Package 'textpress'

March 17, 2026

Type Package

Title A Lightweight and Versatile NLP Toolkit

Version 1.1.1

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

Description A toolkit for web scraping, modular NLP pipelines, and text preparation for large language models. Organized around four core actions: fetching, reading, processing, and searching. Covers the full pipeline from raw web data acquisition to structural text processing and BM25 indexing. Supports multiple retrieval strategies including regex, dictionary matching, and ranked keyword search. Pipe-friendly with no heavy dependencies; all outputs are plain data frames or data.tables.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5)

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

Suggests SnowballC (>= 0.7.0), DT, dplyr

RoxygenNote 7.3.3

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 2026-03-17 22:40:02 UTC

Contents

`abbreviations` *Common abbreviations for NLP*

Description

Common abbreviations for NLP (e.g. sentence splitting). Named list; used by `nlp_split_sentences`.

Usage

```
abbreviations
```

Format

A named list with the following components:

`abbreviations` A character vector of common abbreviations, including titles, months, and standard abbreviations.

Source

Internally compiled linguistic resource.

dict_generations *Demo dictionary of generation-name variants for NER*

Description

A small dictionary of generational cohort terms (Greatest, Silent, Boomers, Gen X, Millennials, Gen Z, Alpha, etc.) and spelling/variant forms, for use with `search_dict`. Built in-package (no `data()`).

Usage

```
dict_generations
```

Format

A data frame with columns `variant` (surface form to match), `TermName` (standardized label), `is_cusp` (logical), `start` and `end` (birth year range; Pew definitions where applicable, see https://github.com/jaytimm/AmericanGenerations/blob/main/data/pew-generations.csv).

Examples

```
head(dict_generations)
# use as term list: search_dict(corpus, by = "doc_id", terms = dict_generations$variant)
```

dict_political *Demo dictionary of political / partisan term variants for NER*

Description

A small dictionary of political party and ideology terms (Democrat, Republican, MAGA, Liberal, Conservative, Christian Nationalist, White Supremacist, etc.) and spelling/variant forms, for use with `search_dict`. Built in-package (no `data()`).

Usage

```
dict_political
```

Format

A data frame with columns `variant` (surface form to match) and `TermName` (standardized label).

Examples

```
head(dict_political)
# search_dict(corpus, by = "doc_id", terms = dict_political$variant)
```

fetch_urls *Fetch URLs from a search engine*

Description

Web (general). Queries a search engine and returns result URLs. Use `read_urls` to get content from these URLs.

Usage

```
fetch_urls(query, n_pages = 1, date_filter = "w")
```

Arguments

`query`	Search query string.
`n_pages`	Number of search result pages to fetch (default 1). ~30 results per page.
`date_filter`	Recency filter: `"d"` (day), `"w"` (week), `"m"` (month), or `"none"` (default `"w"`).

Value

A `data.table` with columns `search_engine`, `url`, `is_excluded`, and optionally `path_depth`.

Examples

```
## Not run:
urls_dt <- fetch_urls("R programming nlp", n_pages = 1)
urls_dt$url

## End(Not run)
```

fetch_wiki_refs *Fetch external citation URLs from Wikipedia article(s)*

Description

Wikipedia. Extracts external citation URLs from the References section of one or more Wikipedia article URLs. Use `read_urls` to scrape content from those URLs.

Usage

```
fetch_wiki_refs(url, n = NULL)
```

Arguments

`url`	Character vector of full Wikipedia article URLs (e.g. from `fetch_wiki_urls`).
`n`	Maximum number of citation URLs to return per source page. Default `NULL` returns all; use a number (e.g. `10`) to limit.

Value

For one URL, a `data.table` with columns `source_url`, `ref_id`, and `ref_url`. For multiple URLs, a named list of such data.tables (names are the Wikipedia article titles); elements are `NULL` for pages with no refs.

