Package 'RKorAPClient'

September 7, 2022

Type Package

Title 'KorAP' Web Service Client Package

Version 0.7.5

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 Corpus DeReKo'
or the 'Corpus of the Contemporary Romanian Language CoRoLa', accessible for linguists 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-mannheim.de/>.

Depends R (>= 3.5.0)

Language en-US

License BSD_2_clause + file LICENSE

URL https://github.com/KorAP/RKorAPClient/,
https://korap.ids-mannheim.de/,
https://www.ids-mannheim.de/digspra/kl/projekte/korap

BugReports https://github.com/KorAP/RKorAPClient/issues

Encoding UTF-8

RoxygenNote 7.2.1

Imports R.cache, broom, ggplot2, tibble, magrittr, tidyr, dplyr, lubridate, highcharter, jsonlite, keyring, utils, httr, curl, methods, PTXQC, purrr, stringr, urltools

Suggests lifecycle, testthat

Collate 'KorAPConnection.R' 'KorAPCorpusStats.R' 'RKorAPClient-package.R' 'KorAPQuery.R' 'association-scores.R' 'ci.R' 'collocationAnalysis.R' 'collocationScoreQuery.R'

'hc_add_onclick_korap_search.R' 'hc_freq_by_year_ci.R' 'misc.R'
'reexports.R'

NeedsCompilation no

Author Marc Kupietz [aut, cre],
Nils Diewald [ctb],
Leibniz Institute for the German Language [cph, fnd]

Maintainer Marc Kupietz <kupietz@ids-mannheim.de>

Repository CRAN

Date/Publication 2022-09-07 18:40:09 UTC

R topics documented:

`association-score-functions`

Association score functions

Description

Functions to calculate different collocation association scores between a node (target word) and words in a window around the it. The functions are primarily used by `collocationScoreQuery()`.

pmi: pointwise mutual information

mi2: pointwise mutual information squared (Daille 1994), also referred to as mutual dependency (Thanopoulos et al. 2002)

mi3: pointwise mutual information cubed (Daille 1994), also referred to as log-frequency biased mutual dependency) (Thanopoulos et al. 2002)

logDice: log-Dice coefficient, a heuristic measure that is popular in lexicography (Rychlý 2008)

ll: log-likelihood (Dunning 1993) using Stefan Evert's (2004) simplified implementation

Usage

```
defaultAssociationScoreFunctions()

pmi(O1, O2, O, N, E, window_size)

mi2(O1, O2, O, N, E, window_size)

mi3(O1, O2, O, N, E, window_size)

logDice(O1, O2, O, N, E, window_size)

ll(O1, O2, O, N, E, window_size)
```

Arguments

O1	observed absolute frequency of node
O2	observed absolute frequency of collocate
O	observed absolute frequency of collocation
N	corpus size
E	expected absolute frequency of collocation (already adjusted to window size)
window_size	total window size around node (left neighbour count + right neighbour count)

Value

```
association score
```

References

Daille, B. (1994): Approche mixte pour l'extraction automatique de terminologie: statistiques lexicales et filtres linguistiques. PhD thesis, Université Paris 7.

Thanopoulos, A., Fakotakis, N., Kokkinakis, G. (2002): Comparative evaluation of collocation extraction metrics. In: Proc. of LREC 2002: 620–625.

Rychlý, Pavel (2008): A lexicographer-friendly association score. In Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN, 6–9. https://www.fi.muni.cz/usr/sojka/download/raslan2008/13.pdf.

Dunning, T. (1993): Accurate methods for the statistics of surprise and coincidence. Comput. Linguist. 19, 1 (March 1993), 61-74.

Evert, Stefan (2004): The Statistics of Word Cooccurrences: Word Pairs and Collocations. PhD dissertation, IMS, University of Stuttgart. Published in 2005, URN urn:nbn:de:bsz:93-opus-23714. Free PDF available from https://purl.org/stefan.evert/PUB/Evert2004phd.pdf

See Also

Other collocation analysis functions: collocationAnalysis,KorAPConnection-method, collocationScoreQuery,KorAP
synsemanticStopwords()

Examples

```
## Not run:

new("KorAPConnection", verbose = TRUE) %>%
collocationScoreQuery("Perlen", c("verziertes", "Säue"),
  scoreFunctions = append(defaultAssociationScoreFunctions(),
     list(localMI = function(O1, O2, O, N, E, window_size) {
                       O * log2(O/E)
                     })))

## End(Not run)
```

`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

Convenience function for converting frequency tables to instances per million.

