fast campus

Fast Campus **Online** | Life Changing **Education**

Chapter 04. 자연어처리 (Natural Language Processing)

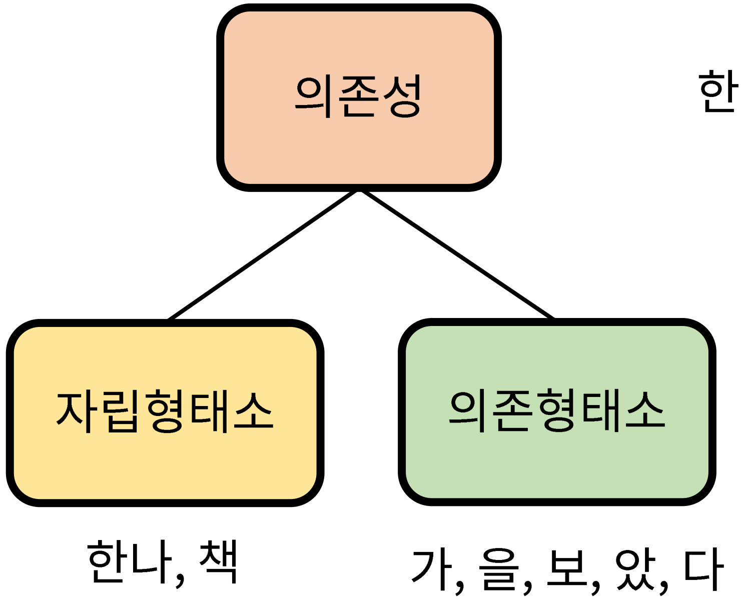
형태소 분석기

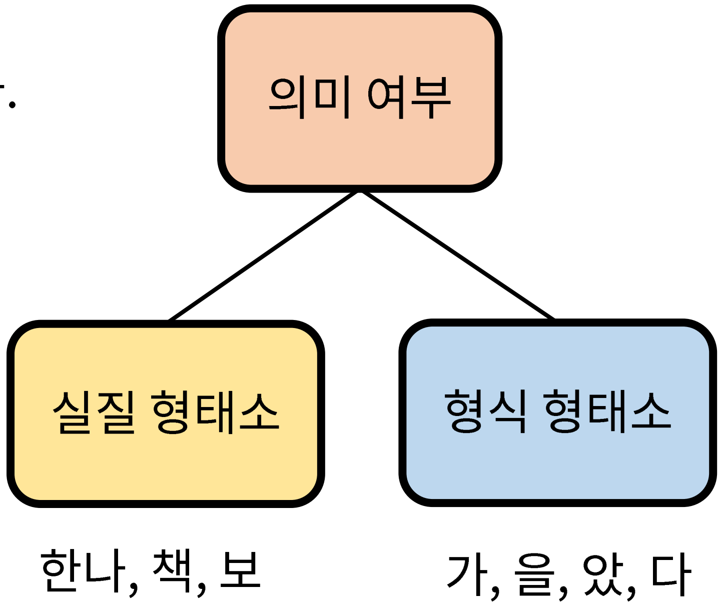
Copyright **FASTCAMPUS** Corp. All Rights Reserved

형태소 Morpheme

형태소: 언어학적으로 말을 분석할 때, 의미가 있는 가장 작은 말의 단위

한나가 책을 보았다.

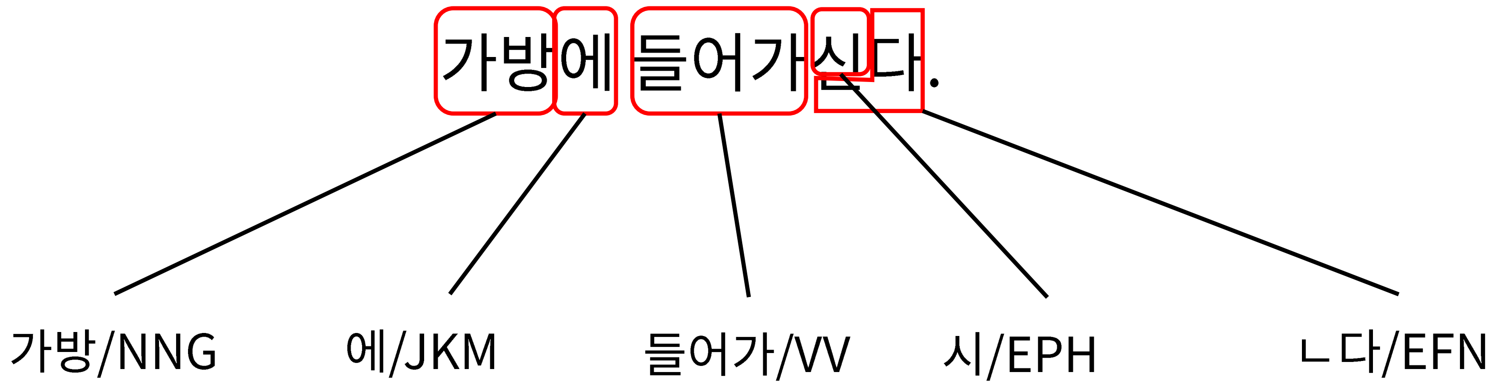




다양한 방법으로 형태소의 종류를 나누어 볼 수 있다.

