CKIPNLP

Release v0.7.0

Mu Yang

Feb 20, 2020

OVERVIEW

CHAPTER

ONE

INTRODUCTION

1.1 Git

https://github.com/ckiplab/ckipnlp

1.2 PyPI

https://pypi.org/project/ckipnlp

1.3 Documentation

https://ckipnlp.readthedocs.io/

1.4 Contributers

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

1.5 External Links

- Online Demo

1.6 Requirements

- Python 3.5+
- Cython 0.29+
- TreeLib 1.5+

Attention: For Python 2 users, please use PyCkip 0.4.2 instead.

1.6.1 CKIPWS (Optional)

- CKIP Word Segmentation Linux version 20190524+

1.6.2 CKIPParser (Optional)

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

CHAPTER

TWO

INSTALLATION

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

2.2 Installation Options

Option	Detail	Default Value
`--[no-]ws`	Enable/disable CKIPWS.	False
`--[no-]parser`	Enable/disable CKIPParser.	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>`	CKIPParser root directory.	
`--parser-lib-dir=<parser-lib-dir>`	CKIPParser libraries directory	`<parser-dir>/lib`
`--parser-share-dir=<parser-share-dir>`	CKIPParser 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

THREE

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 CKIPParser

```
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.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 = '#1:1.[0]␣
↪S(theme:NP(possessor:N(head:Nhaa:|Head:DE:)|Head:Nab(DUMMY1:Nab(DUMMY1:Nab:|Head:Caa:|DUMMY2:Naa:)
↪#'
tree = ParserTree.from_text(tree_text)
tree.show()

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

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

# Get heads of node 2
for node in tree.get_heads(2): print(node)

# Get heads of node 13
for node in tree.get_heads(13): print(node)

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

CHAPTER

FOUR

FAQ

Danger: Due to C code implementation, both `CkipWs` and `CkipParser` can only be instance once.

Tip: 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
```

Tip: 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

FIVE

LICENSE

CHAPTER

SIX

CKIPNLP PACKAGE

6.1 Subpackages

6.1.1 ckipnlp.parser package

class ckipnlp.parser.**CkipParser**(*, *logger=False*, *ini_file=None*, *ws_ini_file=None*, *lex_list=None*, ***kwargs*)

Bases: object

The CKIP sentence parsing driver.

Parameters

- **logger** (*bool*) – enable logger.
- **lex_list** (*Iterable*) – passed to *ckipnlp.util.ini.create_ws_lex()*, overridden **lex_file** for *ckipnlp.util.ini.create_ws_ini()*.
- **ini_file** (*str*) – the path to the INI file.
- **ws_ini_file** (*str*) – the path to the INI file for CKIPWS.

Other Parameters

- ****** – the configs for CKIPParser, passed to *ckipnlp.util.ini.create_parser_ini()*, ignored if **ini_file** is set.
- ****** – the configs for CKIPWS, passed to *ckipnlp.util.ini.create_ws_ini()*, ignored if **ws_ini_file** is set.

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

apply(*text*)

Parse a sentence.

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

Returns *str* – the output sentence.

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

apply_list(*ilist*)

Parse a list of sentences.

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

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

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

Parameters

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

6.1.2 ckipnlp.util package

Submodules

ckipnlp.util.ini module

ckipnlp.util.ini.**create_ws_lex**(**lex_list*)
Generate CKIP word segmentation lexicon file.

Parameters ***lex_list** (*Tuple[str, str]*) – the lexicon word and its POS-tag.

Returns

- **lex_file** (*str*) – the name of the lexicon file.
- **f_lex** (*TextIO*) – the file object.

Attention: Remember to close **f_lex** manually.

ckipnlp.util.ini.**create_ws_ini**(**, data2_dir=None, lex_file=None, new_style_format=False, show_category=True, sentence_max_word_num=80, **options*)
Generate CKIP word segmentation config.

Parameters

- **data2_dir** (*str*) – the path to the folder "Data2/".
- **lex_file** (*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.

Returns

- **ini_file** (*str*) – the name of the config file.
- **f_ini** (*TextIO*) – the file object.

Attention: Remember to close **f_ini** manually.

ckipnlp.util.ini.**create_parser_ini**(*, *ws_ini_file*, *rule_dir=None*, *rdb_dir=None*, *do_ws=True*, *do_parse=True*, *do_role=True*, *sentence_delim='，'*, ***options*)

Generate CKIP parser config.

Parameters

- **rule_dir** (*str*) – the path to "Rule/".
- **rdb_dir** (*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.

Returns

- **ini_file** (*str*) – the name of the config file.
- **f_ini** (*TextIO*) – the file object.

Attention: Remember to close **f_ini** manually.

ckipnlp.util.parser module

class ckipnlp.util.parser.**ParserNodeData**

Bases: `tuple`

A parser node.

property role
str – the role.

property pos
str – the post-tag.

property term
str – the text term.

classmethod from_text(*text*)
Construct an instance from *ckipnlp.parser.CkipParser* output.

Parameters **data** (*str*) – text such as `'Head:Na:'`.

Notes

- `'Head:Na:'` -> role = `'Head'`, pos = `'Na'`, term = `''`
- `'Head:Na'` -> role = `'Head'`, pos = `'Na'`, term = `None`
- `'Na'` -> role = `None`, pos = `'Na'`, term = `None`

to_text()
Transform to plain text.

Returns *str*

classmethod from_dict(*data*)
Construct an instance from python built-in containers.

Parameters **data** (*dict*) – dictionary such as `{ 'role': 'Head', 'pos': 'Na', 'term': '' }`

to_dict()
Transform to python built-in containers.

