Package 'jeopardyNLP'

April 20, 2025

Type Package

Title Analyze Jeopardy! Clue Text with NLP Techniques

Version 0.1.0

Author Chris Reger

Maintainer Chris Reger <kcr28@nau.edu>

Description Provides functions and workflows for preprocessing, exploring, and modeling text data from Jeopardy! clues. Includes tools for text cleaning, exploratory data analysis (EDA) via word clouds and plots, unsupervised topic modeling using Non-negative Matrix Factorization (NMF), and supervised classification of clue difficulty.

License MIT + file LICENSE

Depends R (>= 3.5)

Imports dplyr, forcats, ggplot2, lifecycle, Matrix, NMF, RColorBrewer, readr, scales, slam, stringr, tm, tidyr, wordcloud

Suggests caret, e1071, here, knitr, purrr, rmarkdown, SnowballC, testthat (>= 3.0.0), textstem, tibble

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.2

Contents

`add_clue_difficulty` *Add Clue Difficulty Column*

Description

Adds a factor column 'clue_difficulty' (easy, average, hard) based on the round and dollar value of the clue. Provides two slightly different logic options.

Usage

```
add_clue_difficulty(df, viewer_assumptions = FALSE)
```

Arguments

`df`	Input data frame containing 'round', 'value', and 'daily_double' columns.
`viewer_assumptions`	Logical. If FALSE (default), uses standard logic. If TRUE, uses alternative logic based on viewer perception analysis cited in original project.

Value

The data frame with the added 'clue_difficulty' factor column.

add_question_answer_col

Create Combined Question and Answer Column

Description

Concatenates the 'Question' and 'Answer' columns into a single 'Question And Answer' column.

Usage

```
add_question_answer_col(df)
```

Arguments

df	Input data frame (tibble) with 'Question' and 'Answer' columns.

Value

A tibble with the added 'Question And Answer' column.

calculate_top_categories

Calculate Top N Categories

Description

Determines the top N most frequent J-Categories and estimates the number of episodes they appeared in (assuming 5 clues per category per episode).

Usage

```
calculate_top_categories(df = jeopardyNLP::jeopardy_data, n = 10)
```

Arguments

df	Input data frame (tibble) containing the 'J-Category' column. Defaults to the package's internal 'jeopardy_data'.
n	Number of top categories to return.

Value

A tibble with columns 'J-Category' and 'Estimated Episodes', sorted by frequency.

calculate_word_counts *Calculate Word Counts*

Description

Calculates the number of words in the 'Answer' and 'Question' columns.

Usage

```
calculate_word_counts(df)
```

Arguments

df	Input data frame (tibble) containing 'Clue Difficulty', 'Answer', and 'Question' columns. This data should typically be the result of processing the raw data using 'create_regular_episodes_df' with 'add_difficulty = TRUE'.

Value

A tibble with 'Clue Difficulty', 'Answer Word Count', and 'Question Word Count' columns.

clean_text_corpus *Clean Text Corpus using tm*

Description

Performs a series of cleaning operations on a character vector of text documents using the 'tm' package framework. This includes lowercasing, hyphen replacement, punctuation removal, number removal, stopword removal, and whitespace stripping. Optionally includes stemming or lemmatization. Also applies a post-tm step to remove short tokens.

Usage

```
clean_text_corpus(
  text_vector,
  custom_stopwords_path = NULL,
  stem_words = FALSE,
  lemmatize_words = FALSE,
  remove_short = TRUE,
  min_token_length = 3
)
```

Arguments

text_vector	A character vector where each element is a document.
custom_stopwords_path	Path to a custom stopwords file (one per line). If NULL (default), attempts to load the stopwords.txt file included with the package.
stem_words	Logical. If TRUE, applies Porter stemming using 'tm::stemDocument'.

`lemmatize_words`
: Logical. If TRUE, applies lemmatization using 'textstem::lemmatize_strings'. Requires the 'textstem' package to be installed.

`remove_short`
: Logical. If TRUE (default), removes tokens with 3 or fewer characters after main tm cleaning.

`min_token_length`
: Minimum length for 'remove_short_tokens'.

