KoNLPy Documentation

Release 0.3.3

Lucy Park

September 14, 2014

(https://travis-ci.org/e9t/konlpy) (https://readthedocs.org/projects/konlpy/?badge=latest) KoNLPy (pronounced *"ko en el PIE"*) is a Python package for natural language processing (NLP) of the Korean language. For installation directions, see *here* (page 5).

```
>>> from konlpy.tag import Kkma
>>> from konlpy.utils import pprint
>>> kkma = Kkma()
>>> pprint(kkma.sentences(u'네, 안녕하세요. 반갑습니다.'))
[네, 안녕하세요..,
 반갑습니다.]
>>> pprint(kkma.nouns(u'질문이나 건의사항은 깃헙 이슈 트래커에 남겨주세요.'))
[질문,
 건의,
 건의사항,
 사항,
 깃헙,
 이슈,
 트래커]
>>> pprint(kkma.pos(u'오류보고는 실행환경, 에러메세지와함께 설명을 최대한상세히!^^'))
[(오류, NNG),
 (보고, NNG),
 (는, JX),
 (실행, NNG),
 (환경, NNG),
 (,, SP),
 (에러, NNG),
 (메세지, NNG),
 (와, JKM),
 (함께, MAG),
 (설명, NNG),
 (을, JKO),
 (최대한, NNG),
 (상세히, MAG),
 (!, SF),
 (^^, EMO)]
```

For more on how to use KoNLPy, go see the *API* (page 17).

CHAPTER 1

Standing on the shoulders of giants

Korean, the 13th most widely spoken language in the world (http://www.koreatimes.co.kr/www/news/nation/2014/05/116_157214.ht is a beautiful, yet complex language. Myriad *Korean morpheme analyzer tools* (page 14) were built by numerous researchers, to computationally extract meaningful features from the labyrinthine text.

KoNLPy is not just to create another, but to unify and build upon their shoulders, and see one step further. It is built particularly in the Python (programming) language (http://python.org), not only because of the language's simplicity and elegance, but also the powerful string processing modules and applicability to various tasks - including crawling, Web programming, and data analysis.

The three main philosophies of this project are:

- Keep it simple.
- Make it easy. For humans. [1]
- *"Democracy on the web works."* (page 4)

Please report (https://github.com/e9t/konlpy/issues) when you think any have gone stale.

[1] With clear and brief (http://echojuliett.tumblr.com/post/32108001510/clarity-brevity) documents.

CHAPTER 2

License

- GPL v3 or above (http://gnu.org/licenses/gpl.html) [1]

[1] No, I'm not extremely fond of this either. However, some important depedencies - such as Hannanum, Kkma, MeCab-ko - are GPL licensed, and we want to honor their licenses. (It is also an inevitable choice. We hope things may change in the future.)

CHAPTER 3

Contribute

KoNLPy isn't perfect, but it will continuously evolve and you are invited to participate!

Found a bug? Have a good idea for improving KoNLPy? Visit the KoNLPy GitHub page (https://github.com/e9t/konlpy) and suggest an idea (https://github.com/e9t/konlpy/issues) or make a pull request (https://github.com/e9t/konlpy/pulls).

You are also welcome to join the `#koreannlp` channel at the Ozinger IRC Network (http://ozinger.org), and the mailing list (https://groups.google.com/forum/#!topic/konlpy). The IRC channel is more focused on development discussions and the mailing list is a better place to ask questions, but nobody stops you from going the other way around.

Please note that *asking questions through these channels is also a great contribution*, because it give the community feedback as well as ideas. Don't hesitate to ask.

CHAPTER 4

User guide

4.1 Installation

4.1.1 Linux/Mac OS

```
$ pip install JPype1
$ pip install konlpy
$ bash <(curl -s https://raw.githubusercontent.com/e9t/konlpy/master/scripts/mecab.sh) # (Optiona
```

4.1.2 Windows

1. Download and install jpype (http://www.lfd.uci.edu/ gohlke/pythonlibs/#jpype) [1]
2. From the command prompt, install KoNLPy.

```
C:\> pip install konlpy
```

3. (Optional) Download, extract [2], and install the most recent version of MeCab from the following links:
   - mecab-ko (https://bitbucket.org/eunjeon/mecab-ko/downloads)
   - mecab-ko-dic (https://bitbucket.org/eunjeon/mecab-ko-dic/downloads)
   - mecab-python (https://code.google.com/p/mecab/downloads/list?q=python)

4.2 Morphological analysis and POS tagging

Morphological analysis is the identification of the structure of morphemes and other linguistic units, such as root words, affixes, or parts of speech.

