In [ ]:
```
### Konlpy
 * https://konlpy-ko.readthedocs.io/ko/v0.4.3/
 * KoNLPy의 사용법 가이드 : https://konlpy-ko.readthedocs.io/ko/v0.4.3/#guide
 *
```

## 01. 한국 법률 말뭉치

- kolaw: 한국 법률 말뭉치 : constitution.txt
- kobill: 대한민국 국회 의안 말뭉치. 파일 ID는 의안 번호를 의미 :

    - 1809890.txt - 1809899.txt

In [2]:
```
from konlpy.corpus import kolaw
c = kolaw.open('constitution.txt').read()
print( c[:10] )
```

```
대한민국헌법

유구
```

In [3]:
```
from konlpy.corpus import kobill
d = kobill.open('1809890.txt').read()
print ( d[:15] )
```

```
지방공무원법 일부개정법률안
```

## 사전

- 말뭉치를 이용하여 구축됨.
- 형태소 분석 및 품사 태깅에 이용
- Hannanum 시스템 사전 :
    - ./konlpy/java/data/kE/dic_system.txt

- Kkma 시스템 사전 : 세종 말뭉치 이용한 사전
  - 꼬꼬마 형태소 분석기의 .jar 파일 안에 위치
  - 내용 확인 위해 꼬꼬마 미러 이용 : https://github.com/konlpy/kkma/tree/master/dic (https://github.com/konlpy/kkma/tree/master/dic)
- Mecab 시스템 사전 : 세종 말뭉치로 만들어진 CSV 형태 사전(346MB)
  - 설치시 참조 사이트 : https://bit.ly/2FInf5G (https://bit.ly/2FInf5G)

In [6]:
```
### 문장, 명사, 품사 태깅
from konlpy.tag import Kkma
from konlpy.utils import pprint
kkma = Kkma()
```

In [5]:
```
pprint(kkma.sentences('네 안녕하세요. 반갑습니다'))
pprint(kkma.nouns('질문이나 건의사항은 여기에 남겨주세요.'))
pprint(kkma.pos('우리는 데이터 과학자입니다. 멋진 과학자입니다.'))
```

```
['네 안녕하세요.', '반갑습니다']
['질문', '건의', '건의사항', '사항', '여기']
[('우리', 'NP'),
 ('는', 'JX'),
 ('데이터', 'NNG'),
 ('과학자', 'NNG'),
 ('이', 'VCP'),
 ('ㅂ니다', 'EFN'),
 ('.', 'SF'),
 ('멋지', 'VV'),
 ('ㄴ', 'ETD'),
 ('과학자', 'NNG'),
 ('이', 'VCP'),
 ('ㅂ니다', 'EFN'),
 ('.', 'SF')]
```

## ### 02. 문서 탐색

In [7]:
```
from collections import Counter

from konlpy.corpus import kolaw
from konlpy.tag import Hannanum
from konlpy.utils import concordance, pprint
from matplotlib import pyplot
```

In [8]:
```
doc = kolaw.open('constitution.txt').read()
pos = Hannanum().pos(doc)
cnt = Counter(pos)
```

In [9]:
```
pos
```

Out[9]:
```
[('대한민국헌법', 'N'),
 ('유구', 'N'),
 ('하', 'X'),
 ('ㄴ', 'E'),
 ('역사', 'N'),
 ('와', 'J'),
 ('전통', 'N'),
 ('에', 'J'),
 ('빛', 'N'),
 ('나는', 'J'),
 ('우리', 'N'),
 ('대한국민', 'N'),
 ('은', 'J'),
 ('3·1운동', 'N'),
 ('으로', 'J'),
 ('건립', 'N'),
 ('되', 'X'),
 ('ㄴ', 'E'),
 ('대한민국임시정부', 'N'),
```

In [10]:
```
cnt
```

Out[10]:
```
Counter({('대한민국헌법', 'N'): 1,
         ('유구', 'N'): 1,
         ('하', 'X'): 291,
         ('ㄴ', 'E'): 223,
         ('역사', 'N'): 1,
         ('와', 'J'): 39,
         ('전통', 'N'): 1,
         ('에', 'J'): 283,
         ('빛', 'N'): 1,
         ('나는', 'J'): 1,
         ('우리', 'N'): 1,
         ('대한국민', 'N'): 1,
         ('은', 'J'): 184,
         ('3·1운동', 'N'): 1,
         ('으로', 'J'): 52,
         ('건립', 'N'): 1,
         ('되', 'X'): 94,
         ('대한민국임시정부', 'N'): 1,
         ('의', 'J'): 396,
```

