자연어 처리 2

꼬꼬마에 대하여

```
from konlpy.corpus import kolaw
from konlpy.tag import Kkma
from konlpy.utils import concordance
```

```
# 헌법에 관련된
constitution = kolaw.open('constitution.txt').read()

print(constitution)



# 몇번째 줄에 '민주'라는 단어가 있는지 찾아줌
r = concordance(u'민주', constitution, show=False)
print("show=False => ",r)

# 결과 : show=False =>  [13, 16, 35, 99, 136, 219, 251,
                         1026, 2850, 2854, 3649]
```

고려사항 : 정확성, 속도

```
from konlpy.tag import Kkma
from konlpy.utils import pprint

kkma = Kkma()

text = u'네, 안녕하세요. 반갑습니다.'
```

```
# 문장단위로 찾아냄
text_s = kkma.sentences(text)
print("text_s => ", text_s)



# 리스트에 담겨서 나옴
print("type(text_s) => ", type(text_s))
print("text_s[0] => ", text_s[0])
print("text_s[-1] => ", text_s[-1])
```

tagset : 형식들에 대한 정보 파악

```
# tagset : 형식들에 대한 정보 파악
kkma = Kkma()
print(kkma.tagset())

text = "자연어처리는 재미있습니다. \
    그러나 한국어 분석은 쉽지않습니다."

# 명사 추출기, Noun extractor
text_nouns = kkma.nouns(text)
print(text_nouns)
```

형태소 해석, Parse phrase to morphemes

```
# 나중에 조사들을 추출해서 버리고 의미있는 것들만 분석에 활용한다.
text_morphs = kkma.morphs(text)
print(text_morphs)
```

POS태그

```
pos_tagger = kkma.pos(text)
print(pos_tagger)

print(len(pos_tagger))
print(type(pos_tagger))
print(type(pos_tagger[0]))


# flatten=False : 문장단위에 따라서 묶음이 달라짐
#                  True일때는 하나하나 다 풀어서 저장
pos_tagger_f = kkma.pos(text, flatten=False)

print(pos_tagger_f)
print(len(pos_tagger_f))
print(type(pos_tagger_f))
print(type(pos_tagger_f[0]))
```

```
pos_const = kkma.pos(constitution)
print(len(pos_const))
```

보통 명사만 추출 -> 가나다 순으로

```
pos_const_NNG = [ word[0] for word in pos_const if word[1] == 'NNG']

print(len(pos_const_NNG))

pos_const_NNG.sort()
print(pos_const_NNG[:10])
print(pos_const_NNG[-10:])

# 모든 명사 추출
NN_list = ['NN', 'NNB', 'NNG', 'NNM', 'NNP', 'NP']
```

모든 명사의 개수 파악하기

```
pos_const_NN = [word[0] for word in pos_const if word[1] in NN_list]

print(len(pos_const_NN))

pos_const_NN.sort()
print(pos_const_NN[:10])
print(pos_const_NN[-10:])

# set로 묶어서 유니크한 값 찾기
s = set(pos_const_NN)
print(len(s))
```

어떤 단어개 몇개 있는지 for 구문

```
def getNounCnt(pos_list):
    noun_cnt = {}
    for noun in pos_list:
        if noun_cnt.get(noun):
            noun_cnt[noun] += 1
        else:
            noun_cnt[noun] = 1
```

```
    return noun_cnt

noun_dict = getNounCnt(pos_const_NN)
print(len(noun_dict))
print(noun_dict)
```

constitution = kolaw.open('constitution.txt').read()
pos_sonst = kkma.pos(constitution)

```
from collections import Counter

# most_common : 가장 많이 나온 것들만 뽑아냄
counter = Counter(pos_const)
print(counter.most_common(10))
```

noun_dict = getNounCnt(pos_const_NN)

```
# 의미 있는 것을 찾기 위해서 명사들 중에서 가장 많이 나온 것들을 뽑아냄
counter = Counter(noun_dict)
print(counter.most_common(10))
```

육아 휴직 관련 법안 대한민국 국회 제 1809890호 의안

```
import nltk

from konlpy.corpus import kobill

files_ko = kobill.fileids()
doc_ko = kobill.open('1809890.txt').read()

doc_ko
```

