
CKIPNLP

Release v0.5.2

Jul 25, 2019

1	Introduction	1
1.1	Git	1
1.2	PyPI	1
1.3	Author	1
1.4	Documentation	1
1.5	Requirements	1
2	Installation	3
2.1	Install Using Pip	3
2.2	Installation Options	4
3	Usage	5
3.1	CKIPWS	5
3.2	CKIP-Parser	5
3.3	Utilities	6
4	FAQ	7
5	License	9
6	ckipnlp.ws package	11
7	ckipnlp.parser package	13
8	ckipnlp.util package	15
8.1	Submodules	15
9	Todo List	19
10	Index	21
11	Module Index	23
	Python Module Index	25
	Index	27

CHAPTER 1

Introduction

1.1 Git

<https://github.com/emfomy/ckipnlp>

1.2 PyPI

<https://pypi.org/project/ckipnlp>

1.3 Author

- Mu Yang <emfomy@gmail.com>

1.4 Documentation

<http://ckipnlp.readthedocs.io/>

1.5 Requirements

- Python 3.5+
- Cython 0.29+

Note: For Python 2 users, please use [PyCkip 0.4.2](#) instead.

1.5.1 CkipWs (Optional)

- [CKIP Word Segmentation](#) Linux version 20190524+

1.5.2 CkipParser (Optional)

- [CKIP Parser](#) Linux version 20190506+ (20190725+ recommended)

Denote `<ckipws-linux-root>` as the root path of CKIPWS Linux Version, and `<ckipparser-linux-root>` as the root path of CKIP-Parser Linux Version.

2.1 Install Using Pip

```
pip install --upgrade ckipnlp
pip install --no-deps --force-reinstall --upgrade ckipnlp \
  --install-option='--ws' \
  --install-option='--ws-dir=<ckipws-linux-root>' \
  --install-option='--parser' \
  --install-option='--parser-dir=<ckipparser-linux-root>'
```

Ignore ws/parser options if one doesn't have CKIPWS/CKIP-Parser.

2.2 Installation Options

Option	Detail	Default Value
--[no-]ws	Enable/disable CKIPWS.	False
--[no-]parser	Enable/disable CKIP-Parser.	False
--ws-dir=<ws-dir>	CKIPWS root directory.	
--ws-lib-dir=<ws-lib-dir>	CKIPWS libraries directory	<ws-dir>/lib
--ws-share-dir=<ws-share-dir>	CKIPWS share directory	<ws-dir>
--parser-dir=<parser-dir>	CKIP-Parser root directory.	
--parser-lib-dir=<parser-lib-dir>	CKIP-Parser libraries directory	<parser-dir>/lib
--parser-share-dir=<parser-share-dir>	CKIP-Parser share directory	<parser-dir>
--data2-dir=<data2-dir>	“Data2” directory	<ws-share-dir>/Data2
--rule-dir=<rule-dir>	“Rule” directory	<parser-share-dir>/Rule
--rdb-dir=<rdb-dir>	“RDB” directory	<parser-share-dir>/RDB

See <http://ckipnlp.readthedocs.io/> for API details.

3.1 CKIPWS

```
import ckipnlp.ws
print(ckipnlp.__name__, ckipnlp.__version__)

ws = ckipnlp.ws.CkipWs(logger=False)
print(ws(''))
for l in ws.apply_list(['', '']): print(l)

ws.apply_file(ifile='sample/sample.txt', ofile='output/sample.tag', uwfile='output/
↪sample.uw')
with open('output/sample.tag') as fin:
    print(fin.read())
with open('output/sample.uw') as fin:
    print(fin.read())
```

3.2 CKIP-Parser

```
import ckipnlp.parser
print(ckipnlp.__name__, ckipnlp.__version__)

ps = ckipnlp.parser.CkipParser(logger=False)
print(ps(''))
for l in ps.apply_list(['', '']): print(l)

ps = ckipnlp.parser.CkipParser(logger=False)
print(ps(''))
```

(continues on next page)

(continued from previous page)

```
for l in ps.apply_list(['', '']): print(l)
ps.apply_file(ifile='sample/sample.txt', ofile='output/sample.tree')
with open('output/sample.tree') as fin:
    print(fin.read())
```

3.3 Utilities

```
import ckipnlp
print(ckipnlp.__name__, ckipnlp.__version__)

from ckipnlp.util.ws import *
from ckipnlp.util.parser import *

# Format CkipWs output
ws_text = ['(Na) (T)', '(I) (D)']
for text in ws_text: print(ckipnlp.util.ws.WsSentence.from_text(text))
for text in ws_text: print(repr(ckipnlp.util.ws.WsSentence.from_text(text)))

# Show CkipParser output as tree
tree_text =
↳ 'S(theme:NP (property:N(head:Nhaa:|Head:DE:)|Head:Nad(DUMMY1:Nab:|Head:Caa:|DUMMY2:Naa:))|quantity:I
↳ '
tree = ParserTree.from_text(tree_text)
tree.show()

# Get dummies of node 5
for node in tree.get_dummies(5): print(node)

# Get heads of node 1
for node in tree.get_heads(1): print(node)

# Get relations
for r in tree.get_relations(0): print(r)
```

CHAPTER 4

FAQ

Warning: The CKIPWS throws “what(): locale::facet::_S_create_c_locale name not valid”. What should I do?