Value

A character vector of cleaned documents, with tokens joined by spaces.

`create_regular_episodes_df` *Create Processed Data Frame for Regular Episodes*

Description

Filters the raw Jeopardy data for standard Jeopardy! and Double Jeopardy! rounds, cleans column names, optionally adds a combined Question/Answer column, and optionally adds a clue difficulty classification.

Usage

```
create_regular_episodes_df(
  raw_df,
  add_q_and_a = TRUE,
  add_difficulty = TRUE,
  viewer_assumptions = FALSE,
  filter_notes = TRUE
)
```

Arguments

`raw_df`
: The raw data frame (tibble) read from the source TSV.

`add_q_and_a`
: Logical, if TRUE adds the 'question_and_answer' column.

`add_difficulty`
: Logical, if TRUE adds the 'clue_difficulty' column.

`viewer_assumptions`
: Passed to 'add_clue_difficulty' if 'add_difficulty' is TRUE.

`filter_notes`
: Logical, if TRUE removes rows where the 'notes' column is not NA (default TRUE).

Value

A tibble containing the processed data for regular episodes.

Examples

```
# Assuming jeopardy_raw is your loaded raw data:
# regular_data <- create_regular_episodes_df(jeopardy_raw)
# head(regular_data)
```

`create_special_tournaments_df` *Create Special Tournaments Dataframe*

Description

Filters the raw Jeopardy data for special tournament episodes (notes != '-'), drops unnecessary columns, formats names, and optionally adds combined Q&A and difficulty columns.

Usage

```
create_special_tournaments_df(
  df,
  add_q_and_a = TRUE,
  add_difficulty = FALSE,
  ...
)
```

Arguments

`df`	Raw input data frame from 'read_jeopardy_tsv'.
`add_q_and_a`	Logical. If TRUE, adds the combined 'Question And Answer' column.
`add_difficulty`	Logical. If TRUE, adds the 'Clue Difficulty' column (using default logic unless 'viewer_assumptions' is specified).
`...`	Additional arguments passed to 'add_clue_difficulty' (e.g., 'viewer_assumptions').

Value

A tibble containing processed special tournament data.

`create_tfidf_matrix` *Create TF-IDF Matrix*

Description

Creates a Term-Document Matrix (TDM) with TF-IDF weighting from a vector of cleaned text documents. Assumes input text is already processed (lowercase, punctuation removed, stopwords removed, etc.). The final matrix has terms as rows and documents as columns, as typically expected by the NMF package.

Usage

```
create_tfidf_matrix(text_vector, control = list(weighting = tm::weightTfIdf))
```

Arguments

`text_vector`	A character vector where each element represents a document.
`control`	A list of control parameters passed to 'tm::DocumentTermMatrix'. Defaults include TF-IDF weighting. See '?TermDocumentMatrix' for options (e.g., 'list(weighting = weightTfIdf, bounds = list(global = c(5, Inf)))').

Value

A sparse TermDocumentMatrix (dgTMatrix) with TF-IDF weights. Terms are rows, documents are columns.

`evaluate_nmf_ranks` *Evaluate NMF Ranks (Find Best k)*

Description

Runs NMF for a range of ranks (k) and collects a specified metric (defaulting to Frobenius residuals) to help determine an optimal number of topics.

Usage

```
evaluate_nmf_ranks(
  term_doc_matrix,
  ranks = 7:15,
  seed = 43,
  method = "brunet",
  metric = get_nmf_residuals,
  n_runs = 1,
  ...
)
```

Arguments

`term_doc_matrix`	Input term-document matrix.
`ranks`	A numeric vector of ranks (k values) to evaluate.
`seed`	Random seed for reproducibility.
`method`	NMF algorithm method.
`metric`	Function to extract the evaluation metric from the NMF result object. Defaults to 'get_nmf_residuals'.
`n_runs`	Number of NMF runs per rank (if using methods that benefit from multiple runs).
`...`	Additional arguments passed to 'NMF::nmf'.

Value

A tibble with columns 'k' (rank) and 'metric' (the evaluated metric value).

extract_text_vector *Extract Text Vector from DataFrame Column*

Description

Extracts a specific column from a data frame and returns it as a character vector.

