Package ‘RcppCWB’

February 21, 2019

Type Package

Title 'Rcpp' Bindings for the 'Corpus Workbench' ('CWB')

Version 0.2.8

Date 2019-02-21

Author Andreas Blaette [aut, cre],
Bernard Desgraupes [aut],
Sylvain Loiseau [aut],
Oliver Christ [ctb],
Bruno Maximilian Schulze [ctb],
Stefan Evert [ctb],
Arne Fitschen [ctb]

Maintainer Andreas Blaette <andreas.blaette@uni-due.de>

Description 'Rcpp' Bindings for the C code of the 'Corpus Workbench' ('CWB'), an indexing and query
engine to efficiently analyze large corpora (<http://cwb.sourceforge.net>). 'RcppCWB' is licensed
under the GNU GPL-3, in line with the GPL-3 license of the 'CWB' (<https://www.r-project.org/
Licenses/GPL-3>). The 'CWB' relies on 'pcre' (BSD license, see <https://www.pcre.org/
l licence.txt>) and 'GLib' (LGPL license, see <https://www.gnu.org/licenses/lgpl-3.0.en.
html>). See the file LICENSE.note for further information. The package includes modified code of the
'rcqp' package (GPL-2, see <https://cran.r-
project.org/package=rcqp>). The original work of the authors
of the 'rcqp' package is acknowledged with great respect, and they are listed as authors of this
package. To achieve cross-platform portability (including Windows), using 'Rcpp' for wrapper code
is the approach used by 'RcppCWB'.

License GPL-3

Encoding UTF-8

Copyright For the copyrights for the 'Corpus Workbench' (CWB) and
acknowledgement of authorship, see file COPYRIGHTS.

NeedsCompilation yes

SystemRequirements GNU make, pcre (>= 7), GLib (>= 2.0.0). On Windows,
no prior installations are necessary, as pre-built (i.e.
cross-compiled) binaries of required libraries are downloaded
from a GitHub repository (<https://github.com/PolMine/libcl>)
during installation. On macOS, static libraries of Glib are
downloaded (<https://github.com/PolMine/libglib>) if Glib is
not present.

Imports  Rcpp (>= 0.12.6)
Suggests  knitr, testthat
LinkingTo  Rcpp
Biarch    true

URL  https://www.github.com/PolMine/RcppCWB

BugReports  https://github.com/PolMine/RcppCWB/issues

RoxygenNote  6.1.1

Collate  'RcppCWB_package.R' 'cl.R' 'cqp.R' 'cwb.R' 'checks.R'
         'count.R' 'RcppExports.R' 'decode.R' 'cbow.R' 'region_matrix.R'
         'misc.R' 'zzz.R'

Repository  CRAN

Date/Publication  2019-02-21 13:30:15 UTC

R topics documented:

RcppCWB-package ............................................. 3
check ......................................................... 5
check_pkg_registry_files .................................. 6
CL: p_attributes ............................................. 7
CL: s_attributes ............................................. 9
cl_attribute_size ......................................... 10
cl_charset_name ......................................... 11
cl_delete_corpus ........................................ 11
cl_lexicon_size ......................................... 12
cqp_initialize ........................................... 13
cqp_list_corpora ......................................... 13
cqp_query ................................................. 14
cwb_makeall .............................................. 15
get_cbow_matrix ........................................ 17
get_count_vector ....................................... 17
get_pkg_registry ....................................... 18
get_region_matrix ..................................... 19
ids_to_count_matrix .................................... 19
region_matrix_ops ..................................... 20
s_attribute_decode ..................................... 21
use_tmp_registry ....................................... 22

Index  23
**Description**

The RcppCWB package is a wrapper library to expose core functions of the Open Corpus Workbench (CWB). This includes the low-level functionality of the Corpus Library (CL) as well as capacities to use the query syntax of the Corpus Query Processor (CQP).

**The Idea Behind RcppCWB**

The Open Corpus Workbench (CWB) is an indexing and querying engine popular in corpus-assisted research. Its core aim is to support working efficiently with large, structurally and linguistically annotated corpora. First of all, the CWB includes tools to index and compress corpora. Second, the Corpus Library (CL) offers low-level functionality to retrieve information from CWB indexed corpora. Third, the Corpus Query Processor (CQP) offers a syntax that allows to perform anything from simple to complex queries, using different annotation layers of corpora.

