Package ‘RcppCWB’

March 6, 2022

Type Package

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

Version 0.5.1

Date 2022-03-05

Author Andreas Blaette [aut, cre],
      Bernard Desgraupes [aut],
      Sylvain Loiseau [aut],
      Oliver Christ [ctb],
      Bruno Maximilian Schulze [ctb],
      Stefan Evert [ctb],
      Arne Fitschen [ctb],
      Jeroen Ooms [ctb],
      Marius Bertram [ctb],
      Tomas Kalibera [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 (<https://cwb.sourceforge.io>). '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 <http://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

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 < 10), 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 (>= 1.0.7), fs
Suggests: knitr, testthat
LinkingTo: Rcpp
Biarch: true

URL: https://github.com/PolMine/RcppCWB
BugReports: https://github.com/PolMine/RcppCWB/issues
RoxygenNote: 7.1.2

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

Repository: CRAN
Date/Publication: 2022-03-06 10:50:02 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 ............................................... 11
- cl_charset_name ................................................ 12
- cl_delete_corpus ............................................... 13
- cl_find_corpus .................................................. 14
- cl_lexicon_size ................................................ 14
- cl_list_corpora ............................................... 15
- cl_load_corpus ............................................... 15
- cl_rework ....................................................... 16
- cl_struc_values ............................................... 17
- corpus_data_dir .............................................. 18
- corpus_is_loaded ............................................. 18
- cqp_initialize ............................................... 19
- cqp_list_corpora ............................................ 20
- cqp_query ...................................................... 20
- cwb_charssets ............................................... 21
- cwb_makeall ................................................... 22
RcppCWB-package

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(https://txm.gitpages.huma-num.fr/textometrie/) or CQPweb (https://cwb.sourceforge.io/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 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 (https://cwb.sourceforge.io)


Examples

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

generate_r_ids <- cl_cpos2id("REUTERS", cpos = 1:20, p_attribute = "word", registry = get_tmp_registry())
generate_r_tokens <- cl_id2str("REUTERS", id = generate_r_ids, p_attribute = "word", registry = get_tmp_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_query("REUTERS", query = '/\{5\} oil /\{5\}')
cqp_matrix <- cqp_dump_subcorpus("REUTERS")

# Apply a function to each row of the CQP matrix
# ids <- cl_cpos2id("REUTERS", p_attribute = "word", cpos = row[1]:row[2], registry = get_tmp_registry())
tokens <- cl_id2str("REUTERS", p_attribute = "word", id = ids, registry = get_tmp_registry())
paste(tokens, collapse = " ")

check

Check Input to Rcpp Functions.

Description

A set of functions to check whether the input values to the Rcpp 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_pkg_registry_files

Arguments

registry path to registry directory
    corpus name of a CWB corpus
    s_attribute a structural attribute
    p_attribute a positional attribute
    structs structs (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

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.

---

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

```r
# registry directory and cpos_total will be needed in examples
cpos_total <- cl_attribute_size(
  corpus = "REUTERS", attribute = "word",
  attribute_type = "p", registry = get_tmp_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 = get_tmp_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 = get_tmp_registry()
)
token_stream_str <- cl_id2str(
  corpus = "REUTERS", p_attribute = "word",
  id = token_stream_ids, registry = get_tmp_registry()
)

# get corpus positions of a token
token_to_get <- "oil"
id_oil <- cl_str2id(
  corpus = "REUTERS", p_attribute = "word",
  str = token_to_get, registry = get_tmp_registry()
)
cpos_oil <- cl_id2cpos <- cl_id2cpos(
  corpus = "REUTERS", p_attribute = "word",
  id = id_oil, registry = get_tmp_registry()
)

# get frequency of token
oil_freq <- cl_id2freq(
  corpus = "REUTERS", p_attribute = "word", id = id_oil, registry = get_tmp_registry()
)
```
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")
)

c1_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 An integer vector with corpus positions.
registry path to the registry directory, defaults to the value of the environment variable CORPUS_REGISTRY
struc a struc identifying a region

Examples

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

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

```r
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**

```r
token_no <- cl_attribute_size(
  "REUTERS",
  attribute = "word",
  attribute_type = "p",
  registry = get_tmp_registry()
)
corpus_positions <- seq.int(from = 0, to = token_no - 1)
cp <- cl_cpos2id(
  "REUTERS",
  "word",
  cpos = corpus_positions,
  registry = get_tmp_registry()
)

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

---

**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**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>corpus</td>
<td>Name of a CWB corpus (upper case).</td>
</tr>
<tr>
<td>registry</td>
<td>Path to the registry directory, defaults to the value of the environment variable CORPUS_REGISTRY</td>
</tr>
</tbody>
</table>
cl_delete_corpus

Examples

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.