Usage

```
extract_text_vector(df, col = "Question And Answer")
```

Arguments

df	The input data frame (tibble).
col	The name of the column to extract (string).

Value

A character vector containing the text data from the specified column.

format_column_names *Format Column Names*

Description

Renames columns to a consistent format (e.g., snake_case to PascalCase or adding hyphens).

Usage

```
format_column_names(df)
```

Arguments

df	Input data frame (tibble).

Value

A tibble with renamed columns.

get_nmf_residuals *Get NMF Residuals*

Description

Extracts the Frobenius norm of the residuals (||X - WH||) from a fitted NMF model object. This serves as a proxy for reconstruction error.

Usage

```
get_nmf_residuals(nmf_result)
```

Arguments

nmf_result	An object of class 'NMF' (output from 'run_nmf_model' or 'NMF::nmf').

Value

The Frobenius norm of the residuals.

get_top_words_per_topic *Get Top Words Per Topic*

Description

Extracts the top N most heavily weighted words for each topic from the basis matrix (W) of a fitted NMF model.

Usage

```
get_top_words_per_topic(nmf_result, n_top_words = 10)
```

Arguments

nmf_result	An object of class 'NMF'.
n_top_words	The number of top words to retrieve for each topic.

Value

A list where each element corresponds to a topic and contains a character vector of the top 'n_top_words' for that topic.

get_word_weights_for_cloud

Get Word Weights for Topic Word Cloud

Description

Extracts the top N words and their corresponding weights (scores) from the basis matrix for a specific topic. Applies square root scaling to weights similar to the original Python implementation for word cloud frequency scaling.

Usage

```
get_word_weights_for_cloud(nmf_result, topic_index, n_top_words = 15)
```

Arguments

nmf_result	An object of class 'NMF'.
topic_index	The index (1-based) of the topic to extract weights for.
n_top_words	The number of top words/weights to return.

Value

A named numeric vector where names are the top words and values are their scaled weights (sqrt(score)). Returns NULL if topic_index is invalid or no valid weights are found.

jeopardyNLP-eda-internal

Jeopardy EDA Utilities

Description

Functions for performing exploratory data analysis on the Jeopardy dataset. Includes functions for plotting distributions and generating word clouds.

jeopardyNLP-nmf-internal

Non-Negative Matrix Factorization (NMF) for Topic Modeling

Description

Functions for applying NMF to the Jeopardy dataset to discover underlying topics within the clues. Includes text preparation specific to NMF and visualization of results.

`jeopardyNLP-preprocessor-internal` *Data Preprocessing Utilities*

Description

Functions for reading, cleaning, and preparing the raw Jeopardy TSV data for analysis. Includes column renaming, filtering, and adding derived columns like clue difficulty.

`jeopardyNLP-textcleaner-internal` *Text Cleaning Utilities*

Description

Functions for cleaning text data, primarily designed for preparing Jeopardy clues for NLP tasks. Uses the 'tm' package for efficient corpus-based cleaning.

`jeopardy_data` *Jeopardy! Clues Dataset*

Description

A dataset containing information about Jeopardy! clues from show number 4680 (aired 2004-12-31) onwards, scraped from J! Archive (j-archive.com). This version is pre-processed slightly from the raw source available on Kaggle (https://www.kaggle.com/datasets/tunguz/200000-jeopardy-questions).

Usage

```
jeopardy_data
```

Format

A tibble (data frame) with approximately 216,930 rows and 9 columns:

show_number Unique identifier for the show (character).

air_date Date the show aired (Date object).

round The round the clue appeared in (e.g., "Jeopardy!", "Double Jeopardy!", "Final Jeopardy!") (character).

category The category of the clue (character).

value The dollar value of the clue (e.g., "$200") (character).

question The text of the clue/question (character).

answer The correct answer to the clue (character).

notes Additional notes, often indicating special rounds like Final Jeopardy (character).

daily_double Indicates if the clue was a Daily Double ("yes" or "no") (character).

