Package 'cwbtools'

December 17, 2019

Type Package

Title Tools to create, modify and manage 'CWB' Corpora

Version 0.1.2

Date 2019-12-17

Description
The 'Corpus Workbench' ('CWB', <http://cwb.sourceforge.net/>) offers a classic and mature approach for working with large, linguistically and structurally annotated corpora. The 'CWB' is memory efficient and its design makes running queries fast (Evert and Hardie 2011, <http://www.stefan-evert.de/PUB/EvertHardie2011.pdf>). The 'cwbtools' package offers pure R tools to create indexed corpus files as well as high-level wrappers for the original C implementation of CWB as exposed by the 'RcppCWB' package <https://CRAN.R-project.org/package=RcppCWB>. Additional functionality to add and modify annotations of corpora from within R makes working with CWB indexed corpora much more flexible and convenient. The 'cwbtools' package in combination with the R packages 'RcppCWB' (<https://CRAN.R-project.org/package=RcppCWB>) and 'polmineR' (<https://CRAN.R-project.org/package=polmineR>) offers a lightweight infrastructure to support the combination of quantitative and qualitative approaches for working with textual data.

Imports data.table, R6, xml2, stringi, curl, RcppCWB (>= 0.2.8), pbapply, methods

Suggests tm (>= 0.7.3), knitr, tokenizers (>= 0.2.1), tidytext, SnowballC, janeaustenr, devtools, polmineR, NLP

VignetteBuilder knitr

LazyData yes

License GPL-3

Language en-US

Encoding UTF-8

Collate 'cwbtools.R' 'pkg.R' 'utils.R' 'p_attribute.R' 's_attribute.R' 'registry_file.R' 'CorpusData.R' 'corpus.R' 'cwb.R' 'ner.R'

RoxygenNote 6.1.1

NeedsCompilation no

Author Andreas Blaette [aut, cre],
Christoph Leonhardt [ctb]

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

Repository CRAN

Date/Publication 2019-12-17 13:00:02 UTC

R topics documented:

`cwbtools-package` *cwbtools-package*

Description

Tools to Create, Modify and Manage CWB Corpora.

Details

The *Corpus Workbench* (CWB) offers a classic approach for working with large, linguistically and structurally annotated corpora that ensures memory efficiency and makes running queries fast (Evert and Hardie 2011). Technically, indexing and compressing corpora as suggested by Witten et al. (1999) is the approach implemented in the design of the CWB (Christ 1994).

The maturity of the CWB and the efficiency of the original C implementation notwithstanding, both the convenience and the flexibility of traditional CWB command line tools is limited. Restrictions to the portability of code across platforms inhibits the ideal of reproducible research.

The 'cwbtools' package combines portable pure R tools to create indexed corpus files and convenience wrappers for the original C implementation of CWB as exposed by the RcppCWB package. Additional functionality to add and modify annotations of corpora from within R makes working with CWB indexed corpora much more flexible. "Pure R" workflows to enrich corpora with annotations using standard NLP tools or generated manually can be implemented seamlessly and conveniently.

The *cwbtools* package is a companion of the RcppCWB and the polmineR package and is a building block of an infrastructure to support the combination of quantitative and qualitative approaches when working with textual data.

Author(s)

Andreas Blaette

References

Christ, Oliver (1994): "A Modular and Flexible Architecture for an Integrated Corpus Query System". *Proceedings of COMPLEX'94*, pp.23-32. (available online here)

Evert, Stefan and Andrew Hardie (2011): "Twenty-first century Corpus Workbench: Updating a query architecture for the new millennium." In: *Proceedings of the Corpus Linguistics 2011 conference*, University of Birmingham, UK. (available online here)

Witten, Ian H., Alistair Moffat and Timothy C. Bell (1999): *Managing Gigabytes: Compressing and Indexing Documents and Images.* 2nd edition. San Francisco et al.: Morgan Kaufmann.

