

# 데이터 크롤링과 정제

---

## 7장. 한국어 형태소 분석

# 목차

---

- 자연어 처리
  - 설치 라이브러리
  - 사용 방법
- 크롤링 및 Wordcloud 생성

# 한글 자연어 처리 라이브러리:KoNLPy

---

- 자연어 처리: Natural Language Processing(NLP)
  - 자연어: 우리가 일상 생활에서 사용하는 언어
  - 자연어 처리: 자연어의 의미를 분석하여 컴퓨터가 처리할 수 있도록 하는 일
- NLTK
  - 파이썬 패키지 (아나콘다 패키지에 포함)
  - 영어 텍스트 처리
- 한국어 자연어 처리
  - KoNLPy (코엔엘파이) 포함 모듈들
    - <https://konlpy-ko.readthedocs.io/ko/v0.4.3/>
    - Hannanum (한나눔): KAIST
    - Kkma(꼬꼬마): 서울대학교 IDS 연구실 개발
    - Komoran(코모란): Shineware 개발
    - Mecab(메카브): 일본어용 형태소 분석기를 한국어에 사용하도록 수정
    - Open Korean Text(Oket): 오픈 소스 한국어 분석기
      - Twitter에서 이름 변경(과거 트위터 형태소 분석기)

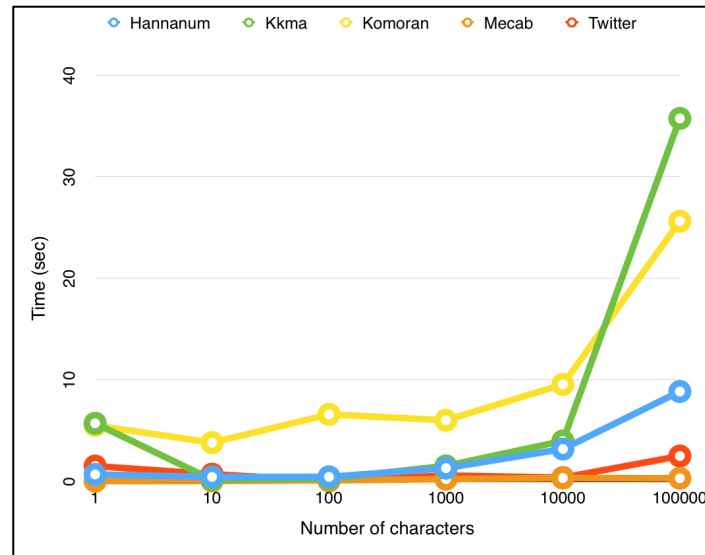
# KoNLPy 성능 비교

## ■ 로딩 시간

- 사전 로딩을 포함하여 클래스를 로딩하는 시간
  - Kkma: 5.6988 secs
  - Komoran: 5.4866 secs
  - Hannanum: 0.6591 secs
  - Okt(Twitter): 1.4870 secs
  - Mecab: 0.0007 secs