The CWB is a classical tool which has inspired a set of developments. A persisting advantage of the CWB is its mature, open source code base that is actively maintained by a community of developers. It is used as a robust and efficient backend for widely used tools such as TXM (http://textometrie.ens-lyon.fr) or CQPweb (http://cwb.sourceforge.net/cqpweb.php). Its uncompromising C implementation guarantees speed and makes it well suited to be integrated with R at the same time.

The package RcppCWB is a follow-up on the rcap package that has pioneered to expose CWB functionality from within R. Indeed, the rcap package, published at CRAN in 2015, offers robust access to CWB functionality. However, the “pure C” implementation of the rcap package creates difficulties to make the package portable to Windows. The primary purpose of the RcppCWB package is to reimplement a wrapper library for the CWB using a design that makes it easier to achieve cross-platform portability.

Even though RcppCWB functions may be used directly, the package is designed to serve as an interface to CWB indexed corpora in packages with higher-level functionality. In this regard, RcppCWB is the backend of the polmineR package. It is deliberately open to be used in other contexts. The package may stimulate using linguistically annotated, indexed and compressed corpora on all platforms. The paradigm of working with text as linguistic data may benefit from RcppCWB.

**Implementation**

When building the package, the first step is to compile the relevant parts of the CWB on Linux and macOS machines. On Windows, cross-compiled binaries are downloaded from a GitHub repository of the PolMine Project (https://github.com/PolMine/libcl). Second, Rcpp wrappers are compiled and make the relevant functions of the Corpus Library and CQP accessible. In addition to genuine CWB functions, RcppCWB offers a set of higher level functions implemented using Rcpp for common performance critical tasks.
Getting Started with RcppCWB

To understand the data storage model of the CWB, in particular the notions of positional and structural attributes (s- and p-attributes), the vignette of the rcqp package is a very good starting point (see references).

The CWB 'Corpus Encoding Tutorial’ explains how to create your own corpus, the 'CQP Query Language Tutorial’ introduces the syntax of CQP (see references).

The RcppCWB package includes a sample corpus (REUTERS, the data also included in the tm package). The examples in the documentation of the functions may be a good starting point to understand how to use RcppCWB.

Digging Deeper

The original paper of Christ (1994) explains the design choices of the CWB. The indexing and compression techniques of the CWB (Huffman coding) are explained in Witten et al. (1999).

Acknowledgements

The work of the all developers of the CWB is gratefully acknowledged. There is a particular intellectual debt to Bernard Desgraupes and Sylvain Loiseau, and the rcqp package they developed as the original R wrapper to expose the functionality of the CWB.

Author(s)

Andreas Blaette (andreas.blaette@uni-due.de)

References


Desgraupes, B.; Loiseau, S. 2012. Introduction to the rcqp package. Vignette of the rcqp package. Available at the CRAN archive at https://cran.r-project.org/src/contrib/Archive/rcqp/


Open Corpus Workbench (http://cwb.sourceforge.net)


Examples

# functions of the corpus library (starting with cl) expose the low-level
# access to the CWB corpus library (CL)

# registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()
registry <- use_tmp_registry()
check

print(registry)
ids <- cl_cpos2id("REUTERS", cpos = 1:20, p_attribute = "word", registry = registry)
tokens <- cl_id2str("REUTERS", id = ids, p_attribute = "word", registry = registry)
print(paste(tokens, collapse = " "))

# To use the corpus query processor (CQP) and its syntax, it is necessary first
# to initialize CQP (example: get concordances of 'oil')

cqp_initialize(registry)
cqp_query("REUTERS", query = '[5] "oil" [5]')
cpos_matrix <- cqp_dump_subcorpus("REUTERS")
concordances_oil <- apply(
  cpos_matrix, 1,
  function(row){
    ids <- cl_cpos2id("REUTERS", p_attribute = "word", cpos = row[1]:row[2])
    tokens <- cl_id2str("REUTERS", p_attribute = "word", id = ids)
    paste(tokens, collapse = " ")
  }
)

check check check_check Input to Repp Functions.

Description
A set of functions to check whether the input values to the Repp wrappers for the C functions of
the Corpus Workbench potentially causing crashes are valid. These auxiliary functions are called
by the cl_ and cqp_ functions.