In [11]:
```
print('nchars  :', len(doc))
print('ntokens :', len(doc.split()))
print('nmorphs :', len(set(pos)))
print('₩nTop 20 frequent morphemes:'); pprint(cnt.most_common(20))
print('₩nLocations of "대한민국" in the document:')
concordance(u'대한민국', doc, show=True)
```

```
nchars  : 18884
ntokens : 4178
nmorphs : 1499

Top 20 frequent morphemes:
[(('의', 'J'), 396),
 (('.', 'S'), 340),
 (('하', 'X'), 291),
 (('에', 'J'), 283),
 (('ㄴ다', 'E'), 241),
 (('ㄴ', 'E'), 223),
 (('이', 'J'), 221),
 (('을', 'J'), 211),
 (('은', 'J'), 184),
 (('어', 'E'), 176),
 (('를', 'J'), 148),
 (('ㄹ', 'E'), 134),
 (('하', 'P'), 124),
 (('는', 'J'), 117),
 (('법률', 'N'), 115),
 ((',', 'S'), 99),
 (('는', 'E'), 97),
 (('있', 'P'), 96),
 (('되', 'X'), 94),
 (('수', 'N'), 91)]

Locations of "대한민국" in the document:
0       대한민국헌법 유구한 역사와
9       대한국민은 3·1운동으로 건립된 대한민국임시정부의 법통과 불의에
98      총강 제1조 ① 대한민국은 민주공화국이다. ②대한민국의
100     ① 대한민국은 민주공화국이다. ②대한민국의 주권은 국민에게
110     나온다. 제2조 ① 대한민국의 국민이 되는
126     의무를 진다. 제3조 대한민국의 영토는 한반도와
```

```
133     부속도서로 한다. 제4조 대한민국은 통일을 지향하며,
147     추진한다. 제5조 ① 대한민국은 국제평화의 유지에
787     군무원이 아닌 국민은 대한민국의 영역안에서는 중대한
1836    파견 또는 외국군대의 대한민국 영역안에서의 주류에
3620    경제 제119조 ① 대한민국의 경제질서는 개인과
```

Out[11]: [0, 9, 98, 100, 110, 126, 133, 147, 787, 1836, 3620]

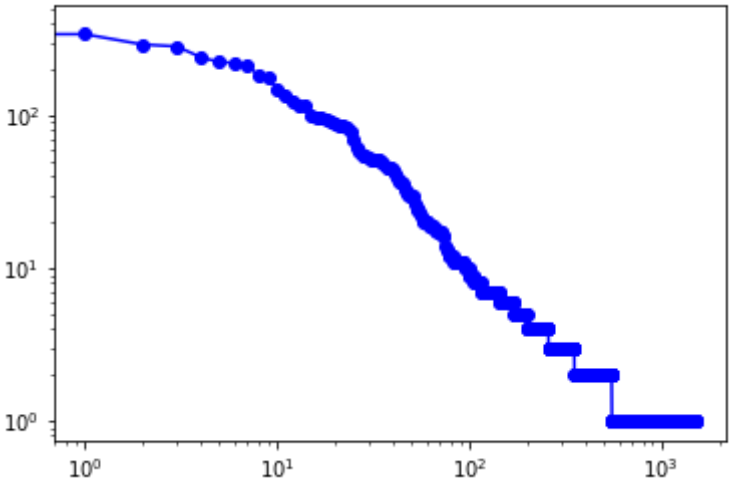
## 지프의 법칙 확인(Zipf's laws)

수학적 통계를 바탕으로 밝혀진 경험적 법칙으로, 물리 및 사회 과학 분야에서 연구된 많은 종류의 정보들이 지프 분포에 가까운 경향을 보인다는 것

지프의 법칙에 따르면 어떠한 자연어 말뭉치 표현에 나타나는 단어들을 그 사용 빈도가 높은 순서대로 나열하였을 때, 모든 단어의 사용 빈도는 해당 단어의 순위에 반비례한다. 따라서 가장 사용 빈도가 높은 단어는 두 번째 단어보다 빈도가 약 두 배 높으며, 세 번째 단어보다는 빈도가 세 배 높다.