Source

https://www.kaggle.com/datasets/tunguz/200000-jeopardy-questions, originally scraped from http://j-archive.com/

`load_stopwords` *Load Stopwords*

Description

Loads stopwords from the standard English list provided by the ‘tm‘ package and extends it with custom stopwords read from a specified file. The custom stopwords file can be either newline-separated (one word per line) or a single line of comma-separated words.

Usage

```
load_stopwords(custom_stopwords_path = NULL)
```

Arguments

`custom_stopwords_path`
: Path to the file containing custom stopwords. If NULL (default), attempts to load the stopwords.txt file included with the package from ‘inst/extdata/stopwords.txt‘.

Value

A character vector containing the combined list of unique stopwords.

`plot_all_topic_clouds` *Plot All Topic Word Clouds*

Description

Generates word clouds for all topics in a fitted NMF model.

Usage

```
plot_all_topic_clouds(
  nmf_result,
  n_top_words = 15,
  color_palette = "Dark2",
  save = FALSE,
  output_dir = NULL,
  filename_prefix = "topic_"
)
```

Arguments

nmf_result	An object of class ‘NMF‘.
n_top_words	Number of top words per topic cloud.
color_palette	RColorBrewer palette name.
save	Logical. If TRUE, saves plots to files.
output_dir	Directory path for saving plots.
filename_prefix	Allow custom prefix, but no full default filename

Value

Nothing. Generates multiple word clouds.

---

plot_nmf_evaluation *Plot NMF Rank Evaluation Metric*

Description

Plots the chosen evaluation metric (e.g., residuals) against the number of topics (k).

Usage

```
plot_nmf_evaluation(
  k_results_df,
  metric_name = "Residuals",
  save = FALSE,
  output_dir = NULL,
  filename = NULL
)
```

Arguments

k_results_df	A data frame/tibble from ‘evaluate_nmf_ranks‘ with columns ’k’ and ’metric’.
metric_name	A string label for the y-axis (e.g., "Frobenius Residuals").
save	Logical. If TRUE, saves the plot to a file.
output_dir	Directory path for saving.
filename	Filename for the saved plot.

Value

Invisibly returns the ggplot object. Displays or saves the plot.

plot_topic_word_cloud *Plot Topic Word Cloud*

Description

Generates and displays/saves a word cloud for a specific NMF topic.

Usage

```
plot_topic_word_cloud(
  nmf_result,
  topic_index,
  n_top_words = 15,
  color_palette = "Dark2",
  save = FALSE,
  output_dir = NULL,
  filename = NULL
)
```

Arguments

nmf_result	An object of class 'NMF'.
topic_index	The index (1-based) of the topic to visualize.
n_top_words	The number of top words to include in the cloud.
color_palette	Name of the RColorBrewer palette to use.
save	Logical. If TRUE, saves the plot to a file.
output_dir	Directory path to save the plot if 'save = TRUE'.
filename	Optional filename. Defaults to '_topic_X_wordcloud.png' (using 1-based X).

Value

Nothing. Displays or saves the word cloud plot.

plot_top_categories *Plot Top N Categories*

Description

Creates a bar chart showing the top N J-Categories based on their estimated episode appearances.

Usage

```
plot_top_categories(
  top_cats_df,
  bar_color = "midnightblue",
  save = FALSE,
  output_dir = NULL,
  filename = NULL
)
```

Arguments

top_cats_df	Data frame output from ‘calculate_top_categories‘.
bar_color	Color for the bars.
save	Logical. If TRUE, saves the plot to a file.
output_dir	Directory path to save the plot if ‘save = TRUE‘.
filename	Name for the saved file.

Value

Invisibly returns the ggplot object. Displays or saves the plot.

---

plot_word_cloud *Plot Word Cloud from Text Column*

Description

Generates and displays/saves a word cloud from a specified text column in a data frame. Note: Performs basic internal text cleaning (lowercase, punctuation, numbers, standard English stopwords) unless ‘perform_cleaning‘ is FALSE. For more control or advanced cleaning, pre-process the text using functions from ‘text_cleaner.R‘ before calling this function.