Copyright FASTCAMPUS Corp. All Rights Reserved

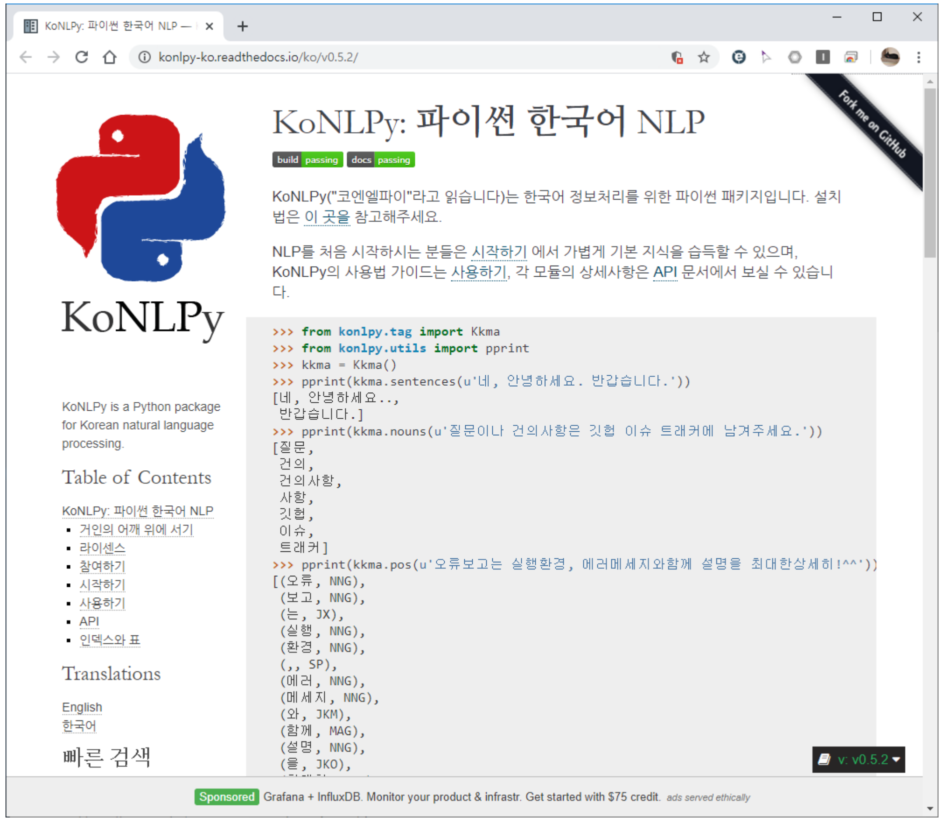
형태소 분석



문장을 형태소 단위로 구분하고, 언어적인 구조를 파악하는 것을 형태소 분석이라 한다.
어근, 접두사/접미사, 품사(POS; Part-of Speech) 등을 구분한다.

Copyright **FASTCAMPUS** Corp. All Rights Reserved

KoNLPy


KoNLPy: 파이썬 한국어 NLP —
konlpy-ko.readthedocs.io/ko/v0.5.2/
Fork me on GitHub
build passing
docs passing
KoNLPy
v: v0.5.2

KoNLPy: 파이썬 한국어 NLP

KoNLPy("코엔엘파이"라고 읽습니다)는 한국어 정보처리를 위한 파이썬 패키지입니다. 설치법은 이 곳을 참고해주세요.

NLP를 처음 시작하시는 분들은 시작하기 에서 가볍게 기본 지식을 습득할 수 있으며, KoNLPy의 사용법 가이드는 사용하기, 각 모듈의 상세사항은 API 문서에서 보실 수 있습니다.

```
>>> 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),
```

KoNLPy is a Python package for Korean natural language processing.

Table of Contents

Translations

English
한국어

빠른 검색

Sponsored Grafana + InfluxDB. Monitor your product & infrastr. Get started with $75 credit. *ads served ethically*

- Hannanum
- Kkma
- Komoran
- MeCab-Ko
- OKT(Twitter)