Usage

check_registry(registry)

check_corpus(corpus, registry)

check_s_attribute(s_attribute, corpus,
  registry = Sys.getenv("CORPUS_REGISTRY"))

check_p_attribute(p_attribute, corpus,
  registry = Sys.getenv("CORPUS_REGISTRY"))

check_strucs(corpus, s_attribute, strucs, registry)

check_region_matrix(region_matrix)

check_cqp_query(query)

check_cpos(corpus, p_attribute = "word", cpos,
check_pkg_registry_files

registry = Sys.getenv("CORPUS_REGISTRY")

check_id(corpus, p_attribute, id,
    registry = Sys.getenv("CORPUS_REGISTRY")
)

Arguments

    registry          path to registry directory
    corpus           name of a CWB corpus
    s_attribute      a structural attribute
    p_attribute      a positional attribute
    strucs           strucs (indices of structural attributes)
    region_matrix    a region matrix
    query            a CQP query
    cpos             vector of corpus positions
    id               id (encoded p-attribute), integer value

_____________________________________________________

check_pkg_registry_files

    Check Paths in Registry Files

_____________________________________________________

Description

    Check Paths in Registry Files

Usage

    check_pkg_registry_files(pkg = system.file(package = "RcppCWB"),
                                set = FALSE)

Arguments

    pkg              Full path to package directory
    set              Logical, whether

Value

    Logical value, whether home directories are set correctly.
Using Positional Attributes.

Description

CWB indexed corpora store the text of a corpus as numbers: Every token in the token stream of the corpus is identified by a unique corpus position. The string value of every token is identified by a unique integer id. The corpus library (CL) offers a set of functions to make the transitions between corpus positions, token ids, and the character string of tokens.

Usage

```r
cl_cpos2str(corpus, p_attribute, 
  registry = Sys.getenv("CORPUS_REGISTRY"), cpos)

cl_cpos2id(corpus, p_attribute, registry = Sys.getenv("CORPUS_REGISTRY"), 
  cpos)

cl_id2str(corpus, p_attribute, registry = Sys.getenv("CORPUS_REGISTRY"), 
  id)

cl_regex2id(corpus, p_attribute, regex, 
  registry = Sys.getenv("CORPUS_REGISTRY"))

cl_str2id(corpus, p_attribute, str, 
  registry = Sys.getenv("CORPUS_REGISTRY"))

cl_id2freq(corpus, p_attribute, id, 
  registry = Sys.getenv("CORPUS_REGISTRY"))

cl_id2cpos(corpus, p_attribute, id, 
  registry = Sys.getenv("CORPUS_REGISTRY"))
```

Arguments

- **corpus**: name of a CWB corpus (upper case)
- **p_attribute**: a p-attribute (positional attribute)
- **registry**: path to the registry directory, defaults to the value of the environment variable CORPUS_REGISTRY
- **cpos**: corpus positions (integer vector)
- **id**: id of a token
- **regex**: a regular expression
- **str**: a character string
Examples

# registry directory and cpos_total will be needed in examples
registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()
Sys.setenv(CORPUS_REGISTRY = registry)
cpos_total <- cl_attribute_size(
  corpus = "REUTERS", attribute = "word",
  attribute_type = "p", registry = registry
)

# decode the token stream of the corpus (the quick way)
token_stream_str <- cl_cpos2str(
  corpus = "REUTERS", p_attribute = "word",
  cpos = seq.int(from = 0, to = cpos_total - 1),
  registry = registry
)

token_stream_str <- cl_cpos2str(
  corpus = "REUTERS", p_attribute = "word",
  cpos = seq.int(from = 0, to = cpos_total - 1),
  registry = registry
)
token_stream_ids <- cl_id2str(
  corpus = "REUTERS", p_attribute = "word",
  id = token_stream_ids, registry = registry
)

token_to_get <- "oil"
id_oil <- cl_str2id(
  corpus = "REUTERS", p_attribute = "word",
  str = token_to_get
)
cpos_oil <- cl_id2cpos <- cl_id2cpos(
  corpus = "REUTERS", p_attribute = "word",
  id = id_oil
)

# get frequency of token
oil_freq <- cl_id2freq(
  corpus = "REUTERS", p_attribute = "word", id = id_oil
)
length(cpos_oil) # needs to be the same as oil_freq

# use regular expressions
ids <- cl_regex2id(
  corpus = "REUTERS", p_attribute = "word",
  regex = "M.*"
)
m_words <- cl_id2str(
  corpus = "REUTERS", p_attribute = "word",
  id = ids
)
Using Structural Attributes.

