Package 'LLMing'

January 8, 2026

Title Large Language Model (LLM) Tools for Psychological Text Analysis

Version 1.1.0

Maintainer Lindley Slipetz <ddj6tu@virginia.edu>

Description A collection of large language model (LLM) text analysis methods designed with psychological data in mind. Currently, LLMing (aka ``lemming'') includes a text anomaly detection method based on the angle-based subspace approach described by Zhang, Lin, and Karim (2015) and a text generation method. <doi:10.1016/j.ress.2015.05.025>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports Rdpack, quanteda, stopwords, stringi, reticulate, text, dbscan, pracma, stats, jsonlite

SystemRequirements Python (>= 3.10) with packages: torch, transformers, pandas, numpy

RdMacros Rdpack

URL https://github.com/sliplr19/LLMing

BugReports https://github.com/sliplr19/LLMing/issues

NeedsCompilation no

Author Lindley Slipetz [aut, cre],
Teague Henry [aut],
Siqi Sun [ctb]

Depends R (>= 4.1.0)

Repository CRAN

Date/Publication 2026-01-08 05:20:13 UTC

Contents

`embed` *Embed texts with a Transformer model*

Description

Cleans a text column and converts it to a dataframe of numeric vectors via BERT embeddings. For the input dataframe, each row is one text entry.

Usage

```
embed(dat, layers, keep_tokens = TRUE, tokens_method = NULL)
```

Arguments

`dat`	A dataframe with text data, one text per row
`layers`	Integer vector specifying which model layers to aggregate from.
`keep_tokens`	Logical, keep token-level embeddings in the returned object or discard them to save memory
`tokens_method`	Character scalar controlling how token-level embeddings are aggregated to word types

Value

A dataframe where each row corresponds to one input text and each column is an embedding dimension

@examples df <- data.frame(text = c("I slept well and feel great today!", "I saw from friends and it went well.", "I think I failed that exam. I'm such a disappointment." "I think I failed that exam. I'm such a disapointment."))

emb_dat <- embed(dat = df, layers = 1, keep_tokens = FALSE, tokens_method = "mean")

G_thres *Thresholding of pCOS dataframe*

Description

Converts each column of a pCOS score matrix into binary indicators

Usage

```
G_thres(pCOS_mat, theta)
```

Arguments

pCOS_mat	Dataframe of pCOS values
theta	Numeric threshold

Value

A matrix of 0s and 1s of which cells meet the threshold

Examples

```
z_dat <- data.frame("A" = rnorm(500,0,1), "B" = rnorm(500,0,1), "C" = rnorm(500,0,1))
snn <- sim_SNN(z_dat, 10, 5)
vec_snn <- vector_SNN(z_dat, snn)
pCOSdat <- pCOS(z_dat, vec_snn)
G <- G_thres(pCOSdat, theta = 0.1)
```

normahalo *Local outlier score*

Description

Computes a normalized Mahalanobis distance score. Only features with nonzero scores in S receive nonzero Mahalanobis scores.

Usage

```
normahalo(z, rs, S)
```

Arguments

z	Dataframe of z scores
rs	List of reference sets
S	Dataframe of numeric values

Value

A dataframe of local outlier scores

`pCOS` *pCOS scores for every row of dataframe*

Description

Applies pCOS_row() to corresponding rows of two data frames, returning one pCOS value per row.

Usage

```
pCOS(z_dat, vec_SNN)
```

Arguments

`z_dat`	Numeric dataframe, typically z-scores
`vec_SNN`	Numeric dataframe, typically the output of vector_SNN

Value

A dataframe with same dimensions as z_dat

`pCOS_row` *Pairwise cosine-style row score*

Description

Given two numeric vectors, computes an average cosine-based similarity.

Usage

```
pCOS_row(z, v_SNN)
```

Arguments

`z`	Numeric vector
`v_SNN`	Numeric vector, same size as z

Value

A numeric vector

rep_set *The vectors of the shared nearest neighbors*

Description

Creates a list of the vectors of the top shared nearest neighbors for each row of the z dataframe

Usage

```
rep_set(z, snn)
```

Arguments

z	Dataframe of values of reference set
snn	Dataframe of shared nearest neighbors indices

Value

A list of dataframes where each row of the dataframe is the vector representation of a given shared nearest neighbor

sim_SNN *Compute shared nearest neighbors*

Description

Builds a shared nearest neighbors matrix and, for each row (observation), returns the indices of the top neighbors with the largest SNN overlap counts

Usage

```
sim_SNN(z_dat, k, tops)
```

Arguments

z_dat	A dataframe with numeric columns
k	An integer representing number of nearest neighbors
tops	An integer representing how many of shared nearest neighbors to return

Value

A dataframe of top rows with shared nearest neighbors

textanomaly *Text anomaly score*

Description

Text anomaly detection method adapted from (Zhang et al. 2015).