Twitter

```
from konlpy.tag import Twitter;
```

```
t = Twitter()
tokens_ko = t.nouns(doc_ko)
tokens_ko

ko = nltk.Text(tokens_ko, name='대한민국 국회 의안 제 1809890호')

print(len(ko.tokens))
print(len(set(ko.tokens)))
ko.vocab()
```

chart 1

```
# 워드 클라우드 폰트 설정
import matplotlib.pyplot as plt
import platform
path = "c:/Windows/Fonts/malgun.ttf"
from matplotlib import font_manager, rc

if platform.system() == 'Darwin':
    rc('font', family='AppleGothic')
elif platform.system() == 'Windows':
    path = "c:/Windows/Fonts/malgun.ttf"
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
else:
    print('Unknown system.... sorry')

plt.figure(figsize=(12, 6))
ko.plot(50)
plt.show()


stop_words = ['.','(',')','만', '액', '세', '제', '위', '의', '호', '안', '것

ko = [each_word for each_word in ko if each_word not in stop_words]
ko
```

chart 2

```
ko = nltk.Text(ko, name='대한민국 국회 의안 제 1809890호')

plt.figure(figsize=(12, 6))
```

```
ko.plot(50)
plt.show()
```

chart 3

```
ko.count('초등학교')

plt.figure(figsize=(12,6))

ko.dispersion_plot(['육아휴직', '초등학교', '공무원'])

ko.concordance('초등학교')


data = ko.vocab().most_common(150)
```

wordcloud

```
# for mac : font_path='/Library/Fonts/AppleGotic.ttf'
from wordcloud import WordCloud, STOPWORDS

wordcloud = WordCloud(font_path='c:/Windows/Fonts/malgun.ttf',
                      relative_scaling=0.2,
                      background_color='white',
                      ).generate_from_frequencies(dict(data))

plt.figure(figsize=(12, 6))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

```
from konlpy.tag import Okt
from nltk.tokenize import word_tokenize
import nltk


pos_tagger = Okt()
```

```
train = [('메리가 좋아', 'pos'),
         ('고양이도 좋아', 'pos'),
         ('난 수업이 지루해', 'neg'),
         ('메리는 이쁜 고야이야', 'pos'),
         ('난 마치고 메리랑 놀거야', 'pos')]
```

문장을 쪼개고 형태를 분석해보자

```
all_words = set(word.lower()
                for sentence in train
                for word in word_tokenize(sentence[0]))
                # word_tokenize : 문장을 띄어쓰기 기준으로 쪼개줌

all_words

# 결과:
{'고야이야',
 '고양이도',
 '난',
 '놀거야',
 '마치고',
 '메리가',
 '메리는',
 '메리랑',
 '수업이',
 '이쁜',
 '좋아',
 '지루해'}
```

```
t = [({word: (word in word_tokenize(x[0]))
       for word in all_words}, x[1]) for x in train]
    # train의 첫 문장을 all_words와 비교하여 있으면 True 없으면 False
    # 그 다음 문장으로 넘어간다.
t

# 결과 :
[({'마치고': False,
   '난': False,
   '이쁜': False,
   '메리가': True,
   '고양이도': False,
   '메리는': False,
```

```
   '놀거야': False,
   '메리랑': False,
   '좋아': True,
   '수업이': False,
   '지루해': False,
   '고야이야': False},
  'pos'),
 ({'마치고': False,
   '난': False,
   '이쁜': False,
   ...
```

```
classifier = nltk.NaiveBayesClassifier.train(t)
classifier.show_most_informative_features()

# 결과:
Most Informative Features
                      난 = True              neg : pos    =      2.5 : 1.0
                    좋아 = False             neg : pos    =      1.5 : 1.0
                  고야이야 = False             neg : pos    =      1.1 : 1.0
                  고양이도 = False             neg : pos    =      1.1 : 1.0
                  놀거야 = False             neg : pos    =      1.1 : 1.0
                  마치고 = False             neg : pos    =      1.1 : 1.0
                  메리가 = False             neg : pos    =      1.1 : 1.0
                  메리는 = False             neg : pos    =      1.1 : 1.0
                  메리랑 = False             neg : pos    =      1.1 : 1.0
                   이쁜 = False              neg : pos    =      1.1 : 1.0
```