Examples

```
## Not run:
library(tm)
reut21578 <- system.file("texts", "crude", package = "tm")
reuters.tm <- VCorpus(DirSource(reut21578), list(reader = readReut21578XMLasPlain))

library(tidytext)
reuters.tibble <- tidy(reuters.tm)
reuters.tibble[["topics_cat"]] <- sapply(
  reuters.tibble[["topics_cat"]],
  function(x) paste(reuters.tibble[["topics_cat"]], collapse = "|")
)
reuters.tibble[["places"]] <- sapply(
  reuters.tibble[["places"]],
  function(x) paste(x, collapse = "|")
)
reuters.tidy <- unnest_tokens(
  reuters.tibble, output = "word", input = "text", to_lower = FALSE
)

cdata <- list(
  tokenstream = as.data.table(reuters.tidy[, c("id", "word")]),
  metadata = as.data.table(reuters.tibble[,c("id", "topics_cat", "places", "language")])
  )
cdata <- add_corpus_positions(cdata)

registry_dir_tmp <- normalizePath(tempdir(), winslash = "/")
data_dir_tmp <- normalizePath(tempdir(), winslash = "/")

encode_corpusdata(
  cdata, corpus = "REUTERS", encoding = "utf8",
  registry_dir = registry_dir_tmp, data_dir = data_dir_tmp
  )

## End(Not run)
```

conll_get_regions *Extract regions from NER annotations (CoNNL format).*

Description

Extract regions from NER annotations (CoNNL format).

Usage

```
conll_get_regions(x)
```

Arguments

x	A data.frame, a data.table, or any other object that can be coerced to a data.table. The input table is expected to have the columns "token" and "ner", and "cpos".

Examples

```
x <- data.frame(
  token = c(
    "Die",
    "Bundeskanzlerin",
    "Angela",
    "Merkel",
    "hält",
    "im",
    "Bundestag",
    "eine",
    "Rede",
    "."
  ),
  ne = c("O", "O", "B-PERS", "I-PERS", "O", "O", "B-ORG", "O", "O", "O"),
  stringsAsFactors = FALSE
)
x[["cpos"]] <- 100L:(100L + nrow(x) - 1L)
tab <- conll_get_regions(x)
```

CorpusData *Manage Corpus Data and Encode CWB Corpus.*

Description

Manage Corpus Data and Encode CWB Corpus.

Usage

```
CorpusData
```

Format

An object of class `R6ClassGenerator` of length 24.

Arguments

`x` A single filename, a character vector of filenames, or a directory with XML files.

`body` An xpath expression defining the body of the xml document.

`meta` A named character vector with xpath expressions.

`mc` A numeric/integer value, number of cores to use.

`compress` Logical, whether to compress corpus.

`encoding` Encoding/charset of the CWB corpus.

`registry_dir` Corpus registry, the directory where registry files are stored.

`corpus` The name of the CWB corpus.

`p_attributes` Positional attributes.

`s_attributes` Columns that will be encoded as structural attributes.

`data_dir` Directory where to create directory for indexed corpus files.

`method` Either "R" or "CWB".

`...` Arguments that are passed into `tokenizers::tokenize_words()`.

`verbose` Logical, whether to be verbose.

`progress` Logical, whether to show progress bar.

Fiels

`chunktable` A `data.table` with column "id" (unique values), columns with metadata, and a column with text chunks.

`tokenstream` A `data.table` with a column "cpos" (corpus position), and columns with positional attributes, such as "word", "lemma", "pos", "stem".

`metadata` A `data.table` with a column "id", to link data with chunks/tokenstream, columns with document-level metadata, and a column "cpos_left" and "cpos_right", which can be generated using method `$add_corpus_positions()`.

`sentences` A `data.table`.

`named_entities` A code `data.table`

Methods

`$new()` Initialize a new instance of class `CorpusData`.

`$print()` Print summary of `CorpusData` object.

`$purge(replacements = list(c("^\s*<.*?>\s*$", ""), c("\u2019", "'")))` Remove patterns from chunkdata that are known to cause problems. This is done most efficiently at the chunkdata level of data preparation as the length of the character vector to handle is much smaller than when tokenization/annotation has been performed.

`$tokenize(verbose = TRUE)` Simple tokenization of text in chunktable.

`$add_corpus_positions(verbose = TRUE)` Add column cpos to tokenstream and columns `cpos_left` and `cpos_right` to metadata.

`$encode(corpus, p_attributes = "word", s_attributes = NULL, encoding, registry_dir = Sys.getenv("CORPUS_`
Encode corpus. If the corpus already exists, it will be removed.

