Package ‘RKorAPClient’

June 8, 2020

Type Package
Title 'KorAP' Web Service Client Package
Version 0.5.9
Description A client package that makes the 'KorAP' web service API accessible from R.
   The corpus analysis platform 'KorAP' has been developed as a scientific tool to make
   potentially large, stratified and multiply annotated corpora, such as the 'German Reference Cor-
   pus DeReKo'
   or the 'Corpus of the Contemporary Romanian Language CoRoLa', accessible for lin-
   guists to let them verify
   hypotheses and to find interesting patterns in real language use.
   The 'RKorAPClient' package provides access to 'KorAP' and the corpora behind it for user-
   created R code,
   as a programmatic alternative to the 'KorAP' web user-interface.
   You can learn more about 'KorAP' and use it directly on 'DeReKo' at <https://korap.ids-
   manheim.de/>.
Depends R (>= 3.5.0)
Language en-US
License BSD_2_clause + file LICENSE
Encoding UTF-8
LazyData false
RoxygenNote 7.1.0
Imports R.cache, broom, ggplot2, tibble, magrittr, tidyr, dplyr,
   lubridate, highcharter, jsonlite, keyring, plotly, htmlwidgets,
   utils, httr, methods, PTXQC
Suggests testthat
Collate 'KorAPConnection.R' 'KorAPCorpusStats.R' 'RKorAPClient.R'
   'KorAPQuery.R' 'ci.R' 'highcharter-helper.R' 'misc.R'
   'reexports.R'
NeedsCompilation no
Author Marc Kupietz [aut, cre],
   Nils Diewald [ctb],
   Leibniz Institute for the German Language [cph, fnd]
**ci** ................................. Add confidence interval and relative frequency variables

### Description

Using `prop.test`, `ci` adds three columns to a data frame: 1. relative frequency (`f`) 2. lower bound of a confidence interval (`ci.low`) 3. upper bound of a confidence interval

#### Usage

```r
ci(df, x = totalResults, N = total, conf.level = 0.95)
```

#### Arguments

- `df` table with columns for absolute and total frequencies.
- `x` column with the observed absolute frequency.
- `N` column with the total frequencies
- `conf.level` confidence level of the returned confidence interval. Must be a single number between 0 and 1.

### See Also

- `ci` is already included in `frequencyQuery`
Examples

```r
library(ggplot2)
kco <- new("KorAPConnection", verbose=TRUE)
expand_grid(year=2015:2018, alternatives=c("Hate Speech", "Hatespeech")) %>%
bind_cols(corpusQuery(kco, .$alternatives, sprintf("pubDate in %d", .$year))) %>%
mutate(total=corpusStats(kco, vc=vc)$tokens) %>%
ci() %>%
ggplot(aes(x=year, y=f, fill=query, color=query, ymin=conf.low, ymax=conf.high)) +
  geom_point() + geom_line() + geom_ribbon(alpha=.3)
```

---

### Method `corpusQuery`

**Description**

Perform a corpus query via a connection to a KorAP-API-server.

**Usage**

```r
## S4 method for signature 'KorAPConnection'
corpusQuery(
  kco,
  query = if (missing(KorAPUrl))
    stop("At least one of the parameters query and KorAPUrl must be specified.", call. = FALSE) else http::parse_url(KorAPUrl)$query$q,
  vc = if (missing(KorAPUrl)) "" else http::parse_url(KorAPUrl)$query$cq,
  KorAPUrl,
  metadataOnly = TRUE,
  ql = if (missing(KorAPUrl)) "poliqlarp" else http::parse_url(KorAPUrl)$query$ql,
  fields = c("corpusSigle", "textSigle", "pubDate", "pubPlace", "availability", "textClass", "snippet"),
  accessRewriteFatal = TRUE,
  verbose = kco@verbose,
  expand = length(vc) != length(query),
  as.df = FALSE
)
```

**Arguments**

- **kco**
  - `KorAPConnection` object (obtained e.g. from `new("KorAPConnection")`)
- **query**
  - string that contains the corpus query. The query language depends on the `ql` parameter. Either `query` must be provided or `KorAPUrl`.
corpusQuery.KorAPConnection-method

vc  
string describing the virtual corpus in which the query should be performed. An empty string (default) means the whole corpus, as far as it is license-wise accessible.