## ■ 실행 시간

- 10만 문자의 문서를 대상으로 각 클래스의 pos 메소드를 실행하는데 소요되는 시간

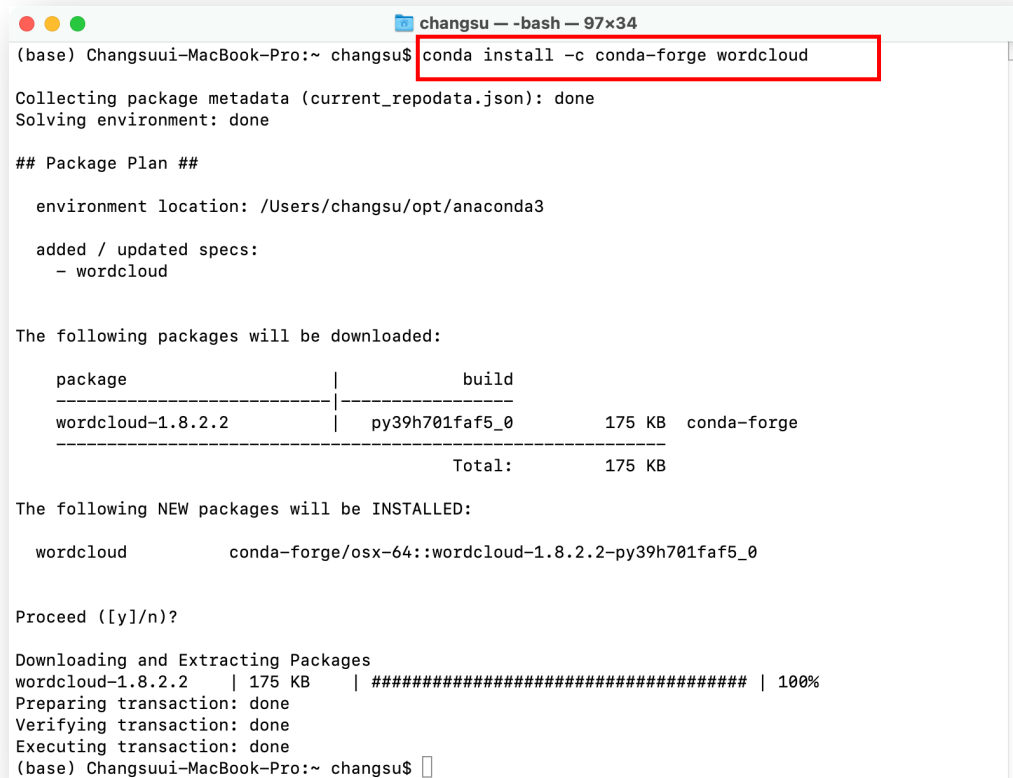


# wordcloud 라이브러리 설치

- anaconda 터미널 창에서 아래 명령어 실행

```
conda install -c conda-forge wordcloud
```

- -c conda-forge
  - -c 옵션: channel
  - conda-forge: anaconda에서 쉽게 설치할 수 있도록 검증된 파이썬 패키지들을 모아 놓은 채널



```
changsu ~ -bash - 97x34
(base) Changsuui-MacBook-Pro:~ changsu$ conda install -c conda-forge wordcloud

Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/changsu/opt/anaconda3

added / updated specs:
- wordcloud

The following packages will be downloaded:

package | build | size | channel
-----|-----|-----|-----
wordcloud-1.8.2.2 | py39h701faf5_0 | 175 KB | conda-forge

Total: 175 KB

The following NEW packages will be INSTALLED:

wordcloud conda-forge/osx-64::wordcloud-1.8.2.2-py39h701faf5_0

Proceed ([y]/n)?

Downloading and Extracting Packages
wordcloud-1.8.2.2 | 175 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(base) Changsuui-MacBook-Pro:~ changsu$
```

# konlpy 라이브러리 설치 #1

## ■ jpype1 라이브러리 설치

```
conda install -c conda-forge jpype1
```

– Python에서 Java 클래스 호출 라이브러리

```
changsu — -bash — 97x34
(base) Changsuui-MacBook-Pro:~ changsu$ conda install -c conda-forge jpype1
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/changsu/opt/anaconda3

  added / updated specs:
    - jpype1

The following packages will be downloaded:



| package      | build          |        |
|--------------|----------------|--------|
| jpype1-1.3.0 | py39haf03e11_0 | 368 KB |
| Total:       |                | 368 KB |



The following NEW packages will be INSTALLED:

  jpype1          pkgs/main/osx-64::jpype1-1.3.0-py39haf03e11_0

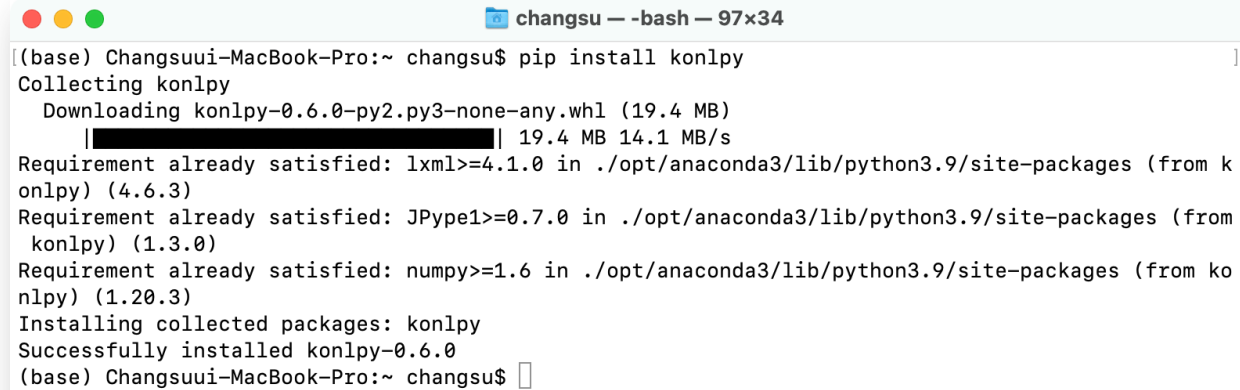
Proceed ([y]/n)? y

Downloading and Extracting Packages
jpype1-1.3.0      | 368 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(base) Changsuui-MacBook-Pro:~ changsu$
```