POS (part-of-speech) tagging is the process of marking up morphemes in a phrase, based on their definitions and contexts. For example.:

```
가방에 들어가신다 -> 가방/NNG + 에/JKM + 들어가/VV + 시/EPH + ㄴ다/EFN
```

4.2.1 Tagging with KoNLPy

In KoNLPy, there are several different options you can choose for POS tagging. All have the same input-output structure; the input is a phrase, and the output is a list of tagged morphemes.

[1] *win-amd64* for 64-bit Windows, *win32* for 32-bit Windows.

[2] Having MinGW/MSYS or Cygwin installed may be more convenient. Otherwise, you can use 7zip (http://7-zip.org) for the extraction of *tar* files.

For detailed usage instructions see the *tag Package* (page 18).

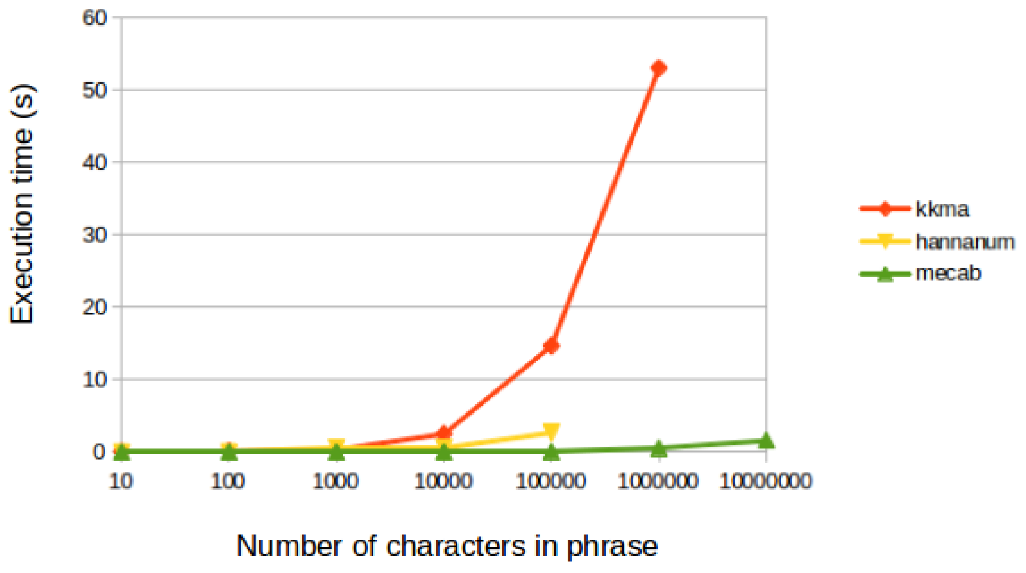
4.2.2 Comparison between tagging classes

Now, we do time and performation analysis for executing the `pos` method for each of the classes in the *tag Package* (page 18).

Time analysis [3]

1. *Loading time*: Class loading time, including dictionary loads. [4]
   - `Kkma` (page 19): 13.2913 *secs*
   - `Hannanum` (page 18): 2.2950 *secs*
   - `Mecab` (page 20): 0.0002 *secs*
2. *Execution time*: Time for executing the `pos` method for each class, with 100K characters. [5]
   - `Kkma` (page 19): 14.6882 *secs*
   - `Hannanum` (page 18): 2.6872 *secs*
   - `Mecab` (page 20): 0.0594 *secs*

   If we test among a various number of characters, all classes' execution times increase in an exponential manner. [6]



Performance analysis

The performance evaluation is replaced with result comparisons for several sample sentences.

1. “저는 대학생이구요. 소프트웨어 관련학과 입니다.”

[3] All time analyses in this document were performed with `time` on a Thinkpad X1 Carbon (2013) and KoNLPy v0.3.

[4] Average of five consecutive runs.

[5] Average of ten consecutive runs.

[6] The current Hannanum class raises a `java.lang.ArrayIndexOutOfBoundsException: 10000` exception if the number of characters is too large.