Usage

```
textanomaly(dat, k, tops, theta)
```

Arguments

dat	A dataframe with text data, one text per row
k	An integer representing number of nearest neighbors
tops	An integer representing how many of shared nearest neighbors to return
theta	Numeric threshold

Value

Dataframe of local outlier score

References

Zhang L, Lin J, Karim R (2015). "An angle-based subspace anomaly detection approach to high-dimensional data: With an application to industrial fault detection." *Reliability Engineering & System Safety*, **142**, 482–497. ISSN 0951-8320, doi:10.1016/j.ress.2015.05.025.

text_datagen *Generate text data via Python LLM*

Description

All prompt components and example texts are provided by the user as function arguments. This function generates text data based on severity score from a given questionnaire.

Usage

```
text_datagen(
  prompts,
  examples,
  scenario = NULL,
  overall_rules = NULL,
  percentile_scaffold = NULL,
  item_rules = NULL,
```

```
  items = NULL,
  structure_rules = NULL,
  percentile_specification = NULL,
  band_specification = NULL,
  example_instruction = NULL,
  what_to_write = NULL,
  task_desc = NULL,
  target_min = 90L,
  target_max = 100L,
  temperature = 0.4,
  top_p = 0.9,
  repetition_penalty = 1.1,
  model_name = "meta-llama/Meta-Llama-3-8B-Instruct",
  batch_size = 2L,
  python = Sys.getenv("RETICULATE_PYTHON", "python"),
  env = NULL,
  output_file = NULL
)
```

Arguments

prompts	A data.frame with one row per diary to generate. Must contain at least a column indicating severity level.
examples	A data.frame of example diary texts with columns: text or character column and any grouping severity variable column).
scenario	Character string used in the SCENARIO section. This describes the situation in which the data is being collected.
overall_rules	Character string describing global writing rules.
percentile_scaffold	Character string describing how percentiles map onto severity.
item_rules	Character string describing how to internally choose symptom patterns.
items	Character string of the battery under study.
structure_rules	Character string describing structural rules (paragraphs, length, etc.).
percentile_specification	Character string describing what the severity percentile means.
band_specification	Character string describing severity bands, that is, what you expect each band of severity to look like in text.
example_instruction	Character string introducing the example texts.
what_to_write	Character string describing what the model should write about.
task_desc	Character string for the system-level role description.
target_min	Integer minimum number of tokens to generate.
target_max	Integer maximum number of tokens to generate.

temperature	Numeric temperature for sampling.
top_p	Numeric top-p nucleus sampling value.
repetition_penalty	Numeric repetition penalty.
model_name	Model identifier string to pass to transformers (e.g., "meta-llama/Meta-Llama-3-8B-Instruct", a local path, etc.).
batch_size	Integer, passed through to the Python script (not heavily used yet).
python	Path to the Python executable. Defaults to Sys.getenv("RETICULATE_PYTHON", "python").
env	Optional named character vector or list of environment variables to set for the duration of the call (e.g., c(HUGGINGFACE_HUB_TOKEN = "xxx", OPENAI_API_KEY = "yyy")). Any variables set here are restored to their previous values on exit.
output_file	Optional path to save the output CSV. If NULL, a temporary file is used and only the data.frame is returned.

Value

A data.frame with columns id, severity, and response. @examples prompts <- data.frame(id = 1:2, severity = c(10, 80), num_examples = c(1, 1)) examples <- data.frame(text = c("Example A", "Example B"), label = c("group1", "group2"), stringsAsFactors = FALSE) out <- text_datagen(prompts = prompts, examples = examples, scenario = "This is an EMA study on depression", overall_rules = "Write 100 tokens of a diary entry collected every 6 hours.", percentile_scaffold = "The 90th percentile corresponds with severe depression and the 10th percentile corresponds with mild depression", item_rules = "For the 90th percentile, you should write as though you scored a 3 on all items", items = "Insert full battery here.", structure_rules = "Short paragraph.", percentile_specification = "Test specification.", band_specification = "Test bands.", example_instruction = "Here are examples.", what_to_write = "Write no less than 100 tokens and no more than 200 tokens", task_desc = "You are a participant in an EMA study on depression scoring in the 90th percentile of X battery.", target_min = 10, target_max = 20, temperature = 0.9, top_p = 0.9, repetition_penalty = 1.0, model_name = "sshleifer/tiny-gpt2", env = NULL # No token needed)

---

vector_SNN *Aggregate dataframe into mean feature vectors Aggregrate dataframe into mean feature vectors*

Description

For each row of the SNN index matrix, this function takes the rows of reference dataframe, z, and computes their column means, yielding one mean vector per observation.

Usage

```
vector_SNN(z, snn)
```

Arguments

z	Numeric dataframe
snn	Dataframe of shared nearest neighbors indices

Value

Dataframe of same dimensions as z

---

z_score *Z-score on columns*

Description

Z-score on columns

Usage

```
z_score(dat)
```

Arguments

dat	A dataframe with numeric cells

Value

A dataframe with numeric cells with the same dimensions as dat

Index