Usage

```
plot_word_cloud(
  df = jeopardyNLP::jeopardy_data,
  col,
  perform_cleaning = TRUE,
  color_palette = "Dark2",
  max_words = 150,
  save = FALSE,
  output_dir = NULL,
  filename = NULL
)
```

Arguments

df	Input data frame (tibble). Defaults to the package’s internal ‘jeopardy_data‘.
col	Name of the column containing the text data.
perform_cleaning	Logical. If TRUE (default), performs basic ‘tm‘ cleaning.
color_palette	Name of the RColorBrewer palette to use.
max_words	Maximum number of words to include in the cloud.
save	Logical. If TRUE, saves the plot to a file instead of displaying.
output_dir	Directory path to save the plot if ‘save = TRUE‘.
filename	Name for the saved file (defaults based on column name).

Value

Invisibly returns the word frequency data frame used for the plot. Displays or saves the word cloud plot.

plot_word_counts_difficulty *Plot Word Counts vs. Clue Difficulty*

Description

Creates a bar chart showing the average word count (for either Answer or Question) for each clue difficulty level.

Usage

```
plot_word_counts_difficulty(
  df_word_counts,
  count_col,
  bar_color = "steelblue",
  save = FALSE,
  output_dir = NULL,
  filename = NULL
)
```

Arguments

df_word_counts	Data frame output from ‘calculate_word_counts‘.
count_col	Name of the word count column to plot (e.g., "Answer Word Count").
bar_color	Color for the bars.
save	Logical. If TRUE, saves the plot to a file.
output_dir	Directory path to save the plot if ‘save = TRUE‘.
filename	Name for the saved file (defaults based on count column name).

Value

Invisibly returns the ggplot object. Displays or saves the plot.

read_jeopardy_tsv *Read Jeopardy TSV Data [Deprecated]*

Description

Reads a TSV file containing Jeopardy data into a tibble. This function is deprecated. Please use the internal ‘jeopardy_data‘ object directly (available after loading the package) or load it via ‘data(jeopardy_data)‘. Use this function only if you need to load a custom, external TSV file with the same structure.

Usage

```
read_jeopardy_tsv(filepath)
```

Arguments

filepath	Path to the TSV file.

Value

A tibble containing the Jeopardy data.

Examples

```
# Access internal data instead:
# library(jeopardyNLP)
# head(jeopardy_data)

# Or load explicitly:
# data(jeopardy_data)
# head(jeopardy_data)

# Only use for external files:
# external_file <- system.file("extdata", "master_season1-35.tsv", package = "jeopardyNLP")
# if (file.exists(external_file)) {
#   external_data <- read_jeopardy_tsv(external_file)
# }
```

---

remove_short_tokens *Remove Short Tokens*

Description

Removes tokens (words) from a character vector that are shorter than a specified minimum length.

Usage

```
remove_short_tokens(tokens, min_length = 3)
```

Arguments

tokens	A character vector of tokens.
min_length	The minimum number of characters a token must have to be kept. Defaults to 3 (keeps tokens with length > 3).

Value

A character vector containing only tokens longer than ‘min_length‘.

replace_hyphens *Replace Hyphens with Spaces*

Description

Replaces hyphens connecting word characters with spaces. Handles patterns like word-word and word-word-word.

Usage

```
replace_hyphens(text_vector)
```

Arguments

text_vector	A character vector.

Value

A character vector with hyphens replaced by spaces.

run_nmf_model *Run NMF Model*

Description

Performs Non-negative Matrix Factorization on a given matrix (typically TF-IDF TDM).

Usage

```
run_nmf_model(term_doc_matrix, k, seed = 43, method = "brunet", ...)
```

Arguments

term_doc_matrix	A numeric matrix where rows are terms/features and columns are documents (e.g., output from ‘create_tfidf_matrix‘). Matrix must be non-negative.
k	The desired number of topics (rank for factorization).
seed	Random seed for reproducibility.
method	The NMF algorithm to use (e.g., "brunet", "lee", "nsNMF"). See ‘?nmf‘.
...	Additional arguments passed to ‘NMF::nmf‘.

Value

An object of class ‘NMF‘ resulting from the factorization.

Index