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

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.

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

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

ipm(df)

percent(df)

queryStringToLabel(data, pubDateOnly = FALSE, excludePubDate = FALSE)

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

Arguments

df	table returned from frequencyQuery()
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.
data	string or vector of query or vc definition strings
pubDateOnly	discard all but the publication date
excludePubDate	discard publication date constraints
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.

Details

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

Value

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

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

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

See Also

ci is already included in frequencyQuery()

Examples

```
## Not run:

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)

## End(Not run)
## Not run:

new("KorAPConnection") %>% frequencyQuery("Test", paste0("pubDate in ", 2000:2002)) %>% ipm()
```

```
## End(Not run)
## Not run:

new("KorAPConnection") %>%
    frequencyQuery(c("Tollpatsch", "Tolpatsch"),
    vc=paste0("pubDate in ", 2000:2002),
    as.alternatives = TRUE) %>%
  percent()

## End(Not run)
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]"))

library(ggplot2)
kco <- new("KorAPConnection", verbose=TRUE)
## Not run:

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

## End(Not run)
```

---

collocationAnalysis,KorAPConnection-method
Collocation analysis

Description

[Experimental]

Performs a collocation analysis for the given node (or query) in the given virtual corpus.

Usage

```
## S4 method for signature 'KorAPConnection'
collocationAnalysis(
  kco,
  node,
  vc = "",
  lemmatizeNodeQuery = FALSE,
  minOccur = 5,
  leftContextSize = 5,
  rightContextSize = 5,
  topCollocatesLimit = 200,
```

```
  searchHitsSampleLimit = 20000,
  ignoreCollocateCase = FALSE,
  withinSpan = ifelse(exactFrequencies, "base/s=s", ""),
  exactFrequencies = TRUE,
  stopwords = append(RKorAPClient::synsemanticStopwords(), node),
  seed = 7,
  expand = length(vc) != length(node),
  maxRecurse = 0,
  addExamples = FALSE,
  thresholdScore = "logDice",
  threshold = 2,
  localStopwords = c(),
  collocateFilterRegex = "^[:alnum:]+-?[:alnum:]*$",
  ...
)
```

Arguments

kco	KorAPConnection() object (obtained e.g. from new("KorAPConnection")
node	target word
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.
lemmatizeNodeQuery	if TRUE, node query will be lemmatized, i.e. x -> [tt/l=x]
minOccur	minimum absolute number of observed co-occurrences to consider a collocate candidate
leftContextSize	size of the left context window
rightContextSize	size of the right context window
topCollocatesLimit	limit analysis to the n most frequent collocates in the search hits sample
searchHitsSampleLimit	limit the size of the search hits sample
ignoreCollocateCase	logical, set to TRUE if collocate case should be ignored
withinSpan	KorAP span specification for collocations to be searched within
exactFrequencies	if FALSE, extrapolate observed co-occurrence frequencies from frequencies in search hits sample, otherwise retrieve exact co-occurrence frequencies
stopwords	vector of stopwords not to be considered as collocates
seed	seed for random page collecting order
expand	if TRUE, node and vc parameters are expanded to all of their combinations
maxRecurse	apply collocation analysis recursively maxRecurse times

addExamples — If TRUE, examples for instances of collocations will be added in a column example. This makes a difference in particular if node is given as a lemma query.

thresholdScore — association score function (see association-score-functions) to use for computing the threshold that is applied for recursive collocation analysis calls

threshold — minimum value of thresholdScore function call to apply collocation analysis recursively

localStopwords — vector of stopwords that will not be considered as collocates in the current function call, but that will not be passed to recursive calls

collocateFilterRegex — allow only collocates matching the regular expression

... — more arguments will be passed to collocationScoreQuery()

Details

The collocation analysis is currently implemented on the client side, as some of the functionality is not yet provided by the KorAP backend. Mainly for this reason it is very slow (several minutes, up to hours), but on the other hand very flexible. You can, for example, perform the analysis in arbitrary virtual corpora, use complex node queries, and look for expression-internal collocates using the focus function (see examples and demo).

