Package: topiclabels (via r-universe)

October 21, 2024

Type Package

Title Automated Topic Labeling with Language Models

Version 0.2.0

Date 2024-10-21

Maintainer Jonas Rieger <rieger@statistik.tu-dortmund.de>

Description Leveraging (large) language models for automatic topic labeling. The main function converts a list of top terms into a label for each topic. Hence, it is complementary to any topic modeling package that produces a list of top terms for each topic. While human judgement is indispensable for topic validation (i.e., inspecting top terms and most representative documents), automatic topic labeling can be a valuable tool for researchers in various scenarios.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.6.0)

Imports checkmate (>= 1.8.5), httr, progress, stats, jsonlite

LazyData true

RoxygenNote 7.3.2

URL https://github.com/PetersFritz/topiclabels

BugReports https://github.com/PetersFritz/topiclabels/issues

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

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

RemoteUrl https://github.com/petersfritz/topiclabels

RemoteRef HEAD

RemoteSha d26037f67eb9cfc6e27c212ab9722ba9a364e5fc

Contents

`topiclabels-package` *Automated Topic Labeling with Language Models*

Description

Leveraging (large) language models for automatic topic labeling. The main function converts a list of top terms into a label for each topic. Hence, it is complementary to any topic modeling package that produces a list of top terms for each topic. While human judgement is indispensable for topic validation (i.e., inspecting top terms and most representative documents), automatic topic labeling can be a valuable tool for researchers in various scenarios.

Labeling function

`label_topics`

Constructor

`as.lm_topic_labels`

Author(s)

Maintainer: Jonas Rieger <rieger@statistik.tu-dortmund.de> (ORCID)

Authors:

- Fritz Peters <fpeters3@sheffield.ac.uk> (ORCID)
- Andreas Fischer <andreasfischer1985@web.de> (ORCID)
- Tim Lauer <tl@leibniz-psychology.org> (ORCID)
- André Bittermann <abi@leibniz-psychology.org> (ORCID)

See Also

Useful links:

- https://github.com/PetersFritz/topiclabels
- Report bugs at https://github.com/PetersFritz/topiclabels/issues

`as.lm_topic_labels` *lm_topic_labels object*

Description

Constructor for lm_topic_labels objects used in this package.

Usage

```
as.lm_topic_labels(
  x,
  terms,
  prompts,
  model,
  params,
  with_token,
  time,
  model_output,
  labels
)

is.lm_topic_labels(obj, verbose = FALSE)
```

Arguments

`x`	[`named list`] `lm_topic_labels` object. Alternatively each element can be passed for individual results. Individually set elements overwrite elements from `x`.
`terms`	[`list(n) of character`] List of `character` vectors, whereas each vector represents the top terms of a topic. Topics may consist of different numbers of top terms.
`prompts`	[`character(n)`] Optional. Each entry of the `character` vector contains the original prompt that was used to obtain the corresponding entry of `model_output`.
`model`	[`character(1)`] The language model used for labeling the topics.
`params`	[`named list`] Optional. Model parameters passed.
`with_token`	[`logical(1)`] Optional. Was the labeling executed using a Huggingface token?
`time`	[`numeric(1)`] Optional. Time needed for the labeling.

model_output	[character(n)] Optional. Each entry of the character vector contains the original model output obtained using the corresponding prompt from prompts.
labels	[character(n)] The extracted labels from model_output.
obj	[R object] Object to test.
verbose	[logical(1)] Should test information be given in the console?

Details

If you call as.lm_topic_labels on an object x which already is of the structure of a lm_topic_labels object (in particular a lm_topic_labels object itself), the additional arguments id, param, ... may be used to override the specific elements.

Value

[named list] lm_topic_labels object.

Examples

```
## Not run:
token = "" # please insert your hf token here
topwords_matrix = matrix(c("zidane", "figo", "kroos",
                           "gas", "power", "wind"), ncol = 2)
obj = label_topics(topwords_matrix, token = token)
obj$model
obj_modified = as.lm_topic_labels(obj, model = "It is possible to modify individual entries")
obj_modified$model

obj_modified$model = 3.5 # example for an invalid modification
is.lm_topic_labels(obj_modified, verbose = TRUE)

obj_manual = as.lm_topic_labels(terms = list(c("zidane", "figo", "kroos"),
                                             c("gas", "power", "wind")),
                                model = "manual labels",
                                labels = c("Football Players", "Energy Supply"))

## End(Not run)
```

---

label_topics *Automatically label topics using language models based on top terms*

Description

Performs an automated labeling process of topics from topic models using language models. For this, the top terms and (optionally) a short context description are used.

