Package 'RcppCWB'

April 21, 2018

Type Package

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

Version 0.2.1

Date 2018-04-09

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/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

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

NeedsCompilation yes

SystemRequirements On Unix-like operating systems, pcre (>= 7) and glib2 (>= 2) need to be present to compile 'RcppCWB'. See README.md for instructions how to install these dependencies. On Windows, no prior installations are necessary, as pre-built (i.e. cross-compiled) binaries of all required libraries are downloaded from a GitHub repository (<https://github.com/PolMine/libcl>) during installation.

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 6.0.1

Collate 'RcppCWB_package.R' 'cl.R' 'cqp.R' 'checks.R' 'count.R' 'RcppExports.R' 'encode.R' 'decode.R' 'registry.R' 'cbow.R' 'region_matrix.R'

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

Repository CRAN

Date/Publication 2018-04-21 21:56:39 UTC

R topics documented:

RcppCWB-package *Rcpp Bindings for the Corpus Workbench (CWB).*

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 rcqp package that has pioneered to expose CWB functionality from within R. Indeed, the rcqp package, published at CRAN in 2015, offers robust access to CWB functionality. However, the "pure C" implementation of the rcqp 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

Christ, O. 1994. "A modular and flexible architecture for an integrated corpus query system", in: Proceedings of COMPLEX '94, pp. 23-32. Budapest. Available online at http://cwb.sourceforge.net/files/Christ1994.pdf

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/

Evert, S. 2005. The CQP Query Language Tutorial. Available online at http://cwb.sourceforge.net/files/CWB_Encoding_Tutorial.pdf

Evert, S. 2005. The IMS Open Corpus Workbench (CWB). Corpus Encoding Tutorial. Available online at http://cwb.sourceforge.net/files/CWB_Encoding_Tutorial.pdf

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

Witten, I.H.; Moffat, A.; Bell, T.C. (1999). Managing Gigabytes. Morgan Kaufmann Publishing, San Francisco, 2nd edition.

Examples

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

regdir <- system.file(package = "RcppCWB", "extdata", "cwb", "registry")
ids <- cl_cpos2id("REUTERS", cpos = 1:20, p_attribute = "word", registry = regdir)
```

```
tokens <- cl_id2str("REUTERS", id = ids, p_attribute = "word", registry = regdir)
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(regdir)
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 = " ")
  }
 )
```

CL: p_attributes *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

```
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 <- system.file(package = "RcppCWB", "extdata", "cwb", "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
  )

# decode the token stream (cpos2id first, then id2str)
token_stream_ids <- cl_cpos2id(
  corpus = "REUTERS", p_attribute = "word",
  cpos = seq.int(from = 0, to = cpos_total - 1),
  registry = registry
  )
token_stream_str <- cl_id2str(
  corpus = "REUTERS", p_attribute = "word",
  id = token_stream_ids, registry = registry
)

# get corpus positions of a token
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
)
```

CL: s_attributes *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

```
cl_cpos2struc(corpus, s_attribute, cpos,
  registry = Sys.getenv("CORPUS_REGISTRY"))

cl_struc2cpos(corpus, s_attribute, registry = Sys.getenv("CORPUS_REGISTRY"),
  struc)

cl_struc2str(corpus, s_attribute, struc,
  registry = Sys.getenv("CORPUS_REGISTRY"))

cl_cpos2lbound(corpus, s_attribute, cpos,
  registry = Sys.getenv("CORPUS_REGISTRY"))

cl_cpos2rbound(corpus, s_attribute, cpos,
  registry = Sys.getenv("CORPUS_REGISTRY"))
```

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

```
# get registry directory
registry <- system.file(package = "RcppCWB", "extdata", "cwb", "registry")

# get metadata for matches of token
# scenario: id of the texts with occurrence of 'oil'
token_to_get <- "oil"
token_id <- cl_str2id("REUTERS", p_attribute = "word", str = "oil")
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_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 <- system.file(package = "RcppCWB", "extdata", "cwb", "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)
cl_cpos2id("REUTERS", "word", cpos = corpus_positions)

places_no <- cl_attribute_size("REUTERS", attribute = "places", attribute_type = "s")
strucs <- seq.int(from = 0, to = places_no - 1)
cl_struc2str("REUTERS", "places", struc = strucs)
```

---

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

```
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

```
registry <- system.file(package = "RcppCWB", "extdata", "cwb", "registry")
Sys.setenv(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

```
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

```
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
regdir <- system.file(package = "RcppCWB", "extdata", "cwb", "registry")
if (cqp_get_registry() != regdir) cqp_reset_registry(registry = regdir)
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

```
cqp_query(corpus, query, subcorpus = "QUERY")

cqp_dump_subcorpus(corpus, subcorpus = "QUERY")

cqp_subcorpus_size(corpus, subcorpus = "QUERY")

cqp_list_subcorpora(corpus)
```

Arguments

`corpus`	a CWB corpus
`query`	a CQP query
`subcorpus`	subcorpus name

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

References

Evert, S. 2005. The CQP Query Language Tutorial. Available online at http://cwb.sourceforge.net/files/CWB_Encoding_Tutorial.pdf

Examples

```
registry <- system.file(package = "RcppCWB", "extdata", "cwb", "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")
```

---

encode *Encode CWB Corpus.*

Description

Create indexed and compressed positional attribute from a character vector of tokens (the token stream), and structural annotations.

Usage

```
p_attribute_encode(token_stream, p_attribute = "word", data_dir)

p_attribute_makeall(token_stream, p_attribute = "word", data_dir)

p_attribute_huffcode(corpus, p_attribute,
  registry_dir = Sys.getenv("CORPUS_REGISTRY"))

p_attribute_compress_rdx(corpus, p_attribute,
  registry_dir = Sys.getenv("CORPUS_REGISTRY"))

s_attribute_encode(values, data_dir, s_attribute, region_matrix)
```

Arguments

token_stream	a character vector with the tokens of the corpus
p_attribute	the positional attribute

data_dir	the data directory for the corpus with the binary files
corpus	the CWB corpus (needed by p_attribute_huffcode and p_attribute_compress_rdx)
registry_dir	registry directory (needed by p_attribute_huffcode and p_attribute_compress_rdx)
values	values for structural annotations
s_attribute	the structural attribute
region_matrix	a matrix with left and right corpus positions (regions) of structural annotations

Details

Four steps generate the binary CWB corpus data format for positional attributes: First, encode a character vector (the token stream) using p_attribute_encode. Second, create reverse index using p_attribute_makeall. Third, compress token stream using p_attribute_huffcode. Fourth, compress index files using p_attribute_compress_rdx.

The implementation for the first two steps (p_attribute_encode and p_attribute_makeall) is a pure R implementation (so far). These two steps are enough to use the CQP functionality.

To run p_attribute_huffcode and p_attribute_compress_rdx, an installation of the CWB is required.

See the CQP Corpus Encoding Tutorial (http://cwb.sourceforge.net/files/CWB_Encoding_Tutorial.pdf) for an explanation of the procedure (section 3, "Indexing and compression without CWB/Perl").

Examples

```
tokens <- readLines(system.file(package = "RcppCWB", "extdata", "examples", "reuters.txt"))

tmpdir <- tempdir()
if (.Platform$OS.type == "windows") tmpdir <- normalizePath(tmpdir, winslash = "/")
registry_tmp <- file.path(tmpdir, "registry")
data_dir_tmp <- file.path(tmpdir, "data_dir")
dir.create (registry_tmp)
dir.create(data_dir_tmp)

p_attribute_encode(
  token_stream = tokens, p_attribute = "word",
  data_dir = data_dir_tmp
  )
p_attribute_makeall(
  token_stream = tokens, p_attribute = "word",
  data_dir = data_dir_tmp
)

regfile <- registry_file_write(
  registry_dir = registry_tmp, corpus = "REUTERS", data_dir = data_dir_tmp,
  corpus_properties = c(encoding = "utf-8", language = "en"), p_attributes = "word"
  )
if (cqp_is_initialized()) cqp_reset_registry(registry_tmp) else cqp_initialize(registry_tmp)

cqp_query(corpus = "REUTERS", query = '[]{3} "oil" []{3};')
regions <- cqp_dump_subcorpus(corpus = "REUTERS")
```

