Package: cleanNLP (via r-universe)

September 17, 2024

Type Package

Title A Tidy Data Model for Natural Language Processing

Version 3.1.0

Author Taylor B. Arnold [aut, cre]

Maintainer Taylor B. Arnold <tarnold2@richmond.edu>

Description Provides a set of fast tools for converting a textual corpus into a set of normalized tables. Users may make use of the 'udpipe' back end with no external dependencies, or a Python back ends with 'spaCy' <https://spacy.io>. Exposed annotation tasks include tokenization, part of speech tagging, named entity recognition, and dependency parsing.

Depends R (>= 3.5.0)

Imports Matrix (>= 1.2), udpipe, reticulate, stringi, stats, methods

Suggests knitr (>= 1.15), rmarkdown (>= 1.4), testthat (>= 1.0.1), covr (>= 2.2.2)

SystemRequirements Python (>= 3.7.0)

License LGPL-2

URL https://statsmaths.github.io/cleanNLP/

BugReports https://github.com/statsmaths/cleanNLP/issues

LazyData true

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.3.1

Repository https://statsmaths.r-universe.dev

RemoteUrl https://github.com/statsmaths/cleannlp

RemoteRef HEAD

RemoteSha 0e6bf7d8f618b62a875a305fe0a038c6564b4641

Contents

`cleanNLP-package` *cleanNLP: A Tidy Data Model for Natural Language Processing*

Description

Provides a set of fast tools for converting a textual corpus into a set of normalized tables. Multiple NLP backends can be used, with the output standardized into a normalized format. Options include stringi (very fast, but only provides tokenization), udpipe (fast, many languages, includes part of speech tags and dependencies), and spacy (python backend; includes named entity recognition).

Details

Once the package is set up, run one of `cnlp_init_stringi`, `cnlp_init_spacy`, or `cnlp_init_udpipe` to load the desired NLP backend. After this function is done running, use `cnlp_annotate` to run the annotation engine over a corpus of text. The package vignettes provide more detailed set-up information.

See Also

Useful links:

- https://statsmaths.github.io/cleanNLP/
- Report bugs at https://github.com/statsmaths/cleanNLP/issues

Examples

```
## Not run:
library(cleanNLP)

# load the annotation engine
cnlp_init_stringi()

# annotate your text
input <- data.frame(
```

```
 text=c(
   "This is a sentence.",
   "Here is something else to parse!"
 ),
 stringsAsFactors=FALSE
)

## End(Not run)
```

---

cnlp_annotate *Run the annotation pipeline on a set of documents*

Description

Runs the clean_nlp annotators over a given corpus of text using the desired backend. The details for which annotators to run and how to run them are specified by using one of: cnlp_init_stringi, cnlp_init_spacy, or cnlp_init_udpipe.

Usage

```
cnlp_annotate(
  input,
  backend = NULL,
  verbose = 10,
  text_name = "text",
  doc_name = "doc_id"
)
```

Arguments

input	an object containing the data to parse. Either a character vector with the texts (optional names can be given to provide document ids) or a data frame. The data frame should have a column named 'text' containing the raw text to parse; if there is a column named 'doc_id', it is treated as a a document identifier. The name of the text and document id columns can be changed by setting text_name and doc_name This conforms with corpus objects respecting the Text Interchange Format (TIF), while allowing for some variation.
backend	name of the backend to use. Will default to the last model to be initalized.
verbose	set to a positive integer n to display a progress message to display every n'th record. The default is 10. Set to a non-positive integer to turn off messages. Logical input is converted to an integer, so it also possible to set to TRUE (1) to display a message for every document and FALSE (0) to turn off messages.
text_name	column name containing the text input. The default is 'text'. This parameter is ignored when input is a character vector.
doc_name	column name containing the document ids. The default is 'doc_id'. This parameter is ignored when input is a character vector.

Details

The returned object is a named list where each element containing a data frame. The document table contains one row for each document, along with with all of the metadata that was passed as an input. The tokens table has one row for each token detected in the input. The first three columns are always "doc_id" (to index the input document), "sid" (an integer index for the sentence number), and "tid" (an integer index to the specific token). Together, these are a primary key for each row.

Other columns provide extracted data about each token, which differ slightly based on which backend, language, and options are supplied.

- **token**: detected token, as given in the original input
- **token_with_ws**: detected token along with white space; in, theory, collapsing this field through an entire document will yield the original text
- **lemma**: lemmatised version of the token; the exact form depends on the choosen language and backend
- **upos**: the universal part of speech code; see https://universaldependencies.org/u/pos/all.html for more information
- **xpos**: language dependent part of speech code; the specific categories and their meaning depend on the choosen backend, model and language
- **feats**: other extracted linguistic features, typically given as Universal Dependencies (https://universaldependencies.org/u/feat/index.html), but can be model dependent; currently only provided by the udpipe backend
- **tid_source**: the token id (tid) of the head word for the dependency relationship starting from this token; for the token attached to the root, this will be given as zero
- **relation**: the dependency relation, usually provided using Universal Dependencies (more information available here https://universaldependencies.org/), but could be different for a specific model

Value

a named list with components "token", "document" and (when running spacy with NER) "entity".

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

Examples

```
cnlp_init_stringi()
cnlp_annotate(un)
```

cnlp_download_spacy *Download model files needed for spacy*

Description

The cleanNLP package does not supply the model files required for using the spacy backend. These files can be downloaded with this function. If you need more control, download directly from Python.

Usage

```
cnlp_download_spacy(model_name = "en_core_web_sm")
```

Arguments

model_name	string giving the model namel. Defaults to "en_core_web_sm".

Examples

```
## Not run:
cnlp_download_spacy(model_name="en_core_web_sm")

## End(Not run)
```

cnlp_init_spacy *Interface for initializing the spacy backend*

Description

This function must be run before annotating text with the spacy backend. It sets the properties for the spacy engine and loads the file using the R to Python interface provided by reticulate.

Usage

```
cnlp_init_spacy(model_name = NULL, disable = NULL, max_length = NULL)
```

Arguments

model_name	string giving the model name for the spacy backend. Defaults to "en_core_web_sm" (English) if set to NULL.
disable	an optional vector of pipes to disable.
max_length	amount of temporary memory provided to Spacy, in characters. The default of 1000000 should work for most applications, but can be increased when working with long documents.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

Examples

```
## Not run:
cnlp_init_spacy(model_name = "en_core_web_sm")

## End(Not run)
```

---

cnlp_init_stringi *Interface for initializing the standard R backend*

Description

This function must be run before annotating text with the tokenizers backend.

Usage

```
cnlp_init_stringi(locale = NULL, include_spaces = FALSE)
```

Arguments

locale	string giving the locale name to pass to the stringi functions. If NULL, the default locale is selected
include_spaces	logical. Should spaces be included as tokens in the output. Defaults to FALSE

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

Examples

```
## Not run:
cnlp_init_stringi()

## End(Not run)
```

`cnlp_init_udpipe` *Interface for initializing the udpipe backend*

Description

This function must be run before annotating text with the udpipe backend. It will parse in English by default, but you can load other models as well.

Usage

```
cnlp_init_udpipe(
  model_name = NULL,
  model_path = NULL,
  tokenizer = "tokenizer",
  tagger = "default",
  parser = "default"
)
```

Arguments

`model_name`	string giving the model namel. Defaults to "english" if NULL. Ignored if `model_path` is not NULL.
`model_path`	provide a full path to a model file.
`tokenizer`	a character string of length 1, which is either 'tokenizer' (default udpipe tokenisation) or a character string with more complex tokenisation options as specified in <URL: http://ufal.mff.cuni.cz/udpipe/users-manual> in which case tokenizer should be a character string where the options are put after each other using the semicolon as separation.
`tagger`	a character string of length 1, which is either 'default' (default udpipe POS tagging and lemmatisation) or 'none' (no POS tagging and lemmatisation needed) or a character string with more complex tagging options as specified in <URL: http://ufal.mff.cuni.cz/udpipe/users-manual> in which case tagger should be a character string where the options are put after each other using the semicolon as separation.
`parser`	a character string of length 1, which is either 'default' (default udpipe dependency parsing) or 'none' (no dependency parsing needed) or a character string with more complex parsing options as specified in <URL: http://ufal.mff.cuni.cz/udpipe/users-manual> in which case parser should be a character string where the options are put after each other using the semicolon as separation.

Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

Examples

```
## Not run:
cnlp_init_udpipe(model_name = "english")

## End(Not run)
```

`cnlp_utils_pca` *Compute Principal Components and store as a Data Frame*

Description

Takes a matrix and returns a data frame with the top principal components extracted. This is a simple but powerful technique for visualizing a corpus of documents.

Usage

```
cnlp_utils_pca(x, k = 2, center = TRUE, scale = TRUE)
```

Arguments

`x`	a matrix object to pass to `prcomp`
`k`	integer. The number of components to include in the output.
`center`	logical. Should the data be centered?
`scale`	logical. Should the data be scaled? Note that this will need to be set to false if any columns in `x` are constant if `center` is also true.

Value

a data frame object containing the top `k` principal components of the data in `x`.

`cnlp_utils_tfidf` *Construct the TF-IDF Matrix from Annotation or Data Frame*

Description

Given annotations, this function returns the term-frequency inverse document frequency (tf-idf) matrix from the extracted lemmas.

Usage

```
cnlp_utils_tfidf(
  object,
  tf_weight = c("lognorm", "binary", "raw", "dnorm"),
  idf_weight = c("idf", "smooth", "prob", "uniform"),
  min_df = 0.1,
  max_df = 0.9,
  max_features = 10000,
  doc_var = "doc_id",
  token_var = "lemma",
  vocabulary = NULL,
  doc_set = NULL
)

cnlp_utils_tf(
  object,
  tf_weight = "raw",
  idf_weight = "uniform",
  min_df = 0,
  max_df = 1,
  max_features = 10000,
  doc_var = "doc_id",
  token_var = "lemma",
  vocabulary = NULL,
  doc_set = NULL
)
```

Arguments

`object`	a data frame containing an identifier for the document (set with `doc_var`) and token (set with `token_var`)
`tf_weight`	the weighting scheme for the term frequency matrix. The selection `lognorm` takes one plus the log of the raw frequency (or zero if zero), `binary` encodes a zero one matrix indicating simply whether the token exists at all in the document, `raw` returns raw counts, and `dnorm` uses double normalization.
`idf_weight`	the weighting scheme for the inverse document matrix. The selection `idf` gives the logarithm of the simple inverse frequency, `smooth` gives the logarithm of one plus the simple inverse frequency, and `prob` gives the log odds of the the token occurring in a randomly selected document. Set to `uniform` to return just the term frequencies.
`min_df`	the minimum proportion of documents a token should be in to be included in the vocabulary
`max_df`	the maximum proportion of documents a token should be in to be included in the vocabulary
`max_features`	the maximum number of tokens in the vocabulary
`doc_var`	character vector. The name of the column in `object` that contains the document ids. Defaults to "doc_id".

token_var	character vector. The name of the column in object that contains the tokens. Defaults to "lemma".
vocabulary	character vector. The vocabulary set to use in constructing the matrices. Will be computed within the function if set to NULL. When supplied, the options min_df, max_df, and max_features are ignored.
doc_set	optional character vector of document ids. Useful to create empty rows in the output matrix for documents without data in the input. Most users will want to keep this equal to NULL, the default, to have the function compute the document set automatically.

Value

a sparse matrix with dimnames giving the documents and vocabular.

---

un *Universal Declaration of Human Rights*

Description

Data frame containing the 30 Articles in the United Nations' Universal Declaration of Human Rights, ratified on 10 December 1948 in Paris, France.

References

https://www.un.org/en/universal-declaration-human-rights/

---

word_frequency *Most frequent English words*

Description

A dataset of the 150k most frequently used English words, extracted by Peter Norvig from the Google Web Trillion Word Corpus. Frequencies are multiplied by 100.

References

https://norvig.com/ngrams/

Index