Value

An integer value 1 is returned invisibly if a previously loaded corpus has been deleted, or 0 if the corpus has not been loaded and has not been deleted.

Examples

cl_attribute_size("UNGA", attribute = "word", attribute_type = "p")
corpus_is_loaded("UNGA")
cl_delete_corpus("UNGA")
corpus_is_loaded("UNGA")
**cl_find_corpus**  
*Load corpus.*

**Description**

Load corpus.

**Usage**

```r
cl_find_corpus(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`

**Value**

A `externalptr` referencing the C representation of the corpus.

---

**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
lexicon_size <- cl_lexicon_size(
  "REUTERS",
  p_attribute = "word",
  registry = get_tmp_registry()
)

token_ids <- seq.int(from = 0, to = lexicon_size - 1)
cl_id2str(
  "REUTERS",
  p_attribute = "word",
  id = token_ids,
  registry = get_tmp_registry()
)
```

---

### cl_list_corpora

**Show CL corpora**

**Description**

Show CL corpora

**Usage**

```r
cl_list_corpora()
```

**Value**

A character vector.

**Examples**

```r
cl_list_corpora()
```

---

### cl_load_corpus

**Load corpus**

**Description**

Load corpus

**Usage**

```r
cl_load_corpus(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))
```
Arguments

- corpus: A length-one character vector with the corpus ID.
- registry: A length-one character vector with the registry directory.

Value

TRUE if corpus could be loaded and FALSE if not.

Examples

cl_load_corpus("REUTERS")

cl_rework

Experimental low-level CL access.

Description

Set of functions with same functionality as cl_* functions to improve the ease of writing code.

Usage

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

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

p_attr_size(p_attr)

s_attr_size(s_attr)

p_attr_lexicon_size(p_attr)

cpos_to_struc(cpos, s_attr)

cpos_to_str(cpos, p_attr)

cpos_to_id(cpos, p_attr)

struc_to_cpos(struc, s_attr)

struc_to_str(struc, s_attr)

regex_to_id(regex, p_attr)

str_to_id(str, p_attr)

id_to_freq(id, p_attr)
**cl_struc_values**

id_to_cpos(id, p_attr)
cpos_to_lbound(cpos, s_attr)
cpos_to_rbound(cpos, s_attr)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>corpus</td>
<td>ID of a CWB corpus (length-one character vector).</td>
</tr>
<tr>
<td>s_attribute</td>
<td>A structural attribute (length-one character vector).</td>
</tr>
<tr>
<td>registry</td>
<td>Registry directory.</td>
</tr>
<tr>
<td>p_attribute</td>
<td>A positional attribute (length-one character vector).</td>
</tr>
<tr>
<td>p_attr</td>
<td>A externalptr referencing a p-attribute.</td>
</tr>
<tr>
<td>s_attr</td>
<td>A externalptr referencing a p-attribute.</td>
</tr>
<tr>
<td>cpos</td>
<td>An integer vector of corpus positions.</td>
</tr>
<tr>
<td>struc</td>
<td>A length-one integer vector with a struc.</td>
</tr>
<tr>
<td>regex</td>
<td>A regular expression.</td>
</tr>
<tr>
<td>str</td>
<td>A character vector.</td>
</tr>
<tr>
<td>id</td>
<td>An integer vector with token ids.</td>
</tr>
</tbody>
</table>

---

**cl_struc_values**  \( \quad \text{Check whether structural attribute has values} \)

**Description**

Structural attributes do not necessarily have values, structural attributes (such as annotations of sentences or paragraphs) may just define regions of corpus positions. Use this function to test whether an attribute has values.

**Usage**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>corpus</td>
<td>Corpus ID, a length-one character vector.</td>
</tr>
<tr>
<td>s_attribute</td>
<td>Structural attribute to check, a length-one character vector.</td>
</tr>
<tr>
<td>registry</td>
<td>The registry directory of the corpus.</td>
</tr>
</tbody>
</table>

**Value**

TRUE if the attribute has values and FALSE if not. NA if the structural attribute is not available.

