

# 데이터 크롤링과 정제

---

## 9장. 한국어 형태소 분석

# 목차

---

- 자연어 처리
  - 설치 라이브러리
- 사용 방법
- 예제

# 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(Okt)**: 오픈 소스 한국어 분석기
      - Twitter에서 이름 변경(과거 트위터 형태소 분석기)

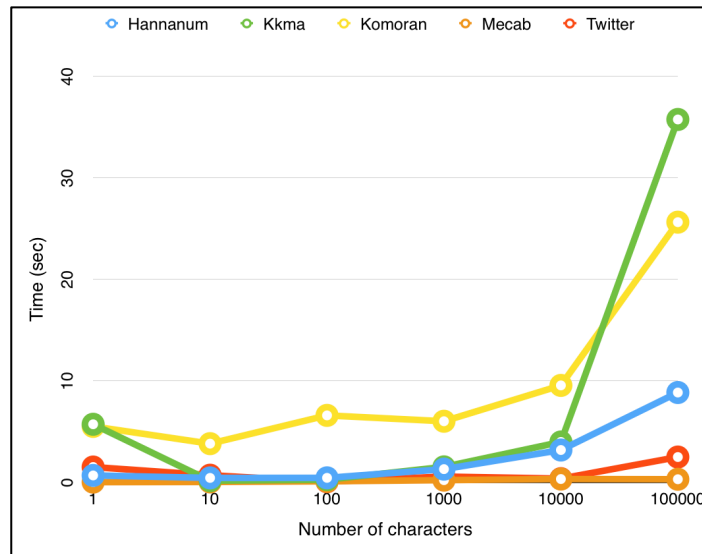
# KoNLPy 성능 비교

## ■ 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

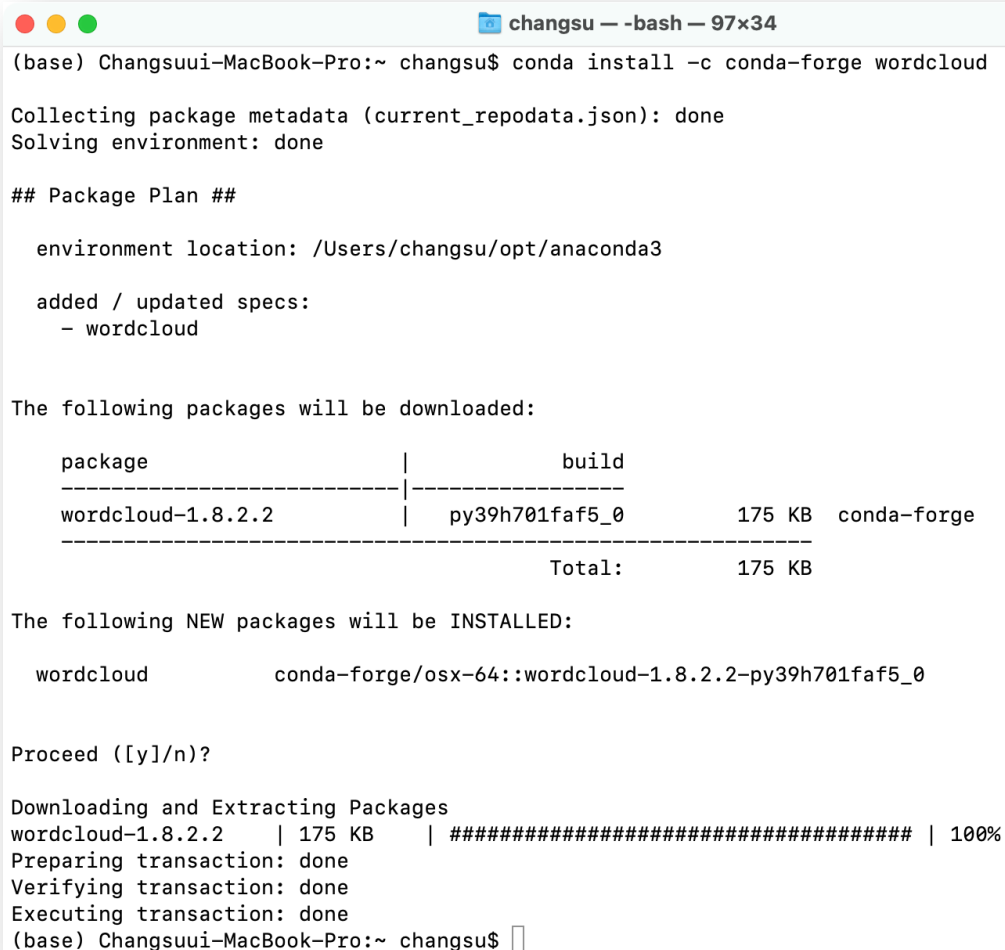
- 10만 문자의 문서를 대상으로 각 클래스의 pos 메소드를 실행하는데 소요되는 시간



