Package 'textpress'

October 14, 2024

Type Package

Title A Lightweight and Versatile NLP Toolkit

Version 1.0.0

Maintainer Jason Timm <JaTimm@salud.unm.edu>

Description A simple Natural Language Processing (NLP) toolkit focused on search-centric workflows with minimal dependencies. The package offers key features for web scraping, text processing, corpus search, and text embedding generation via the 'Hugging-Face API' <https://huggingface.co/docs/api-inference/index>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5)

Imports data.table, httr, Matrix, rvest, stringi, stringr, xml2, pbapply, jsonlite, lubridate

RoxygenNote 7.3.1

URL https://github.com/jaytimm/textpress, https://jaytimm.github.io/textpress/

BugReports https://github.com/jaytimm/textpress/issues

NeedsCompilation no

Author Jason Timm [aut, cre]

Repository CRAN

Date/Publication 2024-10-14 12:30:05 UTC

Contents

.decode_duckduckgo_urls 2
.extract_links 3
.get_site 3
.process_bing 4
.process_duckduckgo 4
.process_yahoo 5

abbreviations 6
api_huggingface_embeddings 6
extract_date 7
nlp_build_chunks 8
nlp_cast_tokens 9
nlp_melt_tokens 9
nlp_split_paragraphs 10
nlp_split_sentences 11
nlp_tokenize_text 12
sem_nearest_neighbors 12
sem_search_corpus 13
standardize_date 14
web_scrape_urls 15
web_search 15

Index **17**

`.decode_duckduckgo_urls` *Decode DuckDuckGo Redirect URLs*

Description

This function decodes the DuckDuckGo search result URLs that are redirected.

Usage

```
.decode_duckduckgo_urls(redirected_urls)
```

Arguments

`redirected_urls`	A vector of DuckDuckGo search result URLs.

Value

A vector of decoded URLs.

.extract_links *Extract links from a search engine result page*

Description

This function extracts all the links (href attributes) from a search engine result page.

Usage

```
.extract_links(search_url)
```

Arguments

search_url	The URL of the search engine result page.

Value

A character vector of URLs.

.get_site *Get Site Content and Extract HTML Elements*

Description

This function attempts to retrieve the HTML content of a URL, extract specific HTML elements (e.g., paragraphs, headings), and extract publication date information using the `extract_date` function.

Usage

```
.get_site(x)
```

Arguments

x	A URL to extract content and publication date from.

Value

A data frame with columns for the URL, HTML element types, text content, extracted date, and date source.

.process_bing *Process Bing search results*

Description

This function retrieves and processes search results from Bing.

Usage

```
.process_bing(
  search_term,
  num_pages,
  time_filter,
  insite,
  intitle,
  combined_pattern
)
```

Arguments

search_term	The search query.
num_pages	Number of result pages to retrieve.
time_filter	Optional time filter ("week", "month", "year").
insite	Restrict search to a specific domain.
intitle	Search within the title.
combined_pattern	A pattern for filtering out irrelevant URLs.

Value

A ‘data.table‘ of search results from Bing.

.process_duckduckgo *Process DuckDuckGo search results*

Description

This function handles the extraction of search results from DuckDuckGo.

Usage

```
.process_duckduckgo(
  search_term,
  num_pages,
  time_filter,
  insite,
  intitle,
  combined_pattern
)
```

Arguments

search_term	The search query.
num_pages	Number of result pages to retrieve.
time_filter	Optional time filter ("week", "month", "year").
insite	Restrict search to a specific domain.
intitle	Search within the title.
combined_pattern	A pattern for filtering out irrelevant URLs.

Value

A ‘data.table‘ of search results from DuckDuckGo.

.process_yahoo *Process Yahoo News search results*

Description

This function retrieves and processes search results from Yahoo News, automatically sorting by the most recent articles.

Usage

```
.process_yahoo(search_term, num_pages, combined_pattern = combined_pattern)
```

Arguments

search_term	The search query.
num_pages	Number of result pages to retrieve.
combined_pattern	A pattern for filtering out irrelevant URLs.

Value

A ‘data.table‘ of search results from Yahoo News.

abbreviations *Common Abbreviations for Sentence Splitting*

Description

A character vector of common abbreviations used in English. These abbreviations are used to assist in sentence splitting, ensuring that sentence boundaries are not incorrectly identified at these abbreviations.

Usage

```
abbreviations
```

Format

A character vector with some common English abbreviations.

Source

Developed internally for sentence splitting functionality.

api_huggingface_embeddings *Call Hugging Face API for Embeddings*

Description

Retrieves embeddings for text data using Hugging Face's API. It can process a batch of texts or a single query. Mostly for demo purposes.

Usage

```
api_huggingface_embeddings(
  tif,
  text_hierarchy,
  api_token,
  api_url = NULL,
  query = NULL,
  dims = 384,
  batch_size = 250,
  sleep_duration = 1,
  verbose = TRUE
)
```

Arguments

tif	A data frame containing text data.
text_hierarchy	A character vector indicating the columns used to create row names.
api_token	Token for accessing the Hugging Face API.
api_url	The URL of the Hugging Face API endpoint (default is all-MiniLM-L6-v2).
query	An optional single text query for which embeddings are required.
dims	The dimension of the output embeddings.
batch_size	Number of rows in each batch sent to the API.
sleep_duration	Duration in seconds to pause between processing batches.
verbose	A boolean specifying whether to include progress bar

Value

A matrix containing embeddings, with each row corresponding to a text input.

Examples

```
## Not run:
tif <- data.frame(doc_id = c('1'), text = c("Hello world."))
embeddings <- api_huggingface_embeddings(tif,
                                         text_hierarchy = 'doc_id',
                                         api_token = api_token)

## End(Not run)
```

---

extract_date *Extract Date from HTML Content*

Description

This function attempts to extract a publication date from the HTML content of a web page using various methods such as JSON-LD, OpenGraph meta tags, standard meta tags, and common HTML elements.

Usage

```
extract_date(site)
```

Arguments

site	An HTML document (as parsed by xml2 or rvest) from which to extract the date.

Value

A data.frame with two columns: ‘date‘ and ‘source‘, indicating the extracted date and the source from which it was extracted (e.g., JSON-LD, OpenGraph, etc.). If no date is found, returns NA for both fields.

`nlp_build_chunks` *Build Chunks for NLP Analysis*

Description

This function processes a data frame for NLP analysis by dividing text into chunks and providing context. It generates chunks of text with a specified size and includes context based on the specified context size.

Usage

```
nlp_build_chunks(tif, text_hierarchy, chunk_size, context_size)
```

Arguments

`tif`	A data.table containing the text to be chunked.
`text_hierarchy`	A character vector specifying the columns used for grouping and chunking.
`chunk_size`	An integer specifying the size of each chunk.
`context_size`	An integer specifying the size of the context around each chunk.

Value

A data.table with the chunked text and their respective contexts.

Examples

```
# Creating a data frame
tif <- data.frame(doc_id = c('1', '1', '2'),
                  sentence_id = c('1', '2', '1'),
                  text = c("Hello world.",
                           "This is an example.",
                           "This is a party!"))

chunks <- nlp_build_chunks(tif,
                           chunk_size = 2,
                           context_size = 1,
                           text_hierarchy = c('doc_id', 'sentence_id'))
```

nlp_cast_tokens *Convert Token List to Data Frame*

Description

This function converts a list of tokens into a data frame, extracting and separating document and sentence identifiers if needed.

Usage

```
nlp_cast_tokens(tok)
```

Arguments

tok	A list where each element contains tokens corresponding to a document or a sentence.

Value

A data frame with columns for token name and token.

Examples

```
tokens <- list(c("Hello", "world", "."),
               c("This", "is", "an", "example", "." ),
               c("This", "is", "a", "party", "!"))
names(tokens) <- c('1.1', '1.2', '2.1')
dtm <- nlp_cast_tokens(tokens)
```

nlp_melt_tokens *Tokenize Data Frame by Specified Column(s)*

Description

This function tokenizes a data frame based on a specified token column and groups the data by one or more specified columns.

Usage

```
nlp_melt_tokens(
  df,
  melt_col = "token",
  parent_cols = c("doc_id", "sentence_id")
)
```

Arguments

df	A data frame containing the data to be tokenized.
melt_col	The name of the column in ‘df‘ that contains the tokens.
parent_cols	A character vector indicating the column(s) by which to group the data.

Value

A list of vectors, each containing the tokens of a group defined by the ‘by‘ parameter.

Examples

```
dtm <- data.frame(doc_id = as.character(c(1, 1, 1, 1, 1, 1, 1, 1)),
                  sentence_id = as.character(c(1, 1, 1, 2, 2, 2, 2, 2)),
                  token = c("Hello", "world", ".", "This", "is", "an", "example", "."))

tokens <- nlp_melt_tokens(dtm, melt_col = 'token', parent_cols = c('doc_id', 'sentence_id'))
```

---

nlp_split_paragraphs *Split Text into Paragraphs*

Description

Splits text from the 'text' column of a data frame into individual paragraphs, based on a specified paragraph delimiter.

Usage

```
nlp_split_paragraphs(tif, paragraph_delim = "\\n+")
```

Arguments

tif	A data frame with at least two columns: ‘doc_id‘ and ‘text‘.
paragraph_delim	A regular expression pattern used to split text into paragraphs.

Value

A data.table with columns: ‘doc_id‘, ‘paragraph_id‘, and ‘text‘. Each row represents a paragraph, along with its associated document and paragraph identifiers.

Examples

```
tif <- data.frame(doc_id = c('1', '2'),
                  text = c("Hello world.\n\nMind your business!",
                           "This is an example.n\nThis is a party!"))
paragraphs <- nlp_split_paragraphs(tif)
```

---

nlp_split_sentences *Split Text into Sentences*

Description

This function splits text from a data frame into individual sentences based on specified columns and handles abbreviations effectively.

Usage

```
nlp_split_sentences(
  tif,
  text_hierarchy = c("doc_id"),
  abbreviations = textpress::abbreviations
)
```

Arguments

tif	A data frame containing text to be split into sentences.
text_hierarchy	A character vector specifying the columns to group by for sentence splitting, usually 'doc_id'.
abbreviations	A character vector of abbreviations to handle during sentence splitting, defaults to textpress::abbreviations.

Value

A data.table with columns specified in 'by', 'sentence_id', and 'text'.

Examples

```
tif <- data.frame(doc_id = c('1'),
                  text = c("Hello world. This is an example. No, this is a party!"))
sentences <- nlp_split_paragraphs(tif)
```

nlp_tokenize_text *Tokenize Text Data (mostly) Non-Destructively*

Description

This function tokenizes text data from a data frame using the 'tokenizers' package, preserving the original text structure like capitalization and punctuation.

Usage

```
nlp_tokenize_text(
  tif,
  text_hierarchy = c("doc_id", "paragraph_id", "sentence_id")
)
```

Arguments

tif	A data frame containing the text to be tokenized and a document identifier in 'doc_id'.
text_hierarchy	A character string specifying grouping column.

Value

A named list of tokens, where each list item corresponds to a document.

Examples

```
tif <- data.frame(doc_id = c('1', '1', '2'),
                  sentence_id = c('1', '2', '1'),
                  text = c("Hello world.",
                           "This is an example.",
                           "This is a party!"))
tokens <- nlp_tokenize_text(tif, text_hierarchy = c('doc_id', 'sentence_id'))
```

sem_nearest_neighbors *Find Nearest Neighbors Based on Cosine Similarity*

Description

This function identifies the nearest neighbors of a given term or vector in a matrix based on cosine similarity.

Usage

```
sem_nearest_neighbors(x, matrix, n = 10)
```

Arguments

x	A character or numeric vector representing the term or vector.
matrix	A numeric matrix or a sparse matrix against which the similarity is calculated.
n	Number of nearest neighbors to return.

Value

A data frame with the ranks, terms, and their cosine similarity scores.

Examples

```
## Not run:
 api_token <- ''
 matrix <- api_huggingface_embeddings(tif,
                                      text_hierarchy = c('doc_id', 'sentence_id'),
                                      api_token = api_token)
 query <- api_huggingface_embeddings(query = "Where's the party at?",
                                     api_token = api_token)
 neighbors <- sem_nearest_neighbors(x = query, matrix = matrix)

## End(Not run)
```

---

sem_search_corpus *NLP Search Corpus*

Description

Searches a text corpus for specified patterns, with support for parallel processing.

Usage

```
sem_search_corpus(
  tif,
  text_hierarchy = c("doc_id", "paragraph_id", "sentence_id"),
  search,
  context_size = 0,
  is_inline = FALSE,
  highlight = c("<b>", "</b>"),
  cores = 1
)
```

Arguments

tif	A data frame or data.table containing the text corpus.
text_hierarchy	A character vector indicating the column(s) by which to group the data.
search	The search pattern or query.
context_size	Numeric, default 0. Specifies the context size, in sentences, around the found patterns.
is_inline	Logical, default FALSE. Indicates if the search should be inline.
highlight	A character vector of length two, default c('', ''). Used to highlight the found patterns in the text.
cores	Numeric, default 1. The number of cores to use for parallel processing.

Value

A data.table with the search results.

Examples

```
tif <- data.frame(doc_id = c('1', '1', '2'),
                  sentence_id = c('1', '2', '1'),
                  text = c("Hello world.",
                           "This is an example.",
                           "This is a party!"))
sem_search_corpus(tif, search = 'This is', text_hierarchy = c('doc_id', 'sentence_id'))
```

---

standardize_date *Standardize Date Format*

Description

This function attempts to parse a date string using multiple formats and standardizes it to "YYYY-MM-DD". It first tries ISO 8601 formats, and then common formats like ymd, dmy, and mdy.

Usage

```
standardize_date(date_str)
```

Arguments

date_str	A character string representing a date.

Value

A character string representing the standardized date in "YYYY-MM-DD" format, or NA if the date cannot be parsed.

web_scrape_urls *Scrape News Data from Various Sources*

Description

Function scrapes content of provided list of URLs.

Usage

```
web_scrape_urls(x, cores = 3)
```

Arguments

x	A character vector of URLs.
cores	The number of cores to use for parallel processing.

Value

A data frame containing scraped news data.

Examples

```
## Not run:
url <- 'https://www.nytimes.com/2024/03/25/nyregion/trump-bond-reduced.html'
article_tif <- web_scrape_urls(x = url, input = 'urls', cores = 1)

## End(Not run)
```

web_search *Process search results from multiple search engines*

Description

This function allows you to query different search engines (DuckDuckGo, Bing, Yahoo News), retrieve search results, and filter them based on predefined patterns.

Usage

```
web_search(
  search_term,
  search_engine,
  num_pages = 1,
  time_filter = NULL,
  insite = NULL,
  intitle = FALSE
)
```

Arguments

search_term	The search query as a string.
search_engine	The search engine to use: "DuckDuckGo", "Bing", or "Yahoo News".
num_pages	The number of result pages to retrieve (default: 1).
time_filter	Optional time filter ("week", "month", "year").
insite	Restrict search to a specific domain (not supported for Yahoo).
intitle	Search within the title (relevant for DuckDuckGo and Bing).

Value

A ‘data.table‘ containing search engine results with columns ‘search_engine‘ and ‘raw_url‘.

Index

∗ **datasets**
abbreviations, 6
.decode_duckduckgo_urls, 2
.extract_links, 3
.get_site, 3
.process_bing, 4
.process_duckduckgo, 4
.process_yahoo, 5

abbreviations, 6
api_huggingface_embeddings, 6

extract_date, 7

nlp_build_chunks, 8
nlp_cast_tokens, 9
nlp_melt_tokens, 9
nlp_split_paragraphs, 10
nlp_split_sentences, 11
nlp_tokenize_text, 12

sem_nearest_neighbors, 12
sem_search_corpus, 13
standardize_date, 14

web_scrape_urls, 15
web_search, 15