KorAPUrl  
instead of providing the query and vc string parameters, you can also simply copy a KorAP query URL from your browser and use it here (and in KorAPConnection) to provide all necessary information for the query.

metadataOnly  
logical that determines whether queries should return only metadata without any snippets. This can also be useful to prevent access rewrites. Note that the default value is TRUE, unless the connection is authorized (currently not possible).

ql  
string to choose the query language (see section on Query Parameters in the Kustvakt-Wiki for possible values.

fields  
(met)adata fields that will be fetched for every match.

accessRewriteFatal  
abort if query or given vc had to be rewritten due to insufficient rights (not yet implemented).

verbose  
print some info

expand  
logical that decides if query and vc parameters are expanded to all of their combinations

as.df  
return result as data frame instead of as S4 object?

Value

Depending on the as.df parameter, a table or a KorAPQuery object that, among other information, contains the total number of results in @totalResults. The resulting object can be used to fetch all query results (with fetchAll) or the next page of results (with fetchNext). A corresponding URL to be used within a web browser is contained in @webUIRequestUrl. Please make sure to check $collection$rewrites to see if any unforeseen access rewrites of the query’s virtual corpus had to be performed.

References

https://ids-pub.bsz-bw.de/frontdoor/index/index/docId/9026

See Also

KorAPConnection, fetchNext, fetchRest, fetchAll, corpusStats

Examples

# Fetch metadata of every query hit for "Ameisenplage" and show a summary
new("KorAPConnection") %>% corpusQuery("Ameisenplage") %>% fetchAll()

# Use the copy of a KorAP-web-frontend URL for an API query of "Ameise" in a virtual corpus
# and show the number of query hits (but don’t fetch them).
new("KorAPConnection", verbose = TRUE) %>%
corpusQuery(KorAPUrl =
  "https://korap.ids-mannheim.de/?q=Ameise&cq=pubDate+since+2017&ql=poliqarp")

# Plot the time/frequency curve of "Ameisenplage"

new("KorAPConnection", verbose=TRUE) %>%
  (. -> kco ) %>%
corpusQuery("Ameisenplage") %>%
  fetchAll() %>%
  slot("collectedMatches") %>%
  mutate(year = lubridate::year(pubDate)) %>%
  dplyr::select(year) %>%
  group_by(year) %>%
  summarise(Count = dplyr::n()) %>%
  mutate(Freq = mapply(function(f, y) f / corpusStats(kco, paste("pubDate in", y))@tokens, Count, year)) %>%
  dplyr::select(-Count) %>%
  complete(year = min(year):max(year), fill = list(Freq = 0)) %>%
  plot(type = "l")

---

corpusStats,KorAPConnection-method

*Fetch information about a (virtual) corpus*

**Description**

Fetch information about a (virtual) corpus

**Usage**

```r
## S4 method for signature 'KorAPConnection'
corpusStats(kco, vc = "", verbose = kco@verbose, as.df = FALSE)
```

**Arguments**

- **kco**: *KorAPConnection* object (obtained e.g. from `new("KorAPConnection")`)
- **vc**: string describing the virtual corpus. An empty string (default) means the whole corpus, as far as it is license-wise accessible.
- **verbose**: logical. If TRUE, additional diagnostics are printed.
- **as.df**: return result as data frame instead of as S4 object?

**Value**

*KorAPCorpusStats* object with the slots documents, tokens, sentences, paragraphs
Examples

```r
corpusStats(new("KorAPConnection"))

c <- new("KorAPConnection")
corpusStats(c, "pubDate in 2017 & articleType=/Zeitung.*)/
```

---

**geom_freq_by_year_ci**  
Experimental: Plot frequency by year graphs with confidence intervals

**Description**

Experimental convenience function for plotting typical frequency by year graphs with confidence intervals using ggplot2. **Warning:** This function may be moved to a new package.