# wordcloud 라이브러리 설치

- anaconda 터미널 창에서 아래 명령어 실행

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



A screenshot of a terminal window titled "changsu — -bash — 97x34". The terminal shows the execution of the command `conda install -c conda-forge wordcloud`. The output includes the following text:

```
(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
```

```
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 | size
-----|-----|-----
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

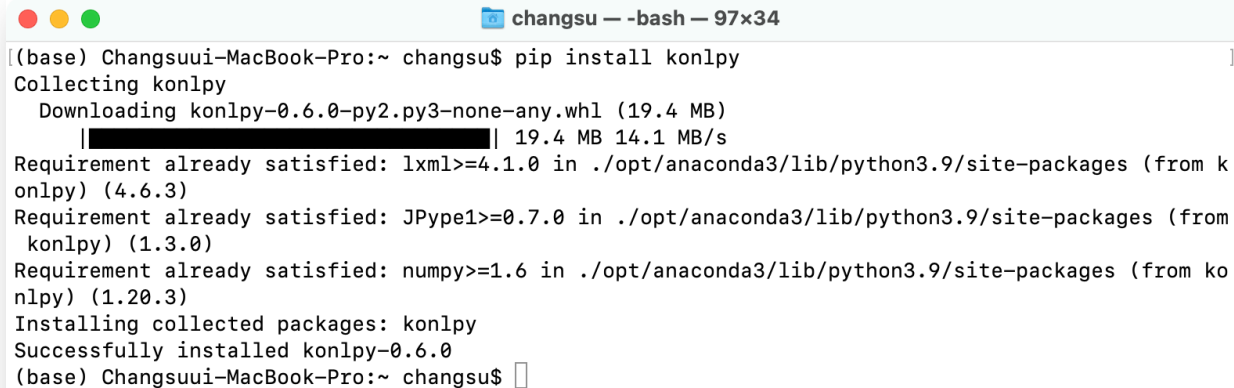
Downloading and Extracting Packages
jpype1-1.3.0 | 368 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(base) Changsuui-MacBook-Pro:~ changsu$
```

# konlpy 라이브러리 설치 #2

---

## ▪ konlpy 라이브러리 설치

```
pip 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), and installed successfully. Dependencies for lxml, JPype1, and numpy are also listed as already satisfied.

```
[(base) Changsuui-MacBook-Pro:~ 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
  - 각 형태소를 품사와 함께 리스트로 반환
  - norm=False: 정규화
    - 같은 의미이면서 표현이 다른 단어를 통합
  - stem=False: 어간 찾기
    - 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)



# Okt 간단 예제 #1

```
from konlpy.tag import Kkma, Komoran, Okt
```

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

```
text = "마음에 쏘인 칼한자루 보다 마음에 쏘인 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'), ('쏘이다', 'Verb'), ('칼', 'Noun'),  
('한', 'Determiner'), ('자루', 'Noun'), ('보다', 'Verb'), ('마음', 'Noun'),  
('에', 'Josa'), ('쏘이다', 'Verb'), ('꽃', 'Noun'), ('한송이', 'Noun'), ('가', 'Josa'),  
('더', 'Noun'), ('아파다', 'Adjective'), ('잠', 'Noun'), ('이', 'Josa'),  
('오지', 'Noun'), ('않다', 'Verb')]
```