Kkma (page 19)	Hannanum (page 18)	Mecab (page 20)
저 / NP	저 / N	저 / NP
는 / JX	는 / J	는 / JX
대학생 / NNG	대학생이구요 / N	대학 / NNG
		생 / XSN
이 / VCP		이 / VCP
구요 / EFN		구요 / EF
. / SF	. / S	. / SF
소프트웨어 / NNG	소프트웨어 / N	소프트웨어 / NNG
관련 / NNG	관련학과 / N	관련 / NNG
학과 / NNG		학과 / NNG
이 / VCP	일 / P	입니다 / VCP+EF
ㅂ니다 / EFN	ㅂ니다 / E	
. / SF	. / S	. / SF

2. “아버지가방에들어가신다”

Kkma (page 19)	Hannanum (page 18)	Mecab (page 20)
아버지 / NNG	아버지가방에들어가 / N	아버지 / NNG
가방 / NNG	이 / J	가 / JKS
에 / JKM		방 / NNG
		에 / JKB
들어가 / VV		
시 / EPH	시ㄴ다 / E	들어가 / VV
ㄴ다 / EFN		신다 / EP+EC

3. “140823 Tofu Music Festival 존잘러에서 귀요미들로 변신ㅋㅋ #GOT7”

Kkma (page 19)	Hannanum (page 18)	Mecab (page 20)
140823 / NR	140823 / N	140823 / SN
Tofu / OL	Tofu / F	Tofu / SL
Music / OL	Music / F	Music / SL
Festival / OL	Festival / F	Festival / SL
존 / NNP	존잘러 / N	존 / VA+JX
잘 / MAG		잘 / VA
러 / NNP		러 / EC
에서 / JKM	에서 / J	에서 / JKB
귀요 / NNG	귀요미들 / N	귀요미 / NNG
미들 / NNG		들 / XSN
로 / JKM	로 / J	로 / JKB
변신 / NNG	변신ㅋㅋ / N	변신 / NNG
ㅋㅋ / EMO		ㅋㅋ / UNKNOWN
# / SW	#GOT7 / N	# / SY
GOT / OL		GOT / SL
7 / NR		7 / SN

4.3 Data

4.3.1 Tags

See also:

Korean POS tags comparison chart (https://docs.google.com/spreadsheets/d/1OGAjUvalBuX-oZvZ_-9tEfYD2gQe7hTGsgUpiiBSXI8/edit#gid=0)

Compare POS tags between several Korean analytic projects. (In Korean)

4.3.2 Dictionaries

Dictionaries are used for *Morphological analysis and POS tagging* (page 5), and are built with *Corpora* (page 15).

`Hannanum` *system dictionary*

A dictionary created with the KAIST corpus. (4.7MB)

Located at `./konlpy/java/data/kE/dic_system.txt`. Part of this file is shown below.:

```
...
나라경제	ncn
나라기획	nqq
나라기획회장	ncn
나라꽃	ncn
나라님	ncn
나라도둑	ncn
나라따르	pvg
나라링링프로덕션	ncn
나라말	ncn
나라망신	ncn
나라박물관	ncn
나라발전	ncpa
나라별	ncn
나라부동산	nqq
나라사랑	ncn
나라살림	ncpa
나라시	nqq
나라시마	ncn
...
```

You can add your own terms, modify `./konlpy/java/data/kE/dic_user.txt`.

`Kkma` *system dictionary*

A dictionary created with the Sejong corpus. (32MB)

It is included within the Kkma `.jar` file, so in order to see dictionary files, check out the KKMA's mirror (https://github.com/e9t/kkma/tree/master/dic). Part of `kcc.dic` is shown below.:

```
아니/IC
후우/IC
그래서/MAC
그러나/MAC
그러니까/MAC
그러면/MAC
그러므로/MAC
그런데/MAC
그리고/MAC
따라서/MAC
하지만/MAC
...
```

`Mecab` *system dictionary*

A CSV formatted dictionary created with the Sejong corpus. (346MB)

