Package 'jiebaR'

March 12, 2019

Type Package

Title Chinese Text Segmentation

Description Chinese text segmentation, keyword extraction and speech tagging For R.

Version 0.10.99

Date 2019-3-11

Author Qin Wenfeng, Wu Yanyi

Maintainer Qin Wenfeng <mail@qinwenfeng.com>

License MIT + file LICENSE

Depends R (>= 3.3), jiebaRD

Imports Rcpp (>= 0.12.1)

LinkingTo Rcpp (>= 0.12.1)

Suggests knitr,testthat,devtools,rmarkdown,roxygen2

URL https://github.com/qinwf/jiebaR/

BugReports https://github.com/qinwf/jiebaR/issues

VignetteBuilder knitr

NeedsCompilation yes

RoxygenNote 6.1.1

Repository CRAN

Date/Publication 2019-03-12 06:00:09 UTC

R topics documented:

`<=.keywords` *Keywords symbol*

Description

Keywords symbol to find keywords.

Usage

```
## S3 method for class 'keywords'
jiebar <= code

## S3 method for class 'keywords'
jiebar[code]
```

Arguments

`jiebar`	jiebaR Worker.
`code`	A Chinese sentence or the path of a text file.

Author(s)

Qin Wenfeng <http://qinwenfeng.com>

Examples

```
## Not run:
words = "hello world"
test1 = worker("keywords",topn=1)
test1 <= words
## End(Not run)
```

`<=.qseg` *Quick mode symbol*

Description

Depreciated.

Usage

```
## S3 method for class 'qseg'
qseg <= code

## S3 method for class 'qseg'
qseg[code]

qseg
```

Arguments

qseg	a qseg object
code	a string

Format

qseg an environment

Details

Quick mode is depreciated, and is scheduled to be remove in v0.11.0. If you want to keep this feature, please submit a issue on GitHub page to let me know.

Quick mode symbol to do segmentation, keyword extraction and speech tagging. This symbol will initialize a `quick_worker` when it is first called, and will do segmentation or other types of work immediately.

You can reset the default model setting by `$`, and it will change the default setting the next time you use quick mode. If you only want to change the parameter temporarily, you can reset the settings of `quick_worker$`. `get_qsegmodel`, `set_qsegmodel`, and `reset_qsegmodel` are also available for setting quick mode settings.

Author(s)

Qin Wenfeng <http://qinwenfeng.com>

See Also

set_qsegmodel worker

Examples

```
## Not run:
qseg <= "This is test"
qseg <= "This is the second test"

## End(Not run)

## Not run:
qseg <= "This is test"
qseg$detect = T
qseg
get_qsegmodel()

## End(Not run)
```

<=.segment *Text segmentation symbol*

Description

Text segmentation symbol to cut words.

Usage

```
## S3 method for class 'segment'
jiebar <= code

## S3 method for class 'segment'
jiebar[code]
```

Arguments

jiebar	jiebaR Worker.
code	A Chinese sentence or the path of a text file.

Author(s)

Qin Wenfeng <http://qinwenfeng.com>

Examples

```
## Not run:
words = "hello world"
test1 = worker()
test1 <= words
## End(Not run)
```

`<=.simhash` *Simhash symbol*

Description

Simhash symbol to compute simhash.

Usage

```
## S3 method for class 'simhash'
jiebar <= code

## S3 method for class 'simhash'
jiebar[code]
```

Arguments

`jiebar`	jiebaR Worker.
`code`	A Chinese sentence or the path of a text file.

Author(s)

Qin Wenfeng <http://qinwenfeng.com>

Examples

```
## Not run:
words = "hello world"
test1 = worker("simhash",topn=1)
test1 <= words
## End(Not run)
```

<=.tagger *Tagger symbol*

Description

Tagger symbol to tag words.

Usage

```
## S3 method for class 'tagger'
jiebar <= code

## S3 method for class 'tagger'
jiebar[code]
```

Arguments

jiebar	jiebaR Worker.
code	A Chinese sentence or the path of a text file.

Author(s)

Qin Wenfeng <http://qinwenfeng.com>

Examples

```
## Not run:
words = "hello world"
test1 = worker("tag")
test1 <= words
## End(Not run)
```

apply_list *Apply list input to a worker*

Description

Apply list input to a worker

Usage

```
apply_list(input, worker)
```

Arguments

input	a list of characters
worker	a worker

Examples

```
cutter = worker()
apply_list(list("this is test", "that is not test"), cutter)
apply_list(list("this is test", list("that is not test","ab c")), cutter)
```

DICTPATH *The path of dictionary*

Description

The path of dictionary, and it is used by segmentation and other function.