**Usage**

```r geom_freq_by_year_ci(mapping = aes(ymin = conf.low, ymax = conf.high), ...)
```

**Arguments**

- `mapping`  
  Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

- `...`  
  Other arguments passed to geom_ribbon, geom_line, and geom_click_point.

**Examples**

```r
library(ggplot2)
c <- new("KorAPConnection", verbose=TRUE)

expand_grid(condition = c("textDomain = /Wirtschaft.*/", "textDomain != /Wirtschaft.*/"),
year = (2005:2011)) %>%
cbind(frequencyQuery(c, "[tt/l=Heuschrecke]",
paste0(.$condition," & pubDate in ", .$year))) %>%
ipm() %>%
ggplot(aes(year, ipm, fill = condition, color = condition)) +
geom_freq_by_year_ci()
```
**ggplotly**

**Experimental: Convert ggplot2 to plotly with hyperlinks to KorAP queries**

**Description**

RKorAPClient::ggplotly converts a ggplot2::ggplot() object to a plotly object with hyperlinks from data points to corresponding KorAP queries. **Warning:** This function may be moved to a new package.

**Usage**

```
ggplotly(p = ggplot2::last_plot(), tooltip = c("x", "y", "colour", "url"), ...)  
```

**Arguments**

- `p` a ggplot object.
- `tooltip` a character vector specifying which aesthetic mappings to show in the tooltip. If you want hyperlinks to KorAP queries you need to include "url" here.
- `...` Other arguments passed to plotly::ggplotly

**Examples**

```r
library(ggplot2)
kco <- new("KorAPConnection", verbose=TRUE)
year <- (2003:2011)
condition <- c("textDomain = /Wirtschaft.*/", "textDomain != /Wirtschaft.*/")
g <- expand_grid(condition, year) %>%
cbind(frequencyQuery(kco, "[tt/l=Heuschrecke]",
  paste0(.\$.condition," & pubDate in ", .\$.year))) %>%
  ipm() %>%
ggplot(aes(year, ipm, fill = condition, color = condition)) +
  # theme_light(base_size = 20) +
  geom_freq_by_year_ci()
p <- ggplotly(g)
print(p)
## saveWidget(p, paste0(tmpdir(), "heuschrecke.html")
```
Experimental convenience function for plotting typical frequency by year graphs with confidence intervals using highcharter. **Warning:** This function may be moved to a new package.

**Usage**

```r
hc_freq_by_year_ci(
  df,
  as.alternatives = FALSE,
  ylabel = if (as.alternatives) "%" else "ipm",
  smooth = FALSE,
  ...
)
```

**Arguments**

- `df` data frame like the value of a `frequencyQuery`
- `as.alternatives` boolean decides whether queries should be treated as mutually exclusive and exhaustive wrt. to some meaningful class (e.g. spelling variants of a certain word form).
- `ylabel` defaults to `%` if `as.alternatives` is true and to "ipm" otherwise.
- `smooth` boolean decides whether the graph is smoothed using the highcharts plot types spline and areasplinerange.
- `...` additional arguments passed to `hc_add_series`

**Examples**

```r
ty <- c(1990:2018)
alternatives <- c("macht \{0,3\} Sinn", "ergibt \{0,3\} Sinn")
new("KorAPConnection", verbose = TRUE) %>% frequencyQuery(query = alternatives,
  vc = paste("textType = /Zeit.*/ & pubDate in", year),
  as.alternatives = TRUE) %>%
  hc_freq_by_year_ci(as.alternatives = TRUE)
```

```r
kco <- new("KorAPConnection", verbose = TRUE)
expand_grid(
  condition = c("textDomain = /Wirtschaft.*/", "textDomain != /Wirtschaft.*/"),
  year = (2005:2011)
) %>%
```
ipm

Convert corpus frequency table to instances per million.