The compiled version is located at `/usr/local/lib/mecab/dic/mecab-ko-dic` (or the path you assigned during installation), and you can see the original files in the source code (https://bitbucket.org/eunjeon/mecab-ko-dic/src/ce04f82ab0083fb24e4e542e69d9e88a672c3325/seed/?at=master). Part of `CoinedWord.csv` is shown below.:

```
가오티,0,0,0,NNG,*,F,가오티,*,*,*,*,*
갑툭튀,0,0,0,NNG,*,F,갑툭튀,*,*,*,*,*
강퇴,0,0,0,NNG,*,F,강퇴,*,*,*,*,*
개드립,0,0,0,NNG,*,T,개드립,*,*,*,*,*
갠소,0,0,0,NNG,*,F,갠소,*,*,*,*,*
고퀄,0,0,0,NNG,*,T,고퀄,*,*,*,*,*
광삭,0,0,0,NNG,*,T,광삭,*,*,*,*,*
광탈,0,0,0,NNG,*,T,광탈,*,*,*,*,*
굉천,0,0,0,NNG,*,T,굉천,*,*,*,*,*
국을,0,0,0,NNG,*,T,국을,*,*,*,*,*
귀요미,0,0,0,NNG,*,F,귀요미,*,*,*,*,*
...
```

To add your own terms, see here (https://bitbucket.org/eunjeon/mecab-ko-dic/src/ce04f82ab0083fb24e4e542e69d9e88a672c3325/final/user-dic/?at=master).

Note: You can add new words either to the system dictionaries or user dictionaries. However, there is a slight difference in the two choices.:

- *Adding to the system dictionary*: When dictionary updates are not frequent, when you do not want to drop the analysis speed.
- *Adding to the user dictionary*: When dictionary updates are not frequent, when you do not have `root` access.

4.4 Examples

4.4.1 Exploring a document

Exploring a document can consist of various components:

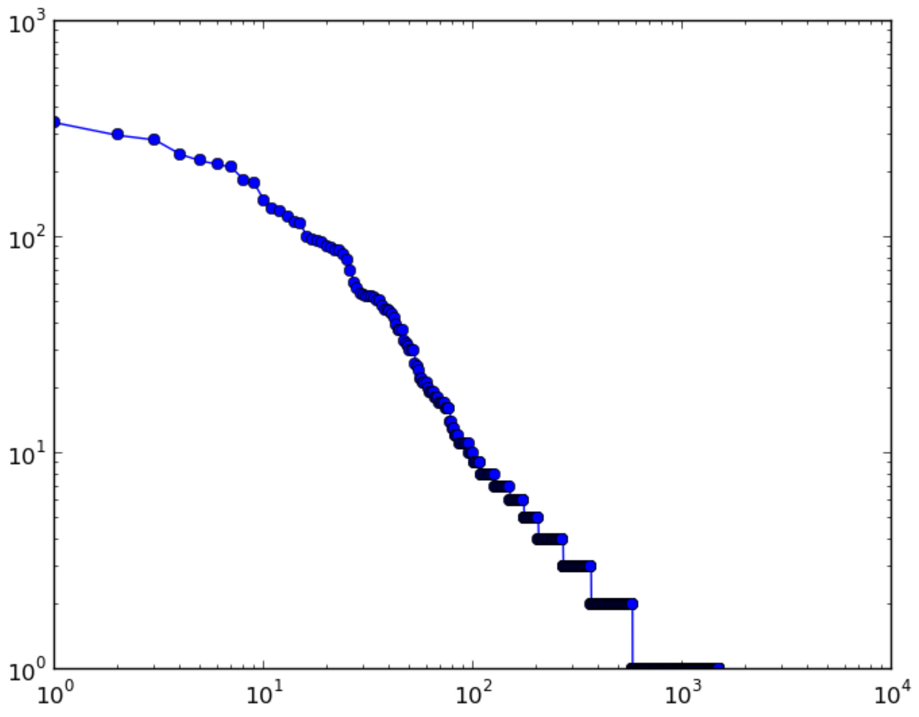
- Counts (characters, words, etc.)
- Checking Zipf's laws: $fr = k$
- Concordances

```
#! /usr/bin/python2.7
# -*- coding: utf-8 -*-

from collections import Counter

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


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)


doc = kolaw.open('constitution.txt').read()
```

```
pos = Hannanum().pos(doc)
cnt = Counter(pos)

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)

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

- Console:

```
nchars  : 19240
ntokens : 4178
nmorphs : 1501

Top 20 frequent morphemes:
[((의, J), 398),
 ((., S), 340),
 ((하, X), 297),
 ((에, J), 283),
 ((ㄴ다, E), 242),
 ((ㄴ, E), 226),
 ((이, J), 218),
 ((을, J), 211),
 ((은, J), 184),
 ((어, E), 177),
 ((를, J), 148),
 ((ㄹ, E), 135),
 ((/, S), 131),
 ((하, P), 124),
 ((는, J), 117),
 ((법률, N), 115),
 ((,, S), 100),
 ((는, E), 97),
 ((있, P), 96),
 ((되, X), 95)]

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
```



- **zipf.png:**

4.4.2 Drawing a word cloud

Below shows a code example that crawls a National Assembly bill from the web, extract nouns and draws a word cloud - from head to tail in Python.

You can change the bill number (i.e., `bill_num`), and see how the word clouds differ per bill. (ex: '1904882', '1904883', 'ZZ19098', etc)

```
#! /usr/bin/python2.7
# -*- coding: utf-8 -*-

from collections import Counter
import urllib
import random
import webbrowser

from konlpy.tag import Hannanum
from lxml import html
import pytagcloud # requires Korean font support
import sys

if sys.version_info[0] >= 3:
    urlopen = urllib.request.urlopen
else:
    urlopen = urllib.urlopen


r = lambda: random.randint(0,255)
color = lambda: (r(), r(), r())

def get_bill_text(billnum):
    url = 'http://pokr.kr/bill/%s/text' % billnum
    response = urlopen(url).read().decode('utf-8')
    page = html.fromstring(response)
    text = page.xpath(".//div[@id='bill-sections']/pre/text()")[0]
    return text

def get_tags(text, ntags=50, multiplier=10):
    h = Hannanum()
    nouns = h.nouns(text)
    count = Counter(nouns)
    return [{ 'color': color(), 'tag': n, 'size': c*multiplier }\
                for n, c in count.most_common(ntags)]

def draw_cloud(tags, filename, fontname='Noto Sans CJK', size=(800, 600)):
    pytagcloud.create_tag_image(tags, filename, fontname=fontname, size=size)
    webbrowser.open(filename)


bill_num = '1904882'
text = get_bill_text(bill_num)
tags = get_tags(text)
print(tags)
draw_cloud(tags, 'wordcloud.png')
```

Note: The PyTagCloud (https://pypi.python.org/pypi/pytagcloud) installed in PyPI may not be sufficient for drawing wordclouds in Korean. You may add eligible fonts - that support the Korean language - manually, or install the Korean supported version here (https://github.com/e9t/PyTagCloud).

행위
동물
학대
영상물
신설
인터넷
판매 · 전시 · 전달 · 상영
고통
다음
게재
동물보호법
제47조(과태료)
호
사
금지
개정법률안
굶주림
고양시
정당
현행법
제1항부터
과태료
목적
제8조
누구
일부
활동
적절
질병
표시
의식
동물보호
일체
방지
신체
위반
홍보
제8조제5항
약
한
이
때
1의2
제47조제1항
제3항까지
-제8조제5항
윤영희 · 김영록 · 이만우
주요내용
개념
제한
발
치
키
말
것

4.4.3 Finding collocations

We can find collocations with the help of NLTK (http://nltk.org).

```
#! /usr/bin/python2.7
# -*- coding: utf-8 -*-

from konlpy.tag import Kkma
from konlpy.corpus import kolaw
from konlpy.utils import pprint
from nltk import collocations
```

```
bigram_measures = collocations.BigramAssocMeasures()
trigram_measures = collocations.TrigramAssocMeasures()

doc = kolaw.open('constitution.txt').read()
pos = Kkma().pos(doc)
words = [s for s, t in pos]
tags = [t for s, t in pos]


print('\nCollocations among tagged words:')
finder = collocations.BigramCollocationFinder.from_words(pos)
pprint(finder.nbest(bigram_measures.pmi, 10)) # top 5 n-grams with highest PMI

print('\nCollocations among 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(bigram_measures.pmi, 10))

print('\nCollocations among tags:')
finder = collocations.BigramCollocationFinder.from_words(tags)
pprint(finder.nbest(bigram_measures.pmi, 5))
```

- Console:

```
Collocations among tagged words:
[((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))]

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

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

4.5 Running tests

KoNLPy has tests to evaulate its quality. To perform a test, use the code below.

```
$ pip install pytest
$ cd konlpy
$ py.test
```

KoNLPy was tested on the below environments:

- Mac OS X 10.9 with Sun/Oracle 1.6.0
- Ubuntu 12.04 with openjdk-7-jdk
- Ubuntu 13.10 with openjdk-7-jdk
- Ubuntu 14.04 with openjdk-7-jdk
- Window 7 with Sun/Oracle JDK 1.7.0

Note: To see known bugs/issues, see here (https://github.com/e9t/konlpy/labels/bug).