**Examples**

```
cl_struc_values("REUTERS", "places") # TRUE - attribute has values
cl_struc_values("REUTERS", "date")  # NA - attribute does not exist
```
corpus_data_dir  Get data directory of a corpus

Description
Extract the data directory from the internal C representation of the content of the registry file for a corpus.

Usage
corpus_data_dir(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

Arguments
  corpus A length-one character vector with the corpus ID.
  registry A length-one character vector with the registry directory.

Value
A length-one character vector stating the data directory.

Examples
corpus_data_dir("REUTERS")

corpus_is_loaded Check whether corpus is loaded

Description
Check whether corpus is loaded

Usage
corpus_is_loaded(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

Arguments
  corpus A length-one character vector with the corpus ID.
  registry A length-one character vector with the registry directory.

Value
TRUE if corpus is loaded and FALSE if not.
**cqp_initialize**

Initializes the 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

```r
# Initialize CQP using the default registry
cqp_initialize()

cqp_is_initialized()

cqp_verbosity(silent = FALSE, verbose = TRUE)

cqp_get_registry()

cqp_reset_registry(registry = Sys.getenv("CORPUS_REGISTRY"))

cqp_load_corpus(corpus, registry)
```

### Arguments

- **registry**: the registry directory
- **silent**: A single logical value, whether to be silent and suppress CQP messages (TRUE), or not (FALSE).
- **verbose**: A single logical value, whether to show verbose parser output (TRUE) or not (FALSE).
- **corpus**: ID of a CWB corpus (length-one character).

### Details

- `cqp_load_corpus` will return a logical value - TRUE if corpus has been loaded successfully, FALSE if not.

### Author(s)

Andreas Blaette, Bernard Desgraupes, Sylvain Loiseau

### Examples

```r
# 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

cqp_load_corpus() # 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**  
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)  
cqp_drop_subcorpus(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


Examples

```r
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_charsets  Character sets supported by CWB

Description

The function returns a character vector with character sets (charsets) supported by the Corpus Workbench (CWB). The vector is derived from the `CorpusCharset` object defined in the header file of the corpus library (CL).

Usage

```r
cwb_charsets()
```

Details

Early versions of the CWB were developed for "latin1", "utf8" support has been introduced with CWB v3.2. Note that RcppCWB is tested only for "latin1" and "utf8" and that R uses "UTF-8" rather than utf8" (CWB) by convention.