To increase speed at the cost of accuracy and possible false negatives, you can decrease searchHitsSampleLimit and/or topCollocatesLimit and/or set exactFrequencies to FALSE.

Note that currently not the tokenization provided by the backend, i.e. the corpus itself, is used, but a tinkered one. This can also lead to false negatives and to frequencies that differ from corresponding ones acquired via the web user interface.

Value

Tibble with top collocates, association scores, corresponding URLs for web user interface queries, etc.

See Also

Other collocation analysis functions: association-score-functions, collocationScoreQuery,KorAPConnection-meth
synsemanticStopwords()

Examples

```
## Not run:

 # Find top collocates of "Packung" inside and outside the sports domain.
 new("KorAPConnection", verbose = TRUE) %>%
  collocationAnalysis("Packung", vc=c("textClass=sport", "textClass!=sport"),
                      leftContextSize=1, rightContextSize=1, topCollocatesLimit=20) %>%
  dplyr::filter(logDice >= 5)

## End(Not run)
```

```
## Not run:

# Identify the most prominent light verb construction with "in ... setzen".
# Note that, currently, the use of focus function disallows exactFrequencies.
new("KorAPConnection", verbose = TRUE) %>%
  collocationAnalysis("focus(in [tt/p=NN] {[tt/l=setzen]})",
    leftContextSize=1, rightContextSize=0, exactFrequencies=FALSE, topCollocatesLimit=20)

## End(Not run)
```

`collocationScoreQuery,KorAPConnection-method`

Query frequencies of a node and a collocate and calculate collocation association scores

---

Description

Computes various collocation association scores based on `frequencyQuery()`s for a target word and a collocate.

Usage

```
## S4 method for signature 'KorAPConnection'
collocationScoreQuery(
  kco,
  node,
  collocate,
  vc = "",
  lemmatizeNodeQuery = FALSE,
  lemmatizeCollocateQuery = FALSE,
  leftContextSize = 5,
  rightContextSize = 5,
  scoreFunctions = defaultAssociationScoreFunctions(),
  smoothingConstant = 0.5,
  observed = NA,
  ignoreCollocateCase = FALSE,
  withinSpan = "base/s=s"
)
```

Arguments

kco	`KorAPConnection()` object (obtained e.g. from `new("KorAPConnection")`
node	target word
collocate	collocate of target word
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.

`lemmatizeNodeQuery`
: logical, set to TRUE if node query should be lemmatized, i.e. `x -> [tt/l=x]`

`lemmatizeCollocateQuery`
: logical, set to TRUE if collocate query should be lemmatized, i.e. `x -> [tt/l=x]`

`leftContextSize`
: size of the left context window

`rightContextSize`
: size of the right context window

`scoreFunctions`
: named list of score functions of the form function(O1, O2, O, N, E, window_size), see e.g. pmi

`smoothingConstant`
: smoothing constant will be added to all observed values

`observed`
: if collocation frequencies are already known (or estimated from a sample) they can be passed as a vector here, otherwise: NA

`ignoreCollocateCase`
: logical, set to TRUE if collocate case should be ignored

`withinSpan`
: KorAP span specification for collocations to be searched within

Value

tibble with query KorAP web request URL, all observed values and association scores

See Also

Other collocation analysis functions: association-score-functions, collocationAnalysis,KorAPConnection-method, synsemanticStopwords()