Description

Structural attributes store the metadata of texts in a CWB corpus and/or any kind of annotation of a region of text. The fundamental unit are so-called strucs, i.e. indices of regions identified by a left and a right corpus position. The corpus library (CL) offers a set of functions to make the translations between corpus positions (cpos) and strucs (struc).

Usage

\[
\begin{align*}
\text{cl\_cpos2struc} & \text{(corpus, s\_attribute, cpos,} \\
& \quad \text{registry = Sys.getenv("CORPUS\_REGISTRY"))} \\
\text{cl\_struc2cpos} & \text{(corpus, s\_attribute,} \\
& \quad \text{registry = Sys.getenv("CORPUS\_REGISTRY"), struc)} \\
\text{cl\_struc2str} & \text{(corpus, s\_attribute,} \\
& \quad \text{registry = Sys.getenv("CORPUS\_REGISTRY"))} \\
\text{cl\_cpos2lbound} & \text{(corpus, s\_attribute, cpos,} \\
& \quad \text{registry = Sys.getenv("CORPUS\_REGISTRY"))} \\
\text{cl\_cpos2rbound} & \text{(corpus, s\_attribute, cpos,} \\
& \quad \text{registry = Sys.getenv("CORPUS\_REGISTRY"))}
\end{align*}
\]

Arguments

- **corpus**: name of a CWB corpus (upper case)
- **s\_attribute**: name of structural attribute (character vector)
- **cpos**: corpus positions (integer vector)
- **registry**: path to the registry directory, defaults to the value of the environment variable CORPUS\_REGISTRY
- **struc**: a struc identifying a region

Examples

\[
\begin{align*}
\text{registry} & \leftarrow \text{if (!check\_pkg\_registry\_files()) use\_tmp\_registry()} \text{ else get\_pkg\_registry()} \\
\text{# get metadata for matches of token} \\
\text{# scenario: id of the texts with occurrence of 'oil'} \\
\text{token\_to\_get} & \leftarrow \text{"oil"} \\
\text{token\_id} & \leftarrow \text{cl\_str2id("REUTERS", p\_attribute = "word", str = \"oil\")}
\end{align*}
\]
cl_attribute_size

Get Attribute Size (of Positional/Structural Attribute).

Description

Use cl_attribute_size to get the total number of values of a positional attribute (param attribute_type = "p"), or structural attribute (param attribute_type = "s"). Note that indices are zero-based, i.e. the maximum position of a positional / structural attribute is attribute size minus 1 (see examples).

Usage

cl_attribute_size(corpus, attribute, attribute_type, registry = Sys.getenv("CORPUS_REGISTRY"))

Arguments

- corpus: name of a CWB corpus (upper case)
- attribute: name of a p- or s-attribute
- attribute_type: either "p" or "s", for structural/positional attribute
- registry: path to the registry directory, defaults to the value of the environment variable CORPUS_REGISTRY

Examples

registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()
Sys.setenv(CORPUS_REGISTRY = registry)
token_no <- cl_attribute_size("REUTERS", attribute = "word", attribute_type = "p")
corpus_positions <- seq.int(from = 0, to = token_no - 1)
c1_cpos2id("REUTERS", "word", cpos = corpus_positions)

# code block

token_cpos <- cl_id2cpos("REUTERS", p_attribute = "word", id = token_id)
strucs <- cl_cpos2struc("REUTERS", s_attribute = "id", cpos = token_cpos)
strucs_unique <- unique(strucs)
text_ids <- cl_struc2str("REUTERS", s_attribute = "id", struc = strucs_unique)

# get the full text of the first text with match for 'oil'
left_cpos <- cl_cpos2lbound("REUTERS", s_attribute = "id", cpos = min(token_cpos))
right_cpos <- cl_cpos2rbound("REUTERS", s_attribute = "id", cpos = min(token_cpos))
txt <- cl_cpos2str("REUTERS", p_attribute = "word", cpos = left_cpos:right_cpos)
fulltext <- paste(txt, collapse = " ")

# alternativ approach to achieve same result
first_struc_match_oil <- cl_cpos2struc("REUTERS", s_attribute = "id", cpos = min(token_cpos))
cpos_struc <- cl_struc2cpos("REUTERS", s_attribute = "id", struc = first_struc_match_oil)
txt <- cl_cpos2str("REUTERS", p_attribute = "word", cpos = cpos_struc[1]:cpos_struc[2])
fulltext <- paste(txt, collapse = " ")
Cl_charset_name

Get charset of a corpus.

