Introduction to the rcqp package

Bernard Desgraupes and Sylvain Loiseau
<bernard.desgraupes@u-paris10.fr>, <sylvain.loiseau@univ-paris13.fr>

October 18, 2012

Abstract

The `rcqp` R library is a wrapper on the CWB software. The CWB software, used in the field of corpus linguistics, lets index and query large annotated corpora. The `rcqp` library includes the CWB code and allows using R to execute CWB functions and import their output into statistical analyses.

Contents

1 Introduction

The CWB (Corpus Workbench) software[1] is a set of tools for corpus linguistics, providing a powerful indexation and query engine for annotated corpora. An interactive command line program called CQP (*corpus query processor*) is provided, as well as a client/server architecture.

CWB offers access to corpora through manipulation of vectors of positions (offsets of the adressed / requested word(s) in the corpus) which can be turned into the corresponding word forms, lemmas, or parts of speech if the corpus provides these pieces of information.

In **rcqp**, CWB is turned into an R library. As a result:

- the CWB data structures are mainly vectors, which are very convenient to manipulate in R;
- R vectors are wrappers on the inner CWB C arrays, thus providing efficient access;
- `rcqp` provides an easy way to run and query CWB, without having to separately compile and install the CWB software;
- `rcqp` lets you take advantage of the R statistical capacities for analyzing the complex CWB data.

In **rcqp**, two different ways of calling CWB are provided:

- You can call CWB through the **cqi_*** set of functions. These functions implement an interface defined by CWB (CQi).
- You can use a set of functions trying to help producing quantitative structures (frequency lists, cross-tabulated frequency tables) for statistical analyses of CWB corpora with R.

1.1 CWB data-model and CQP syntax

Beside token attributes, called positionnal attribute, CWB corpora may have spans of tokens corresponding to various unit: phrases, clauses, sentences, paragraph, chapter, book, . . . Each of these groups of span corresponding to an unit are called structural attribute.

A corpus may be represented as an array where each line represents a token and each column represents an **attribute**. Here are the first 20 lines of such an array, representing the DICKENS demo corpus, with 38 columns.

```
Using registry '/home/sloiseau/corpus/CWB/registry'.
```

[1] http://cwb.sourceforge.net/

```
   file file_name novel novel_title titlepage book book_num chapter chapter_num
0     0         0     0           0         0   -1       -1      -1          -1
1     0         0     0           0         0   -1       -1      -1          -1
2     0         0     0           0         0   -1       -1      -1          -1
3     0         0     0           0         0   -1       -1      -1          -1
4     0         0     0           0         0   -1       -1      -1          -1
5     0         0     0           0         0   -1       -1      -1          -1
6     0         0     0           0         0   -1       -1      -1          -1
7     0         0     0           0         0   -1       -1      -1          -1
8     0         0     0           0         0   -1       -1      -1          -1
9     0         0     0           0         0   -1       -1      -1          -1
10    0         0     0           0         0   -1       -1      -1          -1
11    0         0     0           0         0   -1       -1      -1          -1
12    0         0     0           0         0   -1       -1      -1          -1
13    0         0     0           0         0   -1       -1      -1          -1
14    0         0     0           0         0   -1       -1      -1          -1
15    0         0     0           0         0   -1       -1      -1          -1
16    0         0     0           0         0   -1       -1      -1          -1
17    0         0     0           0         0   -1       -1      -1          -1
18    0         0     0           0         0   -1       -1      -1          -1
19    0         0     0           0         0   -1       -1      -1          -1
20    0         0     0           0         0   -1       -1      -1          -1
   chapter_title title title_len p p_len s s_len np np1 np2 np_h np_h1 np_h2
0             -1     0         0 0     0 0     0  0  -1  -1    0    -1    -1
1             -1     0         0 0     0 0     0  0  -1  -1    0    -1    -1
2             -1     0         0 0     0 0     0 -1  -1  -1   -1    -1    -1
3             -1     0         0 1     1 1     1 -1  -1  -1   -1    -1    -1
4             -1     0         0 1     1 1     1  1  -1  -1    1    -1    -1
5             -1     0         0 1     1 1     1  1  -1  -1    1    -1    -1
6             -1    -1        -1 2     2 2     2  2  -1  -1    2    -1    -1
7             -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
8             -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
9             -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
10            -1    -1        -1 2     2 2     2  3  -1  -1    3    -1    -1
11            -1    -1        -1 2     2 2     2  3  -1  -1    3    -1    -1
12            -1    -1        -1 2     2 2     2  3  -1  -1    3    -1    -1
13            -1    -1        -1 2     2 2     2  3  -1  -1    3    -1    -1
14            -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
15            -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
16            -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
17            -1    -1        -1 2     2 2     2  4  -1  -1    4    -1    -1
18            -1    -1        -1 2     2 2     2  4  -1  -1    4    -1    -1
19            -1    -1        -1 2     2 2     2  4  -1  -1    4    -1    -1
20            -1    -1        -1 2     2 2     2  4   0  -1    4     0    -1
   np_len np_len1 np_len2 pp pp1 pp2 pp_h pp_h1 pp_h2 pp_len pp_len1 pp_len2
0       0      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
```

```
1       0      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
2      -1      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
3      -1      -1      -1  0  -1  -1    0    -1    -1      0      -1      -1
4       1      -1      -1  0  -1  -1    0    -1    -1      0      -1      -1
5       1      -1      -1  0  -1  -1    0    -1    -1      0      -1      -1
6       2      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
7      -1      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
8      -1      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
9      -1      -1      -1  1   0  -1    1     0    -1      1       0      -1
10      3      -1      -1  1   0  -1    1     0    -1      1       0      -1
11      3      -1      -1  1   0  -1    1     0    -1      1       0      -1
12      3      -1      -1  1   0  -1    1     0    -1      1       0      -1
13      3      -1      -1  1   0  -1    1     0    -1      1       0      -1
14     -1      -1      -1  1  -1  -1    1    -1    -1      1      -1      -1
15     -1      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
16     -1      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
17      4      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
18      4      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
19      4      -1      -1  2  -1  -1    2    -1    -1      2      -1      -1
20      4       0      -1  2  -1  -1    2    -1    -1      2      -1      -1
          word pos     lemma               nbc
0            A  DT         a A Christmas Carol
1    CHRISTMAS  NP Christmas A Christmas Carol
2        CAROL  NN     carol A Christmas Carol
3           by  IN        by A Christmas Carol
4      Charles  NP   Charles A Christmas Carol
5      Dickens  NP   Dickens A Christmas Carol
6            I  PP         I A Christmas Carol
7         have VBP      have A Christmas Carol
8  endeavoured VBN endeavour A Christmas Carol
9           in  IN        in A Christmas Carol
10        this  DT      this A Christmas Carol
11     Ghostly  JJ   ghostly A Christmas Carol
12      little  JJ    little A Christmas Carol
13        book  NN      book A Christmas Carol
14           ,   ,         , A Christmas Carol
15          to  TO        to A Christmas Carol
16       raise  VB     raise A Christmas Carol
17         the  DT       the A Christmas Carol
18       Ghost  NN     ghost A Christmas Carol
19          of  IN        of A Christmas Carol
20          an  DT        an A Christmas Carol
```