Usage

```
DICTPATH

HMMPATH

USERPATH

IDFPATH

STOPPATH
```

Format

character

---

distance *Hamming distance of words*

Description

This function uses Simhash worker to do keyword extraction and finds the keywords from two inputs, and then computes Hamming distance between them.

Usage

```
distance(codel, coder, jiebar)

vector_distance(codel, coder, jiebar)
```

Arguments

codel	For distance, a Chinese sentence or the path of a text file, For vector_distance, a character vector of segmented words.
coder	For distance, a Chinese sentence or the path of a text file, For vector_distance, a character vector of segmented words.
jiebar	jiebaR worker

Author(s)

Qin Wenfeng

References

http://en.wikipedia.org/wiki/Hamming_distance

See Also

worker

Examples

```
## Not run:

words = "hello world"
simhasher = worker("simhash", topn = 1)
simhasher <= words
distance("hello world" , "hello world!" , simhasher)

vector_distance(c("hello","world") , c("hello", "world","!") , simhasher)


## End(Not run)
```

---

edit_dict *Edit default user dictionary*

Description

Edit the default user dictionary.

Usage

```
edit_dict(name = "user")
```

Arguments

name	the name of dictionary including user, system, stop_word.

Details

There are three column in the system dictionary. Each column is seperated by space. The first column is the word, and the second column is the frequency of word. The third column is speech tag using labels compatible with ictclas.

There are two column in the user dictionary. The first column is the word, and the second column is speech tag using labels compatible with ictclas. Frequencies of words in the user dictionary is set by user_weight in `worker` function. If you want to provide the frequency of a new word, you can put it in the system dictionary.

Only one column in the stop words dictionary, and it contains the stop words.

References

The ictclas speech tag : http://t.cn/RAEj7e1

`file_coding` *Files encoding detection*

Description

This function detects the encoding of input files. You can also check encoding with checkenc package which is on GitHub.

Usage

```
file_coding(file)

filecoding(file)
```

Arguments

`file`	A file path.

Details

This function will choose the most likely encoding, and it will be more stable for a large input text file.

Value

The encoding of file

Author(s)

Wu Yongwei, Qin wenfeng

References

https://github.com/adah1972/tellenc

See Also

https://github.com/qinwf/checkenc

`filter_segment` *Filter segmentation result*

Description

This function helps remove some words in the segmentation result.

Usage

```
filter_segment(input, filter_words, unit = 50)
```

Arguments

`input`	a string vector
`filter_words`	a string vector of words to be removed.
`unit`	the length of word unit to use in regular expression, and the default is 50. Long list of a words forms a big regular expressions, it may or may not be accepted: the POSIX standard only requires up to 256 bytes. So we use unit to split the words in units.

Examples

```
filter_segment(c("abc","def"," ","."), c("abc"))
```

`freq` *The frequency of words*

Description

This function returns the frequency of words

Usage

```
freq(x)
```

Arguments

`x`	a vector of words

Value

The frequency of words

Author(s)

Qin wenfeng

Examples

```
freq(c("a","a","c"))
```

---

get_idf *generate IDF dict*

Description

Generate IDF dict from a list of documents.

Usage

```
get_idf(x, stop_word = STOPPATH, path = NULL)
```

Arguments

x	a list of character
stop_word	stopword path
path	output path

Details

Input list contains multiple character vectors with words, and each vector represents a document.

Stop words will be removed from the result.

If path is not NULL, it will write the result to the path.

Value

a data.frame or a file

See Also

https://en.wikipedia.org/wiki/Tf-idf#Inverse_document_frequency_2

Examples

```
get_idf(list(c("abc","def"),c("abc"," ")))
```

`get_qsegmodel` *Set quick mode model*

Description

Depreciated.

Usage

```
get_qsegmodel()

set_qsegmodel(qsegmodel)

reset_qsegmodel()
```

Arguments

`qsegmodel`	a list which has the same structure as the return value of get_qsegmodel

Details

These function can get and modify quick mode model. `get_qsegmodel` returns the default model parameters. `set_qsegmodel` can modify quick mode model using a list, which has the same structure as the return value of get_qsegmodel. `reset_qsegmodel` can reset the default model to origin `jiebaR` default model.