Examples

```r
cwb_charsets()
```
**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**

```r
cwb_makeall(
  corpus,
  p_attribute,
  registry = Sys.getenv("CORPUS_REGISTRY"),
  quietly = FALSE
)

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

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

cwb_encode(
  corpus,
  registry = Sys.getenv("CORPUS_REGISTRY"),
  data_dir,
  vrt_dir,
  encoding = "utf8",
  p_attributes = c("word", "pos", "lemma"),
  s_attributes,
  skip_blank_lines = TRUE,
  strip_whitespace = TRUE,
  xml = TRUE,
  quietly = FALSE,
  verbose = FALSE
)```
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
quietly A logical value, whether to turn off messages (including warnings).
data_dir The data directory where cwb_encode will put the binary files of the indexed corpus.
vrt_dir Directory with input corpus files (verticalised format / file ending *.vrt).
encoding The encoding of the files to be encoded. Needs to be an encoding supported by CWB, see cwb_charsets(). "UTF-8" is taken as "utf8". Defaults to "utf8" (recommended charset).
p_attributes Positional attributes (p-attributes) to be declared.
s_attributes A list of named character vectors to declare structural attributes that shall be encoded. The names of the list are the XML elements present in the corpus. Character vectors making up the list declare the attributes that include the metadata of regions. To declare a structural attribute without annotations, provide a zero-length character vector using character() - see examples.
skip_blank_lines A logical value, whether to skip blank lines in the input.
strip_whitespace A logical value, whether to strip whitespace from tokens
xml A logical value, whether input is XML.
verbose A logical value, whether to show progress information (counter of tokens processed).

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.

home_dir <- system.file(package = "RcppCWB", "extdata", "cwb", "indexed_corpora", "unga")

tmp_data_dir <- file.path(tempdir(), "indexed_corpora")
tmp_unga_dir <- file.path(tmp_data_dir, "unga2")
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(
  system.file(package = "RcppCWB", "extdata", "cwbb", "registry", "unga")
)
regfile[grep("^HOME", regfile)] <- sprintf("HOME \"%s\"", tmp_unga_dir)
regfile[grep("^ID", regfile)] <- "ID unga2"
writeLines(text = regfile, con = file.path(get_tmp_registry(), "unga2"))
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 = "UNGA2", p_attribute = "word", registry = get_tmp_registry())
cp_load_corpus("UNGA2", registry = get_tmp_registry())

# see whether it works
ids_sentence_1 <- cl_cpos2id(
  corpus = "UNGA2", p_attribute = "word", registry = get_tmp_registry(),
  cpos = 0:83
)
tokens_sentence_1 <- cl_id2str(
  corpus = "UNGA2", p_attribute = "word",
  registry = get_tmp_registry(), id = ids_sentence_1
)
sentence <- gsub("\s+\([^\.,]\)\", "\1", paste(tokens_sentence_1, collapse = " "))

# perform cwb_huffcode (equivalent to cwb-makeall command line utility)
cwb_huffcode(
  corpus = "UNGA2",
  p_attribute = "word",
  registry = get_tmp_registry()
)
cwb_compress_rdx(
  corpus = "UNGA2",
  p_attribute = "word",
  registry = get_tmp_registry()
)
data_dir <- file.path(tempdir(), "bt_data_dir")
dir.create(data_dir)

cwb_encode(
  corpus = "BTMIN",
  registry = Sys.getenv("CORPUS_REGISTRY"),
  vrt_dir = system.file(package = "RcppCWB", "extdata", "vrt"),
  data_dir = data_dir,
  p_attributes = c("word", "pos", "lemma"),
  s_attributes = list(
    plenary_protocol = c("lp", "protocol_no", "date", "year", "birthday", "version",
      "url", "filetype"),
    speaker = c("id", "type", "lp", "protocol_no", "date", "year", "ai_no", "ai_id"),
    ...),
  )
)
cwb_version

   "ai_type", "who", "name", "parliamentary_group", "party", "role"
 ),
p = character()
)
.unlink(data_dir)
.unlink(file.path(Sys.getenv("CORPUS_REGISTRY"), "btmin"))

cwb_version  Get CWB version

Description
Get the CWB version used and available when compiling the source code.

Usage
cwb_version()

Value
A numeric_version object.

Examples
cwb_version()

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
)
get_count_vector

Arguments

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

Examples

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

```r
y <- get_count_vector(
  corpus = "REUTERS", p_attribute = "word",
  registry = get_tmp_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 = get_tmp_registry()
)
df <- df[,c("token", "token_id", "count")]
head(df)
```

### get_pkg_registry

Get Registry Directory Within Package

#### Description

Get Registry Directory Within Package

#### Usage

```r
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

```r
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**

```r
y <- get_region_matrix(
  corpus = "REUTERS", s_attribute = "id",
  strucs = 0L:5L, registry = get_tmp_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**

```r
ids_to_count_matrix(ids)
```

**Arguments**

- ids: a vector of ids (integer values)

**Examples**

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

Create CWB subcorpus from matrix with regions.

Description

Create CWB subcorpus from matrix with regions.

Usage

matrix_to_subcorpus(region_matrix, corpus, subcorpus)

Arguments

region_matrix A two-column matrix with regions in rows: Start position of region in first column, end position in second column.
corpus A externalptr referencing a corpus such as generated by cl_find_corpus().
subcorpus A length-one character vector providing the name for the subcorpus.