`$import_xml(filenames, body = "//body", meta = NULL, mc = NULL, progress = TRUE)`

Examples

```
library(RcppCWB)
library(data.table)

# this example relies on the R method to write data to disk, there is also a method "CWB"
# that relies on CWB tools to generate the indexed corpus. The CWB can downloaded
# and installed within the package by calling cwb_install()

# create temporary registry file so that data in RcppCWB package can be used

registry_rcppcwb <- system.file(package = "RcppCWB", "extdata", "cwb", "registry")
registry_tmp <- file.path(normalizePath(tempdir(), winslash = "/"), "registry")
if (!dir.exists(registry_tmp)) dir.create(registry_tmp)
r <- registry_file_parse("REUTERS", registry_dir = registry_rcppcwb)
r[["home"]] <- system.file(package = "RcppCWB", "extdata", "cwb", "indexed_corpora", "reuters")
registry_file_write(r, corpus = "REUTERS", registry_dir = registry_tmp)

# decode structural attribute 'places'

s_attrs_places <- RcppCWB::s_attribute_decode(
  corpus = "REUTERS",
 data_dir = system.file(package = "RcppCWB", "extdata", "cwb", "indexed_corpora", "reuters"),
  s_attribute = "places", method = "R"
)
s_attrs_places[["id"]] <- 1L:nrow(s_attrs_places)
setnames(s_attrs_places, old = "value", new = "places")

# decode positional attribute 'word'

tokens <- apply(s_attrs_places, 1, function(row){
  ids <- cl_cpos2id(
    corpus = "REUTERS", cpos = row[1]:row[2],
    p_attribute = "word", registry = registry_tmp
  )
  cl_id2str(corpus = "REUTERS", id = ids, p_attribute = "word", registry = registry_tmp)
})
tokenstream <- rbindlist(
lapply(
  1L:length(tokens),
  function(i) data.table(id = i, word = tokens[[i]]))
  )
tokenstream[["cpos"]] <- 0L:(nrow(tokenstream) - 1L)

# create CorpusData object (see vignette for further explanation)
```

```
CD <- CorpusData$new()
CD$tokenstream <- as.data.table(tokenstream)
CD$metadata <- as.data.table(s_attrs_places)

# Remove temporary registry with home dir still pointing to RcppCWB data dir
# to prevent data from being deleted
file.remove(file.path(registry_tmp, "reuters"))
file.remove(registry_tmp)

# create temporary directories (registry directory and one for indexed corpora)

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

CD$encode(
  corpus = "REUTERS", encoding = "utf8",
  p_attributes = "word", s_attributes = "places",
  registry_dir = registry_tmp, data_dir = data_dir_tmp,
  method = "R"
)
reg <- registry_data(name = "REUTERS", id = "REUTERS", home = data_dir_tmp, p_attributes = "word")
registry_file_write(data = reg, corpus = "REUTERS", registry_dir = registry_tmp)

# see whether it works

cl_cpos2id(corpus = "REUTERS", p_attribute = "word", cpos = 0L:4049L, registry = registry_tmp)
```

---

corpus_install *Install and manage corpora.*

Description

Utitlity functions to keep the installation of indexed CWB corpora wrapped into R data packages simple.

Usage

```
corpus_install(pkg = NULL,
  repo = "http://polmine.sowi.uni-due.de/packages", tarball = NULL,
  lib = .libPaths()[1], verbose = TRUE, user = NULL,
  password = NULL, ...)

corpus_packages()

corpus_rename(old, new, registry_dir = Sys.getenv("CORPUS_REGISTRY"),
```

```
  verbose = TRUE)

corpus_remove(corpus, registry_dir = Sys.getenv("CORPUS_REGISTRY"))

corpus_as_tarball(corpus, registry_dir, tarfile, verbose = TRUE)

corpus_copy(corpus, registry_dir, data_dir = NULL,
  registry_dir_new = file.path(normalizePath(tempdir(), winslash = "/"),
  "cwb", "registry", fsep = "/"),
  data_dir_new = file.path(normalizePath(tempdir(), winslash = "/"),
  "cwb", "indexed_corpora", tolower(corpus), fsep = "/"),
  verbose = interactive(), progress = TRUE)

corpus_recode(corpus, registry_dir = Sys.getenv("CORPUS_REGISTRY"),
  data_dir = registry_file_parse(corpus, registry_dir)[["home"]],
  skip = character(), to = c("latin1", "UTF-8"), verbose = TRUE)
```