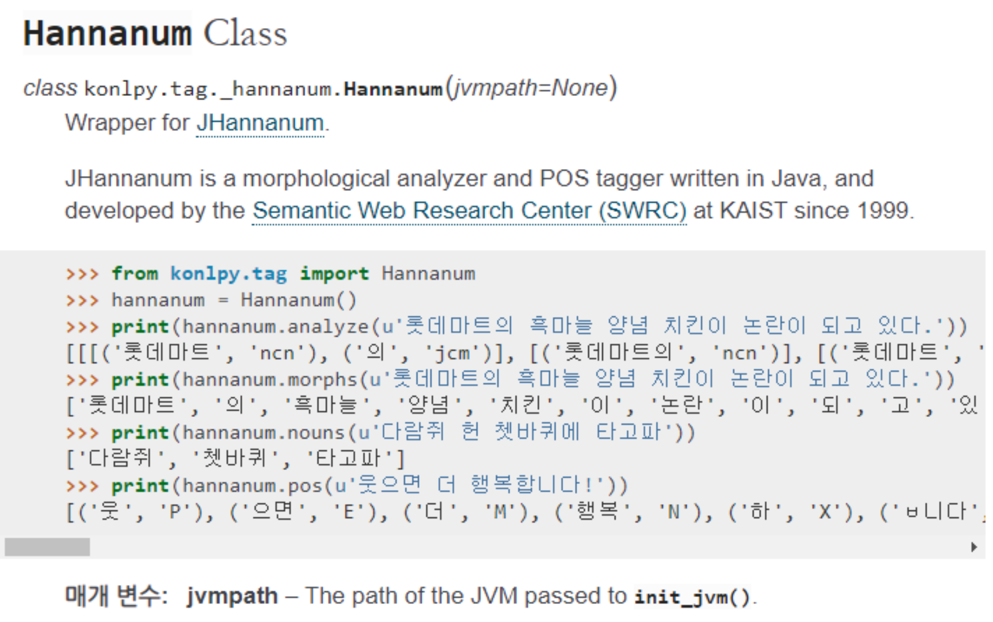
KoNLPy는 Java Script로 개발된 다양한 형태소 분석기의 Python Wrapper이다.

다양한 한글 형태소 분석기를 통일된 방법으로 쉽게 사용할 수 있다.

fast campus

Copyright **FASTCAMPUS** Corp. All Rights Reserved

Hannanum



Hannanum Class

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

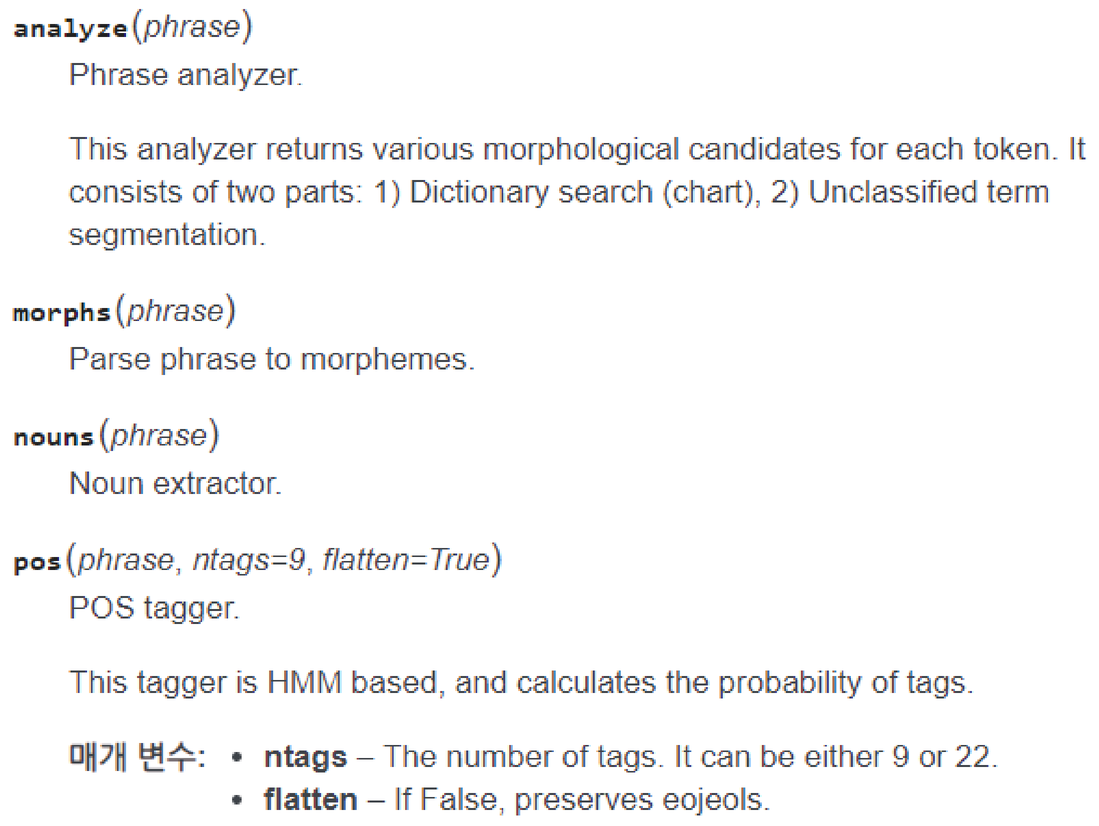
Wrapper for JHannanum.

JHannanum is a morphological analyzer and POS tagger written in Java, and developed by the Semantic Web Research Center (SWRC) at KAIST since 1999.

```
>>> from konlpy.tag import Hannanum
>>> hannanum = Hannanum()
>>> print(hannanum.analyze(u'롯데마트의 흑마늘 양념 치킨이 논란이 되고 있다.'))
[[[('롯데마트', 'ncn'), ('의', 'jcm')], [('롯데마트의', 'ncn')], [('롯데마트', '
>>> print(hannanum.morphs(u'롯데마트의 흑마늘 양념 치킨이 논란이 되고 있다.'))
['롯데마트', '의', '흑마늘', '양념', '치킨', '이', '논란', '이', '되', '고', '있
>>> print(hannanum.nouns(u'다람쥐 헌 쳇바퀴에 타고파'))
['다람쥐', '쳇바퀴', '타고파']
>>> print(hannanum.pos(u'웃으면 더 행복합니다!'))
[('웃', 'P'), ('으면', 'E'), ('더', 'M'), ('행복', 'N'), ('하', 'X'), ('ㅂ니다',
```

매개 변수: **jvmpath** – The path of the JVM passed to `init_jvm()`.