Author(s)

Qin Wenfeng <http://qinwenfeng.com>

See Also

`qseg` `worker`

Examples

```
## Not run:
qseg <= "This is test"
qseg <= "This is the second test"

## End(Not run)

## Not run:
qseg <= "This is test"
qseg$detect = T
qseg
get_qsegmodel()
model = get_qsegmodel()
model$detect = F
```

```
set_qsegmodel(model)
reset_qsegmodel()

## End(Not run)
```

`get_tuple` *get tuple from the segmentation result*

Description

get tuple from the segmentation result

Usage

```
get_tuple(x, size = 2, dataframe = T)
```

Arguments

`x`	a character vector or list
`size`	a integer >= 2
`dataframe`	return data.frame

Examples

```
get_tuple(c("sd","sd","sd","rd"),2)
```

`jiebaR` *A package for Chinese text segmentation*

Description

This is a package for Chinese text segmentation, keyword extraction and speech tagging with Rcpp and cppjieba.

Details

You can use custom dictionary. JiebaR can also identify new words, but adding new words will ensure higher accuracy.

Author(s)

Qin Wenfeng <http://qinwenfeng.com>

References

CppJieba https://github.com/aszxqw/cppjieba;

See Also

JiebaR https://github.com/qinwf/jiebaR;

Examples

```
### Note: Can not display Chinese characters here.
## Not run:
words = "hello world"
engine1 = worker()
segment(words, engine1)

# "./temp.txt" is a file path

segment("./temp.txt", engine1)

engine2 = worker("hmm")
segment("./temp.txt", engine2)

engine2$write = T
segment("./temp.txt", engine2)

engine3 = worker(type = "mix", dict = "dict_path",symbol = T)
segment("./temp.txt", engine3)

## End(Not run)

## Not run:
### Keyword Extraction
engine = worker("keywords", topn = 1)
keywords(words, engine)

### Speech Tagging
tagger = worker("tag")
tagging(words, tagger)

### Simhash
simhasher = worker("simhash", topn = 1)
simhash(words, simhasher)
distance("hello world" , "hello world!" , simhasher)

show_dictpath()

## End(Not run)
```

---

keywords *Keyword extraction*

Description

Keyword Extraction worker uses MixSegment model to cut word and uses TF-IDF algorithm to find the keywords. `dict` , `hmm`, `idf`, `stop_word` and `topn` should be provided when initializing jiebaR worker.

Usage

```
keywords(code, jiebar)

vector_keywords(code, jiebar)
```

Arguments

`code`	For `keywords`, a Chinese sentence or the path of a text file. For `vector_keywords`, a character vector of segmented words.
`jiebar`	jiebaR Worker.

Details

There is a symbol <= for this function.

Value

a vector of keywords with weight.

Author(s)

Qin Wenfeng

References

http://en.wikipedia.org/wiki/Tf-idf

See Also

`<=.keywords` `worker`

Examples

```
## Not run:
### Keyword Extraction
keys = worker("keywords", topn = 1)
keys <= "words of fun"
## End(Not run)
```

new_user_word *Add user word*

Description

Add user word

Usage

```
new_user_word(worker, words, tags = rep("n", length(words)))
```

Arguments

worker	a jieba worker
words	the new words
tags	the new words tags, default "n"

Examples

```
cc = worker()
new_user_word(cc, "test")
new_user_word(cc, "do", "v")
```

print.inv *Print worker settings*

Description

These functoins print the worker settings.

Usage

```
## S3 method for class 'inv'
print(x, ...)

## S3 method for class 'jieba'
print(x, ...)

## S3 method for class 'simhash'
print(x, ...)

## S3 method for class 'keywords'
print(x, ...)

## S3 method for class 'qseg'
print(x, ...)
```

Arguments

x	The jiebaR Worker.
...	Other arguments.

Author(s)

Qin Wenfeng

---

segment *Chinese text segmentation function*

Description

The function uses initialized engines for words segmentation. You can initialize multiple engines simultaneously using `worker()`. Public settings of workers can be got and modified using `$`, such as `WorkerName$symbol = T` . Some private settings are fixed when engine is initialized, and you can get then by `WorkerName$PrivateVarible`.