Usage

```
label_topics(...)

## Default S3 method:
label_topics(
  terms,
  model = "mistralai/Mixtral-8x7B-Instruct-v0.1",
  params = list(),
  token = NA_character_,
  context = "",
  sep_terms = "; ",
  max_length_label = 5L,
  prompt_type = c("json", "plain", "json-roles"),
  max_wait = 0L,
  progress = TRUE,
  ...
)

## S3 method for class 'labelTopics'
label_topics(terms, stm_type = c("prob", "frex", "lift", "score"), ...)
```

Arguments

...	additional arguments
terms	[list (k) of character] List (each list entry represents one topic) of character vectors containing the top terms representing the topics that are to be labeled. If a single character vector is passed, this is interpreted as the top terms of a single topic. If a character matrix is passed, each column is interpreted as the top terms of a topic. The outputs of the packages stm (label_topics object, please specify the type of output using the parameter stm_type) and the BTM package (list of data.frames with entries token and probability each) are also supported.
model	[character(1)] Optional. The language model to use for labeling the topics. The model must be accessible via the Huggingface API. Default is mistralai/Mixtral-8x7B-Instruct-v0.1. Other promising models are HuggingFaceH4/zephyr-7b-beta or tiiuae/falcon-7b-instruct. To find more models see: https://huggingface.co/models?other=conversational&sort=likes.
params	[named list] Optional. Model parameters to pass. Default parameters for common models are given in the details section.
token	[character(1)] Optional. If you want to address the Huggingface API with a Huggingface token, enter it here. The main advantage of this is a higher rate limit.

`context` [character(1)]
Optional.
Explanatory context for the topics to be labeled. Using a (very) brief explanation of the thematic context may greatly improve the usefulness of automatically generated topic labels.

`sep_terms` [character(1)]
How should the top terms of a single topic be separated in the generated prompts? Default is separation via semicolon and space.

`max_length_label`
[integer(1)]
What is the maximum number of words a label should consist of? Default is five words.

`prompt_type` [character(1)]
Which prompt type should be applied. We implemented various prompt types that differ mainly in how the response of the language model is requested. Examples are given in the details section. Default is the request of a json output.

`max_wait` [integer(1)]
In the case that the rate limit on Huggingface is reached: How long (in minutes) should the system wait until it asks the user whether to continue (in other words: to wait). The default is zero minutes, i.e the user is asked every time the rate limit is reached.

`progress` [logical(1)]
Should a nice progress bar be shown? Turning it off, could lead to significantly faster calculation. Default ist `TRUE`.

`stm_type` [character(1)]
For stm topics, which type of word weighting should be used? Default is "prob".

Details

The function builds helpful prompts based on the top terms and sends these prompts to language models on Huggingface. The output is in turn post-processed so that the labels for each topic are extracted automatically. If the automatically extracted labels show any errors, they can alternatively be extracted using custom functions or manually from the original output of the model using the `model_output` entry of the lm_topic_labels object.

Implemented default parameters for the models `HuggingFaceH4/zephyr-7b-beta`, `tiiuae/falcon-7b-instruct`, and `mistralai/Mixtral-8x7B-Instruct-v0.1` are:

`max_new_tokens` 300

`return_full_text FALSE`

Implemented prompt types are:

`json` the language model is asked to respond in JSON format with a single field called 'label', specifying the best label for the topic

`plain` the language model is asked to return an answer that should only consist of the best label for the topic

`json-roles` the language model is asked to respond in JSON format with a single field called 'label', specifying the best label for the topic; in addition, the model is queried using identifiers for <|user|> input and the beginning of the <|assistant|> output

Value

[named `list`] `lm_topic_labels` object.

Examples

```
## Not run:
token = "" # please insert your hf token here
topwords_matrix = matrix(c("zidane", "figo", "kroos",
                           "gas", "power", "wind"), ncol = 2)
label_topics(topwords_matrix, token = token)
label_topics(list(c("zidane", "figo", "kroos"),
                  c("gas", "power", "wind")),
             token = token)
label_topics(list(c("zidane", "figo", "ronaldo"),
                  c("gas", "power", "wind")),
             token = token)

label_topics(list("wind", "greta", "hambach"),
             token = token)
label_topics(list("wind", "fire", "air"),
             token = token)
label_topics(list("wind", "feuer", "luft"),
             token = token)
label_topics(list("wind", "feuer", "luft"),
             context = "Elements of the Earth",
             token = token)

## End(Not run)
```

Index