Arguments

pkg	Name of the data package.
repo	URL of the repository.
tarball	The URL or local path to a tarball with a CWB indexed corpus.
lib	Directory for R packages, defaults to `.libPaths()[1]`.
verbose	Logical, whether to be verbose.
user	A user name that can be specified to download a corpus from a password protected site.
password	A password that can be specified to download a corpus from a password protected site.
...	Further parameters that will be passed into `install.packages`, if argument `tarball` is `NULL`, or into or `download.file`, if `tarball` is specified.
old	Name of the (old) corpus.
new	Name of the (new) corpus.
registry_dir	Directory of registry.
corpus	A CWB corpus.
tarfile	Filename of tarball.
data_dir	The data directory where the files of the CWB corpus live.
registry_dir_new	Target directory with for (new) registry files.
data_dir_new	Target directory for corpus files.
progress	Logical, whether to show a progress bar.
skip	A character vector with s_attributes to skip.
to	Character string describing the target encoding of the corpus.

Details

A data package with a CWB corpus is assumed to include a directory `/extdata/cwb/registry` for registry files and a directory `/extdata/cwb/indexed_corpora` for the inexed corpus files. The `corpus_install` function combines two steps necessary to install a CWB corpus. First, it calls `install.packages`, then it resets the path pointing to the directory with the indexed corpus files in the registry file. The package will be installed to the standard library directory for installing R packages (`.libPaths()[1]`). Another location can be used by stating the param 'lib' explicitly (see documentation for `install.packages`). The function can also be used to install a corpus from a password protected repository. Further parameters are handed over to install.packages, so you might add `method = "wget" extra = "--user donald --password duck"`. See examples how to check whether the directory has been set correctly.

`corpus_packages` will detect the packages that include CWB corpora. Note that the directory structure of all installed packages is evaluated which may be slow on network-mounted file systems.

`corpus_rename` will rename a corpus, affecting the name of the registry file, the corpus id, and the name of the directory where data files reside.

`corpus_remove` can be used to drop a corpus.

`corpus_as_tarball` will create a tarball (.tar.gz-file) with two subdirectories. The 'registry' subdirectory will host the registry file for the tarred corpus. The data files will be put in a subdirectory with the corpus name in the 'indexed_corpora' subdirectory.

`corpus_copy` will create a copy of a corpus (useful for experimental modifications, for instance).

See Also

For managing registry files, see `registry_file_parse` for switching to a packaged corpus.

Examples

```
registry_file_new <- file.path(
  normalizePath(tempdir(), winslash = "/"),
  "cwb", "registry", "reuters", fsep = "/"
  )
if (file.exists(registry_file_new)) file.remove(registry_file_new)
corpus_copy(
  corpus = "REUTERS",
  registry_dir = system.file(package = "RcppCWB", "extdata", "cwb", "registry"),
  data_dir = system.file(
    package = "RcppCWB",
    "extdata", "cwb", "indexed_corpora", "reuters"
  )
)
unlink(file.path(
  normalizePath(tempdir(), winslash = "/"),
  "cwb", fsep = "/"),
  recursive = TRUE)
corpus <- "REUTERS"
pkg <- "RcppCWB"
s_attr <- "places"
Q <- '"oil"'
```

```
registry_dir_src <- system.file(package = pkg, "extdata", "cwb", "registry")
data_dir_src <- system.file(package = pkg, "extdata", "cwb", "indexed_corpora", tolower(corpus))

registry_dir_tmp <- file.path(
  normalizePath(tempdir(), winslash = "/"),
  "cwb", "registry", fsep = "/"
)
registry_file_tmp <- file.path(registry_dir_tmp, tolower(corpus), fsep = "/")
data_dir_tmp <- file.path(
  normalizePath(tempdir(), winslash = "/"),
  "cwb", "indexed_corpora", tolower(corpus), fsep = "/"
)

if (file.exists(registry_file_tmp)) file.remove(registry_file_tmp)
if (!dir.exists(data_dir_tmp)){
   dir.create(data_dir_tmp, recursive = TRUE)
} else {
  if (length(list.files(data_dir_tmp)) > 0L)
    file.remove(list.files(data_dir_tmp, full.names = TRUE))
}

corpus_copy(
  corpus = corpus,
  registry_dir = registry_dir_src,
  data_dir = data_dir_src,
  registry_dir_new = registry_dir_tmp,
  data_dir_new = data_dir_tmp
)

RcppCWB::cl_charset_name(corpus = corpus, registry = registry_dir_tmp)

corpus_recode(
  corpus = corpus,
  registry_dir = registry_dir_tmp,
  data_dir = data_dir_tmp,
  to = "UTF-8"
)

RcppCWB::cl_delete_corpus(corpus = corpus, registry = registry_dir_tmp)
RcppCWB::cqp_initialize(registry_dir_tmp)
RcppCWB::cl_charset_name(corpus = corpus, registry = registry_dir_tmp)

n_strucs <- RcppCWB::cl_attribute_size(
  corpus = corpus, attribute = s_attr, attribute_type = "s", registry = registry_dir_tmp
)
strucs <- 0L:(n_strucs - 1L)
struc_values <- RcppCWB::cl_struc2str(
  corpus = corpus, s_attribute = s_attr, struc = strucs, registry = registry_dir_tmp
)
speakers <- unique(struc_values)

Sys.setenv("CORPUS_REGISTRY" = registry_dir_tmp)
if (RcppCWB::cqp_is_initialized()) RcppCWB::cqp_reset_registry() else RcppCWB::cqp_initialize()
```

```
RcppCWB::cqp_query(corpus = corpus, query = Q)
cpos <- RcppCWB::cqp_dump_subcorpus(corpus = corpus)
ids <- RcppCWB::cl_cpos2id(
  corpus = corpus, p_attribute = "word", registry = registry_dir_tmp, cpos = cpos
)
str <- RcppCWB::cl_id2str(
  corpus = corpus, p_attribute = "word", registry = registry_dir_tmp, id = ids
)
unique(str)

unlink(file.path(normalizePath(tempdir(), winslash = "/"), "cwb", fsep = "/"), recursive = TRUE)
```