Usage

```
segment(code, jiebar, mod = NULL)
```

Arguments

code	A Chinese sentence or the path of a text file.
jiebar	jiebaR Worker.
mod	change default result type, value can be "mix","hmm","query","full" or "mp"

Details

There are four kinds of models:

Maximum probability segmentation model uses Trie tree to construct a directed acyclic graph and uses dynamic programming algorithm. It is the core segmentation algorithm. `dict` and `user` should be provided when initializing jiebaR worker.

Hidden Markov Model uses HMM model to determine status set and observed set of words. The default HMM model is based on People's Daily language library. `hmm` should be provided when initializing jiebaR worker.

MixSegment model uses both Maximum probability segmentation model and Hidden Markov Model to construct segmentation. `dict`, `hmm` and `user` should be provided when initializing jiebaR worker.

QuerySegment model uses MixSegment to construct segmentation and then enumerates all the possible long words in the dictionary. `dict`, `hmm` and `qmax` should be provided when initializing jiebaR worker.

There is a symbol <= for this function.

See Also

<=.segment worker

---

show_dictpath *Show default path of dictionaries*

Description

Show the default dictionaries' path. HMMPATH, DICTPATH , IDFPATH, STOPPATH and USERPATH can be changed in default environment.

Usage

```
show_dictpath()
```

Author(s)

Qin Wenfeng

---

simhash *Simhash computation*

Description

Simhash worker uses the keyword extraction worker to find the keywords and uses simhash algorithm to compute simhash. dict hmm, idf and stop_word should be provided when initializing jiebaR worker.

Usage

```
simhash(code, jiebar)

vector_simhash(code, jiebar)
```

Arguments

code	For simhash, a Chinese sentence or the path of a text file. For vector_simhash, a character vector of segmented words.
jiebar	jiebaR Worker.

Details

There is a symbol <= for this function.

Author(s)

Qin Wenfeng

References

MS Charikar - Similarity Estimation Techniques from Rounding Algorithms

See Also

`<=.simhash` `worker`

Examples

```
## Not run:
### Simhash
words = "hello world"
simhasher = worker("simhash",topn=1)
simhasher <= words
distance("hello world" , "hello world!" , simhasher)

## End(Not run)
```

`simhash_dist` *Compute Hamming distance of Simhash value*

Description

Compute Hamming distance of Simhash value

Usage

```
simhash_dist(x, y)

simhash_dist_mat(x, y)
```

Arguments

x	a character vector of simhash value
y	a character vector of simhash value

Value

a character vector

Examples

```
simhash_dist("1","1")
simhash_dist("1","2")
tobin("1")
tobin("2")
simhash_dist_mat(c("1","12","123"),c("2","1"))
```

`tagging` *Speech Tagging*

Description

The function uses Speech Tagging worker to cut word and tags each word after segmentation using labels compatible with ictclas. `dict` `hmm` and `user` should be provided when initializing jiebaR worker.

Usage

```
tagging(code, jiebar)
```

Arguments

`code`	a Chinese sentence or the path of a text file
`jiebar`	jiebaR Worker

Details

There is a symbol <= for this function.

Author(s)

Qin Wenfeng

References

The ictclas speech tag : http://t.cn/RAEj7e1

See Also

`<=.tagger` `worker`

Examples

```
## Not run:
words = "hello world"

### Speech Tagging
tagger = worker("tag")
tagger <= words

## End(Not run)
```

`tobin` *simhash value to binary*

Description

simhash value to binary

Usage

```
tobin(x)
```

Arguments

`x`	simhash value

`vector_tag` *Tag the a character vector*

Description

Tag the a character vector

Usage

```
vector_tag(string, jiebar)
```

Arguments

`string`	a character vector of segmented words.
`jiebar`	jiebaR Worker.

Examples

```
## Not run:
cc = worker()
(res = cc["this is test"])
vector_tag(res, cc)

## End(Not run)
```

---

worker *Initialize jiebaR worker*

Description

This function can initialize jiebaR workers. You can initialize different kinds of workers including `mix`, `mp`, `hmm`, `query`, `full`, `tag`, `simhash`, and `keywords`. see Detail for more information.