Class initializer



`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.

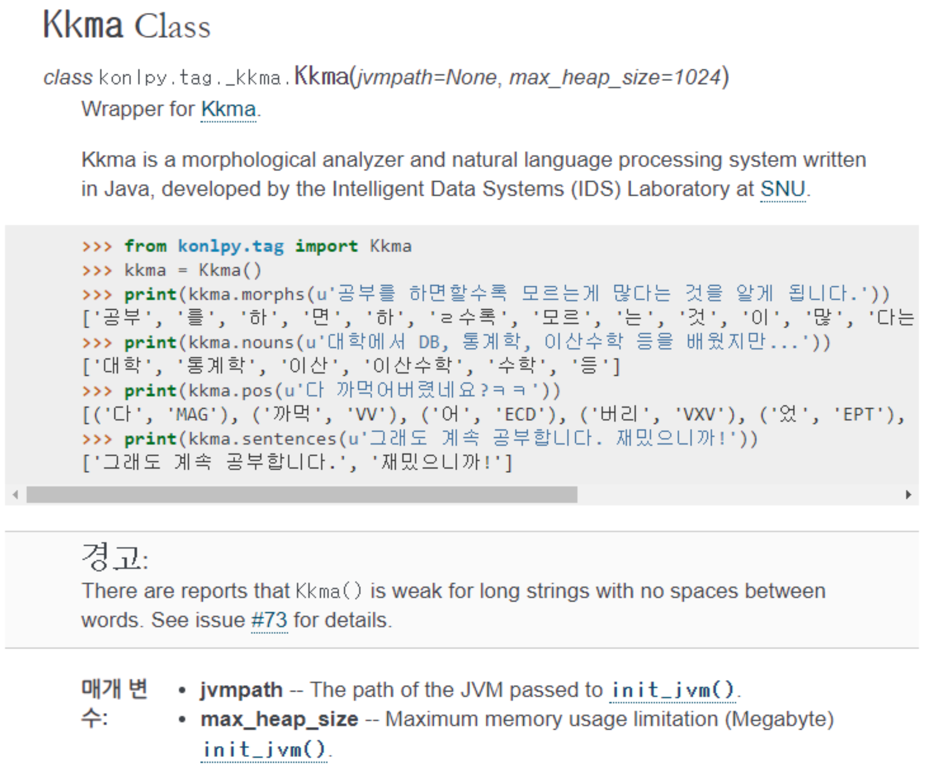
This tagger is HMM based, and calculates the probability of tags.

매개 변수:
- **ntags** – The number of tags. It can be either 9 or 22.
- **flatten** – If False, preserves eojeols.

Methods

Copyright FASTCAMPUS Corp. All Rights Reserved

Kkma



Kkma Class

class konlpy.tag._kkma.Kkma(*jvmpath=None*, *max_heap_size=1024*)

Wrapper for Kkma.

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

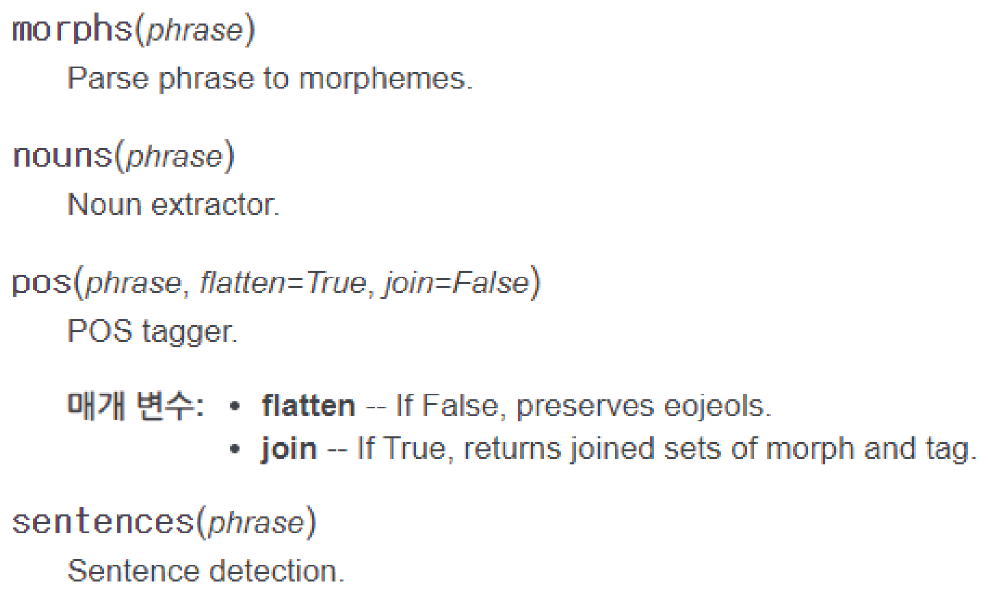
```
>>> from konlpy.tag import Kkma
>>> kkma = Kkma()
>>> print(kkma.morphs(u'공부를 하면할수록 모르는게 많다는 것을 알게 됩니다.'))
['공부', '를', '하', '면', '하', 'ㄹ수록', '모르', '는', '것', '이', '많', '다는
>>> print(kkma.nouns(u'대학에서 DB, 통계학, 이산수학 등을 배웠지만...'))
['대학', '통계학', '이산', '이산수학', '수학', '등']
>>> print(kkma.pos(u'다 까먹어버렸네요?ㅋㅋ'))
[('다', 'MAG'), ('까먹', 'VV'), ('어', 'ECD'), ('버리', 'VXV'), ('었', 'EPT'),
>>> print(kkma.sentences(u'그래도 계속 공부합니다. 재밌으니까!'))
['그래도 계속 공부합니다.', '재밌으니까!']
```

