Package: NatChat (via r-universe)

June 6, 2026

Type Package

Title Chatting with Nature Journals Current Issue using a local Language Model

Version 1.1.0

Description The goal of NatChat is to provide fast, local-language-model-powered summaries of articles from the current issues of Nature journals, making cutting-edge science more accessible and digestible. The package includes functions to identify available journals, retrieve articles from the latest issues, construct prompts for summarization and generate natural-language summaries using large language models (LLMs) via the 'ollama' interface. Output can be formatted for use in markdown tables, reports, or summaries. This tool is particularly useful for researchers, educators, and clinicians who want to stay up to date with the latest literature across multiple disciplines.

License GPL (>= 3)

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports dplyr (>= 1.1.4), ollamar (>= 1.2.2), readr (>= 2.1.5), rvest (>= 1.0.4), tibble (>= 3.2.1), tinytable (>= 0.8.0), tools (>= 4.4.0), utils (>= 4.4.0), xml2 (>= 1.3.8)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/monahton/NatChat, https://monahton.github.io/NatChat/

BugReports https://github.com/monahton/NatChat/issues

Config/pak/sysreqs libicu-dev libxml2-dev libssl-dev libx11-dev

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

Date/Publication 2025-07-11 12:52:10 UTC

RemoteUrl https://github.com/monahton/NatChat

RemoteRef main

RemoteSha 08d85dff72064ac628b10ee7132c97f675153fc1

Contents

`add_prompt` *Add Summarization Prompts to Articles*

Description

Adds a prompt column to a data frame of scientific articles, suitable for use with a language model summarization tool. Prompts are generated using the `build_prompt()` function, based on article titles and abstracts.

Usage

```
add_prompt(article, ...)
```

Arguments

`article`	A data frame or tibble containing at least the columns `"title"` and `"abstract"`.
`...`	Additional arguments passed to `build_prompt()`, such as `nsentences`.

Details

The function checks for the presence of required columns before proceeding. It applies `build_prompt()` row-wise to generate summarization prompts.

This function is typically used after retrieving articles via `get_articles()` or `get_article()`, to prepare data for summarization by a language model (e.g., using `ollama::generate()`).

Value

A modified data frame of class `article_prompt`, including an additional column `"prompt"` containing structured summarization prompts.

Examples

```
## Not run:
papers <- get_articles("Nature Medicine")
papers_with_prompts <- add_prompt(papers, nsentences = 3)
cat(papers_with_prompts$prompt[1])

## End(Not run)
```

`add_summary` *Generate Summaries for Articles*

Description

This function generates concise summaries for each article in a given data frame, using the specified language model (LLM). The summaries are generated based on the prompts previously added to the data frame.

Usage

```
add_summary(article, model = "llama3.1", host = NULL)
```

Arguments

`article`	A data frame or tibble containing at least a `"prompt"` column, which is created using `build_prompt()` or `add_prompt()`.
`model`	Character string. The name of the LLM model to use for generating summaries. Default is `"llama3.1"`.
`host`	Character string or NULL. The host to be used for the `ollamar::generate` function. Default is NULL.

Details

The function iterates over each article and generates a summary using the specified LLM model. A progress bar is shown to track the summarization process. Any newlines within the text fields are removed to ensure clean formatting. This function is typically used after applying `add_prompt()` to prepare a dataset for summarization.

The progress bar updates for each article as the summaries are being generated. The final `summary` column will contain the output of the summarization process, ready for further processing or analysis.

Value

A modified data frame of class `article_summary`, including an additional column `"summary"` containing the generated summaries.

Examples

```
## Not run:
papers <- get_article(journal = "Nature Medicine")
papers_with_prompts <- add_prompt(papers, nsentences = 3)
summarized_papers <- add_summary(papers_with_prompts)

## End(Not run)
```

---

build_prompt *Construct a Text Prompt for Summarizing an Article*

Description

Builds a structured prompt from an article's title and abstract, designed for input to a language model. The prompt emphasizes extracting key findings, methodology, and tone, and is customizable via instructions.

Usage

```
build_prompt(
  title,
  abstract,
  nsentences = 3L,
  instructions = c("You will receive a paper's title and abstract as input.",
    "Provide a concise summary with exactly the number of sentences specified.",
    "Do not include introductory phrases or preamble text.",
    "Start directly with the summary; avoid any framing statements.",
    "Focus on key findings, especially of last two sentences of the abstract.",
   "If the abstract is missing, reply explicitly with 'Abstract is not available.'",
    "Highlight any novel contributions, claims, or innovations in the abstract.",
   "Mention main methods or datasets only if explicitly stated in the abstract.",

    "Indicate the strength and tone of evidence.",
    "Optionally,add a one-sentence lay summary for a non-specialist audience.")
)
```

Arguments

`title`	Character string. The title of the article.
`abstract`	Character string. The abstract of the article. If unavailable, include a default message.

nsentences	Integer. The number of sentences required in the summary. Default is 3. Must be a positive whole number.
instructions	Character vector. A set of instructions guiding the summarization. Defaults to a structured template emphasizing main findings, methods, novelty, and tone.

Details

The generated prompt follows a structured format:

- Lists the instructions (customizable via `instructions`).
- States the number of summary sentences required (`nsentences`).
- Embeds the article title and abstract.
- If the abstract is missing or not available, the prompt explicitly states this.

The default `instructions` vector can be modified to adapt the tone or focus of the summary, such as prioritizing method, dataset, confidence tone, or accessibility for non-specialists.

Value

A character string representing a structured prompt for use with a language model summarization tool.

Examples

```
title <- "Deep Learning for Genomic Data Analysis"
abstract <- "This study explores deep learning in diverse tasks highlighting predictive accuracy."
prompt <- build_prompt(title, abstract, nsentences = 3)
cat(prompt)
```

---

check_ollama *Check Ollama Installation and List Available Models*

Description

Verify whether the Ollama backend is properly installed and running by testing the connection. If successful, retrieve and print the list of available local models.

Usage

```
check_ollama(verbose = TRUE)
```

Arguments

verbose	Logical. Should informative messages and the list of available models be printed to the console? Default is TRUE.

Details

The function calls `ollamar::test_connection()` to verify the Ollama service is running, then calls `ollamar::list_models()` to check for installed local models. If verbose, it prints detailed diagnostic messages and the model names.

Value

Logical TRUE if Ollama is installed, running, and at least one model is available; otherwise FALSE.

Examples

```
## Not run:
check_ollama()

## End(Not run)
```

---

filter_articles *Filter Articles Based on Whitelist Terms*

Description

This function filters a data frame of articles, retaining only those that contain at least one of the specified whitelist terms in either the title or abstract. This allows for easy extraction of articles relevant to a set of predefined topics.

Usage

```
filter_articles(article, whitelist_terms)
```

Arguments

`article`	A data frame or tibble containing at least the `"title"` and `"abstract"` columns.
`whitelist_terms`	A character vector of terms that are used to filter articles by matching the title or abstract.

Details

The function combines the "title" and "abstract" columns into a single text string and uses regular expression matching to search for the presence of any of the specified whitelist terms. The search is case-insensitive. Only the articles that match one or more of the whitelist terms will be retained in the output data frame.

Value

A filtered data frame containing only articles where at least one of the whitelist terms is found in the title or abstract.

Examples

```
## Not run:
papers <- get_article(journal = "Nature Medicine")
filtered_papers <- filter_articles(papers, whitelist_terms = c("CRISPR", "gene therapy"))

## End(Not run)
```

`get_articles` *Retrieve Articles from a Nature Journal's Current Issue*

Description

This function scrapes articles from the current issue of a specified Nature journal, extracting article titles, URLs, and abstracts with robust fallback handling.

Usage

```
get_articles(journal, article_selector = ".c-card.c-card--flush",
             title_selector = "h3 a", url_selector = "h3 a",
             abstract_selector = ".c-card__summary", verbose = FALSE)
```

Arguments

`journal`	Character string. The full name of the Nature journal (e.g., "Nature Biotechnology", "Nature Medicine").
`article_selector`	Character string. CSS selector for locating articles on the journal's webpage. Default is ".c-card.c-card–flush".
`title_selector`	Character string. CSS selector for extracting article titles. Default is "h3 a".
`url_selector`	Character string. CSS selector for extracting article URLs. Default is "h3 a".
`abstract_selector`	Character string. CSS selector for extracting article abstracts. Default is ".c-card__summary".
`verbose`	Logical. If TRUE, prints messages about progress and internal steps. Default is FALSE.

Details

The journal argument is matched (case-insensitively) against available entries from `nat_journals()`. If not found, an informative error is thrown. Abstracts that are missing are replaced with "Abstract not available". If titles, URLs, and abstracts differ in length, they are truncated to the shortest length with a warning.

Value

A tibble with columns: `title`, `url`, `abstract`, and `source`. If no articles are found, returns an empty tibble.

Examples

```
get_articles("Nature Biotechnology")
get_articles("Nature Reviews Genetics", verbose = TRUE)
```

---

nat_journals *List Available Nature Journals and Their Slugs*

Description

This function returns a data frame of *Nature* journals supported by the `Natchat` package, including their full names and URL slugs (used in links or programmatic access). Optionally, users can provide a journal name to filter and display only the matching journal and its slug.

Usage

```
nat_journals(journal = NULL)
```

Arguments

`journal`	Optional character string. The full name of a Nature journal (case-insensitive) to filter the list. If `NULL` (default), returns the full table of available journals and slugs.

Details

- The `slug` corresponds to the subdirectory used in Nature URLs (e.g., `"https://www.nature.com/nbt/"` for *Nature Biotechnology*).
- Journal name matching is case-insensitive and supports exact matches only (no partial or fuzzy matching).

Value

A tibble with two columns:

journal The full name of the journal

slug The short URL identifier (slug) used in Nature journal web addresses

Examples

```
nat_journals()
nat_journals("Nature Medicine")
nat_journals("nature biotechnology")
```

save_report *Save Report as CSV and HTML*

Description

Save a data frame of article metadata as both a CSV file and an HTML file with a markdown-styled table. Options are provided to control file format outputs and verbosity.

Usage

```
save_report(input, filename, save_csv, save_html, title, cols, width, verbose, outdir)
```

Arguments

input	A data frame containing article data (e.g., "title", "summary", "url").
filename	A character string specifying the base filename.
save_csv	Logical. Save output as a CSV file? Default is TRUE.
save_html	Logical. Save output as an HTML file? Default is TRUE.
title	A character string specifying the HTML page title. Default is "Article Summary Report".
cols	A character vector of column names to include in the output. Default is c("title", "summary").
width	A numeric vector of column widths for the HTML table. Default is c(1, 3).
verbose	Logical. Should messages be printed? Default is TRUE.
outdir	A character string specifying the directory to save files. Default is current working directory ".".

Details

A timestamp is appended to the base filename to uniquely identify each output. If both save_csv and save_html are FALSE, no files are saved and a message is issued (if verbose = TRUE).

Value

Files are written to disk in the specified formats. The function returns (invisibly) a list of saved file paths.

Examples

```
## Not run:
papers <- get_articles(journal = "Nature Medicine")
papers_with_summary <- add_summary(papers)
save_report(papers_with_summary, save_csv = TRUE, save_html = TRUE)

## End(Not run)
```

summarize_journal *Summarize a Nature Journal Issue*

Description

Retrieve and summarize abstracts from the current issue of a selected Nature Portfolio journal using a local LLM and save the output as CSV and/or HTML. Optionally filter the articles by a set of whitelist terms.

Usage

```
summarize_journal(journal, filename, outdir, model,save_csv,save_html,verbose, whitelist)
```

Arguments

journal	A character string indicating the name of the supported Nature journal (e.g., "Nature Biotechnology").
filename	A character string specifying the base filename for saving the report. Default is "natchat_summary".
outdir	A character string specifying the directory to save output files. Default is current working directory ".".
model	A character string specifying the local Ollama model to use for summarization (e.g., "llama3:instruct").
save_csv	Logical. Save the results as a CSV file? Default is TRUE.
save_html	Logical. Save the results as an HTML file? Default is TRUE.
verbose	Logical. Should informative messages be printed to the console? Default is TRUE.
whitelist	Optional character vector of terms used to filter articles based on title and abstract. Default is NULL (no filtering).

Details

This function is a convenience wrapper around `get_articles()`, `add_prompt()`, `add_summary()`, and `save_report()`. It scrapes the current issue, optionally filters articles using a whitelist of terms, summarizes abstracts using a local LLM, and exports the result.

Value

Invisibly returns a list of file paths (if saved). Generates summarized article metadata and optionally saves it to disk.

Examples

```
## Not run:
summarize_journal(
  journal = "Nature Medicine",
  model = "llama3",
  whitelist = c("CRISPR", "gene therapy"),
  save_csv = TRUE,
  save_html = TRUE
)

## End(Not run)
```

`tt_article` *Format Articles as a TinyTable*

Description

This function formats a data frame of articles into a markdown-styled table using the `tinytable` package. It allows you to select specific columns from the article data and adjust the column widths for a clean, formatted output.

Usage

```
tt_article(article, cols = c("title", "summary"), width = c(1, 3))
```

Arguments

`article`	A data frame containing article information (e.g., "title", "summary", "url").
`cols`	A character vector of column names to include in the table. Default is `c("title", "summary")`.
`width`	A numeric vector specifying column widths. Default is `c(1, 3)`.

Details

The function first ensures that the specified columns exist in the data frame and that the length of `cols` matches the length of `width`. It then formats the "title" column as a markdown link, using the article's URL (if provided), and selects the requested columns to be displayed in the table. The resulting table is formatted as markdown for easy integration into markdown environments.

Value

A formatted markdown table as a character string, ready to be displayed in markdown-supported environments.

Examples

```
## Not run:
papers <- get_articles(journal = "Nature Biotechnology")
papers_with_summary <- add_summary(papers)
tt_article(papers_with_summary)

## End(Not run)
```

Index