Description

The encoding of a corpus is declared in the registry file (corpus property "charset"). Once a corpus is loaded, this information is available without parsing the registry file again and again. The cl_charset_name offers a quick access to this information.

Usage

cl_charset_name(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

Arguments

- corpus: Name of a CWB corpus (upper case).
- registry: Path to the registry directory, defaults to the value of the environment variable CORPUS_REGISTRY.

Examples

```r
cl_charset_name(
  corpus = "REUTERS",
  registry = system.file(package = "RcppCWB", "extdata", "cwb", "registry")
)
```

cl_delete_corpus

Drop loaded corpus.

Description

Remove a corpus from the list of loaded corpora of the corpus library (CL).

Usage

cl_delete_corpus(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

Arguments

- corpus: Name of a CWB corpus (upper case).
- registry: Path to the registry directory, defaults to the value of the environment variable CORPUS_REGISTRY.
Details

The corpus library (CL) internally maintains a list of corpora including information on positional and structural attributes so that the registry file needs not be parsed again and again. However, when an attribute has been added to the corpus, it will not yet be visible, because it is not part of the data that has been loaded. The cl_delete_corpus function exposes a CL function named identically, to force reloading the corpus (after it has been deleted), which will include parsing an updated registry file.

---

**cl_lexicon_size**

*Get Lexicon Size.*

Description

Get the total number of unique tokens/ids of a positional attribute. Note that token ids are zero-based, i.e. when iterating through tokens, start at 0, the maximum will be cl_lexicon_size() minus 1.

Usage

```r
cl_lexicon_size(corpus, p_attribute,
registry = Sys.getenv("CORPUS_REGISTRY"))
```

Arguments

- **corpus**
  name of a CWB corpus (upper case)
- **p_attribute**
  name of positional attribute
- **registry**
  path to the registry directory, defaults to the value of the environment variable CORPUS_REGISTRY

Examples

```r
registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()
Sys.getenv(CORPUS_REGISTRY = registry)
lexicon_size <- cl_lexicon_size("REUTERS", p_attribute = "word")
token_ids <- seq.int(from = 0, to = lexicon_size - 1)
cl_id2str("REUTERS", p_attribute = "word", id = token_ids)
```
cqp_initialize

Initialize Corpus Query Processor (CQP).

Description

CQP needs to know where to look for CWB indexed corpora. To initialize CQP, call cqp_initialize. To reset the registry, use the function cqp_reset_registry. To get the registry used by CQP, use cqp_get_registry. To get the initialization status, use cqp_is_initialized

Usage

```c
 cqp_initialize(registry = Sys.getenv("CORPUS_REGISTRY"))
 cqp_is_initialized()
 cqp_get_registry()
 cqp_reset_registry(registry = Sys.getenv("CORPUS_REGISTRY"))
```

Arguments

registry the registry directory

Author(s)

Andreas Blaette, Bernard Desgraupes, Sylvain Loiseau

Examples

```c
 cqp_is_initialized() # check initialization status
 if (!cqp_is_initialized()) cqp_initialize()
 cqp_is_initialized() # check initialization status (TRUE now?)
 cqp_get_registry() # get registry dir used by CQP

 registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()
 if (cqp_get_registry() != registry) cqp_reset_registry(registry = registry)
 cqp_list_corpora() # get list of corpora
```

---

cqp_list_corpora

List Available CWB Corpora.

Description

List the corpora described by the registry files in the registry directory that is currently set.
Usage

cqp_list_corpora()

Author(s)

Andreas Blaette, Bernard Desgraupes, Sylvain Loiseau

Examples

if (!cqp_is_initialized()){
    registry <- system.file(package = "RcppCWB", "extdata", "cwb", "registry")
    cqp_initialize(registry)
}
cqp_list_corpora()

---

**cqp_query**

*Execute CQP Query and Retrieve Results.*

Description

Using CQP queries requires a two-step procedure: At first, you execute a query using *cqp_query*. Then, *cqp_dump_subcorpus* will return a matrix with the regions of the matches for the query.

