Package 'NLPclient'

December 13, 2019

Version 1.0

Title Stanford 'CoreNLP' Annotation Client

Description Stanford 'CoreNLP' annotation client.
Stanford 'CoreNLP' <https://stanfordnlp.github.io/CoreNLP/index.html> integrates all NLP tools from the Stanford Natural Language Processing Group, including a part-of-speech (POS) tagger, a named entity recognizer (NER), a parser, and a coreference resolution system, and provides model files for the analysis of English. More information can be found in the README.

Imports NLP (>= 0.1-10), utils, xml2, curl

Depends R (>= 3.2.0)

License GPL-2

RoxygenNote 6.1.1

NeedsCompilation no

Author Florian Schwendinger [aut, cre],
Kurt Hornik [aut]

Maintainer Florian Schwendinger <FlorianSchwendinger@gmx.at>

Repository CRAN

Date/Publication 2019-12-13 14:20:15 UTC

R topics documented:

nlp_server_docker_run *Create the Docker Run Command*

Description

This function helps to create the docker run command which can be copied into the terminal (Unix) or docker toolbox (Windows).

Usage

```
nlp_server_docker_run(name = "coreNLP", docker = "schwe/corenlp",
  memory = "-mx8g", threads = NA, port = 9000L,
  status_port = 9000L, timeout = 15000L, strict = FALSE,
  quiet = FALSE, ssl = FALSE,
  key = "edu/stanford/nlp/pipeline/corenlp.jks", username = "",
  password = "", annotators = "all", preload = "",
  server_properties = "", default_properties = "", cleanup = TRUE)
```

Arguments

name	a character string giving the name of the docker container.
docker	a character string giving the image name.
memory	a character string giving the java memory options.
threads	an integer giving the number of threads to be used. The default is NA in which case CoreNLP will choose the number of threads to be used.
port	an integer giving the server port.
status_port	an integer giving the port to run the liveness and readiness server on.
timeout	an integer giving the maximum amount of time, in milliseconds, to wait for an annotation to finish before cancelling it.
strict	a logical controlling whether server strictly follows the HTTP standards.
quiet	a logical controlling whether the incoming requests are logged to STDOUT.
ssl	a logical controlling whether an SSL should be run.
key	a character string giving the classpath or filepath to the *.jks key to use for creating an SSL connection.
username	a character string giving the username of the server.
password	a character string giving the password of the server.
annotators	a character string giving the default annotators (e.g. "tokenize,ssplit,pos").
preload	a character string giving the set of annotators to warm up in the cache when the server boots up.
server_properties	a character giving the path to the default properties.
default_properties	a character string giving the server properties, to be written into the file "default.properties".
cleanup	a logical giving if docker should automatically clean up the container and remove the file system when the container exits.

Value

A character string which can be copied into the terminal (or docker toolbox) to start the coreNLP server.

Examples

```
nlp_server_docker_run(memory = "-mx6g")
```

---

ping_nlp_client *Ping Stanford CoreNLP Server*

Description

Ping Stanford CoreNLP server.

Usage

```
ping_nlp_client(port = 9000L, host = "localhost",
  user_agent = "NLPclient")
```

Arguments

port	an integer giving the port (default is 9000L).
host	a character string giving the hostname of the server.
user_agent	a character string giving the client name.

Value

Returns "pong" if the server is online "" otherwise.

Examples

```
ping_nlp_client()
```

StanfordCoreNLP_Pipeline

Stanford CoreNLP *annotator pipeline*

Description

Create a Stanford CoreNLP annotator pipeline.

Usage

```
StanfordCoreNLP_Pipeline(annotators = c("pos", "lemma"),
  language = "en", control = list(), port = 9000L,
  host = "localhost")
```

Arguments

annotators	a character string specifying the annotators to be used in addition to 'ssplit' (sentence token annotation) and 'tokenize' (word token annotations), with elements "pos" (POS tagging), "lemma" (lemmatizing), "ner" (named entity recognition), "regexner" (rule-based named entity recognition over token sequences using Java regular expressions), "parse" (constituency parsing), "depparse" (dependency parsing), "sentiment" (sentiment analysis), "coref" (coference resolution), "dcoref" (deterministic coference resolution), "cleanxml" (clean XML tags), or "relation" (relation extraction), or unique abbreviations thereof. Ignored for languages other than English.
language	a character string giving the ISO-639 code of the language being processed by the annotator pipeline.
control	a named or empty (default) list vector with annotator control options, with the names giving the option names. See https://stanfordnlp.github.io/CoreNLP/annotators.html for available control options.
port	an integer giving the port (default is 9000L).
host	a character string giving the hostname of the server.

Value

An Annotator object providing the annotator pipeline.

Note

See https://stanfordnlp.github.io/CoreNLP/#citing-stanford-corenlp-in-papers for information on citing Stanford CoreNLP.

Using the parse annotator requires considerable amounts of (Java) memory. The Stanford CoreNLP documentation suggests starting the JVM with at least 3GB of memory on 64-bit systems (and in fact, not using 32-bit systems), and hence have the JVM started with '-Xmx3g' unless option java.parameters is set to something non-empty (hence, this option should be set appropriately to accommodate different memory requirements or constraints).

Using the coreference annotators nowadays requires even more (Java) memory. The Stanford CoreNLP documentation suggests starting the JVM with at least 5GB of memory; we find 4GB sufficient. Hence, to use these annotators one needs to set option `java.parameters` as appropriate before starting the JVM.

See Also

https://stanfordnlp.github.io/CoreNLP/ for more information about the Stanford CoreNLP tools.

Examples

```
require("NLP")
s <- as.String(paste("Stanford University is located in California.",
                     "It is a great university."))
s

## Annotators: ssplit, tokenize:
if ( ping_nlp_client() == "pong" ) {
p <- StanfordCoreNLP_Pipeline(NULL)
a <- p(s)
a

## Annotators: ssplit, tokenize, pos, lemma (default):
p <- StanfordCoreNLP_Pipeline()
a <- p(s)
a

## Equivalently:
annotate(s, p)

## Annotators: ssplit, tokenize, parse:
p <- StanfordCoreNLP_Pipeline("parse")
a <- p(s)
a

## Respective formatted parse trees using Penn Treebank notation
## (see <https://catalog.ldc.upenn.edu/docs/LDC95T7/cl93.html>):
ptexts <- sapply(subset(a, type == "sentence")$features, `[[`, "parse")
ptexts

## Read into NLP Tree objects.
ptrees <- lapply(ptexts, Tree_parse)
ptrees

## Basic dependencies:
depends <- lapply(subset(a, type == "sentence")$features, `[[`,
                  "basic-dependencies")
depends
## Note that the non-zero ids (gid for governor and did for dependent)
## refer to word token positions within the respective sentences, and
## not the ids of these token in the annotation: these can easily be
```

```
## matched using the sentence constituents features:
lapply(subset(a, type == "sentence")$features, `[[`, "constituents")

## (Similarly for sentence ids used in dcoref document features.)

## Note also that the dependencies are returned as a data frame
## inheriting from class "Stanford_typed_dependencies" which has print
## and format methods for obtaining the usual formatting.
depends[[1L]]
## Use as.data.frame() to obtain strip this class:
as.data.frame(depends[[1L]])
}
```

Index