# konlpy 라이브러리 설치 #2

- konlpy 라이브러리 설치

```
pip3 install konlpy
```



A terminal window titled 'changsu - bash - 97x34' showing the command 'pip install konlpy' being executed. The output shows the package being collected, downloaded (19.4 MB at 14.1 MB/s), and installed successfully. Dependencies like lxml, JPype1, and numpy are also listed as already satisfied.

```
changsu$ pip install konlpy
Collecting konlpy
  Downloading konlpy-0.6.0-py2.py3-none-any.whl (19.4 MB)
    |████████████████████| 19.4 MB 14.1 MB/s
Requirement already satisfied: lxml>=4.1.0 in ./opt/anaconda3/lib/python3.9/site-packages (from k
onlpy) (4.6.3)
Requirement already satisfied: JPype1>=0.7.0 in ./opt/anaconda3/lib/python3.9/site-packages (from
konlpy) (1.3.0)
Requirement already satisfied: numpy>=1.6 in ./opt/anaconda3/lib/python3.9/site-packages (from ko
nlpy) (1.20.3)
Installing collected packages: konlpy
Successfully installed konlpy-0.6.0
(base) Changsuui-MacBook-Pro:~ changsu$
```

# KoNLPy 사용법

## ■ Okt (Twitter) class

- **morphs**(텍스트)
  - 텍스트에서 형태소를 반환
- **nouns**(텍스트)
  - 텍스트에서 명사만 반환
- **phrases**(텍스트)
  - 텍스트에서 어절을 반환
- **pos**(텍스트, [norm=False, stem=False])
  - 텍스트에서 품사 정보를 부착하여 반환 (Part-Of-Speech tagging)
  - 각 형태소를 품사와 함께 리스트로 반환
  - norm=False: 정규화 여부
    - 같은 의미이면서 표현이 다른 단어를 통합
  - stem=False: 어간 찾기 여부
    - 단어의 의미를 담고 있는 단어의 핵심 부분 추출

형태소: 뜻을 가진 가장 작은 말의 단위

– ‘책가방’: ‘책’, ‘가방’이 형태소

어간: 활용어가 활용할 때 변하지 않는 부분

– ‘보다’의 경우, 보았다, 보니, 보고 등으로 활용

– 어간은 ‘보’가 됨

어절: 문장을 구성하는 각각의 마디 (띄어쓰기 단위)



# pos(norm, stem) 사용

## ■ pos(norm, stem) 사용 비교

<okt\_00.py>

```
from konlpy.tag import Okt
okt = Okt()
```

```
list1 = okt.pos("아버지 가방에 들어가신다.", norm=True, stem=True)
list2 = okt.pos("아버지 가방에 들어가신다.", norm=False, stem=False)
print(list1)
print(list2)
```

stem=True  
- 단어의 어간을 리턴

```
word1 = okt.pos("그래요ㅋㅋ?", norm=True, stem=True)
word2 = okt.pos("그래옉ㅋㅋ?", norm=False, stem=True)
word3 = okt.pos("그래옉ㅋㅋ?", norm=True, stem=False)
print(word1)
print(word2)
print(word3)
```