Usage

```r

cqp_query(corpus, query, subcorpus = "QUERY")
cqp_dump_subcorpus(corpus, subcorpus = "QUERY")
cqp_subcorpus_size(corpus, subcorpus = "QUERY")
cqp_list_subcorpora(corpus)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>corpus</td>
<td>a CWB corpus</td>
</tr>
<tr>
<td>query</td>
<td>a CQP query</td>
</tr>
<tr>
<td>subcorpus</td>
<td>subcorpus name</td>
</tr>
</tbody>
</table>

Details

The *cqp_query* function executes a CQP query. The *cqp_subcorpus_size* function returns the number of matches for the CQP query. The *cqp_dump_subcorpus* function will return a two-column matrix with the left and right corpus positions of the matches for the CQP query.

Author(s)

Andreas Blaette, Bernard Desgraupes, Sylvain Loiseau
cwb_makeall

References

Examples

```r
registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()

if (!cqp_is_initialized()){
  cqp_initialize(registry = registry)
} else {
  if (cqp_get_registry() != registry) cqp_reset_registry(registry)
}
cqp_query(corpus = "REUTERS", query = "oil;'

cqp_subcorpus_size("REUTERS")
cqp_dump_subcorpus("REUTERS")

cqp_query(corpus = "REUTERS", query = "crude "oil;'

cqp_subcorpus_size("REUTERS", subcorpus = "QUERY")
cqp_dump_subcorpus("REUTERS")
```

cwb_makeall  CWB Tools for Creating Corpora

Description
Wrappers for the CWB tools (cwb-makeall, cwb-huffcode, cwb-compress-rdx). Unlike the 'original' command line tools, these wrappers will always perform a specific indexing/compression step on one positional attribute, and produce all components.

Usage
```
cwb_makeall(corpus, p_attribute,
  registry = Sys.getenv("CORPUS_REGISTRY"))

cwb_huffcode(corpus, p_attribute,
  registry = Sys.getenv("CORPUS_REGISTRY"))

cwb_compress_rdx(corpus, p_attribute,
  registry = Sys.getenv("CORPUS_REGISTRY"))
```

Arguments
```
corpus name of a CWB corpus (upper case)
p_attribute name p-attribute
registry path to the registry directory, defaults to the value of the environment variable CORPUS_REGISTRY
```
Examples

# The package includes and 'unfinished' corpus of debates in the UN General
# Assembly ("UNGA"), i.e. it does not yet include the reverse index, and it is
# not compressed.
#
# The first step in the following example is to copy the raw
# corpus to a temporary place.

registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()
home_dir <- system.file(package = "RcppCWB", "extdata", "cwb", "indexed_corpora", "unga")

tmpdir <- tempdir()
win <- if (Sys.info()["sysname"] == "Windows") TRUE else FALSE
if (win) tmpdir <- normalizePath(tmpdir)
tmp_regdir <- file.path(tmpdir, "registry", fsep = if (win) "\\" else "/")
tmp_data_dir <- file.path(tmpdir, "indexed_corpora", fsep = if (win) "\\" else "/")
tmp_unga_dir <- file.path(tmp_data_dir, "unga", fsep = if (win) "\\" else "/")
if (!file.exists(tmp_regdir)) dir.create(tmp_regdir)
if (!file.exists(tmp_data_dir)) dir.create(tmp_data_dir)
if (!file.exists(tmp_unga_dir)){
  dir.create(tmp_unga_dir)
} else {
  file.remove(list.files(tmp_unga_dir, full.names = TRUE))
}
regfile <- readLines(file.path(registry, "unga"))
homedir_line <- grep("\^home\B", regfile)
regfile[homedir_line] <- sprintf("HOME \%s", tmp_unga_dir)
writelines(text = regfile, con = file.path(tmp_regdir, "unga"))
for (x in list.files(home_dir, full.names = TRUE)){
  file.copy(from = x, to = tmp_unga_dir)
}

# perform cwb_makeall (equivalent to cwb-makeall command line utility)
cwb_makeall(corpus = "UNGA", p_attribute = "word", registry = tmp_regdir)

# see whether it works
ids_sentence_1 <- cl_cpos2id(
  corpus = "UNGA", p_attribute = "word", registry = tmp_regdir,
  cpos = 0:83
)
tokens_sentence_1 <- cl_id2str(
  corpus = "UNGA", p_attribute = "word",
  registry = tmp_regdir, id = ids_sentence_1
)
sentence <- gsub("\\s*([\\\.
])", "\\1", paste(tokens_sentence_1, collapse = " "))
## Not run:
cwb_huffcode(corpus = "UNGA", p_attribute = "word", registry = tmp_regdir)
## End(Not run)
cwb_compress_rdx(corpus = "UNGA", p_attribute = "word", registry = tmp_regdir)
**get_cbow_matrix**

*Get CBOW Matrix.*

**Description**

Get matrix with moving windows. Negative integer values indicate absence of a token at the respective position.

**Usage**