`cwb_install` *Utilities to install Corpus Workbench.*

Description

Some steps for encoding corpora can be performed by calling CWB utilities from the command line, which requires an installation of the CWB, either as part of the CWB package, or using the default installation location of the CWB.

Usage

```
cwb_install(url_cwb = cwb_get_url())

cwb_get_url()

cwb_get_bindir()

cwb_is_installed()
```

Arguments

`url_cwb`	The URL from where the CWB can be downloaded.

Details

`cwb_get_url` will return the URL for downloading the appropriate binary (Linux / macOS / Windows) of the Corpus Workbench.

`cwb_get_bindir` will return the directory where the cwb utility programs reside. If `cwb_install()` has been used to install the CWB, the function returns the directory within the `cwbtools` package. Alternatively, a check for a local installation is performed.

`cwb_is_installed` will check whether the CWB is installed.

get_encoding *Get Encoding of Character Vector.*

Description

Get Encoding of Character Vector.

Usage

```
get_encoding(x, verbose = FALSE)
```

Arguments

x	a character vector
verbose	logical, whether to output messages

pkg_utils *Create and manage packages with corpus data.*

Description

Putting CWB indexed corpora into R data packages is a convenient way to ship and share corpora, and to keep documentation and supplementary functionality with the data.

Usage

```
pkg_create_cwb_dirs(pkg = ".", verbose = TRUE)

pkg_add_corpus(pkg = ".", corpus,
  registry = Sys.getenv("CORPUS_REGISTRY"), verbose = TRUE)

pkg_add_configure_scripts(pkg = ".")

pkg_add_description(pkg = ".", package = NULL, version = "0.0.1",
  date = Sys.Date(), author, maintainer = NULL, description = "",
  license = "", verbose = TRUE)

pkg_add_creativecommons_license(pkg = ".", license = "CC-BY-NC-SA",
  file = system.file(package = "cwbtools", "txt", "licenses",
  "CC_BY-NC-SA_3.0.txt"))

pkg_add_gitattributes_file(pkg = ".")
```

Arguments

pkg	Path to directory of data package or package name.
verbose	A logical value, whether to be verbose.
corpus	Name of the CWB corpus to insert into the package.
registry	Registry directory.
package	The package name (character), may not include special chars, and no underscores ('_').
version	The version number of the corpus (defaults to "0.0.1")
date	The date of creation, defaults to Sys.Date().
author	The author of the package, either character vector or object of class person.
maintainer	Maintainer, R package style, either character vector or person.
description	description of the data package.
license	The license.
file	Path to file with fulltext of Creative Commons license.