Returns *dict*

classmethod from_json(*data*, ***kwargs*)
Construct an instance from JSON format.

Parameters **data** (*str*) – please refer *from_dict()* for format details.

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

Returns *str*

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.

data_class
alias of *ParserNodeData*

to_dict()
Transform to python built-in containers.

Returns *dict*

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

Returns *str*

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

A parser relation.

property head
ParserNode – the head node.

property tail
ParserNode – the tail node.

property relation
str – the relation.

to_dict()
Transform to python built-in containers.

Returns *dict*

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

Returns *str*

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 `#`.

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** (*bool*) – Do text normalization using *normalize_text()*.

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

Returns *str*

classmethod from_dict (*data*)
Construct an instance from python built-in containers.

Parameters **data** (*dict*) – dictionary such as `{ 'id': 0, 'data': { ... }, 'children': [ ... ] }`, where `'data'` is a dictionary with the same format as *ParserNodeData.to_dict()*, and `'children'` is a list of dictionaries of subtrees with the same format as this tree.

to_dict (*node_id=0*)
Transform to python built-in containers.

Returns *dict*

classmethod from_json (*data*, ***kwargs*)
Construct an instance from JSON format.

Parameters **data** (*str*) – please refer *from_dict()* for format details.

to_json (*node_id=0*, ***kwargs*)
Transform to JSON format.

Returns *str*

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

get_children (*node_id*, ***, *role*)
Get children of a node with given role.

Parameters

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

Yields *ParserNode* – the children nodes with given role.

get_heads(*root_id=0*, ***, *semantic=True*, *deep=True*)
Get all head nodes of a subtree.

Parameters

- **root_id** (*int*) – ID of the root node of target subtree.
- **semantic** (*bool*) – use semantic/syntactic policy. For semantic mode, return `DUMMY` or `head` instead of syntactic `Head`.
- **deep** (*bool*) – find heads recursively.

Yields *ParserNode* – the head nodes.

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

Parameters

- **root_id** (*int*) – ID of the subtree root node.
- **semantic** (*bool*) – please refer *get_heads()* for policy detail.

Yields *ParserRelation* – the relations.

ckipnlp.util.ws module

class ckipnlp.util.ws.**WsWord**
Bases: `tuple`

A word-segmented word.

property word
str – the word.

property pos
str – the post-tag.

classmethod from_text(*data*)
Construct an instance from *ckipnlp.ws.CkipWs* output.

Parameters **data** (*str*) – text such as `'(Na)'`.

Notes

- `'(Na)'` -> word = `''`, pos = `'Na'`
- `''` -> word = `''`, pos = `None`

to_text()
Transform to plain text.

Returns *str*

classmethod from_dict(*data*)
Construct an instance from python built-in containers.

Parameters **data** (*dict*) – dictionary such as `{ 'word': '', 'pos': 'Na' }`

to_dict()
Transform to python built-in containers.

Returns *dict*

classmethod from_json(*data*, ***kwargs*)
Construct an instance from JSON format.

Parameters **data** (*str*) – please refer *from_dict()* for format details.

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

Returns *str*

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

A word-segmented sentence.

item_class
alias of *WsWord*

classmethod from_text(*data*)
Construct an instance from *ckipnlp.ws.CkipWs* output.

Parameters **data** (*str*) – text such as `'(Na)\u3000(T)'`.

to_text()
Transform to plain text.

Returns *str*

classmethod from_dict(*data*)
Construct an instance a from python built-in containers.

Parameters **data** (*Sequence[dict]*) – list of objects as *WsWord.from_dict()* input.

to_dict()
Transform to python built-in containers.

Returns *List[dict]*

classmethod from_json(*data*, ***kwargs*)
Construct an instance from JSON format.

Parameters **data** (*str*) – please refer *from_dict()* for format details.

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

Returns *str*

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

A list of word-segmented sentence.

item_class
alias of *WsSentence*

classmethod from_text(*data*)
Construct an instance from *ckipnlp.ws.CkipWs* output.

Parameters **data** (*Sequence[str]*) – list of texts as *WsSentence.from_text()* input.

to_text()
Transform to plain text.

Returns *List[str]*

classmethod from_dict(*data*)
Construct an instance a from python built-in containers.

Parameters **data** (*Sequence[Sequence[dict]]*) – list of objects as *WsSentence.from_dict()* input.

to_dict()
Transform to python built-in containers.

Returns *List[List[dict]]*

classmethod from_json(*data*, ***kwargs*)
Construct an instance from JSON format.

Parameters **data** (*str*) – please refer *from_dict()* for format details.

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

Returns *str*

6.1.3 ckipnlp.ws package

class ckipnlp.ws.**CkipWs**(*, *logger=False*, *ini_file=None*, *lex_list=None*, ***kwargs*)
Bases: object

The CKIP word segmentation driver.

Parameters

- **logger** (*bool*) – enable logger.
- **lex_list** (*Iterable*) – passed to *ckipnlp.util.ini.create_ws_lex()* overridden **lex_file** for *ckipnlp.util.ini.create_ws_ini()*.
- **ini_file** (*str*) – the path to the INI file.

Other Parameters ****** – the configs for CKIPWS, passed to *ckipnlp.util.ini.create_ws_ini()*, ignored if **ini_file** is set.

Danger: 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.

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

apply_list(*ilist*)
Segment a list of sentences.

Parameters **ilist** (*List[str]*) – 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

SEVEN

TODO LIST

CHAPTER

EIGHT

INDEX

CHAPTER

NINE

MODULE INDEX

PYTHON MODULE INDEX

C

INDEX