Examples

## Not run:
# First we generate a subcorpus from a query result
oil_context <- cqp_query("REUTERS", subcorpus = "OIL", query = '/\{3\}"oil" [3]/\{3}\')
m <- subcorpus_get_ranges(oil_context)
reuters <- cl_find_corpus("REUTERS", registry = get_tmp_registry())
p <- matrix_to_subcorpus(subcorpus = "OIL2", corpus = reuters, region_matrix = m)
cqp_list_subcorpora("REUTERS")
x <- cqp_query("REUTERS:OIL2", query = '"crude";\', subcorpus = "CRUDEOIL")
subcorpus_get_ranges(x)

# clean up
qc_drop_subcorpus("REUTERS:OIL")
qc_drop_subcorpus("REUTERS:OIL2")
qc_drop_subcorpus("REUTERS:CRUDEOIL")

## End(Not run)

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_REGISTERY"),
    matrix
)

region_matrix_to_count_matrix(
    corpus,
    p_attribute,
    registry = Sys.getenv("CORPUS_REGISTERY"),
    matrix
)

region_matrix_context(
    corpus,
    registry = Sys.getenv("CORPUS_REGISTERY"),
    matrix,
    p_attribute,
    s_attribute,
    boundary,
    left,
    right
)

Arguments

corpus a CWB corpus
p_attribute a positional attribute
registry registry directory
matrix a regions matrix
s_attribute If not NULL, a structural attribute (length-one character vector), typically indicating a sentence ("s").
boundary Structural attribute (length-one character vector) that serves as a boundary and that shall not be transgressed.
left An integer value, number of strucs to move to the left.
right An integer value, number of strucs to move to the right.

Examples

# Scenario 1: Get full text for a subcorpus defined by regions
m <- get_region_matrix(
    corpus = "REUTERS", s_attribute = "places",
    strucs = 4L:5L, registry = get_tmp_registry()
)
ids <- region_matrix_to_ids(
subcorpus_get_ranges

Get ranges of subcorpus

Description

Get ranges of subcorpus

Usage

subcorpus_get_ranges(subcorpus_pointer)

Arguments

subcorpus_pointer

A pointer (class externalptr) referencing a CWB subcorpus.

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 (ID in upper case).
data_dir The data directory where the binary files of the corpus are stored.
s_attribute A structural attribute (length 1 character vector).
encoding Encoding of the values ("latin-1" or "utf-8")
registry The CWB registry directory.
method A length-one character vector, whether to use "R" or "Rcpp" implementation for decoding structural attribute.

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, if the s-attribute has values, or two columns, if not. 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.

Examples

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

# Using Rcpp wrappers for CWB C code
b <- s_attribute_decode(
  corpus = "REUTERS",
  data_dir = system.file(package = "RcppCWB", "extdata", "cwb", "indexed_corpora", "reuters"),
  s_attribute = "places",
  method = "Rcpp",
  registry = get_tmp_registry()
s_attr_is_descendent

)  

s_attr_is_descendent  Explore XML structure of CWB corpus

Description

The data format of the Corpus Workbench (CWB) allows nested XML as import data. Auxiliary
functions assist detecting whether two structural attributes are nested or at the same level (i.e. defining
the same regions).

Usage

s_attr_is_descendent(
  x,
  y,
  corpus,
  registry = Sys.getenv("CORPUS_REGISTRY"),
  sample = NULL
)

s_attr_is_sibling(x, y, corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

s_attr_relationship(x, y, corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

Arguments

  x            A structural attribute, stated as length-one character vector.
  y            Another structural attribute, stated as length-one character vector.
  corpus       A corpus ID (length-one character vector).
  registry     The directory with the registry file for the corpus.
  sample       An integer vector with a sample number of strucs to evaluate. Evaluating only
                a sample may be an efficient choice for large corpora. If NULL (default), all strucs
                are evaluated.

Details

s_attr_is_descendent() will evaluate whether s_attribute x is a child of s_attribute y. The return
value is TRUE (a single logical value) if all regions defined by x are within the regions defined by
y. If not, FALSE is returned. The return values is also FALSE if all regions of x and y are idential.
Attributes will be siblings in this case, and not in an ancestor-sibling relationship.

s_attr_is_sibling() will test whether the regions defined for structural attribute x and structural
attribute y are identical. If yes, TRUE is returned, assuming that both attributes are at the same level
(siblings). If not, FALSE is returned.

s_attr_relationship() will return 0 if s-attributes x and y are siblings in the sense that they
define identical regions. The return value is 0 if x is an ancestor of y and 1 if x is a descendent of
y.
s_attr_regions

Get regions defined by a structural attribute

Description

Get all regions defined by a structural attribute. Unlike get_region_matrix() that returns a region matrix for a defined subset of strucs, all regions are returned. As it is the fastest option, the function reads the binary *.rng file for the structural attribute directly. The corpus library (CL) is not used in this case.

