴
        # attrs["href"]는 attrs 결과 중 key가 href인 것의 값만 불러옴
        page_url = page_url_base + td_url[0].attrs["href"]
        if "png" not in page_url:
            page_urls.append(page_url)
            print(page_urls)
```

```
['https://namu.wiki/w/%EB%AC%B4%ED%95%9C%EB%8F%84%EC%A0%84%20TV']
['https://namu.wiki/w/%EB%AC%B4%ED%95%9C%EB%8F%84%EC%A0%84%20TV', 'https://namu.wi
ki/w/%EB%B6%84%EB%A5%98:%EB%B8%94%EB%9E%99%EC%8A%A4%ED%83%80%20-Theater%20Starless
-']
['https://namu.wiki/w/%EB%AC%B4%ED%95%9C%EB%8F%84%EC%A0%84%20TV', 'https://namu.wi
ki/w/%EB%B6%84%EB%A5%98:%EB%B8%94%EB%9E%99%EC%8A%A4%ED%83%80%20-Theater%20Starless
-', 'https://namu.wiki/w/%EC%95%84%EC%8A%A4%ED%83%80%EB%A1%9C%EC%8A%A4']
['https://namu.wiki/w/%EB%AC%B4%ED%95%9C%EB%8F%84%EC%A0%84%20TV', 'https://namu.wi
ki/w/%EB%B6%84%EB%A5%98:%EB%B8%94%EB%9E%99%EC%8A%A4%ED%83%80%20-Theater%20Starless
-', 'https://namu.wiki/w/%EC%95%84%EC%8A%A4%ED%83%80%EB%A1%9C%EC%8A%A4', 'https://
namu.wiki/w/Good%20Times%20Bad%20Times']
['https://namu.wiki/w/%EB%AC%B4%ED%95%9C%EB%8F%84%EC%A0%84%20TV', 'https://namu.wi
ki/w/%EB%B6%84%EB%A5%98:%EB%B8%94%EB%9E%99%EC%8A%A4%ED%83%80%20-Theater%20Starless
-', 'https://namu.wiki/w/%EC%95%84%EC%8A%A4%ED%83%80%EB%A1%9C%EC%8A%A4', 'https://
namu.wiki/w/Good%20Times%20Bad%20Times', 'https://namu.wiki/w/2022%20%EB%B2%A0%EC%
9D%B4%EC%A7%95%20%EB%8F%99%EA%B3%84%EC%98%AC%EB%A6%BC%ED%94%BD']
['https://namu.wiki/w/%EB%AC%B4%ED%95%9C%EB%8F%84%EC%A0%84%20TV', 'https://namu.wi
ki/w/%EB%B6%84%EB%A5%98:%EB%B8%94%EB%9E%99%EC%8A%A4%ED%83%80%20-Theater%20Starless
-', 'https://namu.wiki/w/%EC%95%84%EC%8A%A4%ED%83%80%EB%A1%9C%EC%8A%A4', 'https://
```

In [10]:

```
td_url[0].attrs
```

Out[10]:

```
{'data-v-17dde2e0': '', 'href': '/w/%EB%9E%B4%EC%98%A4%EB%94%A9%EC%96%91'}
```

In [11]:

```
page_urls
```

Out[11]:


```
['https://namu.wiki/w/%EB%AC%B4%ED%95%9C%EB%8F%84%EC%A0%84%20TV',
 'https://namu.wiki/w/%EB%B6%84%EB%A5%98:%EB%B8%94%EB%9E%99%EC%8A%A4%ED%83%80%20-T
heater%20Starless-',
 'https://namu.wiki/w/%EC%95%84%EC%8A%A4%ED%83%80%EB%A1%9C%EC%8A%A4',
 'https://namu.wiki/w/Good%20Times%20Bad%20Times',
 'https://namu.wiki/w/2022%20%EB%B2%A0%EC%9D%B4%EC%A7%95%20%EB%8F%99%EA%B3%84%EC%9
8%AC%EB%A6%BC%ED%94%BD',
 'https://namu.wiki/w/%EB%AA%A8%EC%8A%A4%20%ED%85%8C%ED%81%AC%EB%86%80%EB%A1%9C%E
C%A7%80%206502',
 'https://namu.wiki/w/%EA%B9%80%EC%84%B1%ED%9A%8C(%EB%B0%A9%EC%86%A1%EC%9D%B8)',
 'https://namu.wiki/w/%EC%9B%8C%EA%B8%80',
 'https://namu.wiki/w/%EC%9C%A0%ED%9D%AC%EC%99%95%20%EB%A7%88%EC%8A%A4%ED%84%B0%2
0%EB%93%80%EC%96%BC/Live2D%EA%B0%80%20%EC%A0%81%EC%9A%A9%EB%90%9C%20%EC%B9%B4%EB%9
3%9C%EB%93%A4',
 'https://namu.wiki/w/%EB%94%A5%20%EC%9E%84%ED%8C%A9%ED%8A%B8(%EB%A7%90)',
 'https://namu.wiki/w/%EC%84%B1%EA%B0%80%EC%88%98%EB%8F%84%ED%9A%8C',
 'https://namu.wiki/w/2022%EB%85%84%203%EC%9B%94%20%EC%9E%AC%EB%B3%B4%EA%B6%90%EC%
84%A0%FA%B1%B0',
```


## [각 링크 페이지내 텍스트 구조를 확인하여 제목, 카테고리, 내용 출력]

In [12]:

```
# 윈도우용 크롬 웹드라이버 실행 경로 (Windows) 지정
excutable_path = "chromedriver.exe"
driver = webdriver.Chrome(executable_path=excutable_path)
# 크롬 드라이버를 통해 page_urls[0]번째 사이트의 HTML 문서 가져옴
driver.get(page_urls[0])  # page_urls[0] 의 정보를 가져옴
req = driver.page_source # 페이지 소스를 req에 저장
soup = BeautifulSoup(req, 'html.parser') # html.parser로 파싱
contents_table = soup.find(name="article") #  불러온 소스에서 태그명이 article인 요소 하나만 추출

### 타이틀 추출
title = contents_table.find_all('h1')[0] # 태그명이 h1인 모든 태그 추출, article h1

### 카테고리 추출
category = contents_table.find_all('ul')[0]

### 내용 추출
#contents_table.find_all(name="div", attrs={"class":"wiki-paragraph"})
#div  태그 중  class 속성값이  wiki-paragraph인 요소를 추출
content_paragraphs = contents_table.select("div.wiki-paragraph")

#  내용으로 추출한 리스트를 하나의 문자열로 전처리
content_corpus_list = [] # 내용 중 텍스트만 담을 빈 리스트 생성
# content_paragraphs 리스트의 값을 순서대로 paragraphs에 대입
for paragraphs in content_paragraphs: # content_paragraphs 리스트의 값을 순서대로 paragraphs에 대입
    content_corpus_list.append(paragraphs.text)  # 가져온 결과 태그 중 텍스트만 추출하여 content_cor
content_corpus =" ".join(content_corpus_list) #"텍스트".join(리스트명) => 리스트의 요소를 "텍스트"로

print(title.text) # 제목 출력
print("Wn")
print(category.text) # 카테고리 출력
print("Wn")
print(content_corpus) # 내용 출력

# 크롤링에 사용한 브라우저를 종료합니다.
driver.close()
```

```
무한도전 TV


무한도전/2009년로그 누락 문서


무한도전의 역대 에피소드 품절남 특집 → 무한도전 TV → 무한도전 벼농사 특집  문서
가 존재하는 무한도전 특집 이 외의 특집은 해당 문서 참조 1. 개요2. 프로그램 목록2.
1. 화면조정시간2.2. MBC 뉴스투데이2.3. 지구촌 리포트2.4. 경제매거진 M2.5. 무한도전
마이너2.6. 특선영화 취권2.7. 찾아라! 맛있는 TV2.8. 무릎팍도사2.9. 밥줘2.10. 쇼! 음
악중심2.11. 노안선발대회2.12. 정준하의 푸드 이상형 월드컵2.13. MBC 뉴스데스크2.14.
스포츠뉴스2.15. 세바퀴2.16. 특선대작 스타워즈2.17. 애국가3. 기타  무한도전 TV는 방
송위원회의 심의 규정을 준수하려고 합니다만... 무한도전 173, 174회 에피소드로 2009
년 10월 3일과 10일 2회에 걸쳐 방영. MBC의 추석 당시 방영한 프로그램들을 콩트 형식
으로 패러디한 특집이자 무한도전 B급 감성의 마지막 특집이기도 하다.[1] 무한도전 제
작진들이 자체적으로 만든 추석특집 편성표에 따라 멤버들이 그대로 찍는 것. 자연스럽
게 MBC에서 방영하는 거의 모든 프로그램을 패러디하게 된다. 박명수가 예전에 한번 말
했던 하루종일 TV에 내 얼굴만 나왔으면...을 실제로 실현해 주진 못했다. 제작진이 이
유는 그걸로 위장했지만 현실은 시궁창.모든 편성을 다 패러디하는 것이라서 화면조정부
```

## [각각 링크 페이지를 크롤링하여 제목, 카테고리, 내용 출력]

In [13]:

```
# 크롤링한 데이터를 데이터 프레임으로 만들기 위해 준비
columns = ["title", "category", "content_text"]
df = pd.DataFrame(columns=columns)

#for page_url in page_urls:
for i in range(10):
    # 윈도우용 크롬 웹드라이버 실행 경로 (Windows) 지정
    excutable_path = "chromedriver.exe"
    driver = webdriver.Chrome(executable_path=excutable_path)
    # 크롬 드라이버를 통해 page_urls[0]번째 사이트의 HTML 문서 가져옴
    #driver.get(page_url)  # page_urls[i],  page_url의 정보를 가져옴
    driver.get(page_urls[i])  # page_urls[i],  page_url의 정보를 가져옴
    req = driver.page_source # 페이지 소스를 req에 저장
    soup = BeautifulSoup(req, 'html.parser') # html.parser로 파싱
    contents_table = soup.find(name="article") #  불러온 소스에서 태그명이 article인 요소 하나만 추출

    ### 타이틀 추출
    title = contents_table.find_all('h1')[0] # 태그명이 h1인 모든 태그 추출, article h1
    if title is not None:
        row_title = title.text.replace("₩n", " ")
    else:
        row_title = ""

    ### 카테고리 추출
    # 카테고리 정보가 없는 경우를 확인합니다.
    if len(contents_table.find_all("ul")) > 0: # article ul 로 검색한 결과 여러 ul 결과가 나올 경우
        category = contents_table.find_all("ul")[0] # 제일 첫번째 article ul 을 category로 설정
    else:
        category = None

    if category is not None:
        row_category = category.text.replace("₩n", " ")
    else:
        row_category = ""

    ### 내용 추출
    #contents_table.find_all(name="div", attrs={"class":"wiki-paragraph"})
    #div  태그 중  class 속성값이  wiki-paragraph인 요소를 추출
    content_paragraphs = contents_table.select("div.wiki-paragraph")
    #  내용으로 추출한 리스트를 하나의 문자열로 전처리
    content_corpus_list = [] # 내용 중 텍스트만 담을 빈 리스트 생성

    # content_paragraphs 리스트의 값을 순서대로 paragraphs에 대입
    if content_paragraphs is not None:
        for paragraphs in content_paragraphs:
            if paragraphs is not None:
                content_corpus_list.append(paragraphs.text.replace("₩n", " "))
            else:
                content_corpus_list.append("")
    else:
        content_corpus_list.append("")

    # 모든 정보를 하나의 데이터 프레임에 저장하기 위해서 시리즈 생성
    # 각 페이지의 정보를 추출하여 제목, 카테고리, 내용 순으로 행을 생성
    row = [row_title, row_category, "".join(content_corpus_list)]
    # 시리즈로 만듦
    series = pd.Series(row, index=df.columns)
    # 데이터 프레임에 시리즈를 추가, 한 페이지 당 하나의 행 추가
    df = df.append(series, ignore_index=True)
```

```
    # 크롤링에 사용한 브라우저를 종료합니다.
    driver.close()
```

In [14]:

```
# 데이터 프레임을 출력합니다.
df
```

Out[14]:

| | title | category | content_text |
|---|---|---|---|
| **0** | 무한도전 TV | 무한도전/2009년로그 누락 문서 | 무한도전의 역대 에피소드품절남 특집→무한도전 TV→무한도전 벼농사 특집문서가 존재하... |
| **1** | 분류:블랙스타 -Theater Starless- | 2019년 모바일 게임리듬 게임일본 게임 | 이 분류에 대한 설명은 블랙스타 -Theater Starless- 문서나 문... |
| **2** | 아스타로스 | 동음이의어소울 칼리버 시리즈/등장인물 | 은(는) 여기로 연결됩니다. 마계왕자의 아스타로스에 대한 내용은 아스타로스(마... |
| **3** | Good Times Bad Times | 1968년 노래1969년 싱글록하드 록프로토 메탈헤비메탈레드 제플린 노래 | Led ZeppelinTrack listingLed ZeppelinTrack lis... |
| **4** | 2022 베이징 동계올림픽 | 2022 베이징 동계올림픽 | 은(는) 여기로 연결됩니다. (한시적 넘겨주기) 대규모 외교적 보이콧에 대한 ... |
| **5** | 모스 테크놀로지 6502 | 중앙처리장치/제품1975년 출시명령어 집합코모도어 | 명령어 집합CISCAMD64 x86 · M68K · VAX CISC · MOS 65... |
| **6** | 김성회(방송인) | 샌드박스 네트워크/소속 크리에이터1978년 출생은평구 출신 인물대한민국의 남성 방송... | 로그인 후 편집 가능한 문서입니다.김성회金聖會출생1978년 12월 13일 (4... |
| **7** | 워글 | 5세대 포켓몬노말타입 포켓몬비행타입 포켓몬에스퍼타입 포켓몬리전 폼 | 포켓몬 도감 나열 순서 626 버프론 627 수리둥보 628 워글 629 벌차이지역... |
| **8** | 유희왕 마스터 듀얼/Live2D가 적용된 카드들 | 유희왕 마스터 듀얼 | 1. 개요2. ㄱ3. ㄴ4. ㄷ5. ㄹ6. ㅁ7. ㅂ8. ㅅ9. ㅇ10. ㅈ11. ... |
| **9** | 딥 임팩트(말) | 죽은 경주마2002년 출생2019년 죽은 동물 | 딥 임팩트 관련 틀 [ 펼치기 · 접기 ] JRA 경마의 전당 현창마 [ 펼치기 ... |

# [명사만을 추출하여 워드 클라우드 그리기]

## [코엔엘파이(konlpy)를 이용한 형태소 분석]

- 품사란 단어를 기능, 형태, 의미에 따라 나눈 갈래

| 분류 기준 | 형태 변화에 따라 | 기능에 따라 | 공통된 의미에 따라 |
|---|---|---|---|
| 단어 | 불변어 | 체언 | 명사 |
| | | | 대명사 |
| | | | 수사 |
| | | 수식언 | 관형사 |
| | | | 부사 |
| | | 관계언 | 조사 |
| | | 독립언 | 감탄사 |
| | 가변어 | 용언 | 동사 |
| | | | 형용사 |

**[형태소 분석과 품사태깅]**

- 형태소 : 더 이상 분리를 할 수 없는 의미를 갖는 최소 단어를 의미
- 형태소 분석 : 형태소를 비롯하여, 어근,접두사/접미사, 품사(POS,part-of-search)등 다양한 언어적 속성의 구조를 파악하는 것
- 
  - 품사태깅 : 형태소와 품사를 매칭시키는 것

**[빈도 분석 : 문장 형태소 분석 - KoNLPy]**

- KoNLPy : 파이썬 한국어 형태소 분석 라이브러리

# <Step2. 추출> : 키워드 추출

## [텍스트 데이터 전처리] 정규식을 사용하여 한글과 띄어쓰기만 가져오기

**파이썬 정규표현식(re) 사용법**

- 정규표현식 : 컴파일 => re.compile, 컴파일을 미리 해 두고 이를 저장
- 정규표현식 : 치환 => re.subn(pattern,repl,string,count,flags)

#한글 코드 범위

#ㄱ~ㅎ : 0x3131~0x314e

#ㅏ~ㅣ : 0x314f~0x3163

#가~힣 : 0xac00~0xd7a3

In [15]:

```
import re

#사용자 정의 함수 선언
def text_cleaning(text):
    hangul=re.compile('[^ ㄱ-ㅣ가-힣]+')#[^ㄱ-ㅣ가-힣+] 한글과 띄어쓰기의 정규식 패턴 ^ => [안에 범위
    result=hangul.sub('',text) #한글과 띄어쓰기를 제외한 모든 글자 패턴을 ''빈 문자로 치환
    return result

print(text_cleaning('반갑습니다'))
```

반갑습니다

In [16]:

```
print(text_cleaning(df['title'][0]))
```

무한도전

In [17]:

```
print(text_cleaning(df['category'][0]))
```

무한도전년로그 누락 문서

In [18]:

```
print(text_cleaning(df['content_text'][0]))
```

무한도전의 역대 에피소드품절남 특집무한도전 무한도전 벼농사 특집문서가 존재하는 무한도전 특집 이 외의 특집은 해당 문서 참조 개요 프로그램 목록 화면조정시간 뉴스투데이 지구촌 리포트 경제매거진 무한도전 마이너 특선영화 취권 찾아라 맛있는 무릎팍도사 밥줘 쇼 음악중심 노안선발대회 정준하의 푸드 이상형 월드컵 뉴스데스크 스포츠뉴스 세바퀴 특선대작 스타워즈 애국가 기타무한도전 는 방송위원회의 심의 규정을 준수하려고 합니다만무한도전 회 에피소드로 년 월 일과 일 회에 걸쳐 방영 의 추석 당시 방영한 프로그램들을 콩트 형식으로 패러디한 특집이자 무한도전 급 감성의 마지막 특집이기도 하다 무한도전 제작진들이 자체적으로 만든 추석특집 편성표에 따라 멤버들이 그대로 찍는 것 자연스럽게 에서 방영하는 거의 모든 프로그램을 패러디하게 된다 박명수가 예전에 한번 말했던 하루종일 에 내 얼굴만 나왔으면을 실제로 실현해 주진 못했다 제작진이 이유는 그걸로 위장했지만 현실은 시궁창모든 편성을 다 패러디하는 것이라서 화면조정부터 음악방송 특선영화에 애국가까지 무도 멤버들이 제작했다 과거에 기자들이 이걸 가지고 프로그램을 모두 무한도전이 채운다라고 부풀려서 기사를 띄운 적 있는데 실상은 무도 주 분량에서 조금씩만 내보내는 것 실제로 이 분량을 채우기 위해 평소엔 하지 않던 메이킹 필름 중간삽입까지 동원하면서 분량 불려먹기를 하는 등 분량 부족에 고생하는 모습을 보였다분량이 짧았던 터라 방송을 그대로 재현한 퀄리티는 보여주지 못했고 상당수를 콩트 형식으로 패러디했다 취권의 경우 기준 낮은 개그의 평은 호불호가 갈렸고 눈 버리는 장면이 자주 나와서 황금시간대에 채널을 돌린 시청자들이 많았다는 이야기가 있다 이외에도 주간 시청률이 를 맴돌며 동시간대 시청률 위 자리를 놀라운

In [19]:

```
#각 피처마다 데이터 전처리 적용 한글과 띄어쓰기를 제외한 모든 부분을 제거
df['title']=df['title'].apply(lambda x : text_cleaning(x))
df['category']=df['category'].apply(lambda x : text_cleaning(x))
df['content_text']=df['content_text'].apply(lambda x : text_cleaning(x))
df.head()
```

Out[19]:

| | title | category | content_text |
|---|---|---|---|
| **0** | 무한도전 | 무한도전년로그 누락 문서 | 무한도전의 역대 에피소드품절남 특집무한도전 무한도전 벼농사 특집문서가 존재하는 무한... |
| **1** | 분류블랙스타 | 년 모바일 게임리듬 게임일본 게임 | 이 분류에 대한 설명은 블랙스타 문서나 문서를 참고하십시오 |
| **2** | 아스타로스 | 동음이의어소울 칼리버 시리즈등장인물 | 은는 여기로 연결됩니다 마계왕자의 아스타로스에 대한 내용은 아스타로스마계왕자 문... |
| **3** | | 년 노래년 싱글록하드 록프로토 메탈헤비메탈레드 제플린 노래 | 펼치기 접기 트랙곡명러닝타임 트랙곡명러닝타임 ... |
| **4** | 베이징 동계올림픽 | 베이징 동계올림픽 | 은는 여기로 연결됩니다 한시적 넘겨주기 대규모 외교적 보이콧에 대한 내용은 베... |

In [20]:

```
#각 피처마다 말뭉치를 생성
title_corpus="".join(df['title'].tolist())
category_corpus="".join(df['category'].tolist())
content_corpus="".join(df['content_text'].tolist())
print(title_corpus)
print(category_corpus)
print(content_corpus)
```

```
무한도전  분류블랙스타  아스타로스  베이징 동계올림픽 모스 테크놀로지  김성회 방송인 워글 유희왕 마스터 듀얼가 적용된 카드들 딥 임팩트말
무한도전년로그 누락 문서년 모바일 게임리듬 게임일본 게임동음이의어소울 칼리버 시리즈등장인물년 노래년 싱글록하드 록프로토 메탈헤비메탈레드 제플린 노래 베이징 동계올림픽중앙처리장치제품년 출시명령어 집합코모도어샌드박스 네트워크소속 크리에이터년 출생은평구 출신 인물대한민국의 남성 방송인대한민국의 게임 개발자유튜버대한민국의 무종교인중앙대학교 출신게임 리뷰어게임 콘텐츠 유튜버안동 김씨세대 포켓몬노말타입 포켓몬비행타입 포켓몬에스퍼타입 포켓몬리전 폼유희왕 마스터 듀얼죽은 경주마년 출생년 죽은 동물
무한도전의 역대 에피소드품절남 특집무한도전 무한도전 벼농사 특집문서가 존재하는 무한도전 특집 이 외의 특집은 해당 문서 참조 개요 프로그램 목록 화면조정시간  뉴스투데이 지구촌 리포트 경제매거진  무한도전 마이너 특선영화 취권 찾아라 맛있는  무릎팍도사 밥줘 쇼 음악중심 노안선발대회 정준하의 푸드 이상형 월드컵  뉴스데스크 스포츠뉴스 세바퀴 특선대작 스타워즈 애국가 기타무한도전 는 방송위원회의 심의 규정을 준수하려고 합니다만무한도전  회 에피소드로 년 월 일과 일 회에 걸쳐 방영 의 추석 당시 방영한 프로그램들을 콩트 형식으로 패러디한 특집이자 무한도전 급 감성의 마지막 특집이기도 하다 무한도전 제작진들이 자체적으로 만든 추석특집 편성표에 따라 멤버들이 그대로 찍는 것 자연스럽게 에서 방영하는 거의 모든 프로그램을 패러디하게 된다 박명수가 예전에 한번 말했던 하루종일 에 내 얼굴만 나왔으면을 실제로 실현해 주진 못했다
```

In [21]:

```
from konlpy.tag import Okt
from collections import Counter
#konlpy의 형태소 분석기로 명사 단위의 키워드를 추출합니다.
nouns_tagger=Okt()
nouns=nouns_tagger.nouns(content_corpus)
count=Counter(nouns)
count
```

Out[21]:

```
Counter({'무한도전': 23,
         '역대': 11,
         '에피소드': 5,
         '품절남': 1,
         '특집': 25,
         '벼농사': 2,
         '문서': 185,
         '존재': 17,
         '이': 121,
         '외': 14,
         '해당': 11,
         '참조': 4,
         '개요': 8,
         '프로그램': 16,
         '목록': 6,
         '화면': 7,
         '조정': 6,
         '시간': 23,
```

In [22]:

```
#한 글자 키워드를 제거합니다.
#count 딕셔너리를 돌면서 길이가 1보다 큰것만 remove_char_counter에 저장.
remove_char_counter = Counter({x : count[x] for x in count if len(x)>1})
print(remove_char_counter)
```

```
Counter({'문단': 200, '게임': 195, '문서': 185, '이전': 164, '역사': 162, '아스
타': 130, '로스': 129, '영상': 89, '임팩트': 85, '심볼': 79, '김성회': 75, '베이
징': 71, '부분': 69, '때문': 66, '워글': 66, '스피드': 64, '방송': 63, '자신': 56,
'중국': 54, '사용': 51, '올림픽': 50, '출연': 48, '접기': 48, '동계올림픽': 48,
'이후': 47, '대한': 47, '라이트': 47, '내용': 45, '전승': 44, '사쿠라': 43, '캐릭
터': 42, '보이': 42, '일본': 42, '스케이팅': 42, '정도': 41, '포켓몬': 41, '진행':
40, '이상': 40, '사건': 40, '기술': 40, '종목': 40, '하나': 39, '블랙': 39, '박명
수': 38, '스테이크': 38, '비트': 38, '대회': 37, '위해': 37, '관련': 36, '엣지': 3
6, '개발자': 36, '마이어': 36, '노홍철': 35, '수리둥보': 35, '나리타': 35, '히카
루': 35, '정준하': 34, '파괴': 34, '승도': 34, '공격': 33, '기념': 33, '아이': 33,
'히카리': 33, '루돌프': 33, '골드': 32, '모습': 31, '유재석': 31, '스키': 31, '브
라이언': 31, '스타': 30, '또한': 29, '다른': 29, '대해': 29, '정보': 29, '본인': 2
9, '경우': 28, '시작': 28, '에스': 28, '오페라': 28, '당시': 27, '광고': 27, '사
토': 27, '히데': 27, '카츠': 27, '문제': 26, '한국': 26, '오르페브르': 26, '니노':
26, '특집': 25, '등장': 25, '경기': 25, '경마': 25, '세이': 25, '티엠': 25, '콘트
레일': 25, '무스': 24, '다년': 24, '마법사': 24, '무한도전': 23, '시간': 23, '참
고': 23, '이름': 23, '미스터': 23, '이미지': 23, '성기사': 23, '에이스': 23, '우
승': 23, '메이지': 23, '메이': 23, '통해': 22, '도전': 22, '쇼트트랙': 22, '클래
식': 22, '레지스터': 22, '특성': 22, '지방': 22, '치카라': 22, '시비': 22, '어드':
```

In [23]:

```
#한국어 약식 불용어 사전 예시 파일
korean_stopwords_path = 'stopwords.txt'

#텍스트 파일 오픈
with open(korean_stopwords_path,encoding='utf8') as f:
    stopwords = f.readlines()
stopwords=[x.strip() for x in stopwords]
print(stopwords[:10])

#불용어 추가
namu_wiki_stopwords=['상위','문서','내용','누설','아래','해당','설명','표기','추가','모든',
                     '사용','매우','가장','줄거리','요소','상황', '편집', '틀', '경우',
                     '때문', '모습', '정도', '이후', '사실', '생각', '인물', '이름','년월']
for stopword in namu_wiki_stopwords:
    stopwords.append(stopword)

remove_char_counter = Counter({x : remove_char_counter[x] for x in count if x not in stopwords})
remove_char_counter = Counter({x : count[x] for x in count if len(x)>1})
print(remove_char_counter)
```

```
['아', '휴', '아이구', '아이쿠', '아이고', '어', '나', '우리', '저희', '따라']
Counter({'문단': 200, '게임': 195, '문서': 185, '이전': 164, '역사': 162, '아스
타': 130, '로스': 129, '영상': 89, '임팩트': 85, '심볼': 79, '김성회': 75, '베이
징': 71, '부분': 69, '때문': 66, '워글': 66, '스피드': 64, '방송': 63, '자신': 56,
'중국': 54, '사용': 51, '올림픽': 50, '출연': 48, '접기': 48, '동계올림픽': 48,
'이후': 47, '대한': 47, '라이트': 47, '내용': 45, '전승': 44, '사쿠라': 43, '캐릭
터': 42, '보이': 42, '일본': 42, '스케이팅': 42, '정도': 41, '포켓몬': 41, '진행':
40, '이상': 40, '사건': 40, '기술': 40, '종목': 40, '하나': 39, '블랙': 39, '박명
수': 38, '스테이크': 38, '비트': 38, '대회': 37, '위해': 37, '관련': 36, '엣지': 3
6, '개발자': 36, '마이어': 36, '노홍철': 35, '수리둥보': 35, '나리타': 35, '히카
루': 35, '정준하': 34, '파괴': 34, '승도': 34, '공격': 33, '기념': 33, '아이': 33,
'히카리': 33, '루돌프': 33, '골드': 32, '모습': 31, '유재석': 31, '스키': 31, '브
라이언': 31, '스타': 30, '또한': 29, '다른': 29, '대해': 29, '정보': 29, '본인': 2
9, '경우': 28, '시작': 28, '에스': 28, '오페라': 28, '당시': 27, '광고': 27, '사
토': 27, '히데': 27, '카츠': 27, '문제': 26, '한국': 26, '오르페브르': 26, '니노':
26, '특집': 25, '등장': 25, '경기': 25, '경마': 25, '세이': 25, '티엠': 25, '콘트
레일': 25, '무스': 24, '다년': 24, '마법사': 24, '무한도전': 23, '시간': 23, '참
고': 23, '이름': 23, '미스터': 23, '이미지': 23, '성기사': 23, '에이스': 23, '우
승': 23, '메이지': 23, '메이': 23, '통해': 22, '도전': 22, '쇼트트랙': 22, '클래
```

In [82]:

```
import random
import pytagcloud
import webbrowser
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
from collections import Counter
from PIL import Image
```

In [79]:

```
#가장 출현 빈도수가 높은 40개의 단어 선정
ranked_tags=remove_char_counter.most_common(40)

#pytagcloud로 출력할 40개의 단어 입력 최대크기는 80으로 제한
taglist=pytagcloud.make_tags(ranked_tags,maxsize=80)