Details

pkg_creage_cwb_dirs will create the standard directory structure for storing registry files and indexed corpora within a package (./inst/extdata/cwb/registry and ./inst/extdata/cwb/indexed_corpora, respectively).

pkg_add_corpus will add the corpus described in registry directory to the package defined by pkg.

add_configure_script will add standardized and tested configure scripts configure for Linux and macOS, and configure.win for Windows to the top level directory of the data package, and file setpaths.R to tools subdirectory. The configuration mechanism ensures that the data directory is specified correctly in the registry files during the installation of the data package.

pkg_add_description will add a description file to the package.

pkg_add_creativecommons_license will license information to the DESCRIPTION file, and move file LICENSE to top level directory of the package.

pkg_add_gitattributes_file will add a file '.gitattributes' to the package. The file defines types of files that will be tracked by Git LFS, i.e. they will not be under conventional version control. This is suitable for large binary files, which is the scenario applicable for indexed corpus data.

References

Blätte, Andreas (2018). "Using Data Packages to Ship Annotated Corpora of Parliamentary Protocols: The GermaParl R Package", *ParlaCLARIN 2018 Workshop Proceedings*, available online here.

See Also

The use_description function in the usethis-package will also create a DESCRIPTION file.

Examples

```
pkgdir <- normalizePath(tempdir(), winslash = "/")
pkg_create_cwb_dirs(pkg = pkgdir)
pkg_add_description(
  pkg = pkgdir,
  package = "reuters",
  author = "cwbtools",
  description = "Reuters data package"
 )
pkg_add_corpus(
  pkg = pkgdir, corpus = "REUTERS",
  registry = system.file(package = "RcppCWB", "extdata", "cwb", "registry")
)
pkg_add_gitattributes_file(pkg = pkgdir)
pkg_add_configure_scripts(pkg = pkgdir)
pkg_add_creativecommons_license(pkg = pkgdir)
```

---

p_attribute_encode *Encode Positional Attribute(s).*

Description

Pure R implementation to generate positional attribute from a character vector of tokens (the token stream).

Usage

```
p_attribute_encode(token_stream, p_attribute = "word", registry_dir,
  corpus, data_dir, method = c("R", "CWB"), verbose = TRUE,
  encoding = get_encoding(token_stream), compress = NULL)

p_attribute_recode(data_dir, p_attribute, from = c("UTF-8", "latin1"),
  to = c("UTF-8", "latin1"))
```

Arguments

token_stream	A character vector with the tokens of the corpus.
p_attribute	The positional attribute.
registry_dir	Registry directory (needed by p_attribute_huffcode and p_attribute_compress_rdx).
corpus	The CWB corpus (needed by p_attribute_huffcode and p_attribute_compress_rdx).
data_dir	The data directory for the corpus with the binary files.
method	Either 'CWB' or 'R'.
verbose	Logical.
encoding	Encoding as defined in the charset corpus property of the registry file for the corpus ('latin1' to 'latin9', and 'utf8').
compress	Logical.
from	Character string describing the current encoding of the attribute.
to	Character string describing the target encoding of the attribute.

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 may be necessary.

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").

`p_attribute_recode` will recode the values in the avs-file and change the attribute value index in the avx file. The rng-file remains unchanged. The registry file remains unchanged, and it is highly recommended to consider `s_attribute_recode` as a helper for `corpus_recode` that will recode all s-attributes, all p-attributes, and will reset the encoding in the registry file.

Examples

```
library(RcppCWB)

# In this example, we pursue a "pure R" approach. To rely on the "CWB"
# method, you can use the cwb_install() function, which will download and
# install the CWB command line # tools within the package.

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

# create new (and empty) directory structure

tmpdir <- normalizePath(tempdir(), winslash = "/")
if (.Platform$OS.type == "windows") tmpdir <- normalizePath(tmpdir, winslash = "/")
registry_tmp <- file.path(tmpdir, "registry", fsep = "/")
data_dir_tmp <- file.path(tmpdir, "data_dir", fsep = "/")
if (file.exists(file.path(data_dir_tmp, "word.corpus"))){
  file.remove(file.path(data_dir_tmp, "word.corpus"))
}
if (dir.exists(registry_tmp)) unlink(registry_tmp, recursive = TRUE)
if (dir.exists(data_dir_tmp)) unlink(data_dir_tmp, recursive = TRUE)
dir.create (registry_tmp)
dir.create(data_dir_tmp)

p_attribute_encode(
  corpus = "reuters",
  token_stream = tokens, p_attribute = "word",
  data_dir = data_dir_tmp, method = "R",
  registry_dir = registry_tmp,
  compress = FALSE,
  encoding = "utf8"
  )
```

```
regdata <- registry_data(
  id = "REUTERS", name = "Reuters Sample Corpus", home = data_dir_tmp,
  properties = c(encoding = "utf-8", language = "en"), p_attributes = "word"
)
regfile <- registry_file_write(
  data = regdata, corpus = "REUTERS",
  registry_dir = registry_tmp, data_dir = data_dir_tmp,
)
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]
```