경고:

There are reports that Kkma() is weak for long strings with no spaces between words. See issue #73 for details.

매개 변수:
- **jvmpath** -- The path of the JVM passed to init_jvm().
- **max_heap_size** -- Maximum memory usage limitation (Megabyte) init_jvm().

Class initializer



morphs(*phrase*)

Parse phrase to morphemes.

nouns(*phrase*)

Noun extractor.

pos(*phrase*, *flatten=True*, *join=False*)

POS tagger.

매개 변수:
- **flatten** -- If False, preserves eojeols.
- **join** -- If True, returns joined sets of morph and tag.

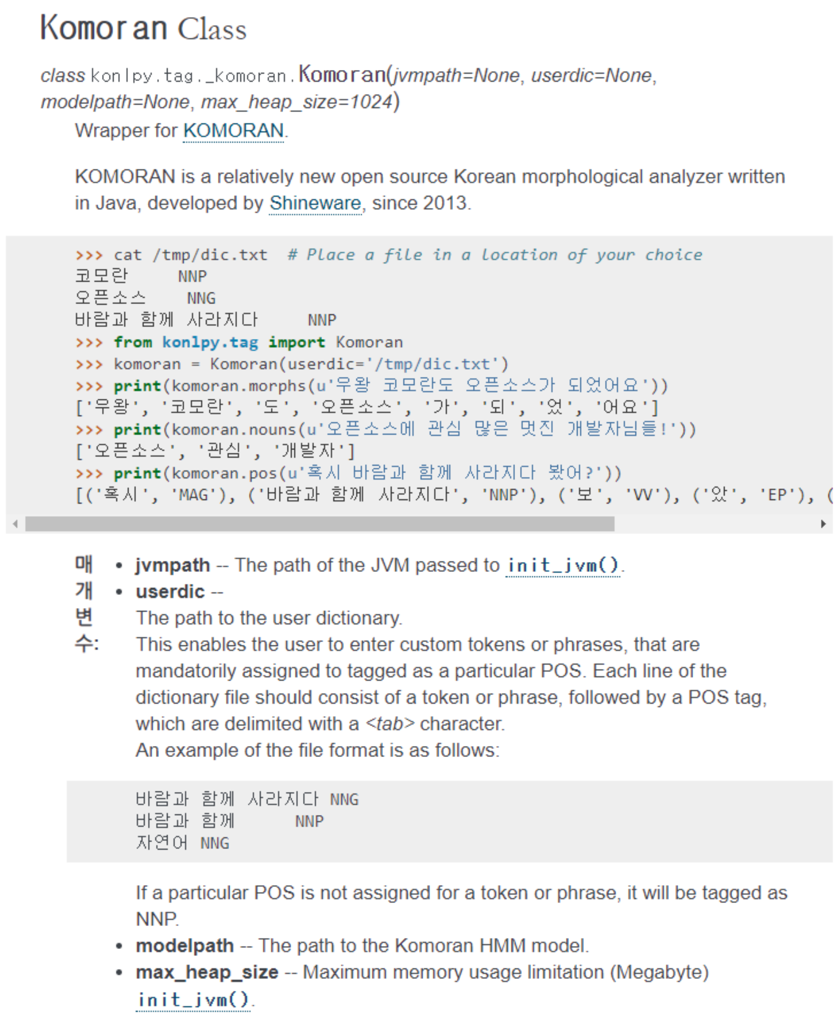
sentences(*phrase*)

Sentence detection.

Methods

Copyright FASTCAMPUS Corp. All Rights Reserved

Komoran



Komoran Class

class konlpy.tag._komoran.Komoran(*jvmpath=None*, *userdic=None*, *modelpath=None*, *max_heap_size=1024*)

Wrapper for KOMORAN.