Examples

```
## Not run:

new("KorAPConnection", verbose = TRUE) %>%
  collocationScoreQuery("Grund", "triftiger")

## End(Not run)

## Not run:

new("KorAPConnection", verbose = TRUE) %>%
collocationScoreQuery("Grund", c("guter", "triftiger"),
  scoreFunctions = list(localMI = function(O1, O2, O, N, E, window_size) { O * log2(O/E) }) )

## End(Not run)

## Not run:

library(highcharter)
library(tidyr)
new("KorAPConnection", verbose = TRUE) %>%
```

```
  collocationScoreQuery("Team", "agil", vc = paste("pubDate in", c(2014:2018)),
                        lemmatizeNodeQuery = TRUE, lemmatizeCollocateQuery = TRUE) %>%
                 pivot_longer(14:last_col(), names_to = "measure", values_to = "score") %>%
  hchart(type="spline", hcaes(label, score, group=measure)) %>%
  hc_add_onclick_korap_search()

## End(Not run)
```

`corpusStats,KorAPConnection-method`
Fetch information about a (virtual) corpus

Description

Fetch information about a (virtual) corpus

Usage

```
## 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

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

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

`hc_add_onclick_korap_search` *Add KorAP search click events to highchart plots*

Description

[Experimental]

Adds on-click events to data points of highcharts that were constructed with `frequencyQuery()` or `collocationScoreQuery()`. Clicks on data points then launch KorAP web UI queries for the given query term and virtual corpus in a separate tab.

Usage

```
hc_add_onclick_korap_search(hc)
```

Arguments

hc	A highchart htmlwidget object generated by e.g. `frequencyQuery()`.

Value

The input highchart object with added on-click events.

See Also

Other highcharter-helpers: `hc_freq_by_year_ci()`

Examples

```
## Not run:

library(highcharter)
library(tidyr)

new("KorAPConnection", verbose = TRUE) %>%
  collocationScoreQuery("Team", "agil", vc = paste("pubDate in", c(2014:2018)),
                        lemmatizeNodeQuery = TRUE, lemmatizeCollocateQuery = TRUE) %>%
                         pivot_longer(c("O", "E")) %>%
  hchart(type="spline", hcaes(label, value, group=name)) %>%
  hc_add_onclick_korap_search()

## End(Not run)
```

`hc_freq_by_year_ci` *Plot interactive frequency curves with confidence intervals*

Description

[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

```
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()`

Value

A highchart htmlwidget object containing the frequency plot.

See Also

Other highcharter-helpers: `hc_add_onclick_korap_search()`

Examples

```
## Not run:

year <- 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)


kco <- new("KorAPConnection", verbose = TRUE)
expand_grid(
  condition = c("textDomain = /Wirtschaft.*/", "textDomain != /Wirtschaft.*/"),
  year = (2005:2011)
) %>%
  cbind(frequencyQuery(
    kco,
    "[tt/l=Heuschrecke]",
    paste0(.$condition, " & pubDate in ", .$year)
  ))  %>%
  hc_freq_by_year_ci()

## End(Not run)
```

KorAPConnection-class *Class KorAPConnection*

Description

KorAPConnection objects represent the connection to a KorAP server. New KorAPConnection objects can be created by new("KorAPConnection").

Usage

```
## 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 = 240,
  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,
  timeout = kco@timeout
)

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

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

Arguments

.Object	KorAPConnection object
KorAPUrl	URL of the web user interface 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 token>")`. 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 keying. 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	tineout in seconds for API requests (this does not influence server internal timeouts).
verbose	logical that decides whether following operations will default to be verbose.
cache	logical that 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)
object	KorAPConnection object

Value

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

Slots

KorAPUrl URL of the web user interface of the KorAP server used in the connection.

apiVersion requested KorAP API version.

indexRevision indexRevision code as reported from API via X-Index-Revision HTTP header.

apiUrl full URL of API including version.

accessToken OAuth2 access token.

userAgent user agent string used for connection the API.

timeout tineout in seconds for API requests (this does not influence server internal timeouts)

verbose logical that decides whether operations will default to be verbose.

cache logical that decides if API calls are cached locally.

welcome list containing HTTP response received from KorAP server welcome function.

Examples

```
## Not run:

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

## End(Not run)

## Not run:

kcon <- new("KorAPConnection", verbose = TRUE, accessToken="e739u6eOzkwADQPdVChxFg")
kq <- corpusQuery(kcon, "Ameisenplage", metadataOnly=FALSE)
kq <- fetchAll(kq)
kq@collectedMatches$snippet

## End(Not run)

## Not run:

kco <- new("KorAPConnection", accessToken="e739u6eOzkwADQPdVChxFg")
persistAccessToken(kco)

## End(Not run)

## Not run:
```

```
kco <- new("KorAPConnection")
clearAccessToken(kco)

## End(Not run)
```

`KorAPCorpusStats-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

This class provides methods to perform different kinds of queries on the KorAP API server. `KorAPQuery` objects, which are typically created by the `corpusQuery()` method, represent the current state of a query to a KorAP server.

`corpusQuery` performs a corpus query via a connection to a KorAP-API-server

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

`fetchAll` fetches all results of a KorAP query.

`frequencyQuery` combines `corpusQuery()`, `corpusStats()` and `ci()` to compute a table with the relative frequencies and confidence intervals of one ore 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 'KorAPConnection'
corpusQuery(
  kco,
  query = if (missing(KorAPUrl))
   stop("At least one of the parameters query and KorAPUrl must be specified.", call. =
    FALSE) else httr::parse_url(KorAPUrl)$query$q,
  vc = if (missing(KorAPUrl)) "" else httr::parse_url(KorAPUrl)$query$cq,
  KorAPUrl,
  metadataOnly = TRUE,
  ql = if (missing(KorAPUrl)) "poliqarp" else httr::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
)

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

## 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,
  ...
)

buildWebUIRequestUrl(
  kco,
  query = if (missing(KorAPUrl))
   stop("At least one of the parameters query and KorAPUrl must be specified.", call. =
    FALSE) else httr::parse_url(KorAPUrl)$query$q,
  vc = if (missing(KorAPUrl)) "" else httr::parse_url(KorAPUrl)$query$cq,
  KorAPUrl,
  metadataOnly = TRUE,
  ql = if (missing(KorAPUrl)) "poliqarp" else httr::parse_url(KorAPUrl)$query$ql,
  fields = c("corpusSigle", "textSigle", "pubDate", "pubPlace", "availability",
    "textClass", "snippet"),
  accessRewriteFatal = TRUE
)

## 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	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.
totalResults	number of hits the query has yielded
nextStartIndex	at what index to start the next fetch of query results
fields	(meta)data fields that will be fetched for every match.
requestUrl	complete URL of the API request

webUIRequestUrl
: URL of a web frontend request corresponding to the API request

apiResponse
: data-frame representation of the JSON response of the API request

hasMoreMatches
: logical that signals if more query results can be fetched

collectedMatches
: matches already fetched from the KorAP-API-server

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.

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.

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

verbose
: print progress information if true

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?

kqo
: object obtained from corpusQuery()

offset
: start offset for query results to fetch

maxFetch
: maximum number of query results to fetch

randomizePageOrder
: fetch result pages in pseudo random order if true. Use set.seed() to set seed for reproducible results.

...
: further arguments passed to or from other methods

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.

x
: KorAPQuery object

object
: KorAPQuery 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.

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

References

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

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

See Also

KorAPConnection(), fetchNext(), fetchRest(), fetchAll(), corpusStats()

Examples

```
## Not run:

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

## End(Not run)

## Not run:

# 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")

## End(Not run)

## Not run:

# 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")

## End(Not run)
## Not run:

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

## End(Not run)

## Not run:

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

## End(Not run)

## Not run:

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

## End(Not run)

## Not run:

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

## End(Not run)
```

`synsemanticStopwords` *Preliminary synsemantic stopwords function*

Description

[Experimental]

Preliminary synsemantic stopwords function to be used in collocation analysis.

Usage

```
synsemanticStopwords(...)
```

Arguments

...	future arguments for language detection

Details

Currently only suitable for German. See stopwords package for other languages.

Value

Vector of synsemantic stopwords.

See Also

Other collocation analysis functions: association-score-functions, collocationAnalysis,KorAPConnection-method
collocationScoreQuery,KorAPConnection-method

Index