`registry_file_parse` *Parse and create registry files.*

Description

A set of functions to parse, create and write registry files.

Usage

```
registry_file_parse(corpus, registry_dir = Sys.getenv("CORPUS_REGISTRY"))

registry_file_compose(x)

registry_data(name, id, home, info = file.path(home, ".info", fsep =
  "/"), properties = c(charset = "utf-8"), p_attributes,
  s_attributes = character())

registry_file_write(data, corpus,
  registry_dir = Sys.getenv("CORPUS_REGISTRY"), ...)
```

Arguments

`corpus`	A CWB corpus indicated by a length-one `character` vector.
`registry_dir`	Directory with registry files.
`x`	An object of class `registry_data`.
`name`	Long descriptive name of corpus (`character` vector).

id	Short name of corpus (character vector).
home	Path with data directory for indexed corpus.
info	A character vector containing path name of info file.
properties	Named character vector with corpus properties, should at least include 'charset'.
p_attributes	A character vector with positional attributes to declare.
s_attributes	A character vector with structural attributes to declare.
data	A registry_data object.
...	further parameters

Details

registry_file_parse will return an object of class registry_data.

See the appendix to the 'Corpus Encoding Tutorial' (http://cwb.sourceforge.net/files/CWB_Encoding_Tutorial.pdf), which includes an explanation of the registry file format.

registry_file_compose will turn an registry_data-object into a character vector with a registry file that can be written to disk.

registry_file_write will compose a registry file from data and write it to disk.

Examples

```
regdata <- registry_file_parse(
  corpus = "REUTERS",
  registry_dir = system.file(package = "RcppCWB", "extdata", "cwb", "registry")
  )
```

---

s_attribute_encode *Read, process and write data on structural attributes.*

Description

Read, process and write data on structural attributes.

Usage

```
s_attribute_encode(values, data_dir, s_attribute, corpus, region_matrix,
  method = c("R", "CWB"), registry_dir = Sys.getenv("CORPUS_REGISTRY"),
  encoding, delete = FALSE, verbose = TRUE)

s_attribute_recode(data_dir, s_attribute, from = c("UTF-8", "latin1"),
  to = c("UTF-8", "latin1"))

s_attribute_files(s_attribute, data_dir)

s_attribute_get_values(s_attribute, data_dir)
```

```
s_attribute_get_regions(s_attribute, data_dir)

s_attribute_merge(x, y)

s_attribute_delete(corpus, s_attribute)
```

Arguments

values	A character vector with the values of the structural attribute.
data_dir	The data directory where to write the files.
s_attribute	Atomic character vector, the name of the structural attribute.
corpus	A CWB corpus.
region_matrix	A two-column matrix with corpus positions.
method	EWither 'R' or 'CWB'.
registry_dir	Path name of the registry directory.
encoding	Encoding of the data.
delete	Logical, whether a call to RcppCWB::cl_delete_corpus is performed.
verbose	Logical.
from	Character string describing the current encoding of the attribute.
to	Character string describing the target encoding of the attribute.
x	Data defining a first s-attribute, a data.table (or an object coercible to a data.table) with three columns ("cpos_left", "cpos_right", "value").
y	Data defining a second s-attribute, a data.table (or an object coercible to a data.table)with three columns ("cpos_left", "cpos_right", "value").

Details

In addition to using CWB functionality, the s_attribute_encode function includes a pure R implementation to add or modify structural attributes of an existing CWB corpus.