KOMORAN is a relatively new open source Korean morphological analyzer written in Java, developed by Shineware, since 2013.

```
>>> cat /tmp/dic.txt  # Place a file in a location of your choice
코모란     NNP
오픈소스     NNG
바람과 함께 사라지다     NNP
>>> from konlpy.tag import Komoran
>>> komoran = Komoran(userdic='/tmp/dic.txt')
>>> print(komoran.morphs(u'우왕 코모란도 오픈소스가 되었어요'))
['우왕', '코모란', '도', '오픈소스', '가', '되', '었', '어요']
>>> print(komoran.nouns(u'오픈소스에 관심 많은 멋진 개발자님들!'))
['오픈소스', '관심', '개발자']
>>> print(komoran.pos(u'혹시 바람과 함께 사라지다 봤어?'))
[('혹시', 'MAG'), ('바람과 함께 사라지다', 'NNP'), ('보', 'VV'), ('았', 'EP'), (
```

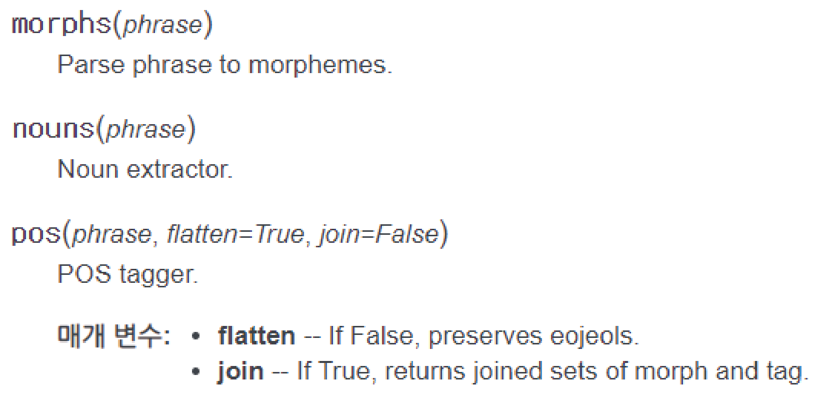
매개변수:

- **jvmpath** -- The path of the JVM passed to `init_jvm()`.
- **userdic** --
  The path to the user dictionary.
  This enables the user to enter custom tokens or phrases, that are mandatorily assigned to tagged as a particular POS. Each line of the dictionary file should consist of a token or phrase, followed by a POS tag, which are delimited with a *<tab>* character.
  An example of the file format is as follows:

  ```
  바람과 함께 사라지다 NNG
  바람과 함께      NNP
  자연어 NNG
  ```

  If a particular POS is not assigned for a token or phrase, it will be tagged as NNP.
- **modelpath** -- The path to the Komoran HMM model.
- **max_heap_size** -- Maximum memory usage limitation (Megabyte) `init_jvm()`.

Class initializer



morphs(*phrase*)

Parse phrase to morphemes.

nouns(*phrase*)

Noun extractor.

pos(*phrase*, *flatten=True*, *join=False*)

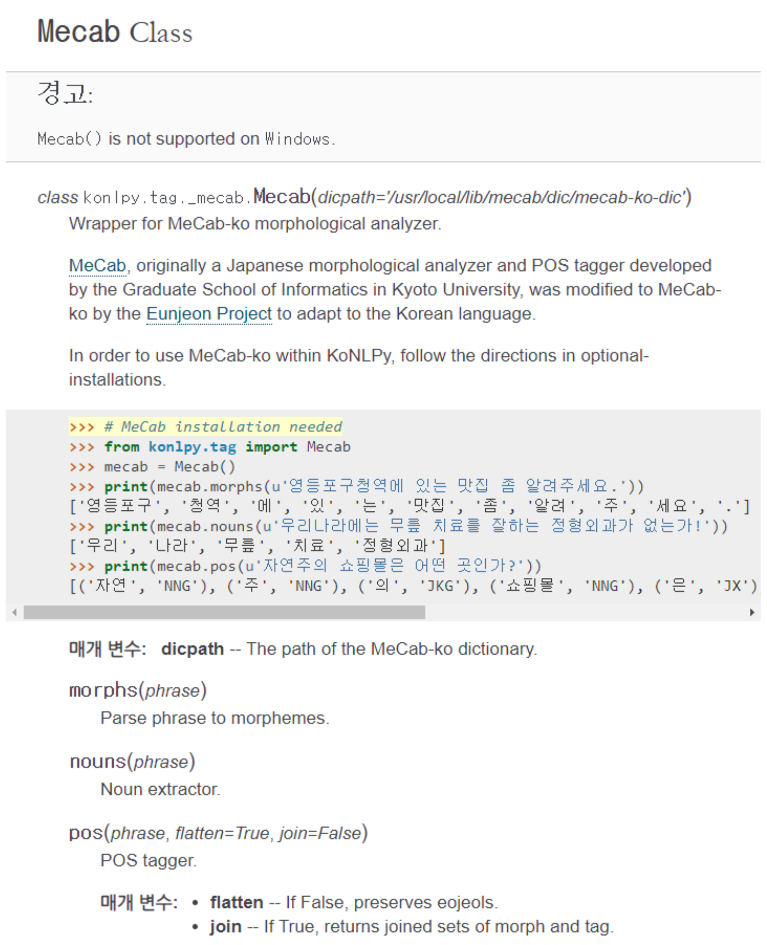
POS tagger.

매개 변수:
- **flatten** -- If False, preserves eojeols.
- **join** -- If True, returns joined sets of morph and tag.