```
kwic <- apply(regions, 1, function(region){
  ids <- cl_cpos2id("REUTERS", "word", registry_tmp, cpos = region[1]:region[2])
  words <- cl_id2str(corpus = "REUTERS", p_attribute = "word", registry = registry_tmp, id = ids)
  paste0(words, collapse = " ")
})
kwic[1:10]
```

`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

```
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

```
registry <- system.file(package = "RcppCWB", "extdata", "cwb", "registry")
m <- get_region_matrix(
  corpus = "REUTERS", s_attribute = "places",
  strucs = 0L: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 <- system.file(package = "RcppCWB", "extdata", "cwb", "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_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("CORPUS_REGISTRY"))
```

Arguments

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

Value

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

Examples

```
registry <- system.file(package = "RcppCWB", "extdata", "cwb", "registry")
y <- get_region_matrix(
  corpus = "REUTERS", s_attribute = "id",
  strucs = 0L:5L, 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 <- system.file(package = "RcppCWB", "extdata", "cwb", "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
  )
```

```
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)
```

`registry file` *Generate and Write Registry File.*

Description

Generate a registry file describing a CWB indexed corpus. `registry_file_make` will create a character vector with the content of the registry file. `registry_file_write` will write also write this content as a registry file to directors `registry_dir`.

Usage

```
registry_file_make(registry_dir, corpus, description = "", data_dir,
  info_file = NULL, corpus_properties = c(language = "en", charset =
  "latin-1"), p_attributes = "word", s_attributes = NULL)

registry_file_write(registry_dir, corpus, ...)
```

Arguments

`registry_dir`	the registry directory
`corpus`	name of the corpus, will be used as ID (using tolower)
`description`	long descriptive name for the corpus
`data_dir`	path to binary data files
`info_file`	optional info file, if NULL, we assume that file ".info.md" is in `data_dir`
`corpus_properties`	named vector of corpus properties; properties `language` and `charset` need to be present
`p_attributes`	positional attributes to be declared
`s_attributes`	structural attributes to be declared
`...`	parameters that are passed into `registry_file_write`

Details

A CWB indexed corpus needs to be described in a registry file. See the CWB Corpus Encoding Tutorial, Appendix A (pp. 21f) for an example registry file (http://cwb.sourceforge.net/files/CWB_Encoding_Tutorial.pdf).

In addition to the corpus properties explicitly declared, the corpus property "drive_letter" will be added on Windows machines to handle the case that the data directory may be on another drive than the drive of the current working directory.

Value

the character vector of the registry file is returned invisibly

Examples

```
dir.create(temp_regdir <- tempfile())
temp_data_dir <- tempfile()
if (.Platform$OS.type == "windows"){
  temp_data_dir <- normalizePath(temp_data_dir, winslash = "/")
}

reuters_regfile <- registry_file_make(
  corpus = "REUTERS", description = "Reuters Example Corpus",
  data_dir = temp_data_dir, info_file = NULL,
  corpus_properties = c(language = "en", charset = "latin-1"),
  p_attributes = "word", s_attributes = c("places", "id")
  )
registry_file_write(
  registry_dir = temp_regdir, corpus = "REUTERS", description = "Reuters Example Corpus",
  data_dir = temp_data_dir, info_file = NULL,
  corpus_properties = c(language = "en", charset = "latin-1"),
  p_attributes = "word", s_attributes = c("places", "id")
  )
regfile <- readLines(file.path(temp_regdir, "reuters"))
```

---

s_attribute_decode *Decode Structural Attribute.*

Description

Get `data.frame` with left and right corpus positions (cpos) for structural attributes and values.

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

Examples

```
registry <- system.file(package = "RcppCWB", "extdata", "cwb", "registry")
Sys.setenv(CORPUS_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