```r
get_cbow_matrix(corpus, p_attribute, 
    registry = Sys.getenv("CORPUS_REGISTRY"), matrix, window)
```

**Arguments**

- **corpus**: a CWB corpus
- **p_attribute**: a positional attribute
- **registry**: the registry directory
- **matrix**: a matrix
- **window**: window size

**Examples**

```r
registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()

m <- get_region_matrix(
    corpus = "REUTERS", s_attribute = "places",
    strucs = 0:5L, registry = registry )
windowsize <- 3L
m2 <- get_cbow_matrix(
    corpus = "REUTERS", p_attribute = "word",
    registry = registry, matrix = m, window = windowsize )

colnames(m2) <- c(-windowsize:-1, "node", 1:windowsize)
```

---

**get_count_vector**

*Get Vector with Counts for Positional Attribute.*

**Description**

The return value is an integer vector. The length of the vector is the number of unique tokens in the corpus / the number of unique ids. The order of the counts corresponds to the number of ids.
Usage

get_count_vector(corpus, p_attribute, registry = Sys.getenv("CORPUS_REGISTRY"))

Arguments

corpus: a CWB corpus
p_attribute: a positional attribute
registry: registry directory

Value

an integer vector

Examples

 registry <- use_tmp_registry()
y <- get_count_vector(
  corpus = "REUTERS", p_attribute = "word",
  registry = registry
)
df <- data.frame(token_id = 0:(length(y) - 1), count = y)
df[["token"]]<- cl_id2str(
  "REUTERS", p_attribute = "word",
  id = df[["token_id"]], registry = registry
)
df <- df[,c("token", "token_id", "count")]
  # reorder columns
df <- df[order(df["count"], decreasing = TRUE),]
head(df)

get_pkg_registry

Get Registry Directory Within Package

Description

Get Registry Directory Within Package

Usage

get_pkg_registry(pkgname = "RcppCWB")

Arguments

pkgname: Name of package (character vector)
get_region_matrix

Get Matrix with Regions for Strucs.

Description

The return value is an integer matrix with the left and right corpus positions of the strucs in columns one and two, respectively.

Usage

get_region_matrix(corpus, s_attribute, strucs,
registry = Sys.getenv("CORPU$REGISTRY"))

Arguments

corpus a CWB corpus
s_attribute a structural attribute
strucs
registry the registry directory

Value

A matrix with integer values indicating left and right corpus positions (columns 1 and 2, respectively).

Examples

registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()
y <- get_region_matrix(
corpus = "REUTERS", s_attribute = "id",
strucs = 0:5, registry = registry
)

ids_to_count_matrix

Perform Count for Vector of IDs.

Description

The return value is a two-column integer matrix. Column one represents the unique ids of the input vector, column two the respective number of occurrences / counts.

Usage

ids_to_count_matrix(ids)
Arguments
ids a vector of ids (integer values)

Examples
ids <- c(1L, 5L, 5L, 7L, 7L, 7L, 7L)
ids_to_count_matrix(ids)
table(ids) # alternative to get a similar result

region_matrix_ops Get IDs and Counts for Region Matrices.

Description
Get IDs and Counts for Region Matrices.

Usage
region_matrix_to_ids(corpus, p_attribute, registry = Sys.getenv("CORPUS_REGISTRY"), matrix)
region_matrix_to_count_matrix(corpus, p_attribute, registry = Sys.getenv("CORPUS_REGISTRY"), matrix)

Arguments
corpus a CWB corpus
p_attribute a positional attribute
registry registry directory
matrix a regions matrix

Examples
registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()

# Scenario 1: Get full text for a subcorpus defined by regions
m <- get_region_matrix(
corpus = "REUTERS", s_attribute = "places",
strucs = 4L:5L, registry = registry
)
ids <- region_matrix_to_ids(
corpus = "REUTERS", p_attribute = "word",
registry = registry, matrix = m
)
tokenstream <- cl_id2str(
corpus = "REUTERS", p_attribute = "word",
registry = registry, id = ids
)
s_attribute_decode

Description

Get data.frame with counts for region matrix.

Usage

s_attribute_decode(corpus, data_dir, s_attribute, encoding = NULL, registry = Sys.getenv("CORPUS_REGISTRY"), method = c("R", "Rcpp"))

Arguments

corpus a CWB corpus
data_dir data directory where binary files for corpus are stored
s_attribute a structural attribute
encoding encoding of the values ("latin-1" or "utf-8")
registry registry directory
method character vector, whether to use "R" or "Rcpp" implementation

Details

Two approaches are implemented: A pure R solution will decode the files directly in the directory specified by data_dir. An implementation using Rcpp will use the registry file for corpus to find the data directory.

Value

A data.frame with three columns. Column cpos_left are the start corpus positions of a structural annotation, cpos_right the end corpus positions. Column value is the value of the annotation. a character vector

txt <- paste(tokenstream, collapse = " ")
txt

# Scenario 2: Get data.frame with counts for region matrix
y <- region_matrix_to_count_matrix(
    corpus = "REUTERS", p_attribute = "word",
    registry = registry, matrix = m
)
df <- as.data.frame(y)
colnames(df) <- c("token_id", "count")
df[["token"]] <- cl_id2str(
    "REUTERS", p_attribute = "word",
    registry = registry, id = df[["token_id"]]
)
df[order(df[["count"]], decreasing = TRUE),]
head(df)
**use_tmp_registry**

**Description**

Use and get temporary registry directory to describe and access the corpora in a package.

**Usage**