norm=True  
- “그래옉”의 다른 형태인 “그래요” 리턴

```
[('아버지', 'Noun'), ('가방', 'Noun'), ('에', 'Josa'), ('들어가다', 'Verb'), ('.', 'Punctuation')]
[('아버지', 'Noun'), ('가방', 'Noun'), ('에', 'Josa'), ('들어가신다', 'Verb'), ('.', 'Punctuation')]
[('그렇다', 'Adjective'), ('ㅋㅋ', 'KoreanParticle'), ('?', 'Punctuation')]
[('그래옉', 'Noun'), ('ㅋㅋ', 'KoreanParticle'), ('?', 'Punctuation')]
[('그래요', 'Adjective'), ('ㅋ', 'KoreanParticle'), ('?', 'Punctuation')]
```

# Okt 간단 예제 #1

```
from konlpy.tag import Okt
```

<okt\_01.py>

```
okt = Okt() # Open Korean Text (과거 트위터 형태소 분석기)
```

```
text = "마음에 꽃힌 칼한자루 보다 마음에 꽃힌 꽃한송이가 더 아파서 잠이 오지 않는다"
```

```
# pos(text): 문장의 각 품사를 태깅
```

```
# norm=True: 문장을 정규화, stem=True: 어간을 추출
```

```
okt_tags = okt.pos(text, norm=True, stem=True)
```

```
print(okt_tags)
```

```
# nouns(text): 명사만 리턴
```

```
okt_nouns = okt.nouns(text)
```

```
print(okt_nouns)
```

```
[('마음', 'Noun'), ('에', 'Josa'), ('꽃히다', 'Verb'), ('칼', 'Noun'), ('한', 'Determiner'), ('자루', 'Noun'),  
('보다', 'Verb'), ('마음', 'Noun'), ('에', 'Josa'), ('꽃히다', 'Verb'), ('꽃', 'Noun'), ('한송이', 'Noun'), ('가',  
'Josa'), ('더', 'Noun'), ('아프다', 'Adjective'), ('잠', 'Noun'), ('이', 'Josa'), ('오지', 'Noun'), ('않다',  
'Verb')]
```

```
['마음', '칼', '자루', '마음', '꽃', '한송이', '더', '잠', '오지']
```

# Okt 예제 #2

<okt\_02.py>

```
from konlpy.tag import Okt

text = """나랏말이 중국과 달라 한자와 서로 통하지 아니하므로,
    우매한 백성들이 말하고 싶은 것이 있어도 마침내 제 뜻을 잘 표현하지 못하는 사람이 많다.
    내 이를 딱하게 여기어 새로 스물여덟 자를 만들었으니,
    사람들로 하여금 쉬 익히어 날마다 쓰는 데 편하게 할 뿐이다."""

okt = Okt()

# morphs(text): 텍스트를 형태소 단위로 나눔
okt_morphs = okt.morphs(text)
print('morphs():\n', okt_morphs)

# 명사만 추출
okt_nouns = okt.nouns(text)
print('nouns():\n', okt_nouns)

# phrases(text): 어절 추출
okt_phrases = okt.phrases(text)
print('phrases():\n', okt_phrases)

# pos(text): 품사를 태깅
okt_pos = okt.pos(text)
print('pos():\n', okt_pos)
```

# Okt 예제 #2 실행 결과

morphs():

['나랏말', '이', '중국', '과', '달라', '한자', '와', '서로', '통', '하지', '아니하므로', ',', '우매', '한', '백성', '들', '이', '말', '하고', '싶은', '것', '이', '있어도', '마침내', '제', '뜻', '을', '잘', '표현', '하지', '못', '하는', '사람', '이', '많다', '.', '내', '이름', '딱하게', '여기어', '새로', '스물', '여덟', '자를', '만들었으니', ',', '사람', '들', '로', '하여금', '쉬', '익히어', '날', '마다', '쓰는', '데', '편하게', '할', '뿐', '이다', '.']

nouns():

['나랏말', '중국', '달라', '한자', '서로', '통', '우매', '백성', '말', '것', '마침내', '제', '뜻', '표현', '사람', '내', '스물', '여덟', '사람', '쉬', '날', '데', '뿐']

phrases():