Install locale data.

```
apt-get install locales-all
```

Warning: The CKIPParser throws “ImportError: libCKIPParser.so: cannot open shared object file: No such file or directory”. What should I do?

Add below command to ~/.bashrc:

```
export LD_LIBRARY_PATH=<ckipparser-linux-root>/lib:$LD_LIBRARY_PATH
```


CHAPTER 5

License

- MIT License

class `ckipnlp.ws.CkipWs` (*, *logger=False*, *infile=None*, ***options*)

Bases: `object`

The CKIP word segmentation driver.

Parameters

- **logger** (*bool*) – enable logger.
- **infile** (*str*) – the path to the INI file.
- **options** – the options, see `ckipnlp.util.ini.create_ws_ini()`.

apply (*text*)

Segment a sentence.

Parameters **text** (*str*) – the input sentence.

Returns *str* – the output sentence.

Notes

One may also call this method as `__call__()`.

apply_list (*ilist*)

Segment a list of sentences.

Parameters **ilist** (*list*) – the list of input sentences (*str*).

Returns **olist** (*list*) – the list of output sentences (*str*).

apply_file (*ifile*, *ofile*, *uwfile=""*)

Segment a file.

Parameters

- **ifile** (*str*) – the input file.
- **ofile** (*str*) – the output file (will be overwritten).

- **uwfile** (*str*) – the unknown word file (will be overwritten).

class `ckipnlp.parser.CkipParser` (*, *logger=False*, *infile=None*, *wsinfile=None*, ***options*)

Bases: `object`

The CKIP sentence parsing driver.

Parameters

- **logger** (*bool*) – enable logger.
- **infile** (*str*) – the path to the INI file.
- **wsinfile** (*str*) – the path to the INI file for CKIPWS.
- **options** – the options, see `ckipnlp.util.ini.create_ws_ini()` and `ckipnlp.util.ini.create_parser_ini()`

apply (*text*)

Segment a sentence.

Parameters **text** (*str*) – the input sentence.

Returns *str* – the output sentence.

Notes

One may also call this method as `__call__()`.

apply_list (*ilist*)

Segment a list of sentences.

Parameters **ilist** (*list*) – the list of input sentences (*str*).

Returns **olist** (*list*) – the list of output sentences (*str*).

apply_file (*ifile*, *ofile*)

Segment a file.

Parameters

- **ifile** (*str*) – the input file.
- **ofile** (*str*) – the output file (will be overwritten).

8.1 Submodules

8.1.1 ckipnlp.util.ini module

```
ckipnlp.util.ini.create_ws_ini(*, data2dir=None, lexfile=None, new_style_format=False,
                               show_category=True, sentence_max_word_num=80, **options)
```

Generate CKIP word segmentation config.

Parameters

- **data2dir** (*str*) – the path to the folder “Data2”.
- **lexfile** (*str*) – the path to the user-defined lexicon file.
- **new_style_format** (*bool*) – split sentences by newline characters (“\n”) rather than punctuations.
- **show_category** (*bool*) – show part-of-speech tags.
- **sentence_max_word_num** (*int*) – maximum number of words per sentence.

```
ckipnlp.util.ini.create_parser_ini(*, wsinfile, ruledir=None, rdbdir=None, do_ws=True,
                                   do_parse=True, do_role=True, sentence_delim=',', **options)
```

Generate CKIP parser config.

Parameters

- **ruledir** (*str*) – the path to “Rule”.
- **rdbdir** (*str*) – the path to “RDB”.
- **do_ws** (*bool*) – do word-segmentation.
- **do_parse** (*bool*) – do parsing.
- **do_role** (*bool*) – do role.

- **sentence_delim** (*str*) – the sentence delimiters.

8.1.2 ckipnlp.util.parser module

class ckipnlp.util.parser.**ParserNode**

Bases: ckipnlp.util.parser._ParserNode

A parser node.

Fields:

- **role** (*str*): the role.
- **pos** (*str*): the post-tag.
- **term** (*str*): the text term.

classmethod **from_text** (*text*)

Create *ParserNode* object from *ckipnlp.parser.CkipParser* output.

class ckipnlp.util.parser.**ParserRelationNode**

Bases: ckipnlp.util.parser._ParserRelationNode

A parser relation node.

Fields:

- **node** (*treelib.Node*): the node.
- **role** (*str*): the relation role.

class ckipnlp.util.parser.**ParserRelation**

Bases: ckipnlp.util.parser._ParserRelation

A parser relation.

Fields:

- **head** (*ParserRelationNode*): the head node.
- **tail** (*ParserRelationNode*): the tail node.

class ckipnlp.util.parser.**ParserTree** (*tree=None, deep=False, node_class=None*)

Bases: *treelib.tree.Tree*

A parsed tree.

classmethod **from_text** (*tree_text*)

Create *ParserTree* object from *ckipnlp.parser.CkipParser* output.

show (*, *key=<function ParserTree.<lambda>>, idhidden=False, **kwargs*)

Show pretty tree.

has_dummies (*node_id*)

Determine if a node has dummies.

