

## Part1. KoNLpy 이해

### [1] KoNLpy 사이트 둘러보기

<https://konlpy-ko.readthedocs.io/ko/v0.4.3/>

<https://konlpy-ko.readthedocs.io/ko/v0.4.3/install/> (설치방법 나옴, 맥은 여기 확인)

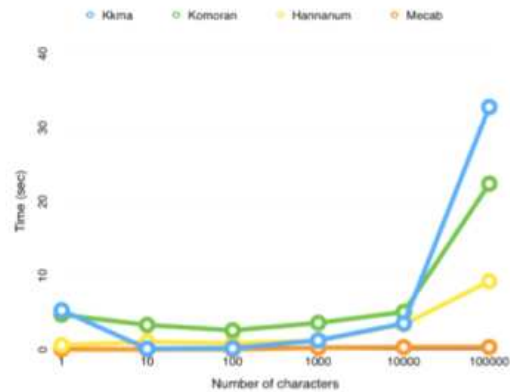
#### 1) KoNLpy(코엔엘파이)패키지

- java 기반의 형태소 분석기 사용
  - 5가지 형태소 분석기 사용가능  
꼬꼬마(Kkma), 한나눔(Hannanum), MeCab-ko(메캡), 코모란(komoran), okt(과거 twitter)
- ✓ <https://konlpy-ko.readthedocs.io/ko/v0.4.3/>

사전별 로딩시 간	• Kkma:	5.6988 secs
	• Komoran:	5.4866 secs
	• Hannanum:	0.6591 secs
	• Twitter:	1.4870 secs
	• Mecab:	0.0007 secs

#### 2) KoNLpy(코엔엘파이)패키지

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



# 품사태그셋

<https://happygrammer.github.io/nlp/postag-set/>

# Hannanum: 한나눔. KAIST Semantic Web Research Center 개발.

# Kkma: 꼬꼬마. 서울대학교 IDS(Intelligent Data Systems) 연구실 개발.

<http://kkma.snu.ac.kr/>

# Komoran: 코모란. Shineware에서 개발.

<https://github.com/shin285/KOMORAN>

# Mecab: 메카브. 일본어용 형태소 분석기를 한국어를 사용할 수 있도록 수정.

<https://bitbucket.org/eunjeon/mecab-ko>

# Open Korean Text: 오픈 소스 한국어 분석기. (구.트위터) 과거 트위터 형태소 분석기.

<https://github.com/open-korean-text/open-korean-text>

## Part3. KoNLpy 실습

### [0] Konlpy.pdf의 9~10페이지를 실습해봄

#### [1] 한글 형태소 분석할 파일 준비 (txt 파일 또는 크롤링한 자료도 가능함)

```
1 f=open('./data/chat.txt', 'r', encoding='utf-8')
2 txt=f.read() #readline(), readlines()
3 f.close
4
5 txt
6
7 'kujefftxt=""\n 카메
8 이더에 연결하세요\n 2
9 요\n 안녕하세요~\n 인
10 카메에서 미메자로 게시중단으로 따요!\n 자료 한번 더 보내주실수 있나요??
11 \n 혹시 북요일 오후에 급격하게 올 것 같은데ㅠㅠ 팀구성에 변화가 있을까
12 요? \n 앓 네!! 그럴 팀원분들께 미리 알려 주하도록 하겠습니다!\n 모르겠
```

문서 열고당 메타태그를 포함함,  
이 글자를 자우려면  
만약 이 글자갯수가 8개라면 txt=txt[9:]  
만약 위치는 가변적인데 Wn이라는 첫표시를  
기준을 나누는 규칙이 있다면

```
1 def myMetaTagDelete(txt):
2     cnt=0
3     for i in range(1000):
4         if txt[i]=="\n":
5             #print(i)
6             cnt+=1
7             if cnt==8:
8                 break
```

사용자정의함수로  
Wn의 첫위치 찾기

```
1 f=open('./data/chat.txt', 'r', encoding='utf-8')
2 txt=f.read() #readline(), readlines()
3 f.close
4
5 myMetaTagDelete(txt)
6
7 txt=txt[i+1:]
8 txt
```

Wn의 첫위치까지 자료  
끊기

#### [2] 형태소분리

```
from konlpy.tag import Okt
f=open('c:/python_data/지식인_답변여행.txt', 'r', encoding='utf-16')
txt=f.read() #readline(), readlines()
f.close

okt=Okt()
re=okt.nouns(txt)
re[:10]
```

#### [3] 글자수제한

```
### 글자의 길이
### 글자의 길이를 구하기 위해서는 각각 리스트에 있을때 하는게 편함
```

```
글자의길이조건준자료=[]
for i in range(len(re)):
    if len(re[i])>=2:
        글자의길이조건준자료.append(re[i])
글자의길이조건준자료[:10]
```

#### [4] 리스트자료 문자열로 변환

```
re1=' '.join(글자의길이조건준자료)
re1[:50]
```

## [5] 워드클라우드 시각화

#stopword 추가 및 png이미지를 넣은 워드클라우드 시각화 가능함.