Usage

ipm(df)

Arguments

df table returned from frequencyQuery

Details

Given a table with columns f, conf.low, and conf.high, ipm adds a column ipm and multiplies conf.low and conf.high with $10^6$.

Value

original table with additional column ipm and converted columns conf.low and conf.high

Examples

new("KorAPConnection") %>% frequencyQuery("Test", paste0("pubDate in ", 2000:2002)) %>% ipm()
KorAPConnection objects represent the connection to a KorAP server. New KorAPConnection objects can be created by `new("KorAPConnection")`.

Usage

```r
## S4 method for signature 'KorAPConnection'
initialize(
  .Object,
  KorAPUrl = "https://korap.ids-mannheim.de/",
  apiVersion = "v1.0",
  apiUrl,
  accessToken = getAccessToken(KorAPUrl),
  userAgent = "R-KorAP-Client",
  timeout = 110,
  verbose = FALSE,
  cache = TRUE
)

## S4 method for signature 'KorAPConnection'
persistAccessToken(kco, accessToken = kco@accessToken)

## S4 method for signature 'KorAPConnection'
clearAccessToken(kco)

## S4 method for signature 'KorAPConnection'
apiCall(kco, url, json = TRUE, getHeaders = FALSE, cache = kco@cache)

## S4 method for signature 'KorAPConnection'
clearCache(kco)

## S4 method for signature 'KorAPConnection'
show(object)
```

Arguments

- `.Object`: KorAPConnection object
- `KorAPUrl`: the URL of the KorAP server instance you want to access.
- `apiVersion`: which version of KorAP's API you want to connect to.
- `apiUrl`: URL of the KorAP web service.
- `accessToken`: OAuth2 access token. To use authorization based on an access token in subsequent queries, initialize your KorAP connection with `kco <- new("KorAPConnection", accessToken="<access`
In order to make the API token persistent for the currently used KorAPUrl (you can have one token per KorAPUrl / KorAP server instance), use persistAccessToken(kco). This will store it in your keyring using the keyring package. Subsequent new("KorAPConnection") calls will then automatically retrieve the token from your keyring. To stop using a persisted token, call clearAccessToken(kco). Please note that for DeReKo, authorized queries will behave differently inside and outside the IDS, because of the special license situation. This concerns also cached results which do not take into account from where a request was issued. If you experience problems or unexpected results, please try kco <- new("KorAPConnection", cache=FALSE) or use clearCache to clear the cache completely.

userAgent  user agent string.
timeout time out in seconds.
verbose logical. Decides whether following operations will default to be verbose.
cache logical. Decides if API calls are cached locally. You can clear the cache with clearCache().
kco KorAPConnection object
url request url
json logical that determines if json result is expected
getHeaders logical that determines if headers and content should be returned (as a list)
oobject KorAPConnection object

Value

KorAPConnection object that can be used e.g. with corpusQuery

Examples

```r
kcon <- new("KorAPConnection", verbose = TRUE)
kq <- corpusQuery(kcon, "Ameisenplage")
kq <- fetchAll(kq)
```

```r
## Not run:
kcon <- new("KorAPConnection", verbose = TRUE, accessToken="e739u6eOzkwADQpVChxFg")
kq <- corpusQuery(kcon, "Ameisenplage", metadataOnly=FALSE)
kq <- fetchAll(kq)
kq@collectedMatches$snippet
```

```r
## End(Not run)
```

```r
## Not run:
kco <- new("KorAPConnection", accessToken="e739u6eOzkwADQpVChxFg")
persistAccessToken(kco)
```

```r
## End(Not run)
```
KorAPQuery-class

Class KorAPCorpusStats

Description

KorAPCorpusStats objects can hold information about a corpus or virtual corpus. KorAPCorpusStats objects can be obtained by the `corpusStats()` method.

Usage

```
## S4 method for signature 'KorAPCorpusStats'
show(object)
```

Arguments

- `object` : KorAPCorpusStats object

Slots

- `vc` : definition of the virtual corpus
- `tokens` : number of tokens
- `documents` : number of documents
- `sentences` : number of sentences
- `paragraphs` : number of paragraphs

KorAPQuery-class

Class KorAPQuery

Description

KorAPQuery objects represent the current state of a query to a KorAP server. New KorAPQuery objects are typically created by the `corpusQuery` method.