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

# Okt 예제 #2

```
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', '싫은', '것', '이', '있어도', '마침내', '제', '뜻', '을', '잘', '표현', '하지', '못', '하는', '사람', '이', '많다', '.', '내', '이를', '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'), ('하지', 'Verb'), ('아니하므로', 'Adjective'), (',', 'Punctuation'), ('우매', 'Noun'), ('한', 'Josa'), ('백성', 'Noun'), ('들', 'Suffix'), ('이', 'Josa'), ('말', 'Noun'), ('하고', 'Josa'), ('싫은', 'Verb'), ('것', 'Noun'), ('이', 'Josa'), ('있어도', 'Adjective'), ('마침내', 'Noun'), ('제', 'Noun'), ('뜻', 'Noun'), ('을', 'Josa'), ('잘', 'Verb'), ('표현', 'Noun'), ('하지', 'Verb'), ('못', 'VerbPrefix'), ('하는', 'Verb'), ('사람', 'Noun'), ('이', 'Josa'), ('많다', 'Adjective'), ('.', 'Punctuation'), ('내', 'Noun'), ('이를', 'Verb'), ('딱하게', 'Adjective'), ('여기어', 'Verb'), ('새로', 'Adjective'), ('스물', 'Noun'), ('여덟', 'Noun'), ('자를', 'Verb'), ('만들었으니', 'Verb'), (',', 'Punctuation'), ('사람', 'Noun'), ('들', 'Suffix'), ('로', 'Josa'), ('하여금', 'Adverb'), ('쉬', 'Noun'), ('익히어', 'Verb'), ('날', 'Noun'), ('마다', 'Josa'), ('쓰는', 'Verb'), ('데', 'Noun'), ('편하게', 'Adjective'), ('할', 'Verb'), ('뿐', 'Noun'), ('이다', 'Josa'), ('.', 'Punctuation')]

# 예제: 단어 분석 및 Word Cloud 생성 #1

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

# open으로 txt파일을 열고 read()를 이용하여 읽는다.
text = open('test.txt').read()
okt = Okt() # Open Korean Text 객체 생성

# okt함수를 통해 읽어 들인 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)

# 가장 많이 나온 단어부터 40개를 저장한다.
counts = Counter(noun_adj_list)
tags = counts.most_common(40)
print(tags)
```

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

```
# WordCloud를 생성한다.  
# 한글을 분석하기 위해 font를 한글로 지정해주어야 된다.  
# macOS는 .otf, window는 .ttf 파일의 위치를 지정해준다. (ex. '/Font/GodoM.otf')  
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'
```

```
wc = WordCloud(font_path=path, background_color="white", max_font_size=60)  
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()
```



```
[('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)]
```

# 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 re
import platform

def get_titles(start_num, end_num, search_word, title_list):
    # start_num ~ end_num까지 크롤링
    while 1:
        if start_num > end_num:
            break
        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')
            # 뉴스제목 뽑아오기
            list_news = soup.find('ul', {'class' : 'list_news'})
            # li_list = list_news.find_all('li', {'id': re.compile('sp_nws.*')})
            li_bxs = list_news.find_all('li', {'class': 'bx'})
            for li_bx in li_bxs:
                news_title = li_bx.find('a', {'class': 'news_tit'})
                title_list.append(news_title['title'])
            start_num += 10
    print(title_list)
```

# 네이버 뉴스 타이틀 Word Cloud 예제 #2

---

```
def make_wordcloud(word_count, title_list):
    okt = Okt()

    sentences_tag = []
    # 형태소 분석하여 리스트에 넣기
    for sentence in title_list:
        morph = okt.pos(sentence)
        sentences_tag.append(morph)
        print(morph)
        print('-' * 30)

    print(sentences_tag)
    print('\n' * 3)

    noun_adj_list = []
    # 명사와 형용사만 구분하여 리스트에 넣기
    for sentence1 in sentences_tag:
        for word, tag in sentence1:
            if tag in ['Noun', 'Adjective']:
                noun_adj_list.append(word)