Methods

Copyright FASTCAMPUS Corp. All Rights Reserved

Mecab



Mecab Class

경고:

`Mecab()` is not supported on `Windows`.

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

Wrapper for MeCab-ko morphological analyzer.

MeCab, originally a Japanese morphological analyzer and POS tagger developed by the Graduate School of Informatics in Kyoto University, was modified to MeCab-ko by the Eunjeon Project to adapt to the Korean language.

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

```
>>> # MeCab installation needed
>>> from konlpy.tag import Mecab
>>> mecab = Mecab()
>>> print(mecab.morphs(u'영등포구청역에 있는 맛집 좀 알려주세요.'))
['영등포구', '청역', '에', '있', '는', '맛집', '좀', '알려', '주', '세요', '.']
>>> print(mecab.nouns(u'우리나라에는 무릎 치료를 잘하는 정형외과가 없는가!'))
['우리', '나라', '무릎', '치료', '정형외과']
>>> print(mecab.pos(u'자연주의 쇼핑몰은 어떤 곳인가?'))
[('자연', 'NNG'), ('주', 'NNG'), ('의', 'JKG'), ('쇼핑몰', 'NNG'), ('은', 'JX')
```

매개 변수: **dicpath** -- The path of the MeCab-ko dictionary.

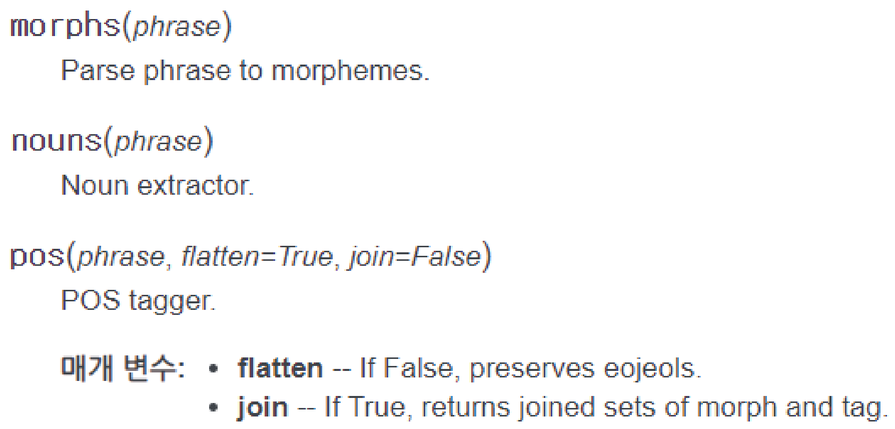
morphs(*phrase*)
Parse phrase to morphemes.

nouns(*phrase*)
Noun extractor.

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

매개 변수:
- **flatten** -- If False, preserves eojeols.
- **join** -- If True, returns joined sets of morph and tag.

Class initializer



morphs(*phrase*)
Parse phrase to morphemes.

nouns(*phrase*)
Noun extractor.

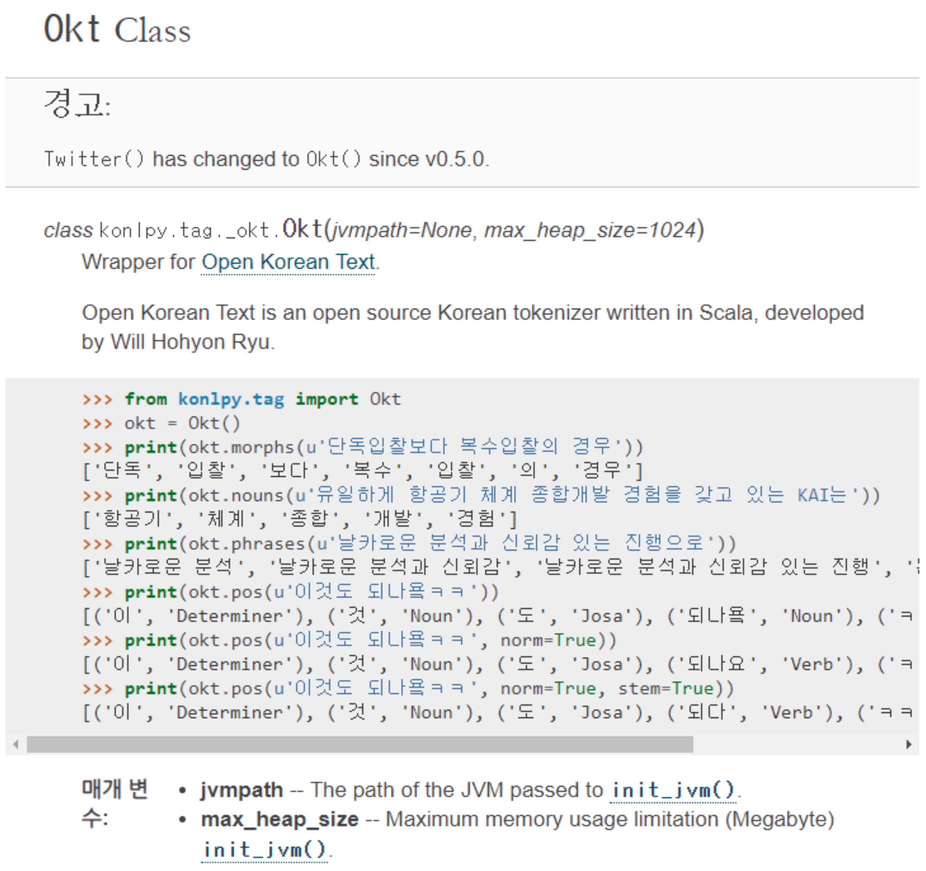
pos(*phrase*, *flatten=True*, *join=False*)
POS tagger.