In [16]:
```
def draw_zipf(count_list, filename, color='blue', marker='o'):
    sorted_list = sorted(count_list, reverse=True)
    pyplot.plot(sorted_list, color=color, marker=marker)
    pyplot.xscale('log')
    pyplot.yscale('log')
    pyplot.savefig(filename)

draw_zipf(cnt.values(), 'zipf.png')
```




## 연어 찾기

In [18]:
```
from konlpy.tag import Kkma
from konlpy.corpus import kolaw
from konlpy.utils import pprint
from nltk import collocations


measures = collocations.BigramAssocMeasures()
doc = kolaw.open('constitution.txt').read()
doc[1:10]
```

Out[18]: '한민국헌법₩n₩n유구'

## collocations 패키지 참조

- http://www.nltk.org/howto/collocations.html (http://www.nltk.org/howto/collocations.html)

In [22]:
```
print('₩nCollocations among tagged words:')
tagged_words = Kkma().pos(doc)  # 품사 태깅
finder = collocations.BigramCollocationFinder.from_words(tagged_words)
finder
# pprint(finder.nbest(measures.pmi, 10)) # top 5 n-grams with highest PMI
```

```
Collocations among tagged words:
```

Out[22]: <nltk.collocations.BigramCollocationFinder at 0x27e441b5ac8>

In [23]:
```
pprint(finder.nbest(measures.pmi, 10)) # top 5 n-grams with highest PMI
```

```
[(('가부', 'NNG'), ('동수', 'NNG')),
 (('강제', 'NNG'), ('노역', 'NNG')),
 (('경자', 'NNG'), ('유전', 'NNG')),
 (('고', 'ECS'), ('채취', 'NNG')),
 (('공무', 'NNG'), ('담임', 'NNG')),
 (('공중', 'NNG'), ('도덕', 'NNG')),
 (('과반', 'NNG'), ('수가', 'NNG')),
 (('교전', 'NNG'), ('상태', 'NNG')),
 (('그러', 'VV'), ('나', 'ECE')),
 (('기본적', 'NNG'), ('인권', 'NNG'))]
```

In [20]:
```
print('₩nCollocations among words:')
words = [w for w, t in tagged_words]
ignored_words = [u'안녕']
finder = collocations.BigramCollocationFinder.from_words(words)
finder.apply_word_filter(lambda w: len(w) < 2 or w in ignored_words)
finder.apply_freq_filter(3) # only bigrams that appear 3+ times
pprint(finder.nbest(measures.pmi, 10))
```

```
Collocations among words:
[('현행', '범인'),
 ('형의', '선고'),
 ('내부', '규율'),
 ('정치적', '중립성'),
 ('누구', '든지'),
 ('회계', '연도'),
 ('지체', '없이'),
 ('평화적', '통일'),
 ('형사', '피고인'),
 ('지방', '자치')]
```

In [21]:
```
print('₩nCollocations among tags:')
tags = [t for w, t in tagged_words]
finder = collocations.BigramCollocationFinder.from_words(tags)
pprint(finder.nbest(measures.pmi, 5))
```

```
Collocations among tags:
[('XR', 'XSA'), ('JKC', 'VCN'), ('EPT', 'EPT'), ('VCN', 'ECD'), ('ECD', 'VX')]
```

## 03. 구문 분석

In [24]:
```
import konlpy
import nltk
```

In [25]:
```
# POS tag a sentence
sentence = u'만 6세 이하의 초등학교 취학 전 자녀를 양육하기 위해서는'
words = konlpy.tag.Twitter().pos(sentence)
```

```
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩konlpy₩tag₩_okt.py:16: UserWarning: "Twitter" has changed to "Okt" since KoNLPy v0.4.
5.
  warn('"Twitter" has changed to "Okt" since KoNLPy v0.4.5.')
```