`fetchNext` fetches the next bunch of results of a KorAP query.

`frequencyQuery` combines `corpusQuery`, `corpusStats` and `ci` to compute a table with the relative frequencies and confidence intervals of one or multiple search terms across one or multiple virtual corpora.
Usage

## S4 method for signature 'KorAPQuery'
initialize(
  .Object,
  korapConnection = NULL,
  request = NULL,
  vc = "",
  totalResults = 0,
  nextStartIndex = 0,
  fields = c("corpusSigle", "textSigle", "pubDate", "pubPlace", "availability",
             "textClass", "snippet"),
  requestUrl = "",
  webUIRequestUrl = "",
  apiResponse = NULL,
  hasMoreMatches = FALSE,
  collectedMatches = NULL
)

## S4 method for signature 'KorAPQuery'
fetchNext(
  kqo,
  offset = kqo@nextStartIndex,
  maxFetch = maxResultsPerPage,
  verbose = kqo@korapConnection@verbose
)

## S4 method for signature 'KorAPQuery'
fetchAll(kqo, verbose = kqo@korapConnection@verbose)

## S4 method for signature 'KorAPQuery'
fetchRest(kqo, verbose = kqo@korapConnection@verbose)

## S4 method for signature 'KorAPConnection'
frequencyQuery(
  kco,
  query,
  vc = "",
  conf.level = 0.95,
  as.alternatives = FALSE,
  ...
)

## S3 method for class 'KorAPQuery'
format(x, ...)

## S4 method for signature 'KorAPQuery'
show(object)
Arguments

- **.Object**: ...
- **korapConnection**: KorAPConnection object
- **request**: query part of the request URL
- **vc**: definition of a virtual corpus
- **totalResults**: number of hits the query has yielded
- **nextStartIndex**: at what index to start the next fetch of query results
- **fields**: what data / metadata fields should be collected
- **requestUrl**: complete URL of the API request
- **webUIRequestUrl**: URL of a web frontend request corresponding to the API request
- **hasMoreMatches**: logical that signals if more query results can be fetched
- **collectedMatches**: matches already fetched from the KorAP-API-server
- **kqo**: object obtained from corpusQuery
- **offset**: start offset for query results to fetch
- **maxFetch**: maximum number of query results to fetch
- **verbose**: print progress information if true
- **kco**: KorAPConnection object (obtained e.g. from new("KorAPConnection")
- **query**: string that contains the corpus query. The query language depends on the ql parameter. Either query must be provided or KorAPUrl.
- **conf.level**: confidence level of the returned confidence interval (passed through `ci` to `prop.test`).
- **as.alternatives**: LOGICAL that specifies if the query terms should be treated as alternatives. If `as.alternatives` is TRUE, the sum over all query hits, instead of the respective vc token sizes is used as total for the calculation of relative frequencies.
- **...**: further arguments passed to or from other methods
- **x**: KorAPQuery object

Value

The `kqo` input object with updated slots collectedMatches, apiResponse, nextStartIndex, hasMoreMatches

References

https://ids-pub.bsz-bw.de/frontdoor/index/index/docId/9026
Examples

```r
q <- new("KorAPConnection") %>% corpusQuery("Ameisenplage") %>% fetchNext()
q@collectedMatches

q <- new("KorAPConnection") %>% corpusQuery("Ameisenplage") %>% fetchAll()
q@collectedMatches

q <- new("KorAPConnection") %>% corpusQuery("Ameisenplage") %>% fetchRest()
q@collectedMatches

new("KorAPConnection", verbose = TRUE) %>%
  frequencyQuery(c("Mücke", "Schnake"), paste0("pubDate in ", 2000:2003))
```

**percent**

Convert corpus frequency table of alternatives to percent

---

**Description**

Convenience function for converting frequency tables of alternative variants (generated with `as.alternatives=TRUE`) to percent.