매개 변수:
- **flatten** -- If False, preserves eojeols.
- **join** -- If True, returns joined sets of morph and tag.

Methods

Copyright FASTCAMPUS Corp. All Rights Reserved

Twitter



Okt Class

경고:

Twitter() has changed to Okt() since v0.5.0.

class konlpy.tag._okt.Okt(*jvmpath=None*, *max_heap_size=1024*)

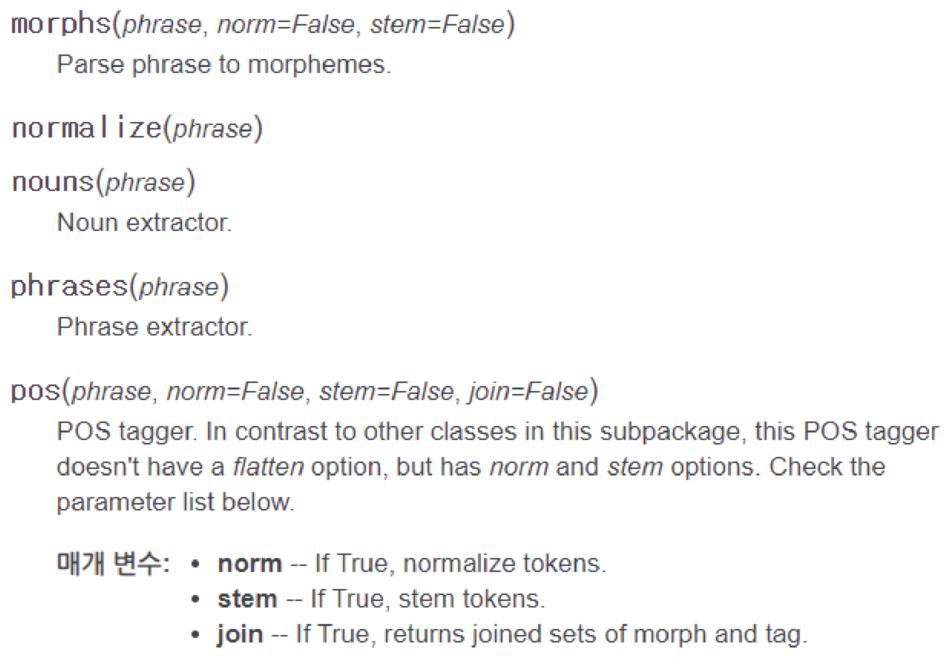
Wrapper for Open Korean Text.

Open Korean Text is an open source Korean tokenizer written in Scala, developed by Will Hohyon Ryu.

```
>>> from konlpy.tag import Okt
>>> okt = Okt()
>>> print(okt.morphs(u'단독입찰보다 복수입찰의 경우'))
['단독', '입찰', '보다', '복수', '입찰', '의', '경우']
>>> print(okt.nouns(u'유일하게 항공기 체계 종합개발 경험을 갖고 있는 KAI는'))
['항공기', '체계', '종합', '개발', '경험']
>>> print(okt.phrases(u'날카로운 분석과 신뢰감 있는 진행으로'))
['날카로운 분석', '날카로운 분석과 신뢰감', '날카로운 분석과 신뢰감 있는 진행', '
>>> print(okt.pos(u'이것도 되나욬ㅋㅋ'))
[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되나욬', 'Noun'), ('ㅋ
>>> print(okt.pos(u'이것도 되나욬ㅋㅋ', norm=True))
[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되나요', 'Verb'), ('ㅋ
>>> print(okt.pos(u'이것도 되나욬ㅋㅋ', norm=True, stem=True))
[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되다', 'Verb'), ('ㅋㅋ
```

매개 변수:
- **jvmpath** -- The path of the JVM passed to init_jvm().
- **max_heap_size** -- Maximum memory usage limitation (Megabyte) init_jvm().

Class initializer



morphs(*phrase*, *norm=False*, *stem=False*)

Parse phrase to morphemes.

normalize(*phrase*)

nouns(*phrase*)

Noun extractor.

phrases(*phrase*)

Phrase extractor.

pos(*phrase*, *norm=False*, *stem=False*, *join=False*)

POS tagger. In contrast to other classes in this subpackage, this POS tagger doesn't have a *flatten* option, but has *norm* and *stem* options. Check the parameter list below.

매개 변수:
- **norm** -- If True, normalize tokens.
- **stem** -- If True, stem tokens.
- **join** -- If True, returns joined sets of morph and tag.

Methods

Copyright FASTCAMPUS Corp. All Rights Reserved