## 정규 표현식을 이용한 확인

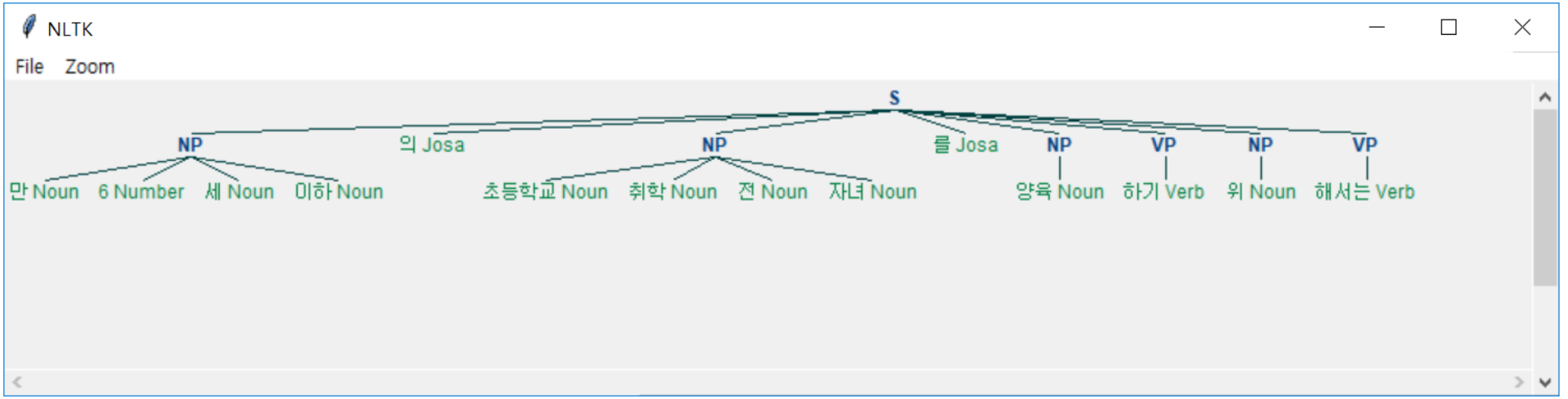
In [26]:
```
# Define a chunk grammar, or chunking rules, then chunk
grammar = """
NP: {<N.*>*<Suffix>?}   # Noun phrase
VP: {<V.*>*}            # Verb phrase
AP: {<A.*>*}            # Adjective phrase
"""
parser = nltk.RegexpParser(grammar)
chunks = parser.parse(words)
print("# Print whole tree")
print(chunks.pprint())
```

```
# Print whole tree
(S
  (NP 만/Noun 6/Number 세/Noun 이하/Noun)
  의/Josa
  (NP 초등학교/Noun 취학/Noun 전/Noun 자녀/Noun)
  를/Josa
  (NP 양육/Noun)
  (VP 하기/Verb)
  (NP 위/Noun)
  (VP 해서는/Verb))
None
```

In [27]:
```
print("₩n# Print noun phrases only")
for subtree in chunks.subtrees():
    if subtree.label()=='NP':
        print(' '.join((e[0] for e in list(subtree))))
        print(subtree.pprint())

# Display the chunk tree
chunks.draw()
```

```
# Print noun phrases only
만 6 세 이하
(NP 만/Noun 6/Number 세/Noun 이하/Noun)
None
초등학교 취학 전 자녀
(NP 초등학교/Noun 취학/Noun 전/Noun 자녀/Noun)
None
양육
(NP 양육/Noun)
None
위
(NP 위/Noun)
None
```



## 04. KoNlpy를 활용한 멀티 스레딩

In [35]:
```
from konlpy.tag import Kkma
from konlpy.corpus import kolaw
from threading import Thread
import jpype

def do_concurrent_tagging(start, end, lines, result):
    jpype.attachThreadToJVM()
    l = [k.pos(lines[i]) for i in range(start, end)]
    result.append(l)
    return

if __name__=="__main__":
    import time

    print('Number of lines in document:')
    k = Kkma()
    lines = kolaw.open('constitution.txt').read().splitlines()
    nlines = len(lines)
    print(nlines)

    print('Batch tagging:')
    s = time.clock()
    result = []
    l = [k.pos(line) for line in lines]
    result.append(l)
    t = time.clock()
    print(t - s)

    print('Concurrent tagging:')
    result = []
    t1 = Thread(target=do_concurrent_tagging, args=(0, int(nlines/2), lines, result))
    t2 = Thread(target=do_concurrent_tagging, args=(int(nlines/2), nlines, lines, result))
    t1.start(); t2.start()
    t1.join(); t2.join()

    m = sum(result, []) # Merge results
    print(time.clock() - t)
```

```
Number of lines in document:
356
```

```
Batch tagging:
16.278313039164598
Concurrent tagging:
8.556062878168998
```

In [ ]: