



KoNLPy

KoNLPy Documentation

Release 0.4.0

Lucy Park

January 18, 2015

1	Standing on the shoulders of giants	2
2	License	3
3	Contribute	4
4	Getting started	5
4.1	What is NLP?	5
4.2	What do I need to get started?	5
5	User guide	6
5.1	Installation	6
5.2	Morphological analysis and POS tagging	7
5.3	Data	10
5.4	Examples	12
5.5	Running tests	22
5.6	References	22
6	API	25
6.1	konlpy Package	25
7	Indices and tables	32
	Python Module Index	33

(<https://travis-ci.org/konlpy/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 6).

For users new to NLP, go to [Getting started](#) (page 5). For step-by-step instructions, follow the [User guide](#) (page 6). For specific descriptions of each module, go see the [API](#) (page 25) documents.

```
>>> 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)]
```

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.htm) (http://www.koreatimes.co.kr/www/news/nation/2014/05/116_157214.htm) is a beautiful, yet complex language. Myriad [Korean morpheme analyzer tools](#) (page 22) 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) (<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/konlpy/konlpy/issues) (<https://github.com/konlpy/konlpy/issues>) when you think any have gone stale.

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

License

KoNLPy is Open Source Software, and is released under the license below:

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

You are welcome to use the code under the terms of the license, however please acknowledge its use with a citation.

- Eunjeong L. Park, Sungzoon Cho. “KoNLPy: Korean natural language processing in Python (<http://dmlab.snu.ac.kr/~lucypark/docs/2014-10-10-hclt.pdf>)”, Proceedings of the 26th Annual Conference on Human & Cognitive Language Technology, Chuncheon, Korea, Oct 2014.

Here is a BibTeX entry.:

```
@inproceedings{park2014konlpy,
  title={KoNLPy: Korean natural language processing in Python},
  author={Park, Eunjeong L. and Cho, Sungzoon},
  booktitle={Proceedings of the 26th Annual Conference on Human & Cognitive Language Technology},
  address={Chuncheon, Korea},
  month={October},
  year={2014}
}
```

¹ No, I'm not extremely fond of this either. However, some important dependencies - 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.)

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/konlpy/konlpy) (<https://github.com/konlpy/konlpy>) and [suggest an idea](https://github.com/konlpy/konlpy/issues) (<https://github.com/konlpy/konlpy/issues>) or [make a pull request](https://github.com/konlpy/konlpy/pulls) (<https://github.com/konlpy/konlpy/pulls>).

You are also welcome to join the `#koreannlp` channel at the [Ozinger IRC Network](http://ozinger.org) (<http://ozinger.org>), and the [mailing list](https://groups.google.com/forum/#!forum/konlpy) (<https://groups.google.com/forum/#!forum/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 gives the community feedback as well as ideas. Don't hesitate to ask.

Getting started

4.1 What is NLP?

NLP (Natural Language Processing) is a set of techniques for analyzing and extracting, and understanding meaningful information text.

We have various NLP applications in our lives. For example:

- Text summarization (ex: [Summly](http://www.summly.com/index.html) (<http://www.summly.com/index.html>))
- Question answering (ex: [Wolfram Alpha](http://www.wolframalpha.com/input/?i=what+is+the+meaning+of+life&lk=4&num) (<http://www.wolframalpha.com/input/?i=what+is+the+meaning+of+life&lk=4&num>))
- Dialogue systems (ex: [Apple Siri](https://www.apple.com/ios/siri/) (<https://www.apple.com/ios/siri/>))
- Machine translation (ex: [Google Translate](http://translate.google.com) (<http://translate.google.com>))

And obviously information retrieval systems such as Web search engines. For a better understanding of NLP techniques, consider referring the so-called “bibles”:

- Jurafsky et al., [Speech and Language Processing](https://www.goodreads.com/book/show/908048) (<https://www.goodreads.com/book/show/908048>), 2nd Edition, 2008.
- Manning and Schutze, [Foundations of Statistical Natural Language Processing](https://www.goodreads.com/book/show/776349) (<https://www.goodreads.com/book/show/776349>), 1999.

KoNLPy will help you to actually carry out some fundamental NLP tasks with Korean text. In case you’re interested in handling English text, check out [NLTK](http://nltk.org) (<http://nltk.org>).

4.2 What do I need to get started?

You have some prerequisites to use KoNLPy.

1. Deep interest in natural languages and some familiarity with the Korean language
2. Understanding of basic Python programming ¹
3. A “good” text editor and terminal (or Python IDE) ²
4. A [Python installation](https://wiki.python.org/moin/BeginnersGuide/Download) (<https://wiki.python.org/moin/BeginnersGuide/Download>)
5. [pip](https://pypi.python.org/pypi/pip) (<https://pypi.python.org/pypi/pip>), the Python package manager

Got ‘em all? Then let’s go.

¹ If you’re new to Python, this tutorial should get you through in minutes: <http://learnxinyminutes.com/docs/python/>. If you’re up to putting in some more time, try [The Hitchhiker’s Guide](http://docs.python-guide.org/en/latest/) (<http://docs.python-guide.org/en/latest/>) or [Learn Python the hard way](http://learnpythonthehardway.org/book/) (<http://learnpythonthehardway.org/book/>).

² Many use [Sublime Text 2](http://www.sublimetext.com/) (<http://www.sublimetext.com/>) for Python programming. Some others use Vim and Terminal. But other than these, there are numerous great [text editors](http://tutorialzine.com/2012/07/battle-of-the-tools-which-is-the-best-code-editor/) (<http://tutorialzine.com/2012/07/battle-of-the-tools-which-is-the-best-code-editor/>) and [Python IDEs](http://pedrokroger.net/choosing-best-python-ide/) (<http://pedrokroger.net/choosing-best-python-ide/>) out there, so take your pick!

User guide

5.1 Installation

Note: For troubleshooting information, see these pages: [Linux](https://github.com/konlpy/konlpy/issues?q=label%3Alinux) (<https://github.com/konlpy/konlpy/issues?q=label%3Alinux>), [Mac OS](https://github.com/konlpy/konlpy/issues?q=label%3A%22mac+os%22) (<https://github.com/konlpy/konlpy/issues?q=label%3A%22mac+os%22>), [Windows](https://github.com/konlpy/konlpy/issues?q=label%3Awindows) (<https://github.com/konlpy/konlpy/issues?q=label%3Awindows>). Please record a “New Issue” (<https://github.com/konlpy/konlpy/issues/new>) if you have an error that is not listed. You can also see testing logs [here](https://docs.google.com/spreadsheets/d/1Ii_L9NF9gSLbsJOGqsf-zfqTtyhhthmJWNC2kgUDIsU/edit#gid=0) (https://docs.google.com/spreadsheets/d/1Ii_L9NF9gSLbsJOGqsf-zfqTtyhhthmJWNC2kgUDIsU/edit#gid=0).

5.1.1 Ubuntu

1. From the command prompt, install KoNLPy.

```
$ sudo apt-get install python-dev python3-dev
$ pip install konlpy      # Python 2.x
$ pip3 install konlpy    # Python 3.x
```

2. Install MeCab (*optional*)

```
$ sudo apt-get install curl
$ bash <(curl -s https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh)
```

5.1.2 Mac OS

1. From the command prompt, install KoNLPy.

```
$ pip install konlpy      # Python 2.x
$ pip3 install konlpy    # Python 3.x
```

2. Install MeCab (*optional*)

```
$ bash <(curl -s https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh) # (Optional)
```

5.1.3 Windows

1. Set `JAVA_HOME` (http://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/index.html).
2. Download and install `JPython` (<http://www.lfd.uci.edu/~gohlke/pythonlibs/#jpython>) ($\geq 0.5.7$).¹ You may have to upgrade your `pip` version in order to install the downloaded `.whl` file.

¹ `win-amd64` for 64-bit Windows, `win32` for 32-bit Windows.

```
> pip install --upgrade pip
> pip install JPype1-0.5.7-cp27-none-win_amd64.whl
```

3. From the command prompt, install KoNLPy.

```
> pip install konlpy
```

4. Download, extract ², and install the most recent version of MeCab from the following links (*optional*):

- [mecab-ko](https://bitbucket.org/eunjeon/mecab-ko/downloads) (<https://bitbucket.org/eunjeon/mecab-ko/downloads>)
- [mecab-ko-dic](https://bitbucket.org/eunjeon/mecab-ko-dic/downloads) (<https://bitbucket.org/eunjeon/mecab-ko-dic/downloads>)
- [mecab-python](https://bitbucket.org/eunjeon/mecab-python-0.996/downloads) (<https://bitbucket.org/eunjeon/mecab-python-0.996/downloads>)

Warning:

- KoNLPy only supports `Kkma()`, `Komorana()` on Windows machines. `Mecab()`, `Hannanum()` are not supported.
- Are you having trouble installing MeCab-ko to your Windows machine? Refer to this blog post by Kim YungYi: [mecab-ko 윈도우에서 빌드하기](https://yungyikim.atlassian.net/wiki/pages/viewpage.action?pageId=2916362) (<https://yungyikim.atlassian.net/wiki/pages/viewpage.action?pageId=2916362>) (In Korean).

5.2 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 + 예/JKM + 들어가/VV + 시/EPH + 다/EFN

5.2.1 POS 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 25).

See also:

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

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

5.2.2 Comparison between POS tagging classes

Now, we do time and performance analysis for executing the `pos` method for each of the classes in the *tag Package* (page 25). The experiments were carried out on a Intel i7 CPU with 4 cores, Python 2.7, and KoNLPy 0.4.0.

² Having MinGW/MSYS or Cygwin installed may be more convenient. If you plan to use Git, [Git BASH](https://msysgit.github.io/) (<https://msysgit.github.io/>) is another good option. Otherwise, you can use [7zip](http://7-zip.org/) (<http://7-zip.org/>) for the extraction of *tar* files.

Time analysis

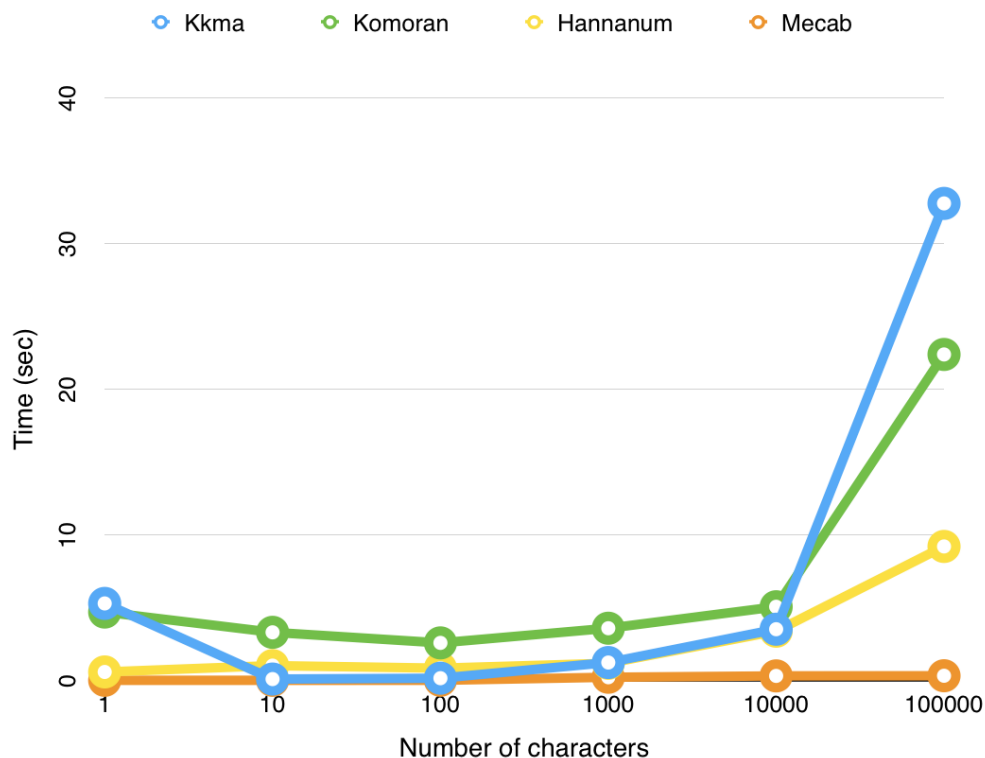
1. *Loading time*: Class loading time, including dictionary loads.

- [Kkma](#) (page 26): 5.1032 secs
- [Komoran](#) (page 26): 4.6852 secs
- [Hannanum](#) (page 25): 0.5736 secs
- [Mecab](#) (page 27): 0.0007 secs

2. *Execution time*: Time for executing the `pos` method for each class, with 100K characters.

- [Kkma](#) (page 26): 32.7300 secs
- [Komoran](#) (page 26): 22.3639 secs
- [Hannanum](#) (page 25): 9.1881 secs
- [Mecab](#) (page 27): 0.3137 secs

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



Performance analysis

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

1. “아버지가방에들어가신다”

We can check the spacing algorithm through this example. Desirably, an analyzer would parse this sentence to “아버지가 방에 들어가신다” (My father enters the room), rather than “아버지 가방에 들어가신다” (My father goes in the bag). [Hannanum](#) (page 25) and [Komoran](#) (page 26) are careful in spacing uncertain terms, and defaults the whole phrase to nouns. [Kkma](#) (page 26) is more confident, but gets undesirable results. For this result, [Mecab](#) (page 27) shows the best results.

Hannanum	Kkma	Komoran	Mecab
아버지가방에들어가 / N 이 / J 시ㄴ 다 / E	아버지 / NNG 가방 / NNG 에 / JKM 들어가 / VV 시 / EPH ㄴ 다 / EFN	아버지가방에들어가신다 / NNP	아버지 / NNG 가 / JKS 방 / NNG 에 / JKB 들어가 / VV 신다 / EP+EC

2. “나는 밥을 먹는다” vs “하늘을 나는 자동차”

If we focus on “나는” in both sentences, we can see whether an analyzer considers the context of words. “나는” in the first sentence should be “나/N + 는/J”, and in the second sentence “나(-ㄴ다)/V + 는/E”.

:py:Kkma properly understands the latter “나는” as a verb, whereas the rest observe it as nouns.

Hannanum	Kkma	Komoran	Mecab
나 / N 는 / J 밥 / N 을 / J 먹 / P 는다 / E	나 / NP 는 / JX 밥 / NNG 을 / JKO 먹 / VV 는 / EPT 다 / EFN	나 / NP 는 / JX 밥 / NNG 을 / JKO 먹 / VV 는다 / EC	나 / NP 는 / JX 밥 / NNG 을 / JKO 먹 / VV 는다 / EC

Hannanum	Kkma	Komoran	Mecab
하늘 / N 을 / J 나 / N 는 / J 자동차 / N	하늘 / NNG 을 / JKO 날 / VV 는 / ETD 자동차 / NNG	하늘 / NNG 을 / JKO 나 / NP 는 / JX 자동차 / NNG	하늘 / NNG 을 / JKO 나 / NP 는 / JX 자동차 / NNG

3. “아이폰 기다리다 지쳐 애플공홈에서 연락폰질러버렸다 6+ 128기가실버ㅋ”

How do each of the analyzers deal with slang, or terms that are not included in the dictionary?

Hannanum	Kkma	Komoran	Mecab
아이폰 / N 기다리 / P 다 / E 지치 / P 어 / E 애플공홈 / N 에서 / J 언락폰질러버렸다 / N 6+ / N 128기가실벌 / N	아이 / NNG 폰 / NNG 기다리 / VV 다 / ECS 지치 / VV 어 / ECS 애플 / NNP 공 / NNG 홈 / NNG 에서 / JKM 언락 / NNG 폰 / NNG 질르 / VV 어 / ECS 버리 / VXV 었 / EPT 다 / ECS 6 / NR • / SW 128 / NR 기가 / NNG 실벌 / NNG ㅋ / UN	아이폰 / NNP 기다리 / VV 다 / EC 지치 / VV 어 / EC 애플 / NNP 공 / NNG 홈 / NNG 에서 / JKB 언 / NNG 락 / NNG 폰 / NNG 지르 / VV 어 / EC 버리 / VX 었 / EP 다 / EC 6 / SN • / SW 128기가실벌 / NA	아이폰 / NNP 기다리 / VV 다 / EC 지쳐 / VV+EC 애플 / NNP 공 / NNG 홈 / NNG 에서 / JKB 언락 / NNG 폰 / NNG 질러버렸 / VV+EC+VX+EP 다 / EC 6 / SN • / SY 128 / SN 기 / NNG 가 / JKS 실벌 / NNP ㅋ / UNKNOWN

5.3 Data

5.3.1 Corpora

The following corpora are currently available:

1. **kolaw: Korean law corpus.**

- constitution.txt

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

- 1809890.txt - 1809899.txt

For more detailed usage of the corpora, see the *corpus Package* (page 27).

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

유구한 역사와

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

5.3.2 Dictionaries

Dictionaries are used for *Morphological analysis and POS tagging* (page 7), and are built with *Corpora* (page 24).

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) (<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) (<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-kodict/src/ce04f82ab0083fb24e4e542e69d9e88a672c3325/final/user-dic/?at=master) (<https://bitbucket.org/eunjeon/mecab-kodict/src/ce04f82ab0083fb24e4e542e69d9e88a672c3325/final/user-dic/?at=master>).

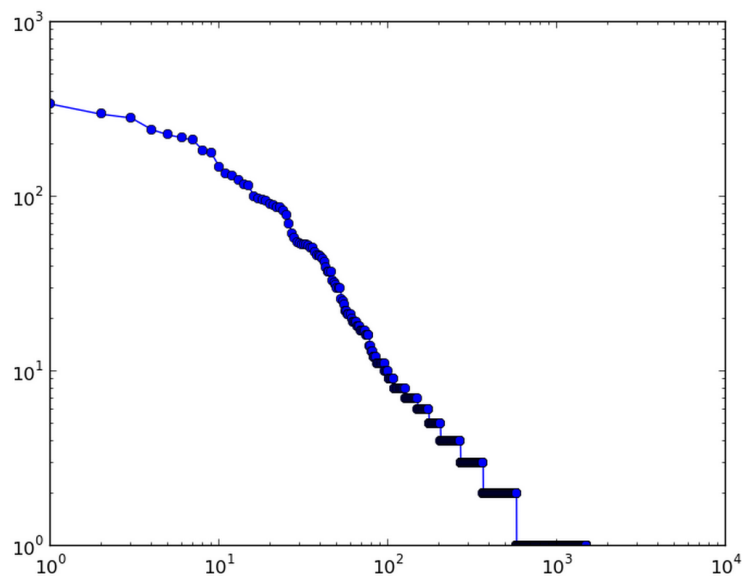
Note: You can add new words either to the system dictionaries or user dictionaries. However, there is a slight difference in the two choices.:

- *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 frequent, when you do not have `root` access.

5.4 Examples

Below are a set of example tasks using KoNLPy.

Exploring a document (page 15)

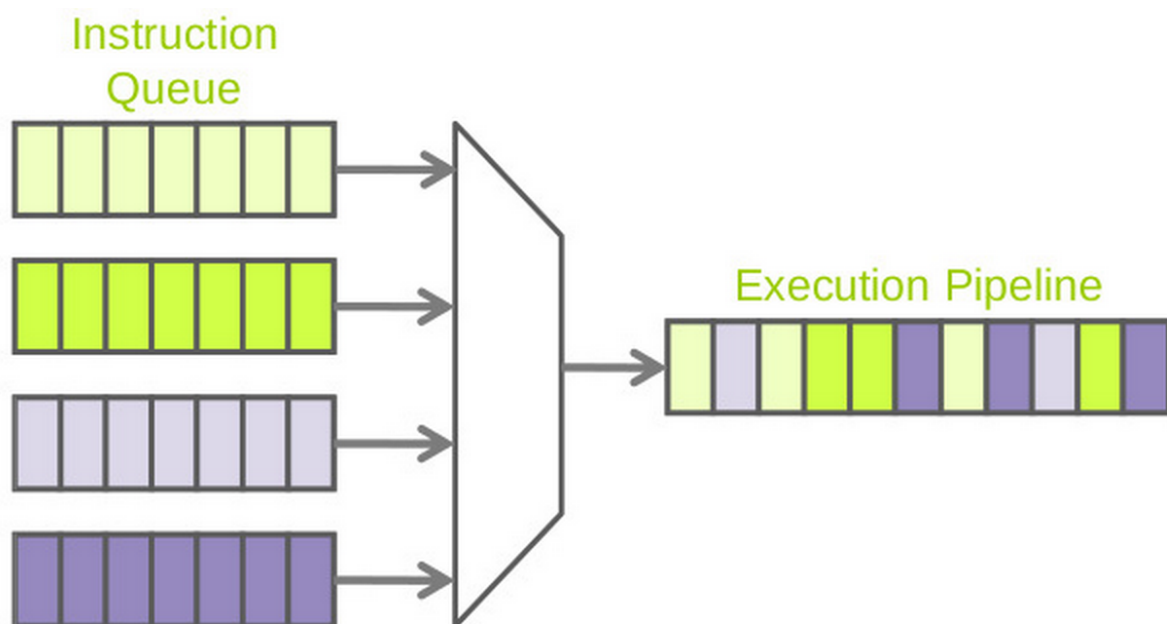
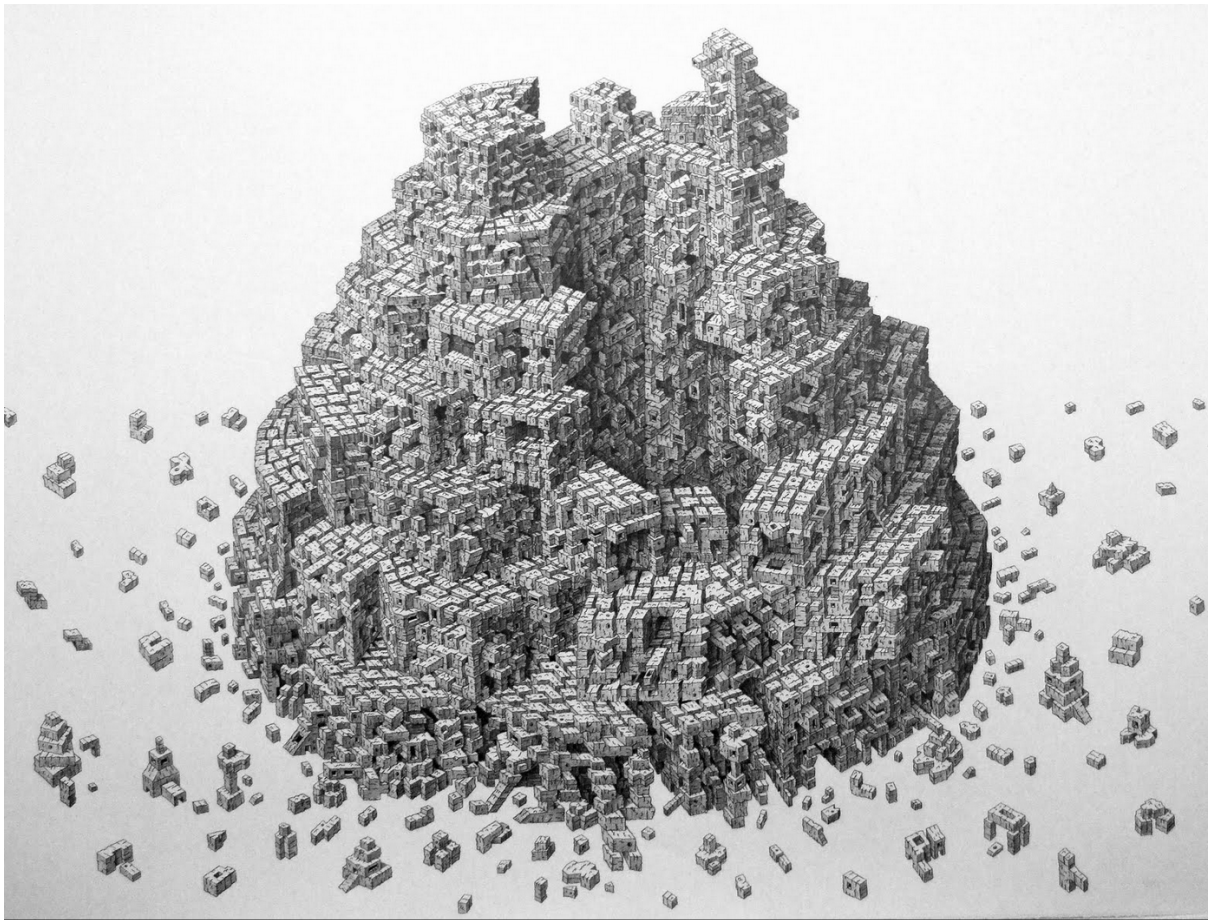


Drawing a word cloud (page 16)

Finding collocations (page 18)

Generating random text (page 19)

Multithreading with KoNLPy (page 21)



5.4.1 Contents

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:

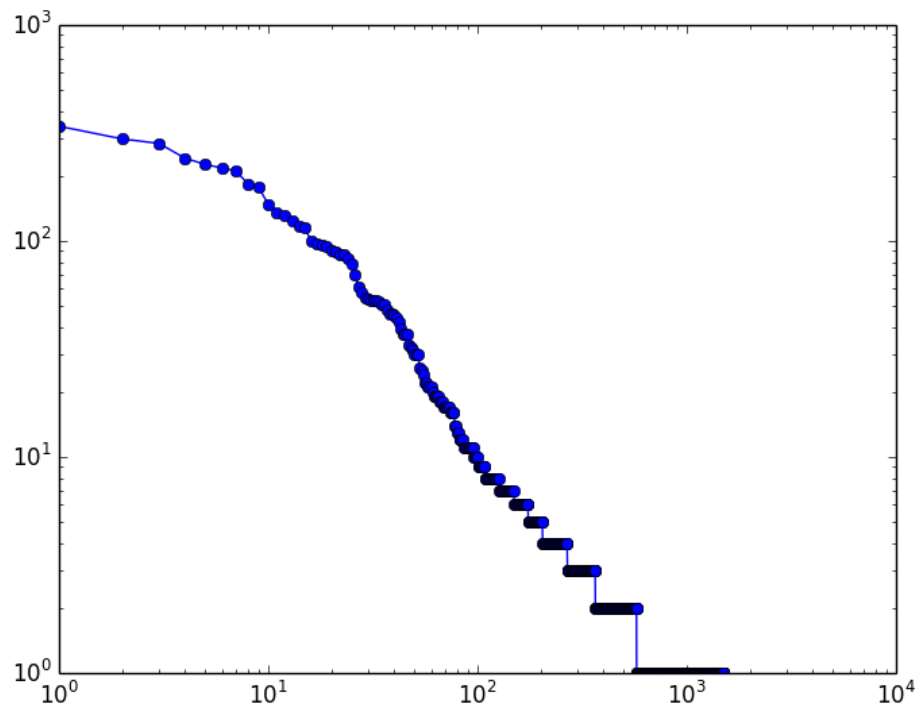
```
nchars  : 19240
ntokens : 4178
nmorphs  : 1501

Top 20 frequent morphemes:
[(('의', J), 398),
 (('.', S), 340),
 (('하', X), 297),
 (('예', J), 283),
 (('다', E), 242),
 (('ㄴ', E), 226),
 (('이', J), 218),
 (('을', J), 211),
 (('은', J), 184),
 (('어', E), 177),
 (('를', J), 148),
 (('르', 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 대한민국은 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:

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) (<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) (<https://github.com/e9t/PyTagCloud>).



In order to find trigram collocations, replace *BigramAssocMeasures* with *TrigramAssocMeasures*, and *BigramCollocationFinder* with *TrigramCollocationFinder*.

- Console:

Our “sentence” above was generated with “bigrams”, or “2-grams”. If we wish to make more sense out of it, we could try “3-grams” (better known as trigrams) or “4-grams”. Or, we could extend the same idea to longer sequences of letters, such as morphemes. Let’s try this with actual code.

Warning: The code below works with Python3, and not with Python2! You can run the code by typing *python3 generate.py* on your terminal.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import bisect
import itertools
import random

import nltk
from konlpy.corpus import kolaw
from konlpy.tag import Mecab # MeCab tends to reserve the original form of morphemes

def generate_sentence(cfdist, word, num=15):
    sentence = []

    # Generate words until we meet a period
    while word != '.':
        sentence.append(word)

        # Generate the next word based on probability
        choices, weights = zip(*cfdist[word].items())
        cumdist = list(itertools.accumulate(weights))
        x = random.random() * cumdist[-1]
        word = choices[bisect.bisect(cumdist, x)]

    return ' '.join(sentence)

def calc_cfd(doc):
    # Calculate conditional frequency distribution of bigrams
    words = [w for w, t in Mecab().pos(doc)]
    bigrams = nltk.bigrams(words)
    return nltk.ConditionalFreqDist(bigrams)

if __name__ == '__main__':
    nsents = 5 # Number of sentences
    initstr = u'국가' # Try replacing with u'국가', u'대통령', etc

    doc = kolaw.open('constitution.txt').read()
    cfd = calc_cfd(doc)

    for i in range(nsents):
        print('%d. %s' % (i, generate_sentence(cfd, initstr)))
```

• Console:

0. 국민은 법률로 인한 배상은 특별한 영장을 청구할 수 있어서 최고 득표자가 제출한 유일한 때에 의하
1. 국민투표의 범죄에 의하여 발언하지 아니한 회의는 요건은 1988년으로 대통령이 의결한다
2. 국민경제자문기구를 타파하기 위하여 긴급한 형태로 정한다
3. 국민은 이 정하는 헌법시행 당시의 심사할 수 있다
4. 국민의 기본질서를 진다

Well, that’s a lot better than the random string typed by our cat! The sentences look a bit ugly because there are whitespaces between all morphemes, whereas in actual Korean text, they would be stuck together. Also note that

this text generation model was built from a single document. If you were to build a model with a much larger corpus, you wouldn't even have to do morpheme analysis because you would have enough data for any potential *initstr*. Other than that, there are much more ways to improve this model! Feel free to experiment.

For more on generating text, you can refer to Jon Bently's [Programming Pearls \(Section 15.3\)](http://www.cs.bell-labs.com/cm/cs/pearls/sec153.html) (<http://www.cs.bell-labs.com/cm/cs/pearls/sec153.html>).

Furthermore, if you use [language models](http://en.wikipedia.org/wiki/Language_model) (http://en.wikipedia.org/wiki/Language_model), you can evaluate your random texts and figure out whether they actually make sense in a statistical point of view.

Multithreading with KoNLPy

Sometimes it gets boring to wait for tagging jobs to end. How about using some concurrency tricks? Python supports multithreading and multiprocessing out-of-the-box, and you can use them with KoNLPy as well. Here's an example using multithreading.

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

from konlpy.tag import Kkma
from konlpy.corpus import kolaw
from threading import Thread
import jpye

def do_concurrent_tagging(start, end, lines, result):
    jpye.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)
```

- Console:

```
Number of lines in document:
356
Batch tagging:
37.758173
```

```
Concurrent tagging:
8.037602
```

Check out how much faster it gets!

Note:

- **Some useful references on concurrency with Python:**

- 장혜식, “파이썬은 멀티코어 줘도 쓰잘데기가 없나요?”에 대한 파이썬 2.6의 대답 (<http://openlook.org/blog/2008/06/28/python-multiprocessing/>), 2008.
 - 하용호, 파이썬으로 클라우드 하고 싶어요 (<http://www.slideshare.net/devparan/h3-2011-c6-python-and-cloud>), 2011.
-

5.5 Running tests

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

```
$ pip install pytest
$ cd konlpy
$ python -m pytest test/* # for Python 2.x
$ python3 -m pytest test/* # for Python 3.x
```

Note: To see testing logs on KoNLPy, see [here](https://docs.google.com/spreadsheets/d/1Ii_L9NF9gSLbsJOGqsf-zfqTtyhhthmJWNC2kgUDIsU/edit?usp=sharing) (https://docs.google.com/spreadsheets/d/1Ii_L9NF9gSLbsJOGqsf-zfqTtyhhthmJWNC2kgUDIsU/edit?usp=sharing).

Note: To see known bugs/issues, see [here](https://github.com/konlpy/konlpy/labels/bug) (<https://github.com/konlpy/konlpy/labels/bug>).

5.6 References

Note: Please [modify this document](https://github.com/konlpy/konlpy/blob/master/docs/references.rst) (<https://github.com/konlpy/konlpy/blob/master/docs/references.rst>) if anything is erroneous or not included. Last updated at January 18, 2015.

5.6.1 Korean morpheme analyzer tools

When you're analyzing Korean text, the most basic task you need to perform is morphological analysis. There are several libraries in various programming languages to achieve this:

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) (<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)
 - Generates morpheme candidates using dynamic programming
 - Tags morphemes by checking neighbors, and employing some heuristics and HMM models
 - Developer blog: [Dongjoo Lee](http://therocks.tistory.com) (<http://therocks.tistory.com>)
- **KOMORAN** (<http://shineware.tistory.com/tag/KOMORAN>) (2013) Apache v2
 - By *shineware*

Python

- **KoNLPy** (<http://konlpy.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)

5.6.2 Other NLP tools

Language parser

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

5.6.3 Corpora

- **Yonsei Corpus, 연세대, 1987.**
 - 42M tokens of Korean since the 1960s
- **Korea University Korean Corpus, 1995.**
 - 10M tokens of Korean of 1970-90s
- **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 (88MB)
- **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.

5.6.4 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)
- **Sejong semantic search system** (<http://sejong21.org>)
- **한국어학회** (<http://koling.org>)
- **한글 및 한국어 정보처리 학술대회**

6.1 konlpy Package

6.1.1 Subpackages

tag Package

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

Hannanum Class

Warning: `Hannanum()` is not supported on Windows 7 [#7 (<https://github.com/konlpy/konlpy/issues/7>)].

```
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 흰 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 29).

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*, *flatten=True*)
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.morphs(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 29).

morphs (*phrase*)
Parse phrase to morphemes.

nouns (*phrase*)
Noun extractor.

pos (*phrase*, *flatten=True*)
POS tagger.

sentences (*phrase*)
Sentence detection.

Komorán Class

Warning: `Komorán()` is not supported on Python 2 + Mac OS [#40] (<https://github.com/konlpy/konlpy/issues/40>).

class `konlpy.tag._komoran.Komorán` (*jvmpath=None*, *dicpath=None*)
Wrapper for **KOMORAN** (<http://shineware.tistory.com/entry/KOMORAN-ver-24>).

KOMORAN is a relatively new open source Korean morphological analyzer written in Java, developed by **Shineware** (<http://shineware.co.kr>), since 2013.

```
from konlpy.tag import Komorán

komoran = Komorán()
print komoran.pos(u'우왕 코모란도 오픈소스가 되었어요')
```

Parameters

- **jvmpath** – The path of the JVM passed to `init_jvm()` (page 29).
- **dicpath** – The path of dictionary files. The KOMORAN system dictionary is loaded by default.

nouns (*phrase*)

Noun extractor.

pos (*phrase, flatten=True*)

POS tagger.

Mecab Class

Warning: `Mecab()` is not supported on 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 쇼핑물은 어떤 곳인가?')
print mecab.morphs(u'영등포구청역에 있는 맛집 좀 알려주세요.')
print mecab.nouns(u'우리나라에는 무릎 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, flatten=True*)

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. For a complete list of corpora available in KoNLPy, refer to *Corpora* (page 24).

```
>>> 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.

6.1.2 data Module

`konlpy.data.find(resource_url)`

Find the path of a given resource URL by searching through directories in `konlpy.data.path`. If the given resource is not found, raise a `LookupError`, whose message gives a pointer to the installation instructions for `konlpy.download()`.

Parameters **resource_url** (*str* (<http://docs.python.org/library/functions.html#str>)) – The URL of the resource to search for. URLs are posix-style relative path names, such as `corpora/kolaw`. In particular, directory names should always be separated by the forward slash character (i.e., `'/'`), which will be automatically converted to a platform-appropriate path separator by KoNLPy.

`konlpy.data.load(resource_url, format='auto')`

Load a given resource from the KoNLPy data package. If no format is specified, `load()` will attempt to determine a format based on the resource name's file extension. If that fails, `load()` will raise a `ValueError` exception.

Parameters

- **resource_url** (*str* (<http://docs.python.org/library/functions.html#str>)) – A URL specifying where the resource should be loaded from.
- **format** – Format type of resource.

`konlpy.data.path = ['/home/docs/konlpy_data', '/usr/share/konlpy_data', '/usr/local/share/konlpy_data', '/usr/lib/konlpy_data']`

A list of directories where the KoNLPy data package might reside. These directories will be checked in order when looking for a resource. Note that this allows users to substitute their own versions of resources.

class `konlpy.data.FileSystemPathPointer` (*path*)

Bases: `konlpy.data.PathPointer` (page 28), *str* (<http://docs.python.org/library/functions.html#str>)

A path pointer that identifies a file by an absolute path.

file_size ()

open (*encoding='utf-8'*)

class `konlpy.data.PathPointer`

Bases: `object` (<http://docs.python.org/library/functions.html#object>)

An abstract base class for path pointers. One subclass exists: 1. `FileSystemPathPointer`: Identifies a file by an absolute path.

file_size ()

open (*encoding='utf-8'*)

6.1.3 downloader Module

class `konlpy.downloader.Downloader` (*download_dir=None*)

Bases: `object` (<http://docs.python.org/library/functions.html#object>)

A class used to access the KoNLPy data server, which can be used to download packages.

INDEX_URL = `'http://konlpy.github.io/konlpy-data/index.json'`

INSTALLED = `'installed'`

NOT_INSTALLED = 'not installed'

PACKAGE_URL = 'http://konlpy.github.io/konlpy-data/packages/%s.%s'

SCRIPT_URL = 'http://konlpy.github.io/konlpy-data/packages/%s.sh'

STALE = 'corrupt or out of date'

download (*id=None, download_dir=None*)

The KoNLPy data downloader. With this module you can download corpora, models and other data packages that can be used with KoNLPy.

Individual packages can be downloaded by passing a single argument, the package identifier for the package that should be downloaded:

```
>>> download('corpus/kobill')
[konlpy_data] Downloading package 'kobill'...
[konlpy_data]   Unzipping corpora/kobill.zip.
```

To download all packages, simply call download with the argument 'all':

```
>>> download('all')
[konlpy_data] Downloading package 'kobill'...
[konlpy_data]   Unzipping corpora/kobill.zip.
...
```

status (*info_or_id=None, download_dir=None*)

Return a constant describing the local status of the given package. Status can be one of **INSTALLED**, **NOT_INSTALLED**, or **STALE**.

konlpy.downloader.default_download_dir ()

Returns the directory to which packages will be downloaded by default. This value can be overridden using the constructor, or on a case-by-case basis using the `download_dir` argument when calling `download()`.

On Windows, the default download directory is `PYTHONHOME/lib/konlpy`, where *PYTHONHOME* is the directory containing Python e.g., `C:\Python27`.

On all other platforms, the default directory is the first of the following which exists or which can be created with write permission: `/usr/share/konlpy_data`, `/usr/local/share/konlpy_data`, `/usr/lib/konlpy_data`, `/usr/local/lib/konlpy_data`, `~/konlpy_data`.

6.1.4 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 `jpye.getDefaultJVMPath()`.

6.1.5 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*)

Overridden 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      대한민국은 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.csvread (f, encoding=u'utf-8')`

Reads a csv file.

Parameters **f** – File object.

```
>>> from konlpy.utils import csvread
>>> with open('some.csv', 'r') as f:
>>>     print csvread(f)
[[u'이 / NR', u'차 / NNB'], [u'나가 / VV', u'네 / EFN']]
```

`konlpy.utils.csvwrite (data, f)`

Writes a csv file.

Parameters **data** – A list of list.

```
>>> from konlpy.utils import csvwrite
>>> d = [[u'이 / NR', u'차 / NNB'], [u'나가 / VV', u'네 / EFN']]
>>> with open('some.csv', 'w') as f:
>>>     csvwrite(d, f)
```

`konlpy.utils.hex2char (h)`

Converts a hex character to unicode.

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

`konlpy.utils.load_txt (filename, encoding=u'utf-8')`

Text file loader. To read a file, use “`read_txt()`” instead.

`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'유니코드', 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 29)

`konlpy.utils.read_txt(filename, encoding=u'utf-8')`

Text file reader.

`konlpy.utils.select(phrase)`

Replaces some ambiguous punctuation marks to simpler ones.

Indices and tables

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

k

`konlpy.corpus`, [27](#)
`konlpy.data`, [28](#)
`konlpy.downloader`, [28](#)
`konlpy.jvm`, [29](#)
`konlpy.tag._hannanum`, [25](#)
`konlpy.tag._kkma`, [26](#)
`konlpy.tag._komoran`, [26](#)
`konlpy.tag._mecab`, [27](#)
`konlpy.utils`, [29](#)

A

abspath() (konlpy.corpus.CorpusLoader method), 27
 analyze() (konlpy.tag._hannanum.Hannanum method), 25

C

char2hex() (in module konlpy.utils), 29
 concat() (in module konlpy.utils), 29
 concordance() (in module konlpy.utils), 30
 CorpusLoader (class in konlpy.corpus), 27
 csvread() (in module konlpy.utils), 30
 csvwrite() (in module konlpy.utils), 30

D

default_download_dir() (in module konlpy.downloader), 29
 download() (konlpy.downloader.Downloader method), 29
 Downloader (class in konlpy.downloader), 28

F

file_size() (konlpy.data.FileSystemPathPointer method), 28
 file_size() (konlpy.data.PathPointer method), 28
 fileids() (konlpy.corpus.CorpusLoader method), 28
 FileSystemPathPointer (class in konlpy.data), 28
 find() (in module konlpy.data), 28
 format() (konlpy.utils.UnicodePrinter method), 29

H

Hannanum (class in konlpy.tag._hannanum), 25
 hex2char() (in module konlpy.utils), 30

I

INDEX_URL (konlpy.downloader.Downloader attribute), 28
 init_jvm() (in module konlpy.jvm), 29
 INSTALLED (konlpy.downloader.Downloader attribute), 28

K

Kkma (class in konlpy.tag._kkma), 26
 Komoran (class in konlpy.tag._komoran), 26
 konlpy.corpus (module), 27

konlpy.data (module), 28
 konlpy.downloader (module), 28
 konlpy.jvm (module), 29
 konlpy.tag._hannanum (module), 25
 konlpy.tag._kkma (module), 26
 konlpy.tag._komoran (module), 26
 konlpy.tag._mecab (module), 27
 konlpy.utils (module), 29

L

load() (in module konlpy.data), 28
 load_txt() (in module konlpy.utils), 30

M

Mecab (class in konlpy.tag._mecab), 27
 morphs() (konlpy.tag._hannanum.Hannanum method), 25
 morphs() (konlpy.tag._kkma.Kkma method), 26
 morphs() (konlpy.tag._mecab.Mecab method), 27

N

NOT_INSTALLED (konlpy.downloader.Downloader attribute), 28
 nouns() (konlpy.tag._hannanum.Hannanum method), 25
 nouns() (konlpy.tag._kkma.Kkma method), 26
 nouns() (konlpy.tag._komoran.Komoran method), 26
 nouns() (konlpy.tag._mecab.Mecab method), 27

O

open() (konlpy.corpus.CorpusLoader method), 28
 open() (konlpy.data.FileSystemPathPointer method), 28
 open() (konlpy.data.PathPointer method), 28

P

PACKAGE_URL (konlpy.downloader.Downloader attribute), 29
 partition() (in module konlpy.utils), 30
 path (in module konlpy.data), 28
 PathPointer (class in konlpy.data), 28
 pos() (konlpy.tag._hannanum.Hannanum method), 25
 pos() (konlpy.tag._kkma.Kkma method), 26
 pos() (konlpy.tag._komoran.Komoran method), 27
 pos() (konlpy.tag._mecab.Mecab method), 27

`pprint()` (in module `konlpy.utils`), [31](#)
`preprocess()` (in module `konlpy.utils`), [31](#)

R

`read_txt()` (in module `konlpy.utils`), [31](#)

S

`SCRIPT_URL` (`konlpy.downloader.Downloader` attribute), [29](#)

`select()` (in module `konlpy.utils`), [31](#)

`sentences()` (`konlpy.tag._kkma.Kkma` method), [26](#)

`STALE` (`konlpy.downloader.Downloader` attribute), [29](#)

`status()` (`konlpy.downloader.Downloader` method), [29](#)

U

`UnicodePrinter` (class in `konlpy.utils`), [29](#)