['나랏말', '중국', '중국과 달라', '중국과 달라 한자', '중국과 달라 한자와 서로', '중국과 달라 한자와 서로 통', '우매', '백성들', '마침내', '마침내 제', '마침내 제 뜻', '표현', '못하는 사람', '스물여덟', '사람들', '달라', '한자', '서로', '사람', '스물', '여덟']

pos():

[('나랏말', 'Noun'), ('이', 'Josa'), ('중국', 'Noun'), ('과', 'Josa'), ('달라', 'Noun'), ('한자', 'Noun'), ('와', 'Josa'), ('서로', 'Noun'), ('통', 'Noun'), ('하다', 'Verb'), ('아니다', 'Adjectiveon'), ('우매', 'Noun'), ('한', 'Josa'), ('백성', 'Noun'), ('들', 'Suffix'), ('이', 'Josa'), ('말', 'Noun'), ('하고', 'Josa'), ('싶다', 'Verb'), ('것', 'Noun'), ('이', 'Josa'), ('있다', 'Adjective'), ('제', 'Noun'), ('뜻', 'Noun'), ('을', 'Josa'), ('자다', 'Verb'), ('표현', 'Noun'), ('하다', 'Verb'), ('못', 'VerbPrefix'), ('하다', 'Verb'), ('사람', 'Noun'), ('이', 'Josa'), ('많다', 'Adjectivuation'), ('\n', 'Foreign'), ('내', 'Noun'), ('이르다', 'Verb'), ('딱하다', 'Adjective'), ('여기다', 'Verb'), ('새롭다', 'Adjective'), ('스물', 'Noun'), ('여덟', 'Noun'), ('자르다', 'Verb'), (' ', 'Punctuation'), ('사람', 'Noun'), ('들', 'Suffix'), ('로', 'Josa'), ('하여금', 'Adverb'), ('쉬', 'Noun'), ('익히다', 'Verb'), ('날', 'Noun'), ('마다', 'Josa'), ('쓰다', 'Verb'), ('데', 'Noun'), (' ', 'Punctuation'), ('하다', 'Verb'), ('뿐', 'Noun'), ('이다', 'Josa'), ('.', 'Punctuation')]

# 예제: 단어 분석 및 Word Cloud 생성 #1

<wordcloud\_01.py>

```
from wordcloud import WordCloud
from konlpy.tag import Okt
from collections import Counter
import matplotlib.pyplot as plt
import platform
import numpy as np
from PIL import Image

text = open('test.txt', encoding='utf-8').read()
okt = Okt() # Open Korean Text 객체 생성

# okt함수를 통해 읽어들이는 내용의 형태소를 분석한다.
sentences_tag = []
sentences_tag = okt.pos(text)

noun_adj_list = []
# tag가 명사이거나 형용사인 단어들만 noun_adj_list에 넣어준다.
for word, tag in sentences_tag:
    if tag in ['Noun', 'Adjective']:
        noun_adj_list.append(word)

print(noun_adj_list)
# 가장 많이 나온 단어부터 50개를 저장한다.
counts = Counter(noun_adj_list)
tags = counts.most_common(50)
print(tags)
```

명사와 형용사만 추가

Counter(리스트)

- 리스트 항목의 개수를 딕셔너리 형태로 리턴
- most\_common(n): 가장 많은 수를 가지는 항목 n개 반환

## 예제: 단어 분석 및 Word Cloud 생성 #2

# 한글을 분석하기위해 font를 한글로 지정, macOS는 .otf , window는 .ttf 파일의 위치를 지정

&lt;wordcloud\_01.py&gt;

```
if platform.system() == 'Windows':
    path = r'c:\Windows\Fonts\malgun.ttf'
elif platform.system() == 'Darwin': # Mac OS
    path = r'/System/Library/Fonts/AppleGothic'
else:
    font = r'/usr/share/fonts/truetype/name/NanumMyeongjo.ttf'
```

```
img_mask = np.array(Image.open('cloud.png'))
wc = WordCloud(font_path=path, width=400, height=400,
               background_color="white", max_font_size=200,
               repeat=True,
               colormap='inferno', mask=img_mask)
```

