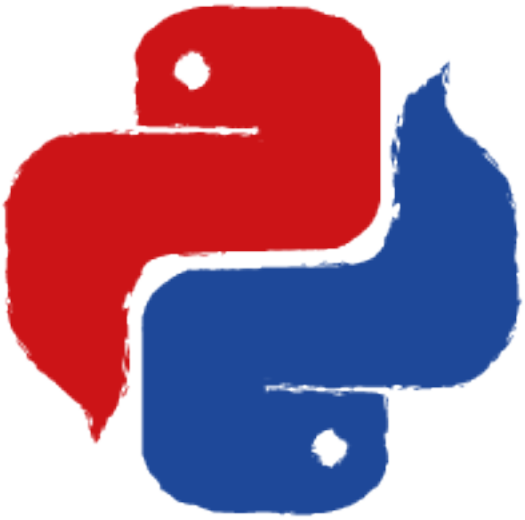


KoNLPy

KoNLPy Documentation

Release 0.3.0

Lucy Park

August 24, 2014

Contents

(https://travis-ci.org/e9t/konlpy) (https://readthedocs.org/projects/konlpy/?badge=latest) KoNLPy 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'저는 대학생이구요. 소프트웨어 관련학과 입니다.'))
[저는 대학생이구요.,
 소프트웨어 관련학과 입니다.]
>>> pprint(kkma.nouns(u'대학에서 DB, 통계학, 이산수학 등을 배웠지만...'))
[대학,
 통계학,
 이산,
 이산수학,
 수학,
 등]
>>> pprint(kkma.pos(u'자주 사용을 안하다보니 모두 까먹은 상태입니다.'))
[(자주, MAG),
 (사용, NNG),
 (을, JKO),
 (안하, VV),
 (다, ECS),
 (보, VXV),
 (니, ECD),
 (모두, MAG),
 (까먹, VV),
 (은, ETD),
 (상태, NNG),
 (이, VCP),
 (ㅂ니다, EFN),
 (., SF)]
```

For more on how to use KoNLPy, go see the *API* (page 14).

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 NLP engines* (page 12) 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 (me@lucypark.kr) 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).

CHAPTER 4

User guide

4.1 Installation

KoNLPy is not registered in PyPI yet. Please install from source until further notice.

- **Requirements**
  - Python 2.6 or higher
  - JRE 1.7 or higher

4.1.1 Linux

Install KoNLPy

```
$ pip install git+https://github.com/e9t/konlpy.git
```

or

```
$ git clone https://github.com/e9t/konlpy.git
$ cd konlpy
$ python setup.py install
```

Optional installations

In order to use the MeCab morpheme analyzer in KoNLPy, install the followings:

- mecab-ko (https://bitbucket.org/eunjeon/mecab-ko/downloads)

```
$ wget https://bitbucket.org/eunjeon/mecab-ko/downloads/mecab-0.996-ko-0.9.1.tar.gz
$ tar zxfv mecab-0.996-ko-0.9.1.tar.gz
$ cd mecab-0.996-ko-0.9.1
$ ./configure
$ make
$ make check
$ sudo make install
```

- mecab-ko-dic (https://bitbucket.org/eunjeon/mecab-ko-dic/downloads)

```
$ wget https://bitbucket.org/eunjeon/mecab-ko-dic/downloads/mecab-ko-dic-1.6.1-20140814.tar.gz
$ tar zxfv mecab-ko-dic-1.6.1-20140814.tar.gz
$ cd mecab-ko-dic-1.6.1-20140814
$ ./configure
$ sudo ldconfig
$ make
```

```
$ sudo sh -c 'echo "dicdir=/usr/local/lib/mecab/dic/mecab-ko-dic" > /usr/local/etc/mecabrc'
$ sudo make install
```

- mecab-python (https://github.com/HiroyukiHaga/mecab-python)

