CKIPNLP

Release v0.6.3

Feb 13, 2020

Overview

CHAPTER 1

Introduction

1.1 Git

https://github.com/ckiplab/ckipnlp

1.2 PyPI

https://pypi.org/project/ckipnlp

1.3 Documentation

http://ckipnlp.readthedocs.io/

1.4 Author / Maintainer

- Mu Yang at CKIP (Author & Maintainer)
- Wei-Yun Ma at CKIP (Maintainer)

1.5 Requirements

- Python 3.5+

- Cython 0.29+
- TreeLib 1.5+

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 CKIP-Parser (Optional)

- CKIP Parser Linux version 20190506+ (20190725+ recommended)

CHAPTER 2

Installation

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`

CHAPTER 3

Usage

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)']

# Show Sentence List
ws_sents = WsSentenceList.from_text(ws_text)
print(repr(ws_sents))
print(ws_sents.to_text())

# Show Each Sentence
for ws_sent in ws_sents: print(repr(ws_sent))
for ws_sent in ws_sents: print(ws_sent.to_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:D
↪'
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 rel in tree.get_relations(0): print(rel)
```

CHAPTER 4

FAQ

Warning: Due to C code implementation, one should not instance more than one `CkipWs` driver object and one `CkipParser` driver object.

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

CHAPTER 6

ckipnlp.ws package

class ckipnlp.ws.**CkipWs**(*, *logger=False*, *inifile=None*, ***kwargs*)

Bases: object

The CKIP word segmentation driver.

Parameters

- **logger** (*bool*) – enable logger.
- **inifile** (*str*) – the path to the INI file.

Other Parameters ** – the configs for CKIPWS, ignored if **inifile** is set. Please refer *ckipnlp.util.ini.create_ws_ini()*.

> **Warning:** Never instance more than one object of this class!

apply(*text*)

Segment a sentence.

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

Returns *str* – the output sentence.

> **Note:** One may also call this method as __call__().

apply_list(*ilist*)

Segment a list of sentences.

Parameters **ilist** – the list of input sentences.

Returns *List[str]* – the list of output sentences.

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).

CHAPTER 7

ckipnlp.parser package

class ckipnlp.parser.**CkipParser**(*, *logger=False*, *inifile=None*, *wsinifile=None*, ***kwargs*)

Bases: object

The CKIP sentence parsing driver.

Parameters

- **logger** (*bool*) – enable logger.
- **inifile** (*str*) – the path to the INI file.
- **wsinifile** (*str*) – the path to the INI file for CKIPWS.

Other Parameters

- ** – the configs for CKIPParser, ignored if **inifile** is set. Please refer *ckipnlp.util.ini.create_parser_ini()*.
- ** – the configs for CKIPWS, ignored if **wsinifile** is set. Please refer *ckipnlp.util.ini.create_ws_ini()*.

Warning: Never instance more than one object of this class!

apply(*text*)

Segment a sentence.

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

Returns *str* – the output sentence.

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

apply_list(*ilist*)

Segment a list of sentences.

Parameters **ilist** – the list of input sentences.

Returns *List[str]* – the list of output sentences.

apply_file(*ifile*, *ofile*)
Segment a file.

Parameters

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

CHAPTER 8

ckipnlp.util package

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**(*, *wsinifile*, *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.**ParserNodeData**
Bases: tuple

A parser node.

role
str – the role.

pos
str – the post-tag.

term
str – the text term.

classmethod from_text(*text*)
Create a *ParserNodeData* object from *ckipnlp.parser.CkipParser* output.

to_text()
Transform to plain text.

to_dict()
Transform to python dict/list.

to_json(***kwargs*)
Transform to JSON format.

class ckipnlp.util.parser.**ParserNode**(*tag=None*, *identifier=None*, *expanded=True*, *data=None*)
Bases: treelib.node.Node

A parser node for tree.

data

Type *ParserNodeData*

See also:

treelib.tree.Node Please refer https://treelib.readthedocs.io/ for built-in usages.

to_text()
Transform to plain text.

to_dict()
Transform to python dict/list.

to_json(***kwargs*)
Transform to JSON format.

class ckipnlp.util.parser.**ParserRelation**
Bases: tuple

A parser relation.

head
ParserNode – the head node.

tail
ParserNode – the tail node.

relation
str – the relation.

head_first

to_dict()
Transform to python dict/list.

to_json(***kwargs*)
Transform to JSON format.

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

A parsed tree.

See also:

treereelib.tree.Tree Please refer https://treelib.readthedocs.io/ for built-in usages.

node_class
alias of *ParserNode*

static normalize_text(*tree_text*)
Text normalization for *ckipnlp.parser.CkipParser* output.

Remove leading number and trailing #. Prepend root: at beginning.

classmethod from_text(*tree_text*, *, *normalize=True*)
Create a *ParserTree* object from *ckipnlp.parser.CkipParser* output.

Parameters

- **text** (*str*) – A parsed tree from *ckipnlp.parser.CkipParser* output.
- **normalize** (*str*) – Do text normalization. Please refer *ParserTree.normalize_text()*.

to_text(*node_id=0*)
Transform to plain text.

to_dict(*node_id=0*)
Transform to python dict/list.

to_json(***kwargs*)
Transform to JSON format.

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[*ParserNode*] – the dummies.

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

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

Returns

- List[*ParserNode*] – the head nodes (when **deep** is set).
- *ParserNode* – the head node (when **deep** is not set).

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

get_relations(*root_id=0*)
Get all relations of a subtree.

Parameters **root_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: tuple

A word-segmented word.

word
str – the word.

pos
str – the post-tag.

classmethod from_text(*text*)
Create a *WsWord* object from *ckipnlp.ws.CkipWs* output.

Parameters **text** (*str*) – A word from *ckipnlp.ws.CkipWs* output.

to_text()
Transform to plain text.

to_dict()
Transform to python dict/list.

to_json(***kwargs*)
Transform to JSON format.

class ckipnlp.util.ws.**WsSentence**(*initlist=None*)
Bases: collections.UserList

A word-segmented sentence.

item_class
alias of *WsWord*

classmethod from_text(*text*)
Create *WsSentence* object from *ckipnlp.ws.CkipWs* output.

Parameters **text** (*str*) – A sentence from *ckipnlp.ws.CkipWs* output.

to_text()
Transform to plain text.

to_dict()
Transform to python dict/list.

to_json(***kwargs*)
Transform to JSON format.

class ckipnlp.util.ws.**WsSentenceList**(*initlist=None*)
Bases: collections.UserList

A list of word-segmented sentence.

item_class
alias of *WsSentence*

classmethod from_text(*text_list*)
Create *WsSentenceList* object from *ckipnlp.ws.CkipWs* output.

Parameters **text_list** (*List[str]*) – A list of sentence from *ckipnlp.ws.CkipWs* output.

to_text()
Transform to plain text.

to_dict()
Transform to python dict/list.

to_json(***kwargs*)
Transform to JSON format.

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/0.6.3/ckipnlp/util/parser.py:docstr
of ckipnlp.util.parser.ParserTree.get_heads, line 11.)

CHAPTER 10

Index

CHAPTER 11

Module Index

Python Module Index

C

Index

S

T

W