```r
use_tmp_registry(pkg = system.file(package = "RcppCWB"))
```

get_tmp_registry()

**Arguments**

- **pkg** Full path to a package.

**Examples**

```r
registry <- if (!check_pkg_registry_files()) use_tmp_registry() else get_pkg_registry()
Sys.setenv(CORPSUS_REGISTRY = registry)

# pure R implementation (Rcpp implementation fails on Windows in vanilla mode)
b <- s_attribute_decode(
  data_dir = system.file(package = "RcppCWB", "extdata", "cwb", "indexed_corpora", "reuters"),
  s_attribute = "places", method = "R"
)
```
Index

*Topic package
   RcppCWB-package, 3

check, 5
check_corpus (check), 5
check_cpos (check), 5
check_cqp_query (check), 5
check_id (check), 5
check_pkg_attribute (check), 5
check_pkg_registry_files, 6
check_region_matrix (check), 5
check_registry (check), 5
check_s_attribute (check), 5
check_strucs (check), 5
CL: p_attributes, 7
CL: s_attributes, 9
c1_attribute_size, 10
c1_charset_name, 11
c1_cpos2id (CL: p_attributes), 7
c1_cpos2lbound (CL: s_attributes), 9
c1_cpos2rbound (CL: s_attributes), 9
c1_cpos2str (CL: p_attributes), 7
c1_cpos2struc (CL: s_attributes), 9
c1_delete_corpus, 11
c1_id2cpos (CL: p_attributes), 7
c1_id2freq (CL: p_attributes), 7
c1_id2str (CL: p_attributes), 7
c1_lexicon_size, 12
c1_regex2id (CL: p_attributes), 7
c1_str2id (CL: p_attributes), 7
c1_struc2cpos (CL: s_attributes), 9
c1_struc2str (CL: s_attributes), 9
cqp_dump_subcorpus (cqp_query), 14
cqp_get_registry (cqp_initialize), 13
cqp_initialize, 13
cqp_is_initialized (cqp_initialize), 13
cqp_list_corpora, 13
cqp_list_subcorpora (cqp_query), 14
cqp_query, 14
cqp_reset_registry (cqp_initialize), 13
cqp_subcorpus_size (cqp_query), 14
cwb_compress_rdx (cwb_makeall), 15
cwb_huffcode (cwb_makeall), 15
cwb_makeall, 15
get_cbow_matrix, 17
get_count_vector, 17
get_pkg_registry, 18
get_region_matrix, 19
get_tmp_registry (use_tmp_registry), 22
ids_to_count_matrix, 19
RcppCWB (RcppCWB-package), 3
RcppCWB-package, 3
region_matrix_ops, 20
region_matrix_to_count_matrix
   (region_matrix_ops), 20
region_matrix_to_ids
   (region_matrix_ops), 20
s_attribute_decode, 21
use_tmp_registry, 22