```
$ git clone https://github.com/HiroyukiHaga/mecab-python.git
$ cd mecab-python
$ python setup.py build
$ sudo python setup.py install
```

4.1.2 Windows

KoNLPy's compatibility with Windows is not stable yet. However, if you would still like to put in some effort you can try the following.

1. Download the most recent release of KoNLPy from GitHub (https://github.com/e9t/konlpy/releases), and extract files. (Otherwise, you can just *clone* it).
   - Current release: 0.3.0
2. In a terminal, go to the KoNLPy directory, and type `python setup.py install`.

Note: If you have installation errors with dependency packages, try installing them with Christoph Gohlke's Windows Binaries (http://www.lfd.uci.edu/ gohlke/pythonlibs/) [1]:

- jpype (http://www.lfd.uci.edu/ gohlke/pythonlibs/#jpype)
- regex (http://www.lfd.uci.edu/ gohlke/pythonlibs/#regex)

---

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.

For detailed usage instructions see the *tag Package* (page 15).

[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.

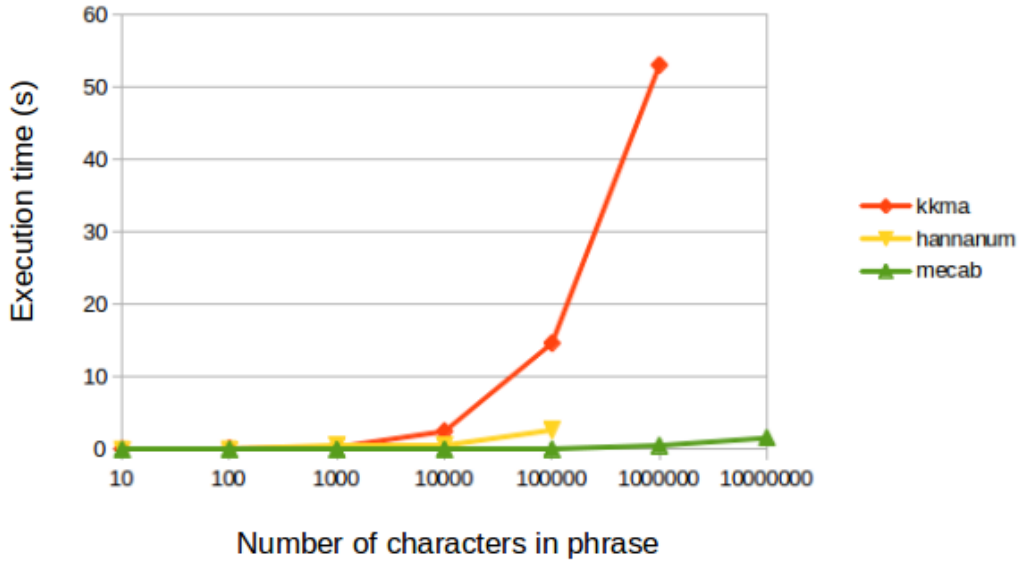
4.2.2 Comparison between tagging modules

Now, we do time and performation analysis for executing the `pos` method for each of the modules in the *tag Package* (page 15).

Time analysis [3]

1. *Loading time*: Module loading time, including dictionary loads. [4]
   - `kkma` (page 16): 13.2913 *secs*
   - `hannanum` (page 15): 2.2950 *secs*
   - `mecab` (page 16): 0.0002 *secs*
2. *Execution time*: Time for executing the `pos` method for each module, with 100K characters. [5]
   - `kkma` (page 16): 14.6882 *secs*
   - `hannanum` (page 15): 2.6872 *secs*
   - `mecab` (page 16): 0.0594 *secs*

   If we test among a various number of characters, all modules' 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 module raises a `java.lang.ArrayIndexOutOfBoundsException: 10000` exception if the number of characters is too large.

kkma (page 16)	hannanum (page 15)	mecab (page 16)
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. “롯데마트의 흑마늘양념치킨이 논란이 되고 있는 가운데, 자연주의쇼핑몰에서부터만큼은 안정적으로 운영되고 있다.”

kkma (page 16)	hannanum (page 15)	mecab (page 16)
롯데 / NNP	롯데마트 / N	롯데마트 / NNP
마트 / NNG		
의 / JKG	의 / J	의 / JKG
흑 / NNG	흑마늘양념치킨 / N	흑마 / NNG
마늘 / NNG		늘 / MAG
양념 / NNG		양념치킨 / NNP
치킨 / NNG		
이 / JKS	이 / J	이 / JKS
논란 / NNG	논란 / N	논란 / NNG
이 / JKC	이 / J	이 / JKS
되 / VV	되 / P	되 / VV
고 / ECE	고 / E	고 / EC
있 / VXV	있 / P	있 / VX
는 / ETD	는 / E	는 / ETM
가운데 / NNG	가운데 / N	가운데 / NNG
, / SP	, / S	, / SC
자연주의 / NNG	자연주의쇼핑몰 / N	자연 / NNG
		주 / NNG
		의 / JKG
쇼핑몰 / NNG		쇼핑몰 / NNG
에서 / JKM	에서부터만큼은 / J	에서부터 / JKB
부터 / JX		
만큼 / NNG		만큼 / JKB
은 / JX		은 / JX
안정적 / NNG	안정적 / N	안정 / NNG
		적 / XSN
으로 / JKM	으로 / J	으로 / JKB
운영 / NNG	운영 / N	운영 / NNG
되 / XSV	되 / X	되 / XSV
고 / ECE	고 / E	고 / EC
있 / VXV	있 / P	있 / VX
다 / EFN	다 / E	다 / EF
. / SF	. / S	. / SF

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 6), and are built with *Corpora* (page 13).

`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 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


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 = urllib.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).

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
```