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

```
wc = WordCloud('c:/malgun.ttf',
               background_color = "white",
               max_words = 2000)
               #stopwords = stopwords)
wc = wc.generate(re1)      # str자료만 가능, list자료는 안됨
plt.figure(figsize = (12, 12))
plt.imshow(wc, interpolation = "bilinear")
plt.axis("off")
plt.show()
```



## [6] 단어집계 - from collections import Counter

- \* collections는 별도 모듈설치없이 사용가능함. 단어의 개수와 순위를 기억함
- \* most\_common() 메서드를 이용하여서 단어수가 많은순으로 정렬하여서 return 받을수 있음.
- \* Counter함수 살펴보기

```
1 from collections import Counter
2 sample='hello world'
3 Counter(sample).most_common()
```

```
[('l', 3),
 ('o', 2),
 ('h', 1),
 ('e', 1),
 (' ', 1),
 ('w', 1),
 ('r', 1),
 ('d', 1)]
```

```
1 sample=['hello', 'hello', 'hi', 'eye']
2 Counter(sample).most_common()
```

```
[('hello', 2), ('hi', 1), ('eye', 1)]
```

```
1 from collections import Counter
2 sample=['hello world hello hi']
3 Counter(sample).most_common()
```

```
[('hello world hello hi', 1)]
```

```
1 sample='hello world hello hi'
2 sample1=sample.split(' ')
3 Counter(sample1).most_common()
```

```
[('hello', 2), ('world', 1), ('hi', 1)]
```

```
1 sample=[['hello', 'hello'], ['hi', 'eye']]
2 Counter(sample).most_common()
```

```
TypeError                                 Traceback (most recent call last)
Input In [5], in <cell line: 2>()
      1 sample=[['hello', 'hello'], ['hi', 'eye']]
----> 2 Counter(sample).most_common()
```

```
TypeError: unhashable type: 'list'
```

```
1 sample=[['hello', 'hello'], ['hi', 'eye']]
2
3 import numpy as np
4 print(np.shape(sample))
5 sample2=np.reshape(sample, 4)
6 print(np.shape(sample2))
7
8 Counter(sample2).most_common()
```

```
(2, 2)
(4,)
```

```
[('hello', 2), ('hi', 1), ('eye', 1)]
```

## \* 작업에 counter함수 적용하기

<pre>re2=re1.split(' ') top=50 cnt = Counter(re2) top_=cnt.most_common(top) top_[:5]</pre>	<pre>[('', 137), ('제주', 71), ('시간', 67), ('코스', 48), ('추천', 45)]</pre>																		
<pre>dict=[] for i in range(top):     dict.append({'word':top_[i][0],'freq':top_[i][1]}) dict[:5]</pre>	<pre>[{'word': '', 'freq': 137},  {'word': '제주', 'freq': 71},  {'word': '시간', 'freq': 67},  {'word': '코스', 'freq': 48},  {'word': '추천', 'freq': 45}]</pre>																		
<pre>import pandas as pd db = pd.DataFrame ( dict )[:5] db</pre>	<table><thead><tr><th></th><th>word</th><th>freq</th></tr></thead><tbody><tr><td>0</td><td></td><td>137</td></tr><tr><td>1</td><td>제주</td><td>71</td></tr><tr><td>2</td><td>시간</td><td>67</td></tr><tr><td>3</td><td>코스</td><td>48</td></tr><tr><td>4</td><td>추천</td><td>45</td></tr></tbody></table>		word	freq	0		137	1	제주	71	2	시간	67	3	코스	48	4	추천	45
	word	freq																	
0		137																	
1	제주	71																	
2	시간	67																	
3	코스	48																	
4	추천	45																	
<pre>db[(db['word']!='제주') &amp; (db['word']!='시간')]</pre>	<table><thead><tr><th></th><th>word</th><th>freq</th></tr></thead><tbody><tr><td>0</td><td></td><td>137</td></tr><tr><td>3</td><td>코스</td><td>48</td></tr><tr><td>4</td><td>추천</td><td>45</td></tr></tbody></table>		word	freq	0		137	3	코스	48	4	추천	45						
	word	freq																	
0		137																	
3	코스	48																	
4	추천	45																	

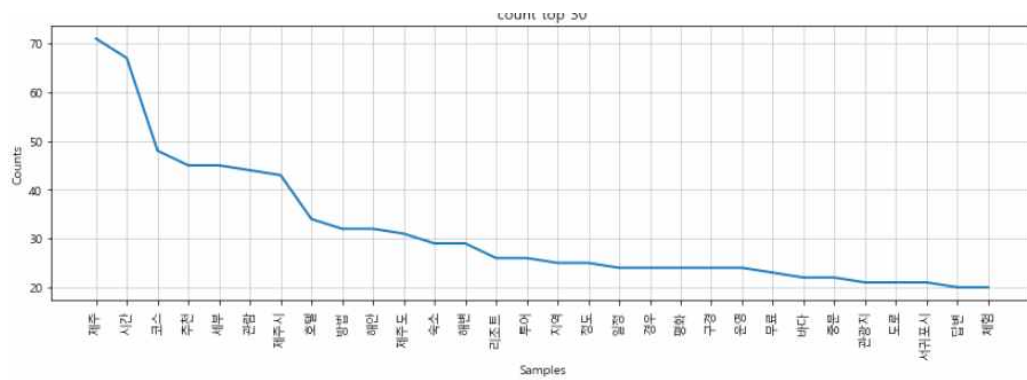
## [7] NLTK함수와 결합

```
from nltk import Text
from nltk.tokenize import RegexpTokenizer
```

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
plt.rcParams["figure.figsize"] = (14,4) #차트 전체크기
plt.rc('font', family='Malgun Gothic')
```

```
top=30
retokenize = RegexpTokenizer("[\w]+")
text = Text(retokenize.tokenize(re1)) # 문자열을 읽어야함.정리다된 자료로 작업함.
```

```
plt.title('count top ' + str(top))
text.plot(30)
plt.show()
```



```
text.dispersion_plot(['자료', '네'])
```