Parameters **node_id** (*int*) – ID of target node.

Returns *bool* – whether or not target node has dummies.

get_dummies (*node_id, deep=True, _check=True*)

Get dummies of a node.

Parameters

- **node_id** (*int*) – ID of target node.

- **deep** (*bool*) – find dummies recursively.

Returns *tuple* – the dummies (*ParserNode*).

Raises *LookupError* – when target node has no dummy (only when **_check** is set).

get_heads (*root_id=0, deep=True*)

Get all head nodes of a subtree.

Parameters

- **node_id** (*int*) – ID of the root node of target subtree.
- **deep** (*bool*) – find heads recursively.

Returns

- *list* – the head nodes (*ParserNode*).
- *ParserNode* – the head node (when **deep** is set).

Todo: Get information of nodes with pos type PP or GP.

get_relations (*root_id=0*)

Get all relations of a subtree.

Parameters **node_id** (*int*) – ID of the subtree root node.

Yields *ParserRelation* – the relation.

8.1.3 ckipnlp.util.ws module

class ckipnlp.util.ws.**WsWord**

Bases: ckipnlp.util.ws._WsWord

A word-segmented word.

Fields:

- **word** (*str*): the word.
- **pos** (*str*): the post-tag.

classmethod **from_text** (*text*)

Create *WsWord* object from *ckipnlp.ws.CkipWs* output.

class ckipnlp.util.ws.**WsSentence** (*initlist=None*)

Bases: collections.UserList

A word-segmented sentence.

Items: *WsWord*: the words.

classmethod **from_text** (*text*)

Create *WsSentence* object from *ckipnlp.ws.CkipWs* output.

CHAPTER 9

Todo List

Todo: Get information of nodes with pos type PP or GP.

(The [original entry](#) is located in `/home/docs/checkouts/readthedocs.org/user_builds/ckipnlp/checkouts/0.5.2/ckipnlp/util/parser.py:docstr` of `ckipnlp.util.parser.ParserTree.get_heads`, line 11.)

CHAPTER 10

Index

CHAPTER 11

Module Index

C

`ckipnlp.parser`, [13](#)
`ckipnlp.util`, [15](#)
`ckipnlp.util.ini`, [15](#)
`ckipnlp.util.parser`, [16](#)
`ckipnlp.util.ws`, [17](#)
`ckipnlp.ws`, [11](#)

A

[apply\(\)](#) (*ckipnlp.parser.CkipParser method*), 13
[apply\(\)](#) (*ckipnlp.ws.CkipWs method*), 11
[apply_file\(\)](#) (*ckipnlp.parser.CkipParser method*), 13
[apply_file\(\)](#) (*ckipnlp.ws.CkipWs method*), 11
[apply_list\(\)](#) (*ckipnlp.parser.CkipParser method*), 13
[apply_list\(\)](#) (*ckipnlp.ws.CkipWs method*), 11

C

[ckipnlp.parser](#) (*module*), 13
[ckipnlp.util](#) (*module*), 15
[ckipnlp.util.ini](#) (*module*), 15
[ckipnlp.util.parser](#) (*module*), 16
[ckipnlp.util.ws](#) (*module*), 17
[ckipnlp.ws](#) (*module*), 11
[CkipParser](#) (*class in ckipnlp.parser*), 13
[CkipWs](#) (*class in ckipnlp.ws*), 11
[create_parser_ini\(\)](#) (*in module ckipnlp.util.ini*), 15
[create_ws_ini\(\)](#) (*in module ckipnlp.util.ini*), 15

F

[from_text\(\)](#) (*ckipnlp.util.parser.ParserNode class method*), 16
[from_text\(\)](#) (*ckipnlp.util.parser.ParserTree class method*), 16
[from_text\(\)](#) (*ckipnlp.util.ws.WsSentence class method*), 17
[from_text\(\)](#) (*ckipnlp.util.ws.WsWord class method*), 17

G

[get_dummies\(\)](#) (*ckipnlp.util.parser.ParserTree method*), 16
[get_heads\(\)](#) (*ckipnlp.util.parser.ParserTree method*), 17

[get_relations\(\)](#) (*ckipnlp.util.parser.ParserTree method*), 17

H

[has_dummies\(\)](#) (*ckipnlp.util.parser.ParserTree method*), 16

P

[ParserNode](#) (*class in ckipnlp.util.parser*), 16
[ParserRelation](#) (*class in ckipnlp.util.parser*), 16
[ParserRelationNode](#) (*class in ckipnlp.util.parser*), 16
[ParserTree](#) (*class in ckipnlp.util.parser*), 16

S

[show\(\)](#) (*ckipnlp.util.parser.ParserTree method*), 16

W

[WsSentence](#) (*class in ckipnlp.util.ws*), 17
[WsWord](#) (*class in ckipnlp.util.ws*), 17