4.6 References

Note: Please modify this document (https://github.com/e9t/konlpy/blob/master/docs/references.rst) if anything is erroneous or not included. Last updated at September 14, 2014.

4.6.1 Korean morpheme analyzer tools

C/C++

- **KTS (http://wiki.kldp.org/wiki.php/KTS) (1995) GPL v2**
  - By 이상호, 서정연, 오영환 (KAIST & 서강대)
  - code (https://github.com/suapapa/kts)
- **MACH (http://cs.sungshin.ac.kr/ shim/demo/mach.html) (2002) custom**
  - By Prof. Kwangseob Shim (성신여대)
- **MeCab-ko (https://bitbucket.org/eunjeon/mecab-ko/) (2013) GPL LGPL BSD**
  - By Yong-woon Lee and Youngho Yoo

Java

- **Arirang (http://cafe.naver.com/korlucene) (2009) Apache v2**
  - By SooMyung Lee
  - code (http://sourceforge.net/projects/lucenekorean)
- **Hannanum (http://semanticweb.kaist.ac.kr/home/index.php/HanNanum) (1999) GPL v3**
  - By Prof. Key-Sun Choi Key's research team (KAIST)
  - code (http://kldp.net/projects/hannanum/src), docs (http://semanticweb.kaist.ac.kr/research/hannanum/j/javadoc/)
- **KKMA (http://kkma.snu.ac.kr) (2010) GPL v2**
  - By Prof. Sang-goo Lee's research team (서울대)

- **KOMORAN (http://shineware.tistory.com/tag/KOMORAN) (2013) custom**
  - By *shineware*

Python

- **KoNLPy (http://konlpy.readthedocs.org) (2014) GPL v3**
  - By Lucy Park (서울대)
- **UMorpheme (https://pypi.python.org/pypi/UMorpheme) (2014) MIT**
  - By Kyunghoon Kim (UNIST)

R

- **KoNLP (https://github.com/haven-jeon/KoNLP) (2011) GPL v3**
  - By Heewon Jeon

Others

- K-LIWC (http://k-liwc.ajou.ac.kr/) (아주대)
- **KRISTAL-IRMS (http://www.kristalinfo.com/) (KISTI)**
  - Development history (http://spasis.egloos.com/9507)
- Korean XTAG (http://www.cis.upenn.edu/ xtag/koreantag/) (UPenn)
- HAM (http://nlp.kookmin.ac.kr/HAM/kor/ham-intr.html) (국민대)
- POSTAG/K (http://nlp.postech.ac.kr/ project/DownLoad/k_api.html) (포스텍)
- Speller (http://speller.cs.pusan.ac.kr/) (부산대)
- UTagger (http://203.250.77.242:5900/) (울산대)
- (No name) (http://cl.korea.ac.kr/Demo/dglee/index.html) (고려대)

4.6.2 Other NLP tools

Language parser

- KoreanParser (http://semanticweb.kaist.ac.kr/home/index.php/KoreanParser) - By DongHyun Choi, Jungyeul Park, Key-Sun Choi (KAIST)

4.6.3 Corpora

- **HANTEC 2.0 (http://www.kristalinfo.com/download/#hantec), KISTI & 충남대, 1998-2003.**
  - 120,000 test documents (237MB)
  - 50 TREC-type questions for QA (48KB)
- **HKIB-40075 (http://www.kristalinfo.com/TestCollections/readme_hkib.html), KISTI & 한국일보, 2002.**
  - 40,075 test documents for text categorization (88 MB)
- KAIST corpus (http://semanticweb.kaist.ac.kr/home/index.php/KAIST_Corpus), KAIST, 1997-2005.
- Sejong corpus (http://www.sejong.or.kr/), National Institute of the Korean Language, 1998-2007.