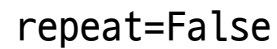
```
cloud = wc.generate_from_frequencies(dict(tags))
# 생성된 WordCloud를 test.jpg로 보낸다.
#cloud.to_file('test.jpg')
plt.figure(figsize=(10, 8))
plt.axis('off')
plt.imshow(cloud)
plt.show()
```

mask를 사용하지 않은  
경우



[('세대', 89), ('소비', 17), ('등', 16), ('이', 14), ('유튜브', 14), ('것', 13), ('명', 12), ('있다', 11), ('수', 11), ('를', 10), ('선호', 10), ('더', 9), ('자신', 9), ('영향', 9), ('크리에이터', 9), ('트렌드', 8), ('점', 8), ('있는', 8), ('중', 8), ('다른', 7), ('특징', 7), ('문화', 7), ('가장', 7), ('취미', 7), ('콘셉트', 7), ('중시', 6), ('현재', 6), ('달리', 6), ('통해', 6), ('브랜드', 6), ('대표', 5), ('같은', 5), ('젊은', 5), ('온라인', 5), ('대비', 5), ('만족', 5), ('편이', 5), ('제품', 5), ('플렉스', 5), ('경우', 4)]

## ■ 실행 결과



## 영문 wordcloud 예제

```
from wordcloud import WordCloud
from wordcloud import STOPWORDS
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
```

```
text = open('alice.txt').read()
STOPWORDS.add('said')
print('STOPWORDS:', STOPWORDS)
```

- 불용어(워드클라우드에서 제외할 단어) 추가

```
img_mask = np.array(Image.open('cloud.png'))
```

```
wordcloud = WordCloud(width=400, height=400,  
                      background_color="white", max_font_size=200,  
                      stopwords=STOPWORDS,  
                      repeat=True,  
                      colormap='inferno', mask=img_mask).generate(text)
```

```
# words_: 객체의 비율 정보가 담긴 딕셔너리 반환
print(wordcloud.words_)
```

```
plt.figure(figsize=(10, 8))
plt.axis('off')
plt.imshow(wordcloud)
plt.show()
```





# 영문 wordcloud 실행 결과

```
STOPWORDS: {"who's", "they'll", 'of', 'it', 'up', 'where', 'again', 'had', 'each', "haven't", 'not', "weren't", 'at', "wasn't", "that's",  
'who', 'why', 'also', "i've", "there's", "she's", "how's", 'no', 'there', 'own', 'with', 'yours', 'all', 'out', 'ourselves', 'this',  
'their', "hasn't", "we'd", 'when', 'because', 'its', 'r', "he's", 'those', 'few', "i'll", 'www', 'so', 'here', 'just', "we've", 'on',  
'which', 'get', "he'd", 'in', 'having', 'should', "they're", 'are', 'whom', 'cannot', 'against', "couldn't", 'otherwise', 'my', "we're",  
'ought', 'before', 'therefore', 'were', "she'll", 'yourself', "we'll", 'but', 'or', 'she', "why's", "won't", 'am', 'through', "they've",  
'hers', 'his', 'by', 'ever', "she'd", 'that', 'than', 'themselves', 'then', "hadn't", 'too', 'other', "mustn't", 'above', 'them', 'is',  
"when's", 'once', 'http', 'as', 'into', 'such', "you're", 'your', 'since', "doesn't", 'like', 'for', "didn't", 'some', 'has', 'was',  
'shall', 'you', 'yourselves', 'the', 'an', "can't", 'did', 'her', "don't", 'we', 'ours', 'over', 'do', 'while', 'does', 'he', 'below',  
'him', 'these', 'about', 'a', "what's", 'to', 'k', 'between', 'doing', "aren't", 'how', 'hence', "isn't", 'any', 'myself', "they'd",  
'theirs', 'down', 'same', 'most', 'com', 'would', 'herself', "where's", 'being', 'however', 'they', 'very', "you'd", "wouldn't",  
'during', 'from', "shan't", 'else', 'be', 'said', 'both', 'off', "i'd", 'until', 'can', "i'm", 'under', 'could', 'further', 'have',  
"it's", "let's", 'me', 'itself', 'more', 'and', 'only', "shouldn't", 'our', 'nor', 'been', "you'll", 'i', 'if', 'what', "here's",  
'himself', "you've", 'after', "he'll"}
```

```
{'Alice': 1.0, 'little': 0.29508196721311475, 'one': 0.27595628415300544, 'know': 0.2459016393442623, 'went': 0.226775956284153, 'thing':  
0.2185792349726776, 'time': 0.2103825136612022, 'Queen': 0.20765027322404372, 'see': 0.1830601092896175, 'King': 0.17486338797814208,  
'well': 0.1721311475409836, 'now': 0.16393442622950818, 'head': 0.16393442622950818, 'began': 0.15846994535519127, 'way':  
0.1557377049180328, 'Hatter': 0.1557377049180328, 'Mock Turtle': 0.15300546448087432, 'say': 0.15027322404371585, 'Gryphon':  
0.15027322404371585, 'think': 0.1448087431693989, 'quite': 0.14207650273224043, 'much': 0.13934426229508196, 'first':  
0.13934426229508196, 'thought': 0.1366120218579235, 'go': 0.1366120218579235, 'come': 0.13114754098360656, 'never': 0.1284153005464481,  
. . . (중간 생략)  
'voice': 0.12568306010928962, 'looked': 0.12295081967213115, 'got': 0.12295081967213115, 'must': 0.12021857923497267, 'Cat':  
0.03551912568306011, 'still': 0.03551912568306011, 'seem': 0.03551912568306011, 'people': 0.03551912568306011, 'behind':  
0.03551912568306011, 'really': 0.03551912568306011, 'grow': 0.03551912568306011, 'far': 0.03551912568306011, 'kept': 0.03551912568306011,  
'used': 0.03551912568306011, 'lesson': 0.03551912568306011, 'always': 0.03551912568306011, 'Dodo': 0.03551912568306011, 'whole':  
0.03551912568306011, 'better': 0.03551912568306011, 'room': 0.03551912568306011, 'gone': 0.03551912568306011, 'remark':  
0.03551912568306011, 'cook': 0.03551912568306011, 'Adventures': 0.03278688524590164, 'CHAPTER': 0.03278688524590164, 'many':  
0.03278688524590164, 'near': 0.03278688524590164, 'among': 0.03278688524590164, 'name': 0.03278688524590164, 'Dinah':  
0.03278688524590164, 'afraid': 0.03278688524590164, 'every': 0.03278688524590164, 'finished': 0.03278688524590164}
```