The first 34 columns represent **structural attributes**: this kind of attribute defines spans of tokens (like XML tags surrounding tokens), called **regions**. A region is made of the tokens sharing a same value for this attribute. A region is always made of consecutive tokens. Thus, while the id is the same in a column, the corresponding tokens belong to the same region. The id identifying a region is called a **struc**. Since regions are defined thanks to a struc value on tokens, there is no recursivity. Tokens between two regions, with respect to a given structural attribute, have a value of -1.

Next there are several columns containing strings. They are the **positional attributes**, giving for each word information such as lemma, word-form, (part of speech),... Each positional attribute has a list of **ids**, which are unique numerical codes for the different possible string forms.

Moreover, certain structural attributes have a string value associated with each *struc* (region). While each struc is unique to a region, string values can be repeated over several regions. For instance, the np_h structural attribute, giving the head of the noun phrase, holds a string value.

In short, in CQi function names, the following types of data are used:

cpos a position, or rank, identifying a unique token in the corpus;

id an id for a form (type) in the lexicon of a positional attribute lexicon;

str the string corresponding to an id in the lexicon of a positional attribute lexicon;

struc the id of a region in a given structural attribute.

A subcorpus is created thanks to the `cqi_query` function. See *CQP Query Language Tutorial*, Stefan Evert & The OCWB Development Team, 17 February 2010, for a complete specification of the CQP query language.

A subcorpus is a collection of sequences of tokens matched by a query and identified by their *cpos*. Since a query may match a sequence of tokens, a subcorpus is a collection of *(match, matchend)* pairs, where **match** is the cpos of the first token and **matchend** the cpos of the last token in the sequence matched. When only one token is addressed by a query, matchend is identical to match.

The **match** and **matchend** positions (together with two other optional pieces of information named **target** and **keyword**) are referred to as the **anchors** (or sometime **fields**: see cqi_fdist1 et cqi_fdist2) available on each hit.

All indices are 0-based: the cpos of the first token is 0, the first id for a positional attribute or the first struc for a structural attribute is 0, etc.

2 The CQi set of functions

2.1 A sample session

```
> sort(cqi_list_corpora())[1:6]

[1] "CFR_FR"            "CFR_RU"            "CHRONIQUES_LATINES"
[4] "CORPUS_ES"         "DEFINITION"        "DESCARTES_CORRESP"
```

```
> # create the subcorpus "Interesting" (it creates the subcorpus internaliy
> # with the given name but does not return any result).
> cqi_query("DICKENS", "Interesting", '"interest.*"');
> # in the CQi API, the qualified name of subcorpus is corpus:subcorpus:
> nbr_hit <- cqi_subcorpus_size("DICKENS:Interesting");
> nbr_hit

[1] 888

> # The subcorpus as a matrix: one line by hit,
> # four columns: match, matchend, target, keyword.
> dump <- cqi_dump_subcorpus("DICKENS:Interesting",0,10)
> dump

       [,1]  [,2] [,3] [,4]
 [1,] 15921 15921   -1   -1
 [2,] 17747 17747   -1   -1
 [3,] 20189 20189   -1   -1
 [4,] 24026 24026   -1   -1
 [5,] 35161 35161   -1   -1
 [6,] 35490 35490   -1   -1
 [7,] 35903 35903   -1   -1
 [8,] 43031 43031   -1   -1
 [9,] 58109 58109   -1   -1
[10,] 63109 63109   -1   -1
[11,] 79532 79532   -1   -1

> # get the lemma of the "match" slot of each hit:
> # Word's attributes (such as "lemma", "word", "pos") are always accessed
> # through qualified name : "corpus.attribute"
> lemma <- cqi_cpos2str("DICKENS.lemma", dump[,1])
> lemma

 [1] "interesting" "interest"    "interest"    "interest"    "interest"
 [6] "interest"    "interest"    "interested"  "interest"    "interest"
[11] "interest"

> # You can acheave the same result in one more steps, using id as an
> # intermediate step:
> ids <- cqi_cpos2id("DICKENS.lemma", dump[,1]);
> lemma <- cqi_id2str("DICKENS.lemma", ids);
> lemma

 [1] "interesting" "interest"    "interest"    "interest"    "interest"
 [6] "interest"    "interest"    "interested"  "interest"    "interest"
[11] "interest"
```

```
> # cqi_fdist1 create a frequency list according to one field (match,
> # matchend...) in a query; cqi_fdist2 a cross tabulated
> # frequency table according to two fields in a query
>
> flist <- cqi_fdist1("DICKENS:Interesting", "match", "word")
> flist

      [,1] [,2]
[1,]  3221  566
[2,]  2892  160
[3,]  5300  125
[4,] 12056   30
[5,] 55879    3
[6,] 43452    2
[7,] 39795    1
[8,] 37414    1

> # cqi_fdist1 et cqi_fdist2 return numeric matrix : (lemma) id -> freq.
> # use id2str in order to turn the (word) id into its form.
> data.frame(cqi_id2str("DICKENS.word", flist[,1]), flist[,2])

  cqi_id2str..DICKENS.word...flist...1.. flist...2.
1                               interest        566
2                            interesting        160
3                             interested        125
4                              interests         30
5                             interestin          3
6                          interestingly          2
7                             interest--          1
8                           interest--or          1
```