#pytagcloud 이미지를 생성합니다.
pytagcloud.create_tag_image(taglist,'cloud_img.jpg',size=(900,600),fontname='NanumGothic',rectangula

from IPython.display import Image
Image(filename='cloud_img.jpg')
```

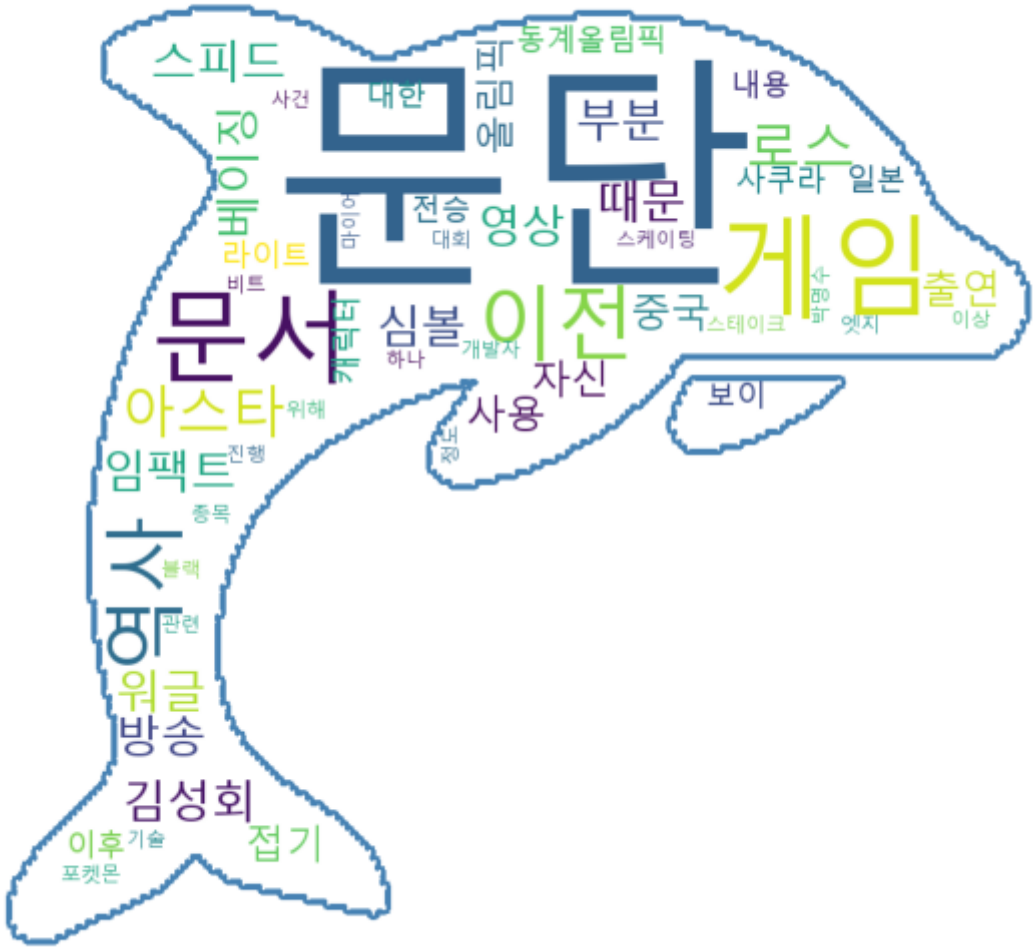
Out[79]:

In [88]:

```
from PIL import Image
img=Image.open('./crap_img.png')
img_array=np.array(img)

wc=WordCloud(font_path='malgun',width=800,height=700,scale=4.0,max_font_size=250,mask=img_array,bac
             contour_width=3,contour_color='red')
gen=wc.generate_from_frequencies(remove_char_counter)

plt.figure(figsize=(20, 10))
plt.imshow(gen)
plt.axis('off')
plt.savefig('word_crap_img.jpg')
```

In [84]:

```
from PIL import Image
img=Image.open('./dolphin_img.png')
img_array=np.array(img)

wc=WordCloud(font_path='malgun',width=800,height=700,scale=4.0,max_font_size=250,mask=img_array,bac
             contour_width=3,contour_color='steelblue')
gen=wc.generate_from_frequencies(remove_char_counter)

plt.figure(figsize=(20, 10))
plt.imshow(gen)
plt.axis('off')
plt.savefig('word_dolphin_img.jpg')
```



In [94]:

```
from PIL import Image
alice_coloring=np.array(Image.open("alice_img.png"))
wc=WordCloud(font_path='malgun',background_color='white',max_words=2000,
             mask=alice_coloring, max_font_size=40,random_state=42)
gen=wc.generate_from_frequencies(remove_char_counter)
from wordcloud import ImageColorGenerator
imge_colors=ImageColorGenerator(alice_coloring)
plt.figure(figsize=(12,12))
plt.imshow(wc.recolor(color_func=imge_colors),interpolation='bilinear')
plt.axis('off')
plt.show()
```

In [ ]: