Package: Goodreader (via r-universe)

October 25, 2024

Title Scrape and Analyze 'Goodreads' Book Data

Version 0.1.2

Description A comprehensive toolkit for scraping and analyzing book data from <https://www.goodreads.com/>. This package provides functions to search for books, scrape book details and reviews, perform sentiment analysis on reviews, and conduct topic modeling. It's designed for researchers, data analysts, and book enthusiasts who want to gain insights from 'Goodreads' data.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Imports cld2, dplyr, ggplot2, httr, lubridate, magrittr, parallel, purrr, rvest, rlang, stringr, tidyr, tidytext, tm, topicmodels, utils, wordcloud2

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/chaoliu-cl/Goodreader, http://liu-chao.site/Goodreader/

BugReports https://github.com/chaoliu-cl/Goodreader/issues

NeedsCompilation no

Author Chao Liu [aut, cre, cph] (<https://orcid.org/0000-0002-9979-8272>)

Maintainer Chao Liu <chaoliu@cedarville.edu>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2024-10-24 10:31:07 UTC

Contents

analyze_sentiment . . . 2
average_book_sentiment . . . 3
fit_lda . . . 4
gen_topic_clouds . . . 5
get_author_info . . . 6
get_book_ids . . . 7
get_book_summary . . . 7
get_format_info . . . 8
get_genres . . . 9
get_num_pages . . . 10
get_published_time . . . 10
get_rating_distribution . . . 11
model_topics . . . 12
plot_topic_heatmap . . . 13
plot_topic_prevalence . . . 13
plot_topic_terms . . . 14
preprocess_reviews . . . 15
replace_special_chars . . . 16
scrape_books . . . 16
scrape_reviews . . . 17
search_goodreads . . . 18
sentiment_histogram . . . 19
sentiment_trend . . . 20
top_terms . . . 21

Index **22**

`analyze_sentiment` *Perform sentiment analysis on book reviews with negation handling*

Description

This function takes the output from scrape_reviews and performs sentiment analysis, including basic negation scope detection.

Usage

```
analyze_sentiment(reviews_df, lexicon = "afinn")
```

Arguments

`reviews_df`	A data frame containing the output from scrape_reviews.
`lexicon`	The sentiment lexicon to use. Options are "afinn", "bing", or "nrc".

Value

A data frame with sentiment scores for each review.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Run the scrape_reviews function
reviews <- scrape_reviews(temp_file, num_reviews = 5, use_parallel = FALSE)

# Check if reviews were successfully scraped
if (nrow(reviews) > 0) {
  # Perform sentiment analysis
  sentiment_results <- analyze_sentiment(reviews, lexicon = "afinn")

  # Display the first few rows of the results
  print(head(sentiment_results))
} else {
  cat("No reviews found. Cannot perform sentiment analysis.\n")
}

# Clean up: remove the temporary file
file.remove(temp_file)
```

---

average_book_sentiment

Calculate average sentiment score per book

Description

This function calculates the average sentiment score for each book.

Usage

```
average_book_sentiment(sentiment_df)
```

Arguments

sentiment_df	A data frame containing the output from analyze_sentiment.

Value

A data frame with average sentiment scores for each book.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)
```

```
# Run the scrape_reviews function
reviews <- scrape_reviews(temp_file, num_reviews = 5, use_parallel = FALSE)

# Check if reviews were successfully scraped
if (nrow(reviews) > 0) {
  # Perform sentiment analysis
  sentiment_results <- analyze_sentiment(reviews, lexicon = "afinn")

  # Calculate average sentiment score per book
  avg_senti <- average_book_sentiment(sentiment_results)

  # Display the results
  print(avg_senti)
} else {
  cat("No reviews found. Cannot calculate average sentiment.\n")
}

# Clean up: remove the temporary file
file.remove(temp_file)
```

`fit_lda` *Perform topic modeling on preprocessed reviews*

Description

This function performs LDA topic modeling on the preprocessed reviews.

Usage

```
fit_lda(dtm, k, method = "Gibbs")
```

Arguments

`dtm`	A document-term matrix
`k`	The number of topics to extract
`method`	The method to use for fitting the model (default: Gibbs)

Value

An LDA model

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Scrape reviews
```

```
reviews <- scrape_reviews(temp_file, num_reviews = 5, use_parallel = FALSE)

# Preprocess the reviews
preprocessed <- preprocess_reviews(reviews, english_only = TRUE)

# Fit LDA model
lda_model <- fit_lda(preprocessed$dtm, k = 2)

# Print model summary
print(lda_model)

# Clean up: remove the temporary file
file.remove(temp_file)
```

`gen_topic_clouds` *Create word cloud for topics*

Description

This function creates a word cloud for each topic.

Usage

```
gen_topic_clouds(model_output, n = 50)
```

Arguments

`model_output`	The output from model_topics function
`n`	The number of top terms to include in the word cloud

Value

A list of ggplot objects, where each element represents a word cloud for a topic.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Scrape reviews
reviews <- scrape_reviews(temp_file, num_reviews = 30, use_parallel = FALSE)

# Model topics
topic_results <- model_topics(reviews, num_topics = 3, num_terms = 50, english_only = TRUE)

# Generate word clouds for each topic
wordcloud_plots <- gen_topic_clouds(topic_results, n = 20)
```

```
# Display the word cloud for the first topic
if (interactive()) {
  print(wordcloud_plots[[1]])
}

# Clean up: remove the temporary file
file.remove(temp_file)
```

`get_author_info` *Get Author Information from Goodreads*

Description

This function takes a file path containing Goodreads book IDs and retrieves the author information for each book.

Usage

```
get_author_info(file_path)
```

Arguments

`file_path`	A character string specifying the path to the file containing Goodreads book IDs.

Value

A named list where each element contains the author information for a book.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)
# Run the function
author_info <- get_author_info(temp_file)
print(author_info)
# Clean up: remove the temporary file
file.remove(temp_file)
```

get_book_ids *Save Book IDs to a Text File*

Description

This function retrieves the book_id values from the input_data and saves them to a specified text file.

Usage

```
get_book_ids(input_data, file_name)
```

Arguments

input_data	A data frame containing a column named book_id.
file_name	A string specifying the name of the text file to save the book_id values.

Value

No return value, the function writes the book_id values to a text file.

Examples

```
# Create sample data
books <- data.frame(title = c("Hamlet", "The Hunger Games", "Jane Eyre"),
                    book_id = c("1420", "2767052", "10210")
                    )
# Create a temporary file path
temp_file <- file.path(tempdir(), "bookids.txt")

# Run the function
get_book_ids(books, temp_file)

# Clean up: remove the temporary file
file.remove(temp_file)
```

get_book_summary *Get Book Summary from Goodreads*

Description

This function takes a file path containing Goodreads book IDs and retrieves the summary for each book.

Usage

```
get_book_summary(file_path)
```

Arguments

file_path	A character string specifying the path to the file containing Goodreads book IDs.

Value

A named list where each element contains the summary for a book.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)
# Run the function
book_summary <- get_book_summary(temp_file)
print(book_summary)
# Clean up: remove the temporary file
file.remove(temp_file)
```

---

get_format_info *Get Format Information from Goodreads*

Description

This function takes a file path containing Goodreads book IDs and retrieves the format information for each book.

Usage

```
get_format_info(file_path)
```

Arguments

file_path	A character string specifying the path to the file containing Goodreads book IDs.

Value

A named list where each element contains the format information for a book.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Run the function
format_info <- get_format_info(temp_file)
print(format_info)
```

```
# Clean up: remove the temporary file
file.remove(temp_file)
```

`get_genres` *Get Genres for Books from Goodreads*

Description

This function reads book IDs from a file, fetches the corresponding Goodreads pages, and extracts the genres for each book.

Usage

```
get_genres(file_path)
```

Arguments

`file_path`	A character string specifying the path to the file containing book IDs.

Value

A named list where each element corresponds to a book ID and contains a character vector of genres for that book.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Run the function
genres <- get_genres(temp_file)

# Display the results
print(genres)

# Clean up: remove the temporary file
file.remove(temp_file)
```

get_num_pages *Get Number of Pages from Goodreads*

Description

This function takes a file path containing Goodreads book IDs and retrieves the number of pages for each book.

Usage

```
get_num_pages(file_path)
```

Arguments

file_path	A character string specifying the path to the file containing Goodreads book IDs.

Value

A named list where each element contains the number of pages for a book.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)
# Run the function
num_pages <- get_num_pages(temp_file)
print(num_pages)
# Clean up: remove the temporary file
file.remove(temp_file)
```

get_published_time *Get Published Time from Goodreads*

Description

This function takes a file path containing Goodreads book IDs and retrieves the published time for each book.

Usage

```
get_published_time(file_path)
```

Arguments

file_path	A character string specifying the path to the file containing Goodreads book IDs.

Value

A named list where each element contains the book information for a book.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)
# Run the function
published_times <- get_published_time(temp_file)
print(published_times)
# Clean up: remove the temporary file
file.remove(temp_file)
```

---

get_rating_distribution

Get Rating Distribution from Goodreads

Description

This function takes a file path containing Goodreads book IDs and retrieves the rating distribution for each book.

Usage

```
get_rating_distribution(file_path)
```

Arguments

file_path	A character string specifying the path to the file containing Goodreads book IDs.

Value

A named list where each element contains the rating distribution for a book.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Run the function
rating_distributions <- get_rating_distribution(temp_file)
print(rating_distributions)

# Clean up: remove the temporary file
file.remove(temp_file)
```

model_topics *Analyze topics in Goodreads reviews*

Description

This function takes the output from scrape_reviews, preprocesses the data, performs topic modeling, and prints the results.

Usage

```
model_topics(reviews, num_topics = 3, num_terms = 10, english_only = TRUE)
```

Arguments

reviews	A data frame containing the scraped reviews
num_topics	The number of topics to extract
num_terms	The number of top terms to display for each topic
english_only	A logical value indicating whether to filter out non-English reviews. Default is FALSE.

Value

A list containing the following elements:

- model: The fitted LDA model object.
- filtered_reviews: The preprocessed and filtered reviews data frame.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Scrape reviews
reviews <- scrape_reviews(temp_file, num_reviews = 5, use_parallel = FALSE)

# Model topics
topic_results <- model_topics(reviews, num_topics = 2, num_terms = 5, english_only = TRUE)

# Print model summary
print(topic_results$model)

# Clean up: remove the temporary file
file.remove(temp_file)
```

plot_topic_heatmap *Visualize topic distribution*

Description

This function creates a heatmap of the topic distribution across documents.

Usage

```
plot_topic_heatmap(model_output)
```

Arguments

model_output	The output from model_topics function

Value

A ggplot object representing the topic distribution heatmap.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Scrape reviews
reviews <- scrape_reviews(temp_file, num_reviews = 10, use_parallel = FALSE)

# Model topics
topic_results <- model_topics(reviews, num_topics = 2, num_terms = 5, english_only = TRUE)

# Visualize topic distribution
plot_topic_heatmap(topic_results)

# Clean up: remove the temporary file
file.remove(temp_file)
```

plot_topic_prevalence *Visualize topic prevalence*

Description

This function creates a bar plot of the overall prevalence of each topic.

Usage

```
plot_topic_prevalence(model_output)
```

Arguments

model_output	The output from model_topics function

Value

A ggplot object representing the bar plot of topic prevalence.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Scrape reviews
reviews <- scrape_reviews(temp_file, num_reviews = 10, use_parallel = FALSE)

# Model topics
topic_results <- model_topics(reviews, num_topics = 2, num_terms = 5, english_only = TRUE)

# Visualize topic distribution
plot_topic_prevalence(topic_results)

# Clean up: remove the temporary file
file.remove(temp_file)
```

---

plot_topic_terms *Visualize top terms for each topic*

Description

This function creates a bar plot of the top terms for each topic.

Usage

```
plot_topic_terms(model_output, n = 10)
```

Arguments

model_output	The output from model_topics function
n	The number of top terms to visualize for each topic

Value

A ggplot object representing the bar plot of top terms for each topic.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Scrape reviews
reviews <- scrape_reviews(temp_file, num_reviews = 10, use_parallel = FALSE)

# Model topics
topic_results <- model_topics(reviews, num_topics = 2, num_terms = 5, english_only = TRUE)

# Visualize top terms for each topic
plot_topic_terms(topic_results, n = 5)

# Clean up: remove the temporary file
file.remove(temp_file)
```

---

preprocess_reviews *Preprocess review text for topic modeling*

Description

This function preprocesses the review text by optionally filtering non-English reviews, removing punctuation, converting to lowercase, removing stopwords, and stemming.

Usage

```
preprocess_reviews(reviews, english_only = TRUE)
```

Arguments

reviews	A data frame containing the scraped reviews
english_only	A logical value indicating whether to filter out non-English reviews. Default is TRUE

Value

A list containing the following elements:

- corpus: The preprocessed corpus object.
- dtm: The document-term matrix.
- filtered_reviews: The filtered reviews data frame.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Scrape reviews
reviews <- scrape_reviews(temp_file, num_reviews = 5, use_parallel = FALSE)

# Preprocess the reviews
preprocessed <- preprocess_reviews(reviews, english_only = TRUE)

# Print the document-term matrix
print(preprocessed$dtm)

# Clean up: remove the temporary file
file.remove(temp_file)
```

---

replace_special_chars *Replace special characters and remove non-ASCII characters*

Description

Replace special characters and remove non-ASCII characters

Usage

```
replace_special_chars(x)
```

Arguments

x	A character vector

Value

A character vector with special characters replaced and non-ASCII characters removed

---

scrape_books *Scrape book details from Goodreads*

Description

This function scrapes details of books using their IDs from Goodreads.

Usage

```
scrape_books(book_ids_path, use_parallel = FALSE, num_cores = 4)
```

Arguments

book_ids_path	Path to a text file containing book IDs.
use_parallel	Logical indicating whether to scrape in parallel (default is FALSE).
num_cores	Number of CPU cores to use for parallel scraping (default is 4).

Value

A data frame containing scraped book details.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Run the function (with a small delay to avoid overwhelming the server)
result <- scrape_books(temp_file, use_parallel = FALSE)
print(head(result))
# Clean up: remove the temporary file
file.remove(temp_file)
```

---

scrape_reviews *Scrape book reviews from Goodreads*

Description

This function scrapes book reviews from Goodreads based on provided book IDs.

Usage

```
scrape_reviews(
  book_ids_path,
  num_reviews = 30,
  use_parallel = FALSE,
  num_cores = 4
)
```

Arguments

book_ids_path	A character string specifying the path to a file containing book IDs.
num_reviews	An integer specifying the number of reviews to scrape per book. Default is 30.
use_parallel	A logical value indicating whether to use parallel processing. Default is FALSE.
num_cores	An integer specifying the number of cores to use for parallel processing. Default is 4.

Value

A data frame containing scraped review information.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)
# Run the function (with a small number of reviews to keep the example quick)
reviews <- scrape_reviews(temp_file, num_reviews = 5, use_parallel = FALSE)
print(head(reviews))
# Clean up: remove the temporary file
file.remove(temp_file)
```

---

search_goodreads *Search Goodreads*

Description

This function searches books on Goodreads.

Usage

```
search_goodreads(
  search_term,
  search_in = c("title", "author"),
  num_books = 10,
  sort_by = "ratings"
)
```

Arguments

search_term	A search term string.
search_in	Where to search (e.g., "title", "author").
num_books	Number of books to return.
sort_by	How to sort the results (e.g., "ratings", "published_year").

Value

A data frame of search results.

Examples

```
search_goodreads("parenting", search_in = "title", num_books = 2)
```

sentiment_histogram *Create a histogram of sentiment scores*

Description

This function creates a histogram of sentiment scores for all reviews.

Usage

```
sentiment_histogram(sentiment_df)
```

Arguments

sentiment_df	A data frame containing the output from analyze_sentiment.

Value

A ggplot object representing the histogram.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Run the scrape_reviews function
reviews <- scrape_reviews(temp_file, num_reviews = 10, use_parallel = FALSE)

# Check if reviews were successfully scraped
if (nrow(reviews) > 0) {
  # Perform sentiment analysis
  sentiment_results <- analyze_sentiment(reviews, lexicon = "afinn")

  # Create histogram of sentiment scores
  sentiment_hist <- sentiment_histogram(sentiment_results)

  # Display the plot
  print(sentiment_hist)

  # Optionally, save the plot
  # ggsave("sentiment_hist.png", sentiment_hist, width = 8, height = 6)
} else {
  cat("No reviews found. Cannot create sentiment histogram.\n")
}

# Clean up: remove the temporary file
file.remove(temp_file)
```

sentiment_trend *Plot sentiment trend over time*

Description

This function plots the average sentiment score over time.

Usage

```
sentiment_trend(sentiment_df, time_period = "month", show_smooth_trend = FALSE)
```

Arguments

sentiment_df	A data frame containing the output from analyze_sentiment.
time_period	A string specifying the time period for grouping ("day", "week", "month", "year").
show_smooth_trend	A logical value indicating whether to show the overall smooth trend line (default: TRUE).

Value

A ggplot object representing the sentiment trend.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Run the scrape_reviews function
reviews <- scrape_reviews(temp_file, num_reviews = 10, use_parallel = FALSE)

# Check if reviews were successfully scraped
if (nrow(reviews) > 0) {
  # Perform sentiment analysis
  sentiment_results <- analyze_sentiment(reviews, lexicon = "afinn")

  # Create histogram of sentiment scores
  senti_trend <- sentiment_trend(sentiment_results)

  # Display the plot
  print(senti_trend)

  # Optionally, save the plot
  # ggsave("senti_trend.png", senti_trend, width = 8, height = 6)
} else {
  cat("No reviews found. Cannot create sentiment trend\n")
}
```

```
# Clean up: remove the temporary file
file.remove(temp_file)
```

---

top_terms *Extract and print top terms for each topic*

Description

This function extracts the top terms for each topic in the LDA model and optionally prints them.

Usage

```
top_terms(lda_model, n = 10, verbose = TRUE)
```

Arguments

lda_model	An LDA model
n	The number of top terms to extract for each topic
verbose	Logical; if TRUE, print the top terms to the console (default is TRUE)

Value

A list of character vectors, each containing the top terms for a topic.

Examples

```
# Create a temporary file with sample book IDs
temp_file <- tempfile(fileext = ".txt")
writeLines(c("1420", "2767052", "10210"), temp_file)

# Scrape reviews
reviews <- scrape_reviews(temp_file, num_reviews = 5, use_parallel = FALSE)

# Preprocess the reviews
preprocessed <- preprocess_reviews(reviews, english_only = TRUE)

# Fit LDA model
lda_model <- fit_lda(preprocessed$dtm, k = 2)

# Print top terms
top_terms(lda_model, n = 5)

# Clean up: remove the temporary file
file.remove(temp_file)
```

Index

analyze_sentiment, 2
average_book_sentiment, 3

fit_lda, 4

gen_topic_clouds, 5
get_author_info, 6
get_book_ids, 7
get_book_summary, 7
get_format_info, 8
get_genres, 9
get_num_pages, 10
get_published_time, 10
get_rating_distribution, 11

model_topics, 12

plot_topic_heatmap, 13
plot_topic_prevalence, 13
plot_topic_terms, 14
preprocess_reviews, 15

replace_special_chars, 16

scrape_books, 16
scrape_reviews, 17
search_goodreads, 18
sentiment_histogram, 19
sentiment_trend, 20

top_terms, 21