Examples

```
## Not run:
wiki_urls <- fetch_wiki_urls("January 6 Capitol attack")
refs_dt <- fetch_wiki_refs(wiki_urls[1])           # single URL: data.table
refs_list <- fetch_wiki_refs(wiki_urls[1:3])      # multiple: named list
articles <- read_urls(refs_dt$ref_url)

## End(Not run)
```

`fetch_wiki_urls` *Fetch Wikipedia page URLs by search query*

Description

Wikipedia. Uses the MediaWiki API to get Wikipedia article URLs matching a search phrase. Does not search your local corpus. Use `read_urls` to get article content from these URLs.

Usage

```
fetch_wiki_urls(query, limit = 10)
```

Arguments

`query`	Search phrase (e.g. "117th Congress").
`limit`	Number of page URLs to return (default 10).

Value

Character vector of full Wikipedia article URLs.

Examples

```
## Not run:
wiki_urls <- fetch_wiki_urls("January 6 Capitol attack")
corpus <- read_urls(wiki_urls[1])

## End(Not run)
```

nlp_cast_tokens *Convert token list to data frame*

Description

Convert the token list returned by nlp_tokenize_text into a data frame (long format), with identifiers and optional spans.

Usage

```
nlp_cast_tokens(tok)
```

Arguments

tok	List with at least a tokens element (and optionally spans), e.g. output of nlp_tokenize_text(..., include_spans = TRUE).

Value

Data frame with columns for unit id, token, and optionally start/end spans.

Examples

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

nlp_index_tokens *Build a BM25 index for ranked keyword search*

Description

Build a weighted BM25 index for ranked keyword search. Creates a searchable index from a named list of token vectors. The unit-id column name is taken from attr(tokens, "id_col") when present (e.g. from nlp_tokenize_text), else "uid".

Usage

```
nlp_index_tokens(tokens, k1 = 1.2, b = 0.75, stem = FALSE)
```

Arguments

tokens	Named list of character vectors (e.g. from nlp_tokenize_text).
k1	BM25 saturation parameter (default 1.2).
b	BM25 length normalization (default 0.75).
stem	Logical. If TRUE, stem tokens (default FALSE).

Value

Data.table with unit-id column, token, score; attr(., "id_col") set for search_index.

---

nlp_roll_chunks *Roll units into fixed-size chunks with optional context*

Description

Roll units (e.g. sentences) into fixed-size chunks with optional context (RAG-style). Groups consecutive rows at the finest by level into chunks and optionally adds surrounding context.

Usage

```
nlp_roll_chunks(corpus, by, chunk_size, context_size, id_col = "uid")
```

Arguments

corpus	Data frame or data.table with a text column and the identifier columns specified in by.
by	Character vector of identifier columns that define the text unit (e.g. doc_id or c("url", "node_id")). The last column is the level rolled into chunks (e.g. sentences).
chunk_size	Integer. Number of units per chunk.
context_size	Integer. Number of units of context around each chunk.
id_col	Character. Name of the column holding the unique chunk id (default "uid").

Value

Data.table with id_col (pasted grouping + chunk index), grouping columns from by, and text (chunk plus context). Unique on by[1] and text.

Examples

```
corpus <- 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_roll_chunks(corpus, by = c('doc_id', 'sentence_id'),
                          chunk_size = 2, context_size = 1)
```

nlp_split_paragraphs *Split text into paragraphs*

Description

Break documents into structural blocks (paragraphs). Splits text from the `text` column by a paragraph delimiter.

Usage

```
nlp_split_paragraphs(corpus, by = c("doc_id"), paragraph_delim = "\\n+")
```

Arguments

`corpus`	Data frame or data.table with a `text` column and the identifier columns specified in `by`.
`by`	Character vector of identifier columns that define the text unit (e.g. `doc_id` or `c("url", "node_id")`). Default `c("doc_id")`.
`paragraph_delim`	Regular expression used to split text into paragraphs (default `"\\n+"`).

Value

Data.table with the `by` columns, `paragraph_id`, and `text`. One row per paragraph.

Examples

```
corpus <- 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(corpus)
```

nlp_split_sentences *Split text into sentences*

Description

Refine blocks into individual sentences. Splits text into sentences with accurate start/end offsets; handles abbreviations (Wikipedia and web optimized).

Usage

```
nlp_split_sentences(
  corpus,
  by = c("doc_id"),
  abbreviations = textpress::abbreviations
)
```

Arguments

corpus	Data frame or data.table with a text column and the identifier columns specified in by.
by	Character vector of identifier columns that define the text unit (e.g. doc_id or c("url", "node_id")). Default c("doc_id").
abbreviations	Character vector of abbreviations to protect (default textpress::abbreviations).

Value

Data.table with by columns, sentence_id, text, start, end.

---

nlp_tokenize_text *Tokenize text into a clean token stream*

Description

Normalize text into a clean token stream. Tokenizes corpus text, preserving structure (capitalization, punctuation). The last column in by determines the tokenization unit.

Usage

```
nlp_tokenize_text(
  corpus,
  by = c("doc_id", "paragraph_id", "sentence_id"),
  id_col = "uid",
  include_spans = TRUE,
  method = "word"
)
```

Arguments

corpus	Data frame or data.table with a text column and the identifier columns specified in by.
by	Character vector of identifier columns that define the text unit (e.g. doc_id or c("url", "node_id")). Default c("doc_id", "paragraph_id", "sentence_id"). The last column is the finest granularity.
id_col	Character. Name of the column (and list names) used for the unit id (default "uid").
include_spans	Logical. Include start/end character spans for each token (default TRUE).
method	Character. "word" or "biber".

Value

Named list of tokens; or list of tokens and spans if include_spans = TRUE.

Examples

```
corpus <- 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(corpus, by = c('doc_id', 'sentence_id'))
```

`read_urls` *Read content from URLs*

Description

Input: character vector of URLs. Output: structured data frame (one row per node: headings, paragraphs, lists). Like `read_csv` or `read_html`: bring an external resource into R. Follows `fetch_urls` or `fetch_wiki_urls` in the pipeline—fetch gets locations, read gets text. Wikipedia uses high-fidelity selectors; use `parent_heading` to see which section each node belongs to. External links and empty text rows are omitted; optionally exclude References/See also/Bibliography/Sources sections for wiki URLs.

Usage

```
read_urls(
  x,
  cores = 1,
  detect_boilerplate = TRUE,
  remove_boilerplate = TRUE,
  exclude_wiki_refs = TRUE
)
```

Arguments

`x`	Character vector of URLs.
`cores`	Number of cores for parallel requests (default 1).
`detect_boilerplate`	Logical. Detect boilerplate (e.g. sign-up, related links).
`remove_boilerplate`	Logical. If `detect_boilerplate` is `TRUE`, remove boilerplate rows; if `FALSE`, keep them and add `is_boilerplate`.
`exclude_wiki_refs`	Logical. For Wikipedia URLs only, drop nodes whose `parent_heading` is References, See also, Bibliography, or Sources. Default `TRUE`.

Value

A list with `text` (node-level data: `doc_id`, `url`, `node_id`, `parent_heading`, `text`, and optionally `type`, `is_boilerplate`) and `meta` (one row per URL: `doc_id`, `url`, `h1_title`, `date`, `source`). `doc_id` is an integer key (1 to number of distinct URLs) in first-appearance order of the input vector.

Examples

```
## Not run:
urls <- fetch_urls("R programming", n_pages = 1)$url
out <- read_urls(urls[1:3], cores = 1)
nodes <- out$text
meta <- out$meta

## End(Not run)
```

`search_dict` *Exact phrase / MWE matcher*

Description

Exact phrase or multi-word expression (MWE) matcher; no partial-match risk. Tokenizes corpus, builds n-grams, and exact-joins against `terms`. Word boundaries respected. N-gram range is set from the min and max word count of `terms`. Good for deterministic entity extraction (e.g. before an LLM call).

Usage

```
search_dict(corpus, by = c("doc_id"), terms)
```

Arguments

`corpus`	Data frame or data.table with a `text` column and the identifier columns specified in `by`.
`by`	Character vector of identifier columns that define the text unit (e.g. `doc_id` or `c("url", "node_id")`). Default `c("doc_id")`.
`terms`	Character vector of terms or phrases to match exactly. N-gram range derived from word counts of `terms`.

Value

Data.table with `id`, `start`, `end`, `n`, `ngram`, `term`.

Examples

```
corpus <- data.frame(doc_id = "1", text = "Gen Z and Millennials use social media.")
search_dict(corpus, by = "doc_id", terms = c("Gen Z", "Millennials", "social media"))
```

search_index *Search the BM25 index*

Description

BM25 ranked retrieval. Search the index produced by `nlp_index_tokens` with a keyword query. The unit-id column in results is taken from `attr(index, "id_col")` when present, else `"uid"`.

Usage

```
search_index(index, query, n = 10, stem = FALSE)
```

Arguments

`index`	Object created by `nlp_index_tokens`.
`query`	Character string (keywords).
`n`	Number of results to return (default 10).
`stem`	Logical; must match the setting used during indexing (default `FALSE`).

Value

Data.table with columns `query`, `method` ("bm25"), `score` (3 significant figures), and the unit-id column (e.g. `uid`), ranked by score.

search_regex *Search corpus by regex*

Description

Search corpus by regex. Specific strings/patterns; good for KWIC-style results. Returns matches with optional highlighting.

Usage

```
search_regex(corpus, query, by = c("doc_id"), highlight = c("<b>", "</b>"))
```

Arguments

`corpus`	Data frame or data.table with a `text` column and the identifier columns specified in `by`.
`query`	Search pattern (regex).
`by`	Character vector of identifier columns that define the text unit (e.g. `doc_id` or `c("url", "node_id")`). Default `c("doc_id")`.
`highlight`	Length-two character vector for wrapping matches (default `c("<b>", "</b>")`).

Value

Data.table with id, by columns, text, start, end, pattern.

---

search_vector *Semantic search by cosine similarity*

Description

Semantic search by cosine similarity. Returns top-n matches from an embedding matrix for one or more query vectors. Subject-first: embeddings (haystack) then query (needle). Pipe-friendly.

Usage

```
search_vector(embeddings, query, n = 10)
```

Arguments

embeddings	Numeric matrix of embeddings; rows are searchable units (row names used as identifiers).
query	Row name in embeddings, a numeric vector (single query), or a numeric matrix (multiple queries).
n	Number of results to return per query (default 10).

Value

Data frame with columns query, method ("cosine"), score (3 significant figures), and the unit-id column (e.g. uid). For multiple queries, a list of such data frames.

---

util_fetch_embeddings *Fetch embeddings from a Hugging Face inference endpoint*

Description

Builds a numeric matrix of embeddings for each text unit. Row names come from by (data frame) or from names(corpus) / corpus (character vector). Use the result with search_vector for semantic search.

Usage

```
util_fetch_embeddings(
  corpus,
  by = NULL,
  api_token,
 api_url = "https://router.huggingface.co/hf-inference/models/BAAI/bge-small-en-v1.5"
)
```

Arguments

corpus	A data frame with text and by columns, or a character vector of texts. If a named character vector, names become row names; if unnamed, the strings themselves are used as row names.
by	Character vector of identifier columns; required when corpus is a data frame (row names), ignored when corpus is a character vector.
api_token	Hugging Face API token.
api_url	Inference endpoint URL (default BAAI/bge-small-en-v1.5).

Value

Numeric matrix with row names (unit ids).

Index