4.6.4 General NLP resources

- Google NLP publications (http://research.google.com/pubs/NaturalLanguageProcessing.html)
- Lingpipe (http://alias-i.com/lingpipe/)
- Microsoft NLP group (Redmond) (http://research.microsoft.com/en-us/groups/nlp/)
- 부산대 NLP 관련사이트 목록 (http://borame.cs.pusan.ac.kr/ai_home/site/site1.html)

API

5.1 konlpy Package

5.1.1 `jvm` Module

`konlpy.jvm.`**`init_jvm`**(*jvmpath=None*)

Initializes the Java virtual machine (JVM).

Parameters jvmpath – The path of the JVM. If left empty, inferred by `jpype.getDefaultJVMPath()`.

5.1.2 `utils` Module

class `konlpy.utils.`**`UnicodePrinter`**(*indent=1*, *width=80*, *depth=None*, *stream=None*)

Bases: `pprint.PrettyPrinter` (http://docs.python.org/library/pprint.html#pprint.PrettyPrinter)

`format`(*object*, *context*, *maxlevels*, *level*)

Overrided method to enable Unicode pretty print.

`konlpy.utils.`**`char2hex`**(*c*)

Converts a unicode character to hex.

```
>>> char2hex(u'음')
'0xc74c'
```

`konlpy.utils.`**`concat`**(*phrase*)

Concatenates lines into a unified string.

`konlpy.utils.`**`concordance`**(*phrase*, *text*, *show=False*)

Find concordances of a phrase in a text.

The farmost left numbers are indices, that indicate the location of the phrase in the text (by means of tokens). The following string, is part of the text surrounding the phrase for the given index.

Parameters

- **phrase** – Phrase to search in the document.
- **text** – Target document.
- **show** – If `True`, shows locations of the phrase on the console.

```
>>> from konlpy.corpus import kolaw
>>> from konlpy.tag import Mecab
>>> from konlpy import utils
>>> constitution = kolaw.open('constitution.txt').read()
>>> idx = utils.concordance(u'대한민국', constitution, show=True)
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
>>> idx
[0, 9, 98, 100, 110, 126, 133, 147, 787, 1836, 3620]
```

konlpy.utils.**hex2char**(*h*)

Converts a hex character to unicode.

```
>>> print hex2char('c74c')
음
>>> print hex2char('0xc74c')
음
```

konlpy.utils.**load_txt**(*filename*)

Text file loader.

konlpy.utils.**partition**(*list_*, *indices*)

Partitions a list to several parts using indices.

Parameters

- **list** – The target list.
- **indices** – Indices to partition the target list.

konlpy.utils.**pprint**(*obj*)

Unicode pretty printer.

```
>>> import pprint, konlpy
>>> pprint.pprint([u"Print", u"유니코드", u"easily"])
[u'Print', u'\uc720\ub2c8\ucf54\ub4dc', u'easily']
>>> konlpy.utils.pprint([u"Print", u"유니코드", u"easily"])
['Print', '유니코드', 'easily']
```

konlpy.utils.**preprocess**(*phrase*)

Preprocesses a phrase in the following steps:.

•concat() (page 17)

konlpy.utils.**select**(*phrase*)

Replaces some ambiguous punctuation marks to simpler ones.

5.1.3 Subpackages

tag Package

Note: Initial runs of each class method may require some time to load dictionaries (< 1 *min*). Second runs should be faster.

Hannanum Class

class konlpy.tag._hannanum.**Hannanum**(*jvmpath=None*)

Wrapper for JHannanum (http://semanticweb.kaist.ac.kr/home/index.php/HanNanum).

JHannanum is a morphological analyzer and POS tagger written in Java, and developed by the Semantic Web Research Center (SWRC) (http://semanticweb.kaist.ac.kr/) at KAIST since 1999.

```
from konlpy.tag import Hannanum

hannanum = Hannanum()
print hannanum.analyze(u'롯데마트의 흑마늘 양념 치킨이 논란이 되고 있다.')
print hannanum.nouns(u'다람쥐 헌 쳇바퀴에 타고파')
print hannanum.pos(u'웃으면 더 행복합니다!')
print hannanum.morphs(u'웃으면 더 행복합니다!')
```

Parameters **jvmpath** – The path of the JVM passed to `init_jvm()` (page 17).