**Usage**

```r
percent(df)
```

**Arguments**

- `df` table returned from `frequencyQuery`

**Value**

original table with converted columns `f`, `conf.low` and `conf.high`

**Examples**

```r
new("KorAPConnection") %>>%
frequencyQuery(c("Tollpatsch", "Tolpatsch"),
  vc=paste0("pubDate in ", 2000:2002),
  as.alternatives = TRUE) %>>%
percent()
```
**queryStringToLabel**  
Convert query or vc strings to plot labels

**Description**  
Converts a vector of query or vc strings to typically appropriate legend labels by clipping off prefixes and suffixes that are common to all query strings.

**Usage**  
`queryStringToLabel(data, pubDateOnly = FALSE, excludePubDate = FALSE)`

**Arguments**  
- `data`: string or vector of query or vc definition strings
- `pubDateOnly`: discard all but the publication date
- `excludePubDate`: discard publication date constraints

**Value**  
string or vector of strings with clipped off common prefixes and suffixes

**Examples**  
```r
queryStringToLabel(paste("textType = /Zeit.*/ & pubDate in", c(2010:2019)))
queryStringToLabel(c("[marmot/m=mood:subj]", "[marmot/m=mood:ind]"))
queryStringToLabel(c("wegen dem [tt/p=NN]", "wegen des [tt/p=NN]"))
```

---

**R KorAP Client**

**Description**  
R package to access the KorAP web service API.

**Details**  
See the README.md on github
Index

apiCall (KorAPConnection-class), 10
apiCall, KorAPConnection-method (KorAPConnection-class), 10

ci, 2, 12, 14
clearAccessToken (KorAPConnection-class), 10
clearAccessToken, KorAPConnection-method (KorAPConnection-class), 10
clearCache, 11
clearCache (KorAPConnection-class), 10
clearCache, KorAPConnection-method (KorAPConnection-class), 10
corpusQuery, 11, 12, 14
corpusQuery (corpusQuery, KorAPConnection-method), 3
corpusQuery, KorAPConnection-method, 3
corpusStats, 4, 12
corpusStats (corpusStats, KorAPConnection-method), 5
corpusStats, KorAPConnection-method, 5

corpusQuery (KorAPQuery-class), 12
corpusQuery, KorAPQuery-method (KorAPQuery-class), 12
corpusStats (KorAPQuery-class), 12
corpusStats, KorAPQuery-method, 12
fetchAll, 4
cfetchAll (KorAPQuery-class), 12
cfetchNext, 4
cfetchNext (KorAPQuery-class), 12
cfetchNext, KorAPQuery-method (KorAPQuery-class), 12
cfetchRest, 4
cfetchRest (KorAPQuery-class), 12
cfetchRest, KorAPQuery-method (KorAPQuery-class), 12
format (KorAPQuery-class), 12
frequencyQuery, 2, 8, 9, 15
frequencyQuery (KorAPQuery-class), 12

frequencyQuery (KorAPQuery-class), 12
geom_freq_by_year_ci, 6
ggplotly, 7
hc_add_series, 8
hc_freq_by_year_ci, 8
initialize, KorAPConnection-method (KorAPConnection-class), 10
initialize, KorAPQuery-method (KorAPQuery-class), 12
ipm, 9
keyring, 11
KorAPConnection, 3–5, 11, 14
KorAPConnection (KorAPConnection-class), 10
KorAPCorpusStats-class, 12
KorAPQuery, 4
KorAPQuery (KorAPQuery-class), 12
KorAPQuery-class, 12

percent, 15
persistAccessToken (KorAPConnection-class), 10
persistAccessToken, KorAPConnection-method (KorAPConnection-class), 10
prop.test, 2, 14
queryStringToLabel, 16
RKorAPClient, 16
show, KorAPConnection-method (KorAPConnection-class), 10
show, KorAPQuery-method (KorAPQuery-class), 12

show, KorAPQuery-method (KorAPQuery-class), 12