Usage

```
worker(type = "mix", dict = DICTPATH, hmm = HMMPATH,
  user = USERPATH, idf = IDFPATH, stop_word = STOPPATH, write = T,
  qmax = 20, topn = 5, encoding = "UTF-8", detect = T,
  symbol = F, lines = 1e+05, output = NULL, bylines = F,
  user_weight = "max")
```

Arguments

`type`	The type of jiebaR workers including `mix`, `mp`, `hmm`, `full`, `query`, `tag`, `simhash`, and `keywords`.
`dict`	A path to main dictionary, default value is `DICTPATH`, and the value is used for `mix`, `mp`, `query`, `full`, `tag`, `simhash` and `keywords` workers.
`hmm`	A path to Hidden Markov Model, default value is `HMMPATH`, `full`, and the value is used for `mix`, `hmm`, `query`, `tag`, `simhash` and `keywords` workers.
`user`	A path to user dictionary, default value is `USERPATH`, and the value is used for `mix`, `full`, `tag` and `mp` workers.
`idf`	A path to inverse document frequency, default value is `IDFPATH`, and the value is used for `simhash` and `keywords` workers.
`stop_word`	A path to stop word dictionary, default value is `STOPPATH`, and the value is used for `simhash`, `keywords`, `tagger` and `segment` workers. Encoding of this file is checked by `file_coding`, and it should be UTF-8 encoding. For `segment` workers, the default `STOPPATH` will not be used, so you should provide another file path.
`write`	Whether to write the output to a file, or return a the result in a object. This value will only be used when the input is a file path. The default value is TRUE. The value is used for segment and speech tagging workers.

qmax	Max query length of words, and the value is used for query workers.
topn	The number of keywords, and the value is used for simhash and keywords workers.
encoding	The encoding of the input file. If encoding detection is enable, the value of encoding will be ignore.
detect	Whether to detect the encoding of input file using file_coding function. If encoding detection is enable, the value of encoding will be ignore.
symbol	Whether to keep symbols in the sentence.
lines	The maximal number of lines to read at one time when input is a file. The value is used for segmentation and speech tagging workers.
output	A path to the output file, and default worker will generate file name by system time stamp, the value is used for segmentation and speech tagging workers.
bylines	return the result by the lines of input files
user_weight	the weight of the user dict words. "min" "max" or "median".

Details

The package uses initialized engines for word segmentation, and you can initialize multiple engines simultaneously. You can also reset the model public settings using $ such as `WorkerName$symbol = T` . Some private settings are fixed when a engine is initialized, and you can get then by `WorkerName$PrivateVarible`.

Maximum probability segmentation model uses Trie tree to construct a directed acyclic graph and uses dynamic programming algorithm. It is the core segmentation algorithm. dict and user should be provided when initializing jiebaR worker.

Hidden Markov Model uses HMM model to determine status set and observed set of words. The default HMM model is based on People's Daily language library. hmm should be provided when initializing jiebaR worker.

MixSegment model uses both Maximum probability segmentation model and Hidden Markov Model to construct segmentation. dict hmm and user should be provided when initializing jiebaR worker.

QuerySegment model uses MixSegment to construct segmentation and then enumerates all the possible long words in the dictionary. dict, hmm and qmax should be provided when initializing jiebaR worker.

FullSegment model will enumerates all the possible words in the dictionary.

Speech Tagging worker uses MixSegment model to cut word and tag each word after segmentation using labels compatible with ictclas. dict, hmm and user should be provided when initializing jiebaR worker.

Keyword Extraction worker uses MixSegment model to cut word and use TF-IDF algorithm to find the keywords. dict ,hmm, idf, stop_word and topn should be provided when initializing jiebaR worker.

Simhash worker uses the keyword extraction worker to find the keywords and uses simhash algorithm to compute simhash. dict hmm, idf and stop_word should be provided when initializing jiebaR worker.

Value

This function returns an environment containing segmentation settings and worker. Public settings can be modified using $.

Examples

```
### Note: Can not display Chinese characters here.
## Not run:
words = "hello world"
engine1 = worker()
segment(words, engine1)

# "./temp.txt" is a file path

segment("./temp.txt", engine1)

engine2 = worker("hmm")
segment("./temp.txt", engine2)

engine2$write = T
segment("./temp.txt", engine2)

engine3 = worker(type = "mix", dict = "dict_path",symbol = T)
segment("./temp.txt", engine3)

## End(Not run)

## Not run:
### Keyword Extraction
engine = worker("keywords", topn = 1)
keywords(words, engine)

### Speech Tagging
tagger = worker("tag")
tagging(words, tagger)

### Simhash
simhasher = worker("simhash", topn = 1)
simhash(words, simhasher)
distance("hello world" , "hello world!" , simhasher)

show_dictpath()

## End(Not run)
```

Index