`analyze` (*phrase*)
Phrase analyzer.

This analyzer returns various morphological candidates for each token. It consists of two parts: 1) Dictionary search (chart), 2) Unclassified term segmentation.

`morphs` (*phrase*)
Parse phrase to morphemes.

`nouns` (*phrase*)
Noun extractor.

`pos` (*phrase*, *ntags=9*)
POS tagger.

This tagger is HMM based, and calculates the probability of tags.

Parameters **ntags** – The number of tags. It can be either 9 or 22.

`Kkma` Class

class `konlpy.tag._kkma.`**`Kkma`** (*jvmpath=None*)
Wrapper for Kkma (http://kkma.snu.ac.kr).

Kkma is a morphological analyzer and natural language processing system written in Java, developed by the Intelligent Data Systems (IDS) Laboratory at SNU (http://snu.ac.kr).

```
from konlpy.tag import Kkma

kkma = Kkma()
print kkma.sentences(u'저는 대학생이구요. 소프트웨어 관련학과 입니다.')
print kkma.nouns(u'대학에서 DB, 통계학, 이산수학 등을 배웠지만...')
print kkma.morph(u'자주 사용을 안하다보니 모두 까먹은 상태입니다.')
print kkma.pos(u'어쩌면 좋죠?')
```

Parameters **jvmpath** – The path of the JVM passed to `init_jvm()` (page 17).

`morphs` (*phrase*)
Parse phrase to morphemes.

`nouns` (*phrase*)
Noun extractor.

`pos` (*phrase*)
POS tagger.

`sentences` (*phrase*)
Sentence detection.

`Mecab` Class

Warning: Mecab is not supported for Python 3 and Windows 7.

class `konlpy.tag._mecab.`**`Mecab`**(*dicpath='/usr/local/lib/mecab/dic/mecab-ko-dic'*)

Wrapper for MeCab-ko morphological analyzer.

MeCab (https://code.google.com/p/mecab/), originally a Japanese morphological analyzer and a POS tagger developed by the Graduate School of Informatics in Kyoto University, was modified to MeCab-ko by the Eunjeon Project (http://eunjeon.blogspot.kr/) to adapt to the Korean language.

In order to use MeCab-ko within KoNLPy, follow the directions in *optional-installations*.

```
from konlpy.tag import Mecab
# MeCab installation needed

mecab = Mecab()
print mecab.pos(u'자연주의 쇼핑몰은 어떤 곳인가?')
print mecab.morphs(u'영등포구청역에 있는 맛집 좀 알려주세요.')
print mecab.nouns(u'우리나라에는 무료 치료를 잘하는 정형외과가 없는가!')
```

Parameters **dicpath** – The path of the MeCab-ko dictionary.

`morphs`(*phrase*)
Parse phrase to morphemes.

`nouns`(*phrase*)
Noun extractor.

`pos`(*phrase*)
POS tagger.

See also:

Korean POS tags comparison chart (https://docs.google.com/spreadsheets/d/1OGAjUvalBuX-oZvZ_-9tEfYD2gQe7hTGsgUpiiBSXI8/edit#gid=0)

Compare POS tags between several Korean analytic projects. (In Korean)

corpus Package

class `konlpy.corpus.`**`CorpusLoader`**(*name=None*)

Loader for corpora. The following corpora are currently available:

- *kolaw*: Korean law corpus.
- *kobill*: Korean National Assembly bill corpus. The file ID corresponds to the bill number.

```
>>> from konlpy.corpus import kolaw
>>> fids = kolaw.fileids()
>>> fobj = kolaw.open(fids[0])
>>> print fobj.read(140)
대한민국헌법

유구한 역사와 전통에 빛나는 우리 대한국민은 3·1운동으로 건립된 대한민국임시정부의 법통과 불의에 항거한 4·19민주
```

`abspath`(*filename=None*)
Absolute path of corpus file. If `filename` is *None*, returns absolute path of corpus.

Parameters **filename** – Name of a particular file in the corpus.

`fileids`()
List of file IDs in the corpus.

open (*filename*)
Method to open a file in the corpus. Returns a file object.

Parameters **filename** – Name of a particular file in the corpus.

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*
- `changelog`