2.2 Functions

For more information about the actual use of these functions, see their respective help pages.

All functions are prefixed with **cqi_**.

cqi_list_corpora List all the corpora available in the registry.

```
> corpora <- cqi_list_corpora()
> corpora[1:5]

[1] "ICHTYA_FR"          "ICHTYA_LAT"         "LITTRE_DEFINITION"
[4] "TOUTMONTESQUIEU"    "DICKENS"
```

cqi_full_name Return the full name of a corpus.

cqi_corpus_info Return various informations about a corpus.

cqi_query Create a subcorpus. A subcorpus is a list of hits. Each hit contains four fields : **match** (the cpos of the first token of the matched sequence), **matchend** (the cpos of the last token of the matched sequence, identical with match if the sequence is one token long), and two optionnal values (see CQP documentation), **target** and **keyword**.

```
> corpora <- cqi_list_corpora()
> cqi_query("DICKENS", "Subcorpus", '"interesting"');
```

The `cqi_query` does not return any value; it creates the subcorpus as an object internally. Use `cqi_dump_subcorpus` for retrieving the subcorpus contents. The subcorpus name must begin with a capital letter.

cqi_list_subcorpora List the created subcorpora.

cqi_drop_subcorpus Delete a subcorpus.

cqi_dump_subcorpus Retrieve the subcorpus created by a call to the `cqi_query` function as a four-column matrix: one row by hit, and one column for each of the four fields (see `cqi_query`).

```
> cqi_query("DICKENS", "Subcorpus", '"interesting"');
> x <- cqi_dump_subcorpus("DICKENS:Subcorpus");
> x[1:10,];
```

```
        [,1]   [,2] [,3] [,4]
 [1,]  15921  15921   -1   -1
 [2,] 131848 131848   -1   -1
 [3,] 176031 176031   -1   -1
 [4,] 248048 248048   -1   -1
 [5,] 248883 248883   -1   -1
 [6,] 270757 270757   -1   -1
 [7,] 470828 470828   -1   -1
 [8,] 514381 514381   -1   -1
 [9,] 514394 514394   -1   -1
[10,] 519640 519640   -1   -1
```

cqi_subcorpus_size Return the number of hits in a subcorpus. This is the same as the number of rows returned by `cqi_dump_subcorpus`.

cqi_attributes Get the list of attributes (positional, structural, or aligned) in a corpus.

```
> positional_attributes <- cqi_attributes("DICKENS", "p");
> positional_attributes
```

```
[1] "word"  "pos"   "lemma" "nbc"
```

```
> structural_attributes <- cqi_attributes("DICKENS", "s");
> structural_attributes
```

```
 [1] "file"          "file_name"     "novel"         "novel_title"
 [5] "titlepage"     "book"          "book_num"      "chapter"
 [9] "chapter_num"   "chapter_title" "title"         "title_len"
[13] "p"             "p_len"         "s"             "s_len"
[17] "np"            "np1"           "np2"           "np_h"
[21] "np_h1"         "np_h2"         "np_len"        "np_len1"
[25] "np_len2"       "pp"            "pp1"           "pp2"
[29] "pp_h"          "pp_h1"         "pp_h2"         "pp_len"
[33] "pp_len1"       "pp_len2"
```

cqi_lexicon_size Number of forms in a positional attribute. Attributes are denoted using their *qualified name*, of the form `corpus:attribute`.

```
> lexicon_size <- cqi_lexicon_size("DICKENS:word");
> lexicon_size
```

```
NULL
```

The greatest id of an attribute is lexicon_size −1.

cqi_structural_attribute_has_values Ask if a structural attribute has a string value associated with its region. For retrieving the actual string value associated with a region id (a struc), see `cqi_struc2str`.

```
> has_values <- cqi_structural_attribute_has_values("DICKENS.np_h");
> has_values
```

```
[1] TRUE
```

cqi_attribute_size Return the number of actual elements (number of occurrences).

- on a positional attribute, it gives the number of tokens.
- on a structural attribute, it gives the number of regions.
- on an alignment attribute, it gives the number of aligned pairs.

cqi_cpos2id Convert from a token cpos to the corresponding id in a given positional attribute.

```
> id <- cqi_cpos2id("DICKENS.word", 0:20);
> id
```

```
 [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

cqi_str2id Get the id corresponding to the specified string in the lexicon of a given positional attribute.

```
> id <- cqi_str2id("DICKENS.word", "interesting");
> id
```

```
[1] 2892
```

cqi_id2cpos Return all the tokens (cpos) corresponding to the specified id of a certain positional attribute.

```
> id <- cqi_str2id("DICKENS.word", "interesting");
> cpos <- cqi_id2cpos("DICKENS.word", id);
> cpos[1:10]
```

```
 [1]  15921 131848 176031 248048 248883 270757 470828 514381 514394 519640
```

```
> length(cpos);
```

```
[1] 160
```

cqi_id2freq Return the number of tokens corresponding to the specified id of a certain positional attribute.

```
> id <- cqi_str2id("DICKENS.word", "interesting");
> freq <- cqi_id2freq("DICKENS.word", id);
> freq
```

```
[1] 160
```

cqi_id2str Return the string corresponding to the specified id of a certain positional attribute.

```
> id <- cqi_str2id("DICKENS.word", "interesting");
> str <- cqi_id2str("DICKENS.word", id);
> str
```

```
[1] "interesting"
```

cqi_cpos2str Return the string of a given positional attribute corresponding to a given id. This is identical to using `cqi_cpos2id` then `cqi_id2str`.

```
> str <- cqi_cpos2str("DICKENS.word", 1:10);
> str
```

```
 [1] "CHRISTMAS"   "CAROL"       "by"          "Charles"     "Dickens"
 [6] "I"           "have"        "endeavoured" "in"          "this"
```

cqi_regex2id Get the id corresponding to the string of a positional attribute matched by a given regex.

```
> id <- cqi_regex2id("DICKENS.word", '"Interest.*"');
> id
```

```
integer(0)
```

cqi_cpos2struc Get the region id (the struc, of a given structural attribute) to which a given token belongs. Below, we are in the sentence with struc 53, then 54.

```
> struc <- cqi_cpos2struc("DICKENS.s", 1010:1020);
> struc
```

```
 [1] 53 53 53 53 53 53 54 54 54 54 54
```

If the token is outside any region in the given structural attribute, -1 is returned.

```
> # In this sequence, tokens are not in nominal phrases.
> cqi_cpos2struc("DICKENS.np", 1000:1010)
```

```
 [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

cqi_struc2cpos Get the first and last cpos (tokens) belonging to a struc (a region id) of a given structural attribute. The second argument is a vector of length 1, the returned value a vector of length 2.

```
> cpos <- cqi_struc2cpos("DICKENS.np_h", 10);
> cpos
```

```
[1] 50 51
```

cqi_struc2str Get the string mapped to a region id (a struc) of a given structural attribute; available only for structural attributes having values.

```
> str <- cqi_struc2str("DICKENS.np_h", 10);
> str
```

```
[1] "house"
```

cqi_cpos2lbound Given a token, return the left-most token belonging to the same region in the given structural attribute.

This is implemented as a simple shortcut for functions `cqi_cpos2struc` and `cqi_struc2cpos[1]`.

```
> str <- cqi_cpos2lbound("DICKENS.np_h", 10);
> str
```

```
[1] 10
```

cqi_cpos2rbound Given a token, return the right-most token belonging to the same region in the given structural attribute.

This is implemented as a simple shortcut for functions `cqi_cpos2struc` and `cqi_struc2cpos(...)[2]`.

```
> str <- cqi_cpos2rbound("DICKENS.np_h", 10);
> str
```

```
[1] 13
```

cqi_alg2cpos Convert from an id denoting a region of an alignment attribute to cpos of tokens contained into this region in the aligned corpora.

Suppose that two parallel corpora **VIE_FR** and **VIE_RU** have been encoded using **tu_id** as the attribute containing aligned chunks of text. The alignment attribute is named **vie_fr** in the corpus **VIE_RU** and **vie_ru** in the corpus **VIE_FR**.

For region 5 of the **tu_id** attribute in **VIE_RU**, the corresponding cpos in corpus **VIE_RU** are obtained with:

```
> cpos <- cqi_alg2cpos("VIE_RU.vie_fr", 5)
> cpos
[1]  89 132 110 166
> str <- cqi_cpos2str("VIE_FR.word", cpos)
[1] "comme"  "dont"   "Jeanne" "."
```

cqi_cpos2alg Convert from a token in corpus A to the corresponding region of an alignment attribute in an aligned corpus B.

cqi_fdist1 Get a frequency list of the strings of a given positional attribute in a subcorpus.

In the following example, get all part-of-speech tags :

```
> cqi_query("DICKENS", "Noun", '[pos="N.*"]')
> fdist <- cqi_fdist1("DICKENS:Noun", "match", "pos")
> cqi_id2str("DICKENS.pos", fdist[,1])
```

```
[1] "NN"  "NP"  "NNS" "NPS"
```

```
> fdist[,2]

[1] 396069 131638  89577    448
```

cqi_fdist2 Get a cross-tabulated table of the string values of a given positional attribute in a subcorpus against the string values of another positional attribute.

3 Integrating CQP into R S3 objects and quantitative structures

A set of high-level functions is aimed at making easier the use of cqp with R and more self-explanatory the data model of CWB. It provides in particular functions for easily creating quantitative data structures.

3.1 A sample session

```
> # create a corpus
> c <- corpus("DICKENS")
> # summary give a quick view of the information available in the corpus,
> # it does not display actual information:
> summary(c)

DICKENS
Number or tokens in the corpus: 3407085
Positional attributes (4)
positional : DICKENS.lemma (41222 types; 3407085 tokens)
                "a", "Christmas", "carol", "by", "Charles", "Dickens", "I", "have", "endeav
positional : DICKENS.nbc (726 types; 3407085 tokens)
                "A Christmas Carol", "A Christmas Carol, Ch. 1", ...
positional : DICKENS.pos (43 types; 3407085 tokens)
                "DT", "NP", "NN", "IN", "PP", "VBP", "VBN", "JJ", ",", "TO", ...
positional : DICKENS.word (57568 types; 3407085 tokens)
                "A", "CHRISTMAS", "CAROL", "by", "Charles", "Dickens", "I", "have", "endeav
Structural attributes (34)
structural : DICKENS.book (17 regions)
structural : DICKENS.book_num (7 types; 17 regions)
                "1", "2", "3", "1", "2", "3", "4".
structural : DICKENS.chapter (696 regions)
structural : DICKENS.chapter_num (73 types; 696 regions)
                "1", "2", "3", "4", "5", "1", "2", "3", "4", "5", ...
structural : DICKENS.chapter_title (559 types; 696 regions)
                "Marley's Ghost", "The First of the Three Spirits", ...
structural : DICKENS.file (14 regions)
structural : DICKENS.file_name (14 types; 14 regions)
                "Source/Dickens:ChristmasCarol.txt.gz", "Source/Dickens:DavidCopperfield.tx
```

```
structural : DICKENS.novel (14 regions)
structural : DICKENS.novel_title (14 types; 14 regions)
                 "A Christmas Carol", "David Copperfield", "Dombey and Son", ...
structural : DICKENS.np (419363 regions)
structural : DICKENS.np1 (90915 regions)
structural : DICKENS.np2 (25640 regions)
structural : DICKENS.np_h (10713 types; 419363 regions)
                 "CHRISTMAS", "Dickens", "I", "book", "ghost", "other", "season", "me", "May
structural : DICKENS.np_h1 (7248 types; 90915 regions)
                 "idea", "burial", "clerk", "undertaker", "mourner", "myself", "ironmongery"
structural : DICKENS.np_h2 (4298 types; 25640 regions)
                 "reader", "humour", "trade", "nose", "cheek", "such", "place", "path", "sym
structural : DICKENS.np_len (39 types; 419363 regions)
                 "2", "2", "1", "4", "18", "2", "2", "1", "1", "1", ...
structural : DICKENS.np_len1 (33 types; 90915 regions)
                 "15", "2", "2", "2", "3", "1", "4", "2", "1", "4", ...
structural : DICKENS.np_len2 (27 types; 25640 regions)
                 "2", "3", "2", "3", "2", "1", "6", "5", "3", "2", ...
structural : DICKENS.p (61177 regions)
structural : DICKENS.p_len (523 types; 61177 regions)
                 "3", "3", "56", "12", "6", "62", "9", "94", "218", "63", ...
structural : DICKENS.pp (116608 regions)
structural : DICKENS.pp1 (38889 regions)
structural : DICKENS.pp2 (9000 regions)
structural : DICKENS.pp_h (91 types; 116608 regions)
                 "by", "in", "of", "with", "about", "of", "by", "of", "about", "of", ...
structural : DICKENS.pp_h1 (76 types; 38889 regions)
                 "in", "of", "with", "with", "with", "of", "in", "on", "on", "on", ...
structural : DICKENS.pp_h2 (64 types; 9000 regions)
                 "with", "of", "on", "with", "in", "of", "in", "of", "in", "from", ...
structural : DICKENS.pp_len (35 types; 116608 regions)
                 "3", "6", "16", "11", "2", "3", "3", "6", "3", "5", ...
structural : DICKENS.pp_len1 (30 types; 38889 regions)
                 "5", "4", "3", "3", "2", "4", "3", "7", "4", "8", ...
structural : DICKENS.pp_len2 (26 types; 9000 regions)
                 "2", "3", "3", "3", "8", "3", "7", "5", "2", "3", ...
structural : DICKENS.s (152455 regions)
structural : DICKENS.s_len (224 types; 152455 regions)
                 "3", "3", "41", "15", "12", "6", "8", "8", "22", "4", ...
structural : DICKENS.title (733 regions)
structural : DICKENS.title_len (48 types; 733 regions)
                 "6", "6", "9", "9", "8", "7", "5", "5", "4", "6", ...
structural : DICKENS.titlepage (14 regions)
Alignement attributes (0)
```

```
> #
> # printing the corpus (by default, first tokens only)
> c

   file file_name novel novel_title titlepage book book_num chapter chapter_num
0     0         0     0           0         0   -1       -1      -1          -1
1     0         0     0           0         0   -1       -1      -1          -1
2     0         0     0           0         0   -1       -1      -1          -1
3     0         0     0           0         0   -1       -1      -1          -1
4     0         0     0           0         0   -1       -1      -1          -1
5     0         0     0           0         0   -1       -1      -1          -1
6     0         0     0           0         0   -1       -1      -1          -1
7     0         0     0           0         0   -1       -1      -1          -1
8     0         0     0           0         0   -1       -1      -1          -1
9     0         0     0           0         0   -1       -1      -1          -1
10    0         0     0           0         0   -1       -1      -1          -1
11    0         0     0           0         0   -1       -1      -1          -1
12    0         0     0           0         0   -1       -1      -1          -1
13    0         0     0           0         0   -1       -1      -1          -1
14    0         0     0           0         0   -1       -1      -1          -1
15    0         0     0           0         0   -1       -1      -1          -1
16    0         0     0           0         0   -1       -1      -1          -1
17    0         0     0           0         0   -1       -1      -1          -1
18    0         0     0           0         0   -1       -1      -1          -1
19    0         0     0           0         0   -1       -1      -1          -1
20    0         0     0           0         0   -1       -1      -1          -1
   chapter_title title title_len p p_len s s_len np np1 np2 np_h np_h1 np_h2
0             -1     0         0 0     0 0     0  0  -1  -1    0    -1    -1
1             -1     0         0 0     0 0     0  0  -1  -1    0    -1    -1
2             -1     0         0 0     0 0     0 -1  -1  -1   -1    -1    -1
3             -1     0         0 1     1 1     1 -1  -1  -1   -1    -1    -1
4             -1     0         0 1     1 1     1  1  -1  -1    1    -1    -1
5             -1     0         0 1     1 1     1  1  -1  -1    1    -1    -1
6             -1    -1        -1 2     2 2     2  2  -1  -1    2    -1    -1
7             -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
8             -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
9             -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
10            -1    -1        -1 2     2 2     2  3  -1  -1    3    -1    -1
11            -1    -1        -1 2     2 2     2  3  -1  -1    3    -1    -1
12            -1    -1        -1 2     2 2     2  3  -1  -1    3    -1    -1
13            -1    -1        -1 2     2 2     2  3  -1  -1    3    -1    -1
14            -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
15            -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
16            -1    -1        -1 2     2 2     2 -1  -1  -1   -1    -1    -1
17            -1    -1        -1 2     2 2     2  4  -1  -1    4    -1    -1
18            -1    -1        -1 2     2 2     2  4  -1  -1    4    -1    -1
```

```
19            -1     -1         -1 2       2 2      2  4  -1  -1     4     -1     -1
20            -1     -1         -1 2       2 2      2  4   0  -1     4      0     -1
   np_len np_len1 np_len2 pp pp1 pp2 pp_h pp_h1 pp_h2 pp_len pp_len1 pp_len2
0       0      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
1       0      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
2      -1      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
3      -1      -1      -1  0  -1  -1    0    -1    -1      0      -1      -1
4       1      -1      -1  0  -1  -1    0    -1    -1      0      -1      -1
5       1      -1      -1  0  -1  -1    0    -1    -1      0      -1      -1
6       2      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
7      -1      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
8      -1      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
9      -1      -1      -1  1   0  -1    1     0    -1      1       0      -1
10      3      -1      -1  1   0  -1    1     0    -1      1       0      -1
11      3      -1      -1  1   0  -1    1     0    -1      1       0      -1
12      3      -1      -1  1   0  -1    1     0    -1      1       0      -1
13      3      -1      -1  1   0  -1    1     0    -1      1       0      -1
14     -1      -1      -1  1  -1  -1    1    -1    -1      1      -1      -1
15     -1      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
16     -1      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
17      4      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
18      4      -1      -1 -1  -1  -1   -1    -1    -1     -1      -1      -1
19      4      -1      -1  2  -1  -1    2    -1    -1      2      -1      -1
20      4       0      -1  2  -1  -1    2    -1    -1      2      -1      -1
          word pos     lemma                nbc
0            A  DT         a A Christmas Carol
1    CHRISTMAS  NP Christmas A Christmas Carol
2        CAROL  NN     carol A Christmas Carol
3           by  IN        by A Christmas Carol
4      Charles  NP   Charles A Christmas Carol
5      Dickens  NP   Dickens A Christmas Carol
6            I  PP         I A Christmas Carol
7         have VBP      have A Christmas Carol
8  endeavoured VBN endeavour A Christmas Carol
9           in  IN        in A Christmas Carol
10        this  DT      this A Christmas Carol
11     Ghostly  JJ   ghostly A Christmas Carol
12      little  JJ    little A Christmas Carol
13        book  NN      book A Christmas Carol
14           ,   ,         , A Christmas Carol
15          to  TO        to A Christmas Carol
16       raise  VB     raise A Christmas Carol
17         the  DT       the A Christmas Carol
18       Ghost  NN     ghost A Christmas Carol
19          of  IN        of A Christmas Carol
20          an  DT        an A Christmas Carol
```

3.2 Functions

3.2.1 Creating a corpus

The first step is creating a corpus object. A `corpus` object is created with the function `corpus()`. This object may be used with the two functions below, as well as for creating `subcorpus`, `cqp_flist` and `cqp_ftable` objects (see below).

print Print all information (but the value of structural attributes having a value) as a dataframe.

summary Give the number of tokens of a corpus, list all the attributes (positional, structural, alignment) ; for each positional attribute (and structural attribute having a value) give the number of types and print some type samples.

write Write into a file with an argument *filename* and optional arguments *from* and *to* denoting token cpos.

region_sizes.cqp_corpus Create a variable containing the size (in tokens) of a given structural attribute.

```
> c <- corpus("DICKENS");
> sentences <- region_sizes(c$s);
> hist(sentences);
```

3.2.2 Accessing attribute

Attribute can be accessed very easily by the "$" and "[[" operator.. The former suppose to type the exact name, the latter allows for using a variable. Here are three identical way of accessing the `word` attribute.

```
> c <- corpus("DICKENS");
> x <- c$word
> x <- c[["word"]]
> attr <- "word"
> x <- c[[ attr ]]
```

Once created, several functions are available.

Positional attribute have the functions `ntype`, `types`, `ntoken` and `tokens`.

```
> c <- corpus("DICKENS");
> a <- c$pos
> ntoken(a)
```

```
[1] 3407085
```

```
> tokens(a)[1:5]
```

```
[1] "DT" "NP" "NN" "DT" "DT"

> ntype(a)

[1] 43

> types(a)

 [1] "DT"   "NP"   "NN"   "IN"   "PP"   "VBP"  "VBN"  "JJ"   ","    "TO"
[11] "VB"   "WDT"  "MD"   "RB"   "PP$"  "NNS"  "RP"   "CC"   "SENT" "CD"
[21] ":"    "POS"  "VBD"  "EX"   "VBZ"  "''"   "WP"   "JJS"  "WRB"  "VBG"
[31] "RBR"  "PDT"  "UH"   "JJR"  "``"   "WP$"  "("    ")"    "RBS"  "NPS"
[41] "FW"   "LS"   "SYM"

> w <- c$word
> ntoken(w)

[1] 3407085

> tokens(w)[1:5]

[1] "A"         "CHRISTMAS" "CAROL"     "by"        "Charles"

> ntype(w)

[1] 57568

> types(w)[1:10]

 [1] "A"           "CHRISTMAS"   "CAROL"       "by"          "Charles"
 [6] "Dickens"     "I"           "have"        "endeavoured" "in"
```

Structural attribute have the functions `nregions`, i.e. the number of regions, `tokens` : the region id (`struc`) of each token of the corpus. Moreover, for structural attribute with value, the function `regions` allows for retreiving the value of each region.

```
> c <- corpus("DICKENS");
> s <- c$s
> nregion(s)

[1] 152455

> np <- c$np_h
> nregion(np)

[1] 419363

> regions(np)[1:10]
```

```
 [1] "CHRISTMAS" "Dickens"   "I"         "book"      "ghost"     "other"
 [7] "season"    "me"        "May"       "it"
```

A function `summary` print information about an attribute:

```
> c <- corpus("DICKENS");
> summary(c$lemma)

positional : DICKENS.lemma (41222 types; 3407085 tokens)
                 "a", "Christmas", "carol", "by", "Charles", "Dickens", "I", "have", "endea

> summary(c$s)

structural : DICKENS.s (152455 regions)

> summary(c$np_h);

structural : DICKENS.np_h (10713 types; 419363 regions)
                 "CHRISTMAS", "Dickens", "I", "book", "ghost", "other", "season", "me", "Ma
```

3.2.3 Creating a subcorpus

An `subcorpus` object is created with the function `subcorpus()`. In the CWB terminology, a subcorpus is the set of sequences matched by a query.

```
> c <- corpus("DICKENS");
> sc <- subcorpus(c, '"interesting" "to" @ []');
> # sc
> #
> # if you want to change the lines printed (0-based);
> # use from/to options:
> print(sc, from=2, to=5);

    270757  ? ' It can hardly be << interesting to you >> , ' said I. ' Yes ,
    639982 he parent of a son is  << interesting to me >> . ' Has Mrs Blimber
    835921  sister as if it were << interesting to him >> to see them together
   1012817 ive security . It was  << interesting to be >> in the quiet old tow

> #
> # if you want more access on the kwic presentation
> # (sorting, printing), you can construct a cqp_kwic object:
> k <- cqp_kwic(sc, right.context=10, left.context=10)
> print(k, from=5, to=10)

   1012817 ty . It was    << interesting to be >> in the qui
   1197341 's not very   << interesting to you >> , and I am
   1903972 se was made   << interesting to the >> public , b
   2521810 proving and  << interesting to hear >> two politi
   3014814 t is always << interesting to trace >> a resembla
   3040285 es , highly     << interesting to a >> bystander
```

```
> k <- sort(k, sort.anchor="target", sort.offset=0, sort.attribute="word")
> print(k, from=5, to=10)

    835921  if it were   << interesting to him >> to see the
    248883 It was very    << interesting to me >> to see the
    639982 of a son is    << interesting to me >> . ' Has Mr
   1903972 se was made   << interesting to the >> public , b
   3014814 t is always << interesting to trace >> a resembla
    270757 n hardly be   << interesting to you >> , ' said I
```

print Print a KWIC (*keyword in context*) form.

summary Get a quick summary of the size and content of the subcorpus.

3.2.4 Creating a frequency list

A frequency list may be created either with a corpus or with a subcorpus.

```
> c <- corpus("DICKENS");
> fl <- cqp_flist(c$lemma);
> summary(fl);

A frequency list
  Number of tokens: 3407085
  Number of types: 41222
  Corpus: DICKENS
  Attribute: lemma

> #
> # get only the 1% most frequent forms
> fl <- cqp_flist(c$lemma, cutoff=0.01);
> summary(fl);

A frequency list
  Number of tokens: 2662681
  Number of types: 412
  Corpus: DICKENS
  Attribute: lemma

> fl[1:30]

     ,    the      .    and     be      '     of     to      a   have      I
282600 142776 114392 100637  94181  74246  74054  72343  63468  63306  51848
    in   that     it    his     he    you   with      ;    say    not     as
 47556  37913  35867  35374  35015  31127  27889  26591  26437  24702  23821
   her     at    for      !     do     on     my      ?
 21531  19770  19763  18134  18023  17441  16932  16032
```

```
> #
> # get only the forms with freq > 100
> fl <- cqp_flist(c$lemma, cutoff=100);
> summary(fl);

A frequency list
  Number of tokens: 3112708
  Number of types: 2245
  Corpus: DICKENS
  Attribute: lemma

> fl[1:30]

         a  Christmas         by    Charles          I       have  endeavour
     63468        168      12594        258      51848      63306        239
        in       this     little       book          ,         to      raise
     47556      12613       6724        649     282600      72343        787
       the      ghost         of         an       idea      which      shall
    142776        253      74054       7878        625      11921       1968
       not        put         my     reader        out     humour       with
     24702       2624      16932        135       7583        207      27889
themselves       each
       726        928
```

With a subcorpus, a lot of options are available in order to construct the frequency list with a particular anchor, an offset for address tokens before or after this anchor, and left and right contexts in order to include tokens in a span.

```
> c <- corpus("DICKENS");
> sc <- subcorpus(c, '"interesting" "to" @ []');
> #
> # Create a cqp_flist with the target anchor
> fl <- cqp_flist(sc, "target", "word");
> summary(fl);

A frequency list
  Number of tokens: 12
  Number of types: 10
  Subcorpus: Uvpydcjgri
  Parent corpus: DICKENS
  anchor: target
  left.context: 0
  right.context: 0
  attribute: word
  offset: 0
```

```
> #
> # Same anchor, but count parts of speech
> fl <- cqp_flist(sc, "target", "pos");
> fl;

 type frequency
   NP         6
   DT         3
   DT         3

> #
> # You can extend the span around the anchor with `left.context' and
> # `right.context'
> fl <- cqp_flist(sc, "match", "pos", left.context=5, right.context=5);
> fl;

 type frequency
   DT         9
   NN        11
   IN         8
   PP        18
  VBN         2
   JJ        17
    ,         8
   TO        14
   VB         7
   MD         1
   RB         9
  NNS         1
   CC         3
 SENT         5
   CD         1
    :         1
  VBD         7
  VBZ         5
   ''         4
  VBG         1

> #
> # or with a match..matchend span :
> fl <- cqp_flist(sc, c("match", "matchend"), "pos");
> fl;

 type frequency
   DT         3
   PP         6
   JJ        12
   TO        12
   VB         3

> #
> # The two can be used together:
> fl <- cqp_flist(sc, c("match", "matchend"), "pos", left.context=5, right.context=5);
> fl;

 type frequency
   DT        11
   NP         4
   NN        13
   IN         9
   PP        20
  VBP         1
  VBN         2
   JJ        19
    ,        10
   TO        14
   VB         7
  WDT         1
   MD         2
   RB        11
  NNS         3
   CC         3
 SENT         5
   CD         1
    :         2
  POS         1
  VBD         7
  VBZ         5
   ''         4
  VBG         1
```

summary.cqp_flist Print information about the frequency list.

3.2.5 Creating a frequency table

The `cqp_ftable` function creates a frequency table: a cross-tabulated frequency count according to two attributes. `cqp_ftable` may be applied either on a corpus, or a subcorpus. It produces a dataframe.

cqp_ftable with corpus `cqp_ftable` lets create frequency tables using a corpus object. The cross-tabulated fields may be positional or structural attributes.

```
> library(reshape);
> c <- corpus("DICKENS");
> f <- cqp_ftable(c, "novel_title", "pos");
> f[1:10,]

         novel_title    pp_h freq
1  A Christmas Carol       '  615
2  A Christmas Carol       , 2759
3  A Christmas Carol       :  514
4  A Christmas Carol       `  259
5  A Christmas Carol       (   17
6  A Christmas Carol       )   17
7  A Christmas Carol      CC 1333
8  A Christmas Carol      CD  189
9  A Christmas Carol      DT 2885
10 A Christmas Carol      EX   91

> #
> # create a contingency table
> t <- cast(f, novel_title ~ pos, value="freq", fun.aggregate=sum )
> #
> # Visual inspection of frequency of various POS in the different novels
> mosaicplot(as.matrix(t));
```

Positional attributes (and structural attributes having values) are represented by their string values rather than by ids. For positional attributes, it is only a matter of presentation, since each id has its own string; but for structural attributes having values, it may entail a different counting: occurrences of phenomena belonging to different strucs but with same value are then counted together. You can force the use of ids rather than string values with the `attribute1.use.id` and `attribute2.use.id` options.

Counts are made on token basis, i.e. each corpus token is an individual on which the two modalities (attributes) are considered. If you use two structural attributes as arguments in `cqp_ftable`, and one of them does not have values, then the third column counts the number of tokens in the smallest region. In the following example, each line gives the length (in number of tokens, third column) of each sentence (second column) in each novel, represented by its title:

```
> f <- cqp_ftable(c, "novel_title", "s")
> f[1:10,]

         novel_title s freq
1  A Christmas Carol 0    3
2  A Christmas Carol 1    3
3  A Christmas Carol 2   41
4  A Christmas Carol 3   15
5  A Christmas Carol 4   12
6  A Christmas Carol 5    6
7  A Christmas Carol 6    8
```

```
8  A Christmas Carol 7    8
9  A Christmas Carol 8   22
10 A Christmas Carol 9    4
```

If both structural attributes have values, you may want to count the number of times the modalities are cooccurring, rather than the total number of tokens included in these cooccurrences. For that purpose, you can use the `structural.attribute.unique.id=TRUE` option. In the following example, we count the number of times each head appears in each novel :

```
> f <- cqp_ftable(c, "novel_title", "pp_h", structural.attribute.unique.id=TRUE)
> f[1:10,]

         novel_title    pp_h freq
1  A Christmas Carol            1
2  A Christmas Carol   about   21
3  A Christmas Carol   above    2
4  A Christmas Carol  across    3
5  A Christmas Carol   after   12
6  A Christmas Carol against    6
7  A Christmas Carol   along    2
8  A Christmas Carol amongst    7
9  A Christmas Carol      as   15
10 A Christmas Carol      at   83
```

Here on the contrary, we count the total number of tokens in each prepositional phrase having a given head :

```
> f <- cqp_ftable(c, "novel_title", "pp_h")
> f[1:10,]

         novel_title    pp_h  freq
1  A Christmas Carol         29265
2  A Christmas Carol   about    83
3  A Christmas Carol   above    12
4  A Christmas Carol  across     9
5  A Christmas Carol   after    58
6  A Christmas Carol against    18
7  A Christmas Carol   along    20
8  A Christmas Carol amongst    24
9  A Christmas Carol      as    42
10 A Christmas Carol      at   287
```

cqp_ftable with subcorpus Applied on a subcorpus, the `cqp_ftable` function is mainly a wrapper on the `cqi_fdist2` function. However, it returns a three columns dataframe with <strings>, <string>, <freq> rather than a three columns matrix with <ids>, <ids>, <freq> like `cqi_fdist2`.

```
> c <- corpus("DICKENS");
> sc <- subcorpus(c, '"from" @ [] "to" []')
> f <- cqp_ftable(sc, "target", "word", "matchend", "word");
> f[1:10,]

   target.word matchend.word freq
1         time          time   87
2         head          foot   70
3          day           day   42
4         side          side   31
5      morning         night   14
6          one       another   13
7          one           the   13
8        place         place   10
9        mouth         mouth    7
10        hour          hour    7
```

Index