    # 형태소별 count
    counts = Counter(noun_adj_list)
    tags = counts.most_common(word_count)
    print(tags)
```

# 네이버 뉴스 타이틀 Word Cloud 예제 #3

```
# wordCloud생성
# 한글깨지는 문제 해결하기위해 font_path 지정
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'

wc = WordCloud(font_path=path, background_color='white', width=800, height=600)
print(dict(tags))
cloud = wc.generate_from_frequencies(dict(tags))
plt.figure(figsize=(10, 8))
plt.axis('off')
plt.imshow(cloud)
plt.show()

if __name__ == '__main__':
    search_word = "빅데이터" # 검색어 지정
    title_list = []
    # 1~200번게시글 까지 크롤링
    get_titles(1, 20, search_word, title_list)
    # 단어 30개까지 wordcloud로 출력
    make_wordcloud(20, title_list)
```



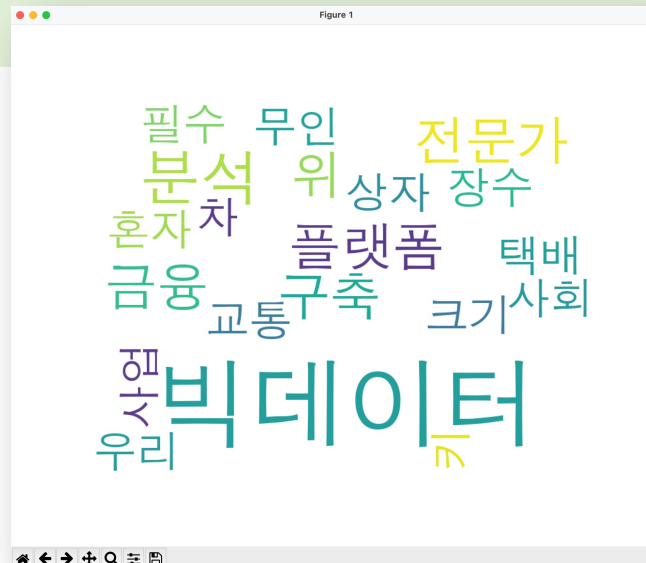
# 네이버 뉴스 타이틀 Word Cloud 예제 실행 결과

```
[('CJ', 'Alpha'), ('의', 'Noun'), ('혁신', 'Noun'), ('...', 'Punctuation'), ('빅데이터', 'Noun'), ('이용', 'Noun'), ('했더니', 'Verb'), ('택배', 'Noun'), ('상자', 'Noun'), ('크기', 'Noun'), ('가', 'Josa'), ('10%', 'Number'), ('줄었다', 'Verb')]
```

```
[('총', 'Modifier'), ('상금', 'Noun'), ('1100만원', 'Number'), (',', 'Punctuation'), ('', 'Foreign'), ('빅데이터', 'Noun'), ('로', 'Josa'), ('한강', 'Noun'), ('수위', 'Noun'), ('예측', 'Noun'), ('하', 'Suffix'), ('라', 'Josa'), ('', 'Punctuation')]
```

...

```
[('빅데이터', 17), ('분석', 4), ('플랫폼', 3), ('구축', 3), ('위', 3), ('금융', 3), ('전문가', 3), ('택배', 2), ('상자', 2), ('크기', 2), ('필수', 2), ('교통', 2), ('사업', 2), ('차', 2), ('혼자', 2), ('장수', 2), ('무인', 2), ('우리', 2), ('사회', 2), ('키', 2)]  
{'빅데이터': 17, '분석': 4, '플랫폼': 3, '구축': 3, '위': 3, '금융': 3, '전문가': 3, '택배': 2, '상자': 2, '크기': 2, '필수': 2, '교통': 2, '사업': 2, '차': 2, '혼자': 2, '장수': 2, '무인': 2, '우리': 2, '사회': 2, '키': 2}
```





# Questions?