Note: KoNLPy was tested on the below environments:

- Ubuntu 12.04 with openjdk-7-jdk
- Ubuntu 13.10 with openjdk-7-jdk
- Window 7 with Sun/Oracle JDK 1.7.0 (`hannanum` (page 15), `mecab` (page 16) have issues)

4.6 References

Note: Please report (me@lucypark.kr) if you know any other NLP engines or corpora that are not included in this list. Last updated at August 24, 2014.

4.6.1 Korean NLP engines

C/C++

- MACH (http://cs.sungshin.ac.kr/ shim/demo/mach.html), Sungshin Women's University custom
- MeCab-ko (https://bitbucket.org/eunjeon/mecab-ko/), Yong-woon Lee and Youngho Yoo GPL LGPL BSD

Java

- Hannanum (http://semanticweb.kaist.ac.kr/home/index.php/HanNanum), KAIST GPL v3
- Kkma (http://kkma.snu.ac.kr), Seoul National University GPL v2
- KOMORAN (http://shineware.tistory.com/tag/KOMORAN), *shineware* custom

Python

- KoNLPy (http://konlpy.readthedocs.org), Lucy Park GPL v3

Others

- K-LIWC (http://k-liwc.ajou.ac.kr/), Ajou University
- **KRISTAL-IRMS (http://www.kristalinfo.com/), KISTI**
  - Development history (http://spasis.egloos.com/9507) (In Korean)
- Korean XTAG (http://www.cis.upenn.edu/ xtag/koreantag/), University of Pennsylvania
- Speller (http://speller.cs.pusan.ac.kr/), Pusan National University
- UTagger (http://203.250.77.242:5900/), University of Ulsan
- (No name) (http://cl.korea.ac.kr/Demo/dglee/index.html), Korea University

4.6.2 Corpora

- **HANTEC 2.0 (http://www.kristalinfo.com/download/#hantec), KISTI and CNU, 1998-2003.**
  - 120,000 test documents (237MB)
  - 50 TREC-type questions for QA (48KB)
- **HKIB-40075 (http://www.kristalinfo.com/TestCollections/readme_hkib.html), 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.

4.6.3 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/)

CHAPTER 5

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*)
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.

```
>>> 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)
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 14)

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 Module

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

Parameters **jvmpath** – The path of the JVM passed to `init_jvm()` (page 14).

morph (*phrase*)
Morphological analyzer.

This analyzer consists of two parts: 1) Dictionary search (chart), 2) Unclassified term segmentation.

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 Module

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.pos(u'자주 사용을 안하다보니 모두 까먹은 상태입니다.')
```

Parameters **jvmpath** – The path of the JVM passed to `init_jvm()` (page 14).

nouns (*phrase*)
Noun extractor.

pos (*phrase*)
POS tagger.

sentences (*phrase*)
Sentence detection.

mecab Module

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* (page 5).

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

mecab = Mecab()
```

```
print mecab.nouns(u'우리나라에는 무료 치료를 잘하는 정형외과가 없는가!')
print mecab.pos(u'자연주의 쇼핑몰은 어떤 곳인가?')
```

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

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.

```
>>> 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 4·19민주이
```

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`

k