If the corpus has been loaded/used before, a new s-attribute may not be available unless RcppCWB::cl_delete_corpus has been called. Use the argument delete for calling this function.

s_attribute_recode will recode the values in the avs-file and change the attribute value index in the avx file. The rng-file remains unchanged. The registry file remains unchanged, and it is highly recommended to consider s_attribute_recode as a helper for corpus_recode that will recode all s-attributes, all p-attributes, and will reset the encoding in the registry file.

s_attribute_files will return a named character vector with the data files (extensions: "avs", "avx", "rng") in the directory indicated by data_dir for the structural attribute s_attribute.

s_attribute_get_values is equivalent to performing the CL function cl_struc2id for all strucs of a structural attribute. It is a "pure R" operation that is faster than using CL, as it processes entire files for the s-attribute directly. The return value is a character vector with all string values for the s-attribute.

s_attribute_get_regions will return a two-column integer matrix with regions for the strucs of a given s-attribute. Left corpus positions are in the first column, right corpus positions in the

second column. The result is equivalent to calling RcppCWB::get_region_matrix for all strucs of a s-attribute, but may be somewhat faster. It is a "pure R" function which is fast as it processes files entirely and directly.

`s_attribute_merge` combines two tables with regions for s-attributes checking for intersections that may cause problems. The heuristic is to keep all non-intersecting annotations and those annotations that define the same region in object `x` and object `y`. Annotations of `x` and `y` which overlap uncleanly, i.e. without an identity of the left and the right corpus position ("cpos_left" / "cpos_right") are dropped. The scenario for using the function is to decode a s-attribute (using `s_attribute_decode`), mix in an additional annotation, and to re-encode the enhanced s-attribute (using `s_attribute_encode`).

Function `s_attribute_delete` is not yet implemented.

See Also

To decode a structural attribute, see `s_attribute_decode`.

Examples

```
require("RcppCWB")
registry_tmp <- file.path(normalizePath(tempdir(), winslash = "/"), "cwb", "registry", fsep = "/")
data_dir_tmp <- file.path(
  normalizePath(tempdir(), winslash = "/"),
  "cwb", "indexed_corpora", "reuters", fsep = "/"
)

corpus_copy(
  corpus = "REUTERS",
  registry_dir = system.file(package = "RcppCWB", "extdata", "cwb", "registry"),
 data_dir = system.file(package = "RcppCWB", "extdata", "cwb", "indexed_corpora", "reuters"),
  registry_dir_new = registry_tmp,
  data_dir_new = data_dir_tmp
)

no_strucs <- cl_attribute_size(
  corpus = "REUTERS",
  attribute = "id", attribute_type = "s",
  registry = registry_tmp
)
cpos_list <- lapply(
  0L:(no_strucs - 1L),
  function(i)
   cl_struc2cpos(corpus = "REUTERS", struc = i, s_attribute = "id", registry = registry_tmp)
)
cpos_matrix <- do.call(rbind, cpos_list)

s_attribute_encode(
  values = as.character(1L:nrow(cpos_matrix)),
  data_dir = data_dir_tmp,
  s_attribute = "foo",
  corpus = "REUTERS",
  region_matrix = cpos_matrix,
```

```
  method = "R",
  registry_dir = registry_tmp,
  encoding = "latin1",
  verbose = TRUE,
  delete = TRUE
)

cl_struc2str(
 "REUTERS", struc = 0L:(nrow(cpos_matrix) - 1L), s_attribute = "foo", registry = registry_tmp
)

unlink(registry_tmp, recursive = TRUE)
unlink(data_dir_tmp, recursive = TRUE)
avs <- s_attribute_get_values(
  s_attribute = "id",
 data_dir = system.file(package = "RcppCWB", "extdata", "cwb", "indexed_corpora", "reuters")
)
rng <- s_attribute_get_regions(
  s_attribute = "id",
 data_dir = system.file(package = "RcppCWB", "extdata", "cwb", "indexed_corpora", "reuters")
)
x <- data.frame(
  cpos_left =  c(1L, 5L, 10L, 20L, 25L),
  cpos_right = c(2L, 5L, 12L, 21L, 27L),
  value = c("ORG", "LOC", "ORG", "PERS", "ORG"),
  stringsAsFactors = FALSE
)
y <- data.frame(
  cpos_left =  c(5, 11, 20, 25L, 30L),
  cpos_right = c(5, 12, 22, 27L, 33L),
  value = c("LOC", "ORG", "ORG", "ORG", "ORG"),
  stringsAsFactors = FALSE
)
s_attribute_merge(x,y)
```

Index