Usage

s_attr_regions(
  corpus,
  s_attr,
  registry = Sys.getenv("CORPUS_REGISTRY"),
  data_dir = corpus_data_dir(corpus = corpus, registry = registry)
)

Arguments

corpus A length-one character vector with a corpus ID.
s_attr A length-one character vector stating a structural attribute.
registry A length-one character vector stating the registry directory (defaults to CORPUS_REGISTY environment variable).
data_dir The data directory of the corpus.

Value

A two-colum matrix with the regions defined by the structural attribute: Column 1 defines left corpus positions and column 2 right corpus positions of regions.

Examples

s_attr_regions("REUTERS", s_attr = "id", registry = get_tmp_registry())
use_tmp_registry  Use Temporary 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.
Index

* package
   RcppCWB-package, 3
   RcppCWB-package, 3

check, 5
check_corpus (check), 5
check_cpos (check), 5
check_id (check), 5
check_p_attribute (check), 5
check_pkg_registry_files, 6
check_query (check), 5
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
cl_attribute_size, 11
cl_charset_name, 12
cl_cpos2id (CL: p_attributes), 7
cloth_cpos2lb (CL: s_attributes), 9
cloth_cpos2rb (CL: s_attributes), 9
cl_cpos2str (CL: p_attributes), 7
cl_cpos2struc (CL: s_attributes), 9
cloth_delete_corpus, 13
cloth_find_corpus, 14
cloth_id2cpos (CL: p_attributes), 7
cloth_id2freq (CL: p_attributes), 7
cloth_id2str (CL: p_attributes), 7
cloth_lexicon_size, 14
cloth_list_corpora, 15
cloth_load_corpus, 15
cloth_regex2id (CL: p_attributes), 7
cloth_rework, 16
cloth_str2id (CL: p_attributes), 7
cloth_struc2cpos (CL: s_attributes), 9
cloth_struc2str (CL: s_attributes), 9
cloth_struc_values, 17
corpus_data_dir, 18
corpus_is_loaded, 18
cpos_to_id (cl_rework), 16

cpp_to_lbound (cl_rework), 16
cpp_to_rbound (cl_rework), 16
cpp_to_str (cl_rework), 16
cpp_to_struc (cl_rework), 16
cap_drop_subcorpus (cap_query), 20
cap_dump_subcorpus (cap_query), 20
cap_get_registry (cap_initialize), 19
cap_initialize, 19
cap_is_initialized (cap_initialize), 19
cap_list_corpora, 20
cap_list_subcorpora (cap_query), 20
cap_load_corpus (cap_initialize), 19

cap_query, 20
cap_reset_registry (cap_initialize), 19
cap_subcorpus_size (cap_query), 20
cap_verbosity (cap_initialize), 19
cwb_charset, 21
cwb_compress_rdx (cwb_makeall), 22
cwb_encode (cwb_makeall), 22
cwb_huffcode (cwb_makeall), 22
cwb_makeall, 22
cwb_version, 25

get_cbow_matrix, 25
get_count_vector, 26
get_pkg_registry, 27
get_region_matrix, 27
get_tmp_registry (use_tmp_registry), 35

id_to_cpos (cl_rework), 16
id_to_freq (cl_rework), 16
ids_to_count_matrix, 28

matrix_to_subcorpus, 29
p_attr (cl_rework), 16
p_attr_lexicon_size (cl_rework), 16
p_attr_size (cl_rework), 16
RcppCWB (RcppCWB-package), 3
RcppCWB-package, 3

36
INDEX

regex_to_id (cl_rework), 16
region_matrix_context
  (region_matrix_ops), 29
region_matrix_ops, 29
region_matrix_to_count_matrix
  (region_matrix_ops), 29
region_matrix_to_ids
  (region_matrix_ops), 29

s_attr (cl_rework), 16
s_attr_is_descendent, 33
s_attr_is_sibling
  (s_attr_is_descendent), 33
s_attr_regions, 34
s_attr_relationship
  (s_attr_is_descendent), 33
s_attr_size (cl_rework), 16
s_attribute_decode, 31
str_to_id (cl_rework), 16
struc_to_cpos (cl_rework), 16
struc_to_str (cl_rework), 16
subcorpus_get_ranges, 31

use_tmp_registry, 35