# 네이버 뉴스 타이틀 Word Cloud 예제 #1

```
from bs4 import BeautifulSoup
import requests
from konlpy.tag import Okt
from collections import Counter
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import time
import platform
import numpy as np
from PIL import Image

def get_titles(start_num, end_num, search_word, title_list):
    # start_num ~ end_num까지 크롤링
    while start_num <= end_num:
        url = ('https://search.naver.com/search.naver?where=news&sm=tab_jum&query={}&start={}'.
              format(search_word, start_num))

        req = requests.get(url)
        time.sleep(1)
        if req.ok: # 정상적인 request 확인
            soup = BeautifulSoup(req.text, 'html.parser')
            news_titles = soup.find_all('a', {'class': 'news_tit'})
            for news in news_titles:
                title_list.append(news['title '])
        start_num += 10
    print('title 개수:', len(title_list))
    print(title_list)
```

<wordcloud\_naver.py>

# 네이버 뉴스 타이틀 Word Cloud 예제 #2

<wordcloud\_naver.py>

```
def make_wordcloud(title_list, stopwords, word_count):
    okt = Okt()
    sentences_tag = []
    # 형태소 분석하여 리스트에 넣기
    for sentence in title_list:
        morph = okt.pos(sentence)
        sentences_tag.append(morph)
        print(morph)
        print('-' * 80)

    noun_adj_list = []
    # 명사와 형용사, 영단어(Alpha)를 리스트에 추가
    for sentence1 in sentences_tag:
        for word, tag in sentence1:
            if tag in ['Noun', 'Adjective', 'Alpha']:
                noun_adj_list.append(word)

    # 형태소별 count
    counts = Counter(noun_adj_list)
    tags = counts.most_common(word_count)
    print('-' * 80)
    print(tags)

    tag_dict = dict(tags)
```

# 검색어 제외 방법 2: dict에서 해당 검색어 제거

```
for stopword in stopwords:
    if stopword in tag_dict:
        tag_dict.pop(stopword)
print(tag_dict)
```

딕셔너리에서 stopwords에  
포함된 Key 삭제

```
if platform.system() == 'Windows':
    path = r'c:\Windows\Fonts\malgun.ttf'
elif platform.system() == 'Darwin': # Mac OS
    path = r'/System/Library/Fonts/AppleGothic'
else:
    path = r'/usr/share/fonts/truetype/name/NanumMyeongjo.ttf'

img_mask = np.array(Image.open('cloud.png'))
wordcloud = WordCloud(font_path=path, width=800, height=600,
                      background_color="white", max_font_size=200,
                      repeat=True,
                      colormap='inferno', mask=img_mask)

cloud = wordcloud.generate_from_frequencies(tag_dict)
plt.figure(figsize=(10, 8))
plt.axis('off')
plt.imshow(cloud)
plt.show()
```

# 네이버 뉴스 타이틀 Word Cloud 예제 #3

<wordcloud\_naver.py>

```
if __name__ == '__main__':  
    search_word = "ChatGPT" # 검색어 지정  
    title_list = []  
    stopwords = [search_word, '데이터'] # wordcloud에서 제외할 단어  
  
    # 1~200번게시글 까지 크롤링  
    get_titles(1, 200, search_word, title_list)  
  
    # 단어 50개까지 wordcloud로 출력  
    make_wordcloud(title_list, stopwords, 50)
```

200개의 뉴스 검색

# 네이버 뉴스 타이틀 Word Cloud 예제 실행 결과

title 개수: 90

['AI로 미국 ETF 투자전략 찾는다... NH증권, 빅데이터 경진대회 개최', "행안부 '빅데이터 경진대회' 개최", "전남도, 빅데이터 활용 공모전서 '섬 여행 플랫폼' 성료", "데이터센터 '코리아 패싱' 시작됐다...韓 발길 돌리는 빅테크", . . .]

pos(): 'AI', 'Alpha'), ('로', 'Noun'), ('미국', 'Noun'), ('ETF', 'Alpha') ('...', 'Punctuation'), ('NH', 'Alpha'), ('증권', 'Noun'), ('', 'Punctuation'), ('Noun')]

[('행안부', 'Noun'), ('', 'Foreign'), ('공공', 'Noun'), ('빅데이터', 'Noun'), ('...', 'Punctuation'), ('공모', 'Noun'), ('에', 'Josa'), ('포함', 'Noun'),

. . . .

tags [('빅데이터', 50), ('데이터', 35), ('위', 29), ('분석', 20), ('브랜드', 13), ('평판', 13), ('빅', 10), ('기업', 8), ('운영', 8), ('AI', 7), ('활용', 7), ('대상', 7), ('센터', 7), ('테크', 7), ('신한카드', 7), ('교육', 7), ('바다', 6), ('오픈', 6), ('결과', 6), ('사업', 6), ('공모전', 5), ('플랫폼', 5), ('마켓', 5), ('개발', 5), ('전북', 5), ('글로벌', 4), ('레이스', 4), ('생명', 4), ('KISTI', 4), ('법무부', 4), ('외국인', 4), ('행정', 4), ('개', 3), ('증권', 3), ('모집', 3), ('아이디어', 3), ('시대', 3), ('이유', 3), ('지구', 3), ('숙명여대', 3), ('취준생', 3), ('리터', 3), ('러시', 3), ('역량', 3)]

tag\_dict: {'위': 29, '분석': 20, '브랜드': 13, '평판': 13, '빅': 10, '기업': 8, '운영': 8, 'AI': 7, '활용': 7, '대상': 7, '센터': 7, '테크': 7, '신한카드': 7, '교육': 7, '경진': 6, '대회': 6, '개최': 6, '바다': 6, '오픈': 6, '마켓': 5, '개발': 5, '전북': 5, '글로벌': 4, '레이스': 4, '생명': 4, 'KISTI': 4, '법무부': 4, '외국인': 4, '행정': 4, '개': 4, 'MOU': 4, '은행': 4, '투자': 3, '증권': 3, '모집': 3, '아이디어': 3, '시대': 3, '이유': 3, '지구': 3, '역량': 3}





**Questions?**