Package 'bertopicr'

January 22, 2026

Title Topic Modeling with 'BERTopic'

Version 0.3.6

Description Provides topic modeling and visualization by interfacing with the 'BERTopic' library for 'Python' via 'reticulate'. See Grootendorst (2022) <doi:10.48550/arXiv.2203.05794>.

Imports dplyr, tidyr, purrr, utils, reticulate, stringr, tibble, htmltools, readr, rlang

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Suggests ggplot2, knitr, rmarkdown, tictoc, wordcloud2

SystemRequirements Python (>= 3.8); BERTopic and Python packages: sentence-transformers, umap-learn, hdbscan, scikit-learn

Config/Needs/website pkgdown, rmarkdown, knitr

VignetteBuilder knitr

URL `https://tpetric7.github.io/bertopicr/`

NeedsCompilation no

Author Teodor Petrič [aut, cre] (ORCID: <`https://orcid.org/0000-0002-4397-9365`>)

Maintainer Teodor Petrič <`teodor.petric@gmail.com`>

Repository CRAN

Date/Publication 2026-01-22 09:00:07 UTC

Contents

`configure_macos_homebrew_zlib`
Configure Homebrew zlib on macOS

Description

Sets DYLD_FALLBACK_LIBRARY_PATH to Homebrew's zlib lib directory. This can help reticulate find compatible libraries on macOS.

Usage

```
configure_macos_homebrew_zlib(quiet = FALSE)
```

Arguments

`quiet`	Logical. If TRUE, suppresses messages.

Value

Logical. TRUE if the environment was updated, FALSE otherwise.

`find_topics_df` *Find Topics DataFrame Function*

Description

This function finds the most similar topics to given keywords using a BERTopic model and returns the results in a data frame or tibble format.

Usage

```
find_topics_df(model, queries, top_n = 10, return_tibble = TRUE)
```

Arguments

`model`	A BERTopic model object. Must be passed from the calling environment.
`queries`	A vector of keywords or phrases to query the topics for.
`top_n`	Number of top similar topics to retrieve for each query. Default is 10.
`return_tibble`	Logical. If TRUE, returns a tibble. If FALSE, returns a data.frame. Default is TRUE.

Value

A data.frame or tibble with columns for the keyword, topics, and similarity scores for each query.

Examples

```
# Example of finding similar topics using a BERTopic model
if (exists("topic_model")) {
  queries <- c("national minority", "minority issues", "nationality issues")
  find_topics_df(model = topic_model, queries = queries, top_n = 10)
} else {
  message("No topic_model found. Please load a BERTopic model and try again.")
}
```

`get_document_info_df` *Get Document Information DataFrame*

Description

This function retrieves document information from a BERTopic model and processes it to unnest list columns, replace NA values, and consolidate columns with the same prefix.

Usage

```
get_document_info_df(model, texts, drop_expanded_columns = TRUE)
```

Arguments

`model`	A BERTopic model object.
`texts`	A character vector containing the preprocessed texts to be passed to the BERTopic model.
`drop_expanded_columns`	Logical. If TRUE, drops the expanded columns after consolidation. Default is TRUE.

Value

A data.frame or tibble with unnested and consolidated columns.

Examples

```
if (exists("topic_model") && exists("texts_cleaned")) {
  document_info_df <- get_document_info_df(
    model = topic_model,
    texts = texts_cleaned,
    drop_expanded_columns = TRUE
  )
  print(document_info_df)
} else {
  message("No topic_model/texts_cleaned found. Please train or load a model first.")
}
```

`get_most_representative_docs`

Get Most Representative Documents for a Specific Topic

Description

This function filters a given data frame to select the most representative documents for a specified topic based on their probability scores. The documents are sorted by relevance in descending order, and the top n documents are returned.

Usage

```
get_most_representative_docs(df, topic_nr, n_docs = 5)
```

Arguments

`df`	A data frame containing at least the columns 'Topic', 'Document', and 'probs'.
`topic_nr`	An integer specifying the topic number to filter the documents.
`n_docs`	An integer specifying the number of top representative documents to return. Defaults to 5.

Value

A vector of the most representative documents corresponding to the specified topic. If the number of documents available is less than `n_docs`, all available documents are returned.

Examples

```
if (exists("df_docs")) {
  # df_docs must contain columns Topic, Document, and probs
  get_most_representative_docs(df_docs, topic_nr = 3, n_docs = 5)
} else {
  message("No df_docs found. Create it before running this example.")
}
```

---

get_representative_docs_custom

Get Representative Documents for a Specific Topic

Description

This function filters a given data frame to select a specified number of representative documents from a particular topic. It uses random sampling to select the documents.

Usage

```
get_representative_docs_custom(df, topic_nr, n_docs)
```

Arguments

`df`	A data frame containing at least the columns 'Topic' and 'Document'.
`topic_nr`	An integer specifying the topic number to filter the documents.
`n_docs`	An integer specifying the number of documents to sample for the specified topic.

Value

A vector of sampled documents corresponding to the specified topic.

Examples

```
if (exists("df_docs")) {
  # df_docs must contain columns Topic, Document, and probs
  get_representative_docs_custom(df_docs, topic_nr = 3, n_docs = 5)
} else {
  message("No df_docs found. Create it before running this example.")
}
```

get_topics_df — *Get Topics DataFrame Function*

Description

This function retrieves all topics from a BERTopic model and converts them into a data frame or tibble format.

Usage

```
get_topics_df(model, return_tibble = TRUE)
```

Arguments

`model`	A BERTopic model object. Must be passed from the calling environment.
`return_tibble`	Logical. If TRUE, returns a tibble. If FALSE, returns a data.frame. Default is TRUE.

Value

A data.frame or tibble with columns for the word, score, and topic number across all topics.

Examples

```
if (exists("topic_model")) {
  topics_df <- get_topics_df(model = topic_model)
  print(topics_df)
} else {
  message("No topic_model found. Please train or load a model first.")
}
```

get_topic_df — *Get Topic DataFrame Function*

Description

This function retrieves a specified number of words with high probability for a given topic number from a BERTopic model and returns the results in a data frame or tibble format.

Usage

```
get_topic_df(model, topic_number = 0, top_n = 10, return_tibble = TRUE)
```

Arguments

`model`	A BERTopic model object. Must be passed from the calling environment.
`topic_number`	The topic number for which words and scores are retrieved.
`top_n`	Number of top words to retrieve for the specified topic. Default is 10. If greater than 10, it will be set to 10 as BERTopic returns a maximum of 10 words.
`return_tibble`	Logical. If TRUE, returns a tibble. If FALSE, returns a data.frame. Default is TRUE.

Value

A data.frame or tibble with columns for the word, score, and topic number.

Examples

```
# Example usage:
if (exists("topic_model")) {
  topic_df <- get_topic_df(model = topic_model, topic_number = 3, top_n = 5)
  print(topic_df)
} else {
  message("No topic_model found. Please load a BERTopic model and try again.")
}
```

`get_topic_info_df` *Get Topic Information DataFrame*

Description

This function retrieves topic information from a BERTopic model and processes it to unnest list columns, replace NA values, and consolidate columns with the same prefix.

Usage

```
get_topic_info_df(model, drop_expanded_columns = TRUE)
```

Arguments

`model`	A BERTopic model object.
`drop_expanded_columns`	Logical. If TRUE, drops the expanded columns after consolidation. Default is TRUE.

Value

A data.frame or tibble with unnested and consolidated columns.

Examples

```
if (exists("topic_model")) {
  topic_info_df <- get_topic_info_df(model = topic_model,
                                     drop_expanded_columns = TRUE)
  print(topic_info_df)
} else {
  message("No topic_model found. Please train or load a model first.")
}
```

`load_bertopic_model` *Load a BERTopic Model Bundle*

Description

Load a BERTopic model saved with `save_bertopic_model()` along with its companion RDS file containing R-side extras.

Usage

```
load_bertopic_model(path, embedding_model = NULL)
```

Arguments

`path`	Directory path where the Python model was saved.
`embedding_model`	Optional embedding model to pass through to `BERTopic$load()` when the embedding model is not serialized.

Value

A list with two elements: `model` (the BERTopic model) and `extras` (the R-side data saved in the companion RDS file).

Examples

```
if (dir.exists("topic_model")) {
  loaded <- load_bertopic_model("topic_model")
  print(loaded$extras)
} else {
  message("No saved model found at 'topic_model'.")
}
```

`save_bertopic_model` *Save a BERTopic Model Bundle*

Description

Persist a trained BERTopic model to disk and store R-side extras in a companion RDS file. This is the recommended way to reuse a model across sessions when working through reticulate.

Usage

```
save_bertopic_model(topic_model, path)
```

Arguments

`topic_model`	A list returned by `train_bertopic_model()`. Must contain a Python BERTopic model at `topic_model$model`. Optional extras such as probabilities, reduced embeddings, topics over time, or topics per class are saved when present and set to `NULL` otherwise.
`path`	Directory path to write the Python model to. The RDS companion file is saved as `paste0(path, "_extras.rds")`.

Value

Invisibly returns `TRUE` after successful write.

Examples

```
if (exists("topic_model")) {
  save_bertopic_model(topic_model, "topic_model")
} else {
  message("No topic_model found. Please train or load a model first.")
}
```

`setup_python_environment` *Set Up Python Environment for BERTopic*

Description

This function sets up a Python environment with all required packages for using the BERTopic model within the R package. It can create and activate a virtualenv or conda environment and then install the bundled requirements.

Usage

```
setup_python_environment(
  envname = "r-bertopic",
  python_path = NULL,
  method = c("virtualenv", "conda"),
  python_version = NULL,
  upgrade = TRUE,
  extra_packages = NULL
)
```

Arguments

`envname`	The name of the Python environment. Default is "r-bertopic".
`python_path`	Optional path to a specific Python executable (virtualenv only).
`method`	Environment type to create and use. One of "virtualenv" or "conda".
`python_version`	Optional Python version for conda (e.g. "3.10").
`upgrade`	Logical. If TRUE, passes –upgrade to pip installs. Default is TRUE.
`extra_packages`	Optional character vector of additional Python packages to install.

Value

Invisibly returns the active Python configuration.

`train_bertopic_model` *Train a BERTopic Model*

Description

This function creates embeddings with sentence-transformers, configures UMAP, HDBSCAN, and CountVectorizer, optionally wires a representation model, and fits a BERTopic model from R. The returned model can be used with bertopicr helpers.

Usage

```
train_bertopic_model(
  docs,
  embedding_model = "Qwen/Qwen3-Embedding-0.6B",
  embeddings = NULL,
  embedding_batch_size = 32,
  embedding_show_progress = TRUE,
  umap_model = NULL,
  umap_n_neighbors = 15,
  umap_n_components = 5,
  umap_min_dist = 0,
  umap_metric = "cosine",
  umap_random_state = 42,
  hdbscan_model = NULL,
  hdbscan_min_cluster_size = 50,
  hdbscan_min_samples = 20,
  hdbscan_metric = "euclidean",
  hdbscan_cluster_selection_method = "eom",
  hdbscan_gen_min_span_tree = TRUE,
  hdbscan_prediction_data = TRUE,
  hdbscan_core_dist_n_jobs = 1,
  vectorizer_model = NULL,
  stop_words = "all_stopwords",
  ngram_range = c(1, 3),
  min_df = 2L,
  max_df = 50L,
  max_features = 10000,
  strip_accents = NULL,
  decode_error = "strict",
  encoding = "UTF-8",
  representation_model = c("none", "keybert", "mmr", "ollama"),
  representation_params = list(),
  ollama_model = NULL,
  ollama_base_url = "http://localhost:11434/v1",
  ollama_api_key = "ollama",
  ollama_client_params = list(),
  ollama_prompt = NULL,
  top_n_words = 200L,
  calculate_probabilities = TRUE,
  verbose = TRUE,
  seed = NULL,
  timestamps = NULL,
  topics_over_time_nr_bins = 20L,
  topics_over_time_global_tuning = TRUE,
  topics_over_time_evolution_tuning = TRUE,
  classes = NULL,
  compute_reduced_embeddings = TRUE,
  reduced_embedding_n_neighbors = 10L,
  reduced_embedding_min_dist = 0,
  reduced_embedding_metric = "cosine",
  compute_hierarchical_topics = TRUE,
  bertopic_args = list()
)
```

Arguments

`docs`	Character vector of documents to model.
`embedding_model`	Sentence-transformers model name or local path.
`embeddings`	Optional precomputed embeddings (matrix or array).

`embedding_batch_size`	Batch size for embedding encoding.
`embedding_show_progress`	Logical. Show embedding progress bar.
`umap_model`	Optional pre-built UMAP Python object. If NULL, one is created.
`umap_n_neighbors`	Number of neighbors for UMAP.
`umap_n_components`	Number of UMAP components.
`umap_min_dist`	UMAP min_dist parameter.
`umap_metric`	UMAP metric.
`umap_random_state`	Random state for UMAP.
`hdbscan_model`	Optional pre-built HDBSCAN Python object. If NULL, one is created.
`hdbscan_min_cluster_size`	HDBSCAN min_cluster_size.
`hdbscan_min_samples`	HDBSCAN min_samples.
`hdbscan_metric`	HDBSCAN metric.
`hdbscan_cluster_selection_method`	HDBSCAN cluster selection method.
`hdbscan_gen_min_span_tree`	HDBSCAN gen_min_span_tree.
`hdbscan_prediction_data`	Logical. Whether to generate prediction data.
`hdbscan_core_dist_n_jobs`	HDBSCAN core_dist_n_jobs.
`vectorizer_model`	Optional pre-built CountVectorizer Python object.
`stop_words`	Stop words for CountVectorizer. Use "all_stopwords" to load the bundled multilingual list, "english", or a character vector.
`ngram_range`	Length-2 integer vector for n-gram range.
`min_df`	Minimum document frequency for CountVectorizer.
`max_df`	Maximum document frequency for CountVectorizer.
`max_features`	Maximum features for CountVectorizer.
`strip_accents`	Passed to CountVectorizer. Use NULL to preserve umlauts.
`decode_error`	Passed to CountVectorizer when decoding input bytes.
`encoding`	Text encoding for CountVectorizer (defaults to "utf-8").
`representation_model`	Representation model to use: "none", "keybert", "mmr", or "ollama".
`representation_params`	Named list of parameters passed to the representation model.

`ollama_model`	Ollama model name when representation_model = "ollama".
`ollama_base_url`	Base URL for the Ollama OpenAI-compatible endpoint.
`ollama_api_key`	API key placeholder for the Ollama OpenAI-compatible endpoint.
`ollama_client_params`	Named list of extra parameters passed to openai$OpenAI().
`ollama_prompt`	Optional prompt template for the Ollama OpenAI representation.
`top_n_words`	Number of top words per topic to keep in the model.
`calculate_probabilities`	Logical. Whether to calculate topic probabilities.
`verbose`	Logical. Verbosity for BERTopic.
`seed`	Optional random seed.
`timestamps`	Optional vector of timestamps (Date/POSIXt/ISO strings or integer) for topics over time. Defaults to NULL (topics over time disabled).
`topics_over_time_nr_bins`	Number of bins for topics_over_time.
`topics_over_time_global_tuning`	Logical. Whether to enable global tuning for topics_over_time.
`topics_over_time_evolution_tuning`	Logical. Whether to enable evolution tuning for topics_over_time.
`classes`	Optional vector of class labels (character or factor) for topics per class. Defaults to NULL (topics per class disabled).
`compute_reduced_embeddings`	Logical. If TRUE, computes 2D and 3D UMAP reductions.
`reduced_embedding_n_neighbors`	Number of neighbors for reduced embeddings.
`reduced_embedding_min_dist`	UMAP min_dist for reduced embeddings.
`reduced_embedding_metric`	UMAP metric for reduced embeddings.
`compute_hierarchical_topics`	Logical. If TRUE, computes hierarchical topics.
`bertopic_args`	Named list of extra arguments passed to BERTopic().

Value

A list with elements model, topics, probabilities, embeddings, reduced_embeddings_2d, reduced_embeddings_3d, hierarchical_topics, topics_over_time, and topics_per_class.

Examples

```
if (requireNamespace("reticulate", quietly = TRUE) &&
    reticulate::py_available(initialize = FALSE) &&
    reticulate::py_module_available("bertopic")) {
  setup_python_environment()
  sample_path <- system.file("extdata", "spiegel_sample.rds", package = "bertopicr")
  df <- readr::read_rds(sample_path)
  texts <- df$text_clean[seq_len(500)]
  fit <- train_bertopic_model(
    texts,
    embedding_model = "Qwen/Qwen3-Embedding-0.6B",
    top_n_words = 3L
  )
  visualize_topics(fit$model, filename = "intertopic_distance_map", auto_open = FALSE)
} else {
  message("Python/bertopic not available. Skipping this example.")
}
```

`visualize_barchart` *Visualize BERTopic Bar Chart*

Description

This function visualizes the topics of a BERTopic model using Plotly and saves the output as an interactive HTML file. It checks for required Python modules and allows for custom file naming.

Usage

```
visualize_barchart(
  model,
  filename = "topics_topwords_interactive_barchart",
  open_file = FALSE
)
```

Arguments

`model`	A BERTopic model object. Must be passed from the calling environment.
`filename`	A character string specifying the name of the HTML file to save the bar chart. Default is "topics_topwords_interactive_barchart". The `.html` extension is added automatically if not provided.
`open_file`	Logical. If TRUE, opens the HTML file after saving. Default is FALSE.

Value

Displays the interactive bar chart within the R environment and saves it as an HTML file.

Examples

```
if (exists("topic_model")) {
  visualize_barchart(model = topic_model, filename = "custom_barchart",
                     open_file = TRUE)
} else {
  message("No topic_model found. Please train or load a model first.")
}
```

visualize_distribution — *Visualize Topic Distribution for a Specific Document using BERTopic*

Description

This function visualizes the topic distribution for a specific document from a BERTopic model using Python's Plotly library. The visualization is saved as an interactive HTML file, which can be opened and viewed in a web browser.

Usage

```
visualize_distribution(
  model,
  text_id = 1,
  probabilities,
  filename = "topic_dist_interactive",
  auto_open = FALSE
)
```

Arguments

`model`	A BERTopic model object. The model must have the method `visualize_distribution`.
`text_id`	An integer specifying the index of the document for which the topic distribution is visualized. Default is 1. Must be a positive integer and a valid index within the `probabilities` matrix.
`probabilities`	A matrix or data frame of topic probabilities, with rows corresponding to documents and columns to topics. Each element represents the probability of a topic for a given document.
`filename`	A character string specifying the name of the HTML file to save the visualization. Default is "topic_dist_interactive". The .html extension will be added automatically.
`auto_open`	Logical. If TRUE, the HTML file will automatically open in the browser. Default is FALSE.

Value

The function does not return a value but saves an HTML file containing the visualization and displays it in the current R environment.

Examples

```
if (exists("topic_model") && exists("probs")) {
  visualize_distribution(
    model = topic_model,
    text_id = 1,
    probabilities = probs,
    filename = "custom_filename",
    auto_open = TRUE
  )
} else {
  message("No topic_model/probs found. Please train or load a model first.")
}
```

`visualize_documents` *Visualize Documents in Reduced Embedding Space*

Description

This function generates a visualization of documents using a pre-trained BERTopic model. It uses UMAP to reduce the dimensionality of embeddings and Plotly for interactive visualizations.

Usage

```
visualize_documents(
  model = topic_model,
  texts = texts_cleaned,
  reduced_embeddings = reduced_embeddings,
  custom_labels = FALSE,
  hide_annotation = TRUE,
  filename = "visualize_documents",
  auto_open = FALSE
)
```

Arguments

`model`	A BERTopic model object. Default is 'topic_model'.
`texts`	A list or vector of cleaned text documents to visualize. Default is 'texts_cleaned'.
`reduced_embeddings`	A matrix of reduced-dimensionality embeddings. Typically generated using UMAP. Default is 'reduced_embeddings'.
`custom_labels`	A logical value indicating whether to use custom labels for topics. Default is FALSE.
`hide_annotation`	A logical value indicating whether to hide annotations in the plot. Default is TRUE.

`filename`	A string specifying the name of the HTML file to save the visualization. Default is "visualize_documents".
`auto_open`	A logical value indicating whether to automatically open the HTML file after saving. Default is FALSE.

Value

A Plotly visualization of the documents, displayed as an HTML file within the R environment.

Examples

```
if (exists("topic_model") && exists("texts_cleaned") && exists("reduced_embeddings")) {
  visualize_documents(model = topic_model,
                      texts = texts_cleaned,
                      reduced_embeddings = reduced_embeddings,
                      custom_labels = FALSE,
                      hide_annotation = TRUE,
                      filename = "visualize_documents",
                      auto_open = FALSE)
} else {
  message("Missing topic_model/texts_cleaned/reduced_embeddings. Train a model first.")
}
```

`visualize_documents_2d` *Visualize Documents in 2D Space using BERTopic*

Description

This function generates a 3D visualization of documents using a pre-trained BERTopic model and UMAP dimensionality reduction. It uses Plotly for interactive visualizations and saves the output as an HTML file.

Usage

```
visualize_documents_2d(
  model,
  texts,
  reduced_embeddings,
  custom_labels = FALSE,
  hide_annotation = TRUE,
  tooltips = c("Topic", "Name", "Probability", "Text"),
  filename = "visualize_documents_2d",
  auto_open = FALSE
)
```

Arguments

`model`	A BERTopic model object. Default is 'topic_model'.
`texts`	A character vector or list of cleaned text documents to visualize.
`reduced_embeddings`	A matrix or data frame of reduced-dimensionality embeddings (2D). Typically generated using UMAP.
`custom_labels`	Logical. If TRUE, custom topic labels are used. Default is FALSE.
`hide_annotation`	Logical. If TRUE, hides annotations on the plot. Default is TRUE.
`tooltips`	A character vector of tooltips for hover information. Default is c("Topic", "Name", "Probability", "Text").
`filename`	A character string specifying the name of the HTML file to save the visualization. Default is "visualize_documents_2d". The `.html` extension is automatically added if not provided.
`auto_open`	Logical. If TRUE, opens the HTML file in the browser after saving. Default is FALSE.

Value

The function does not return a value but saves an HTML file containing the visualization and displays it in the current R environment.

Examples

```
if (exists("topic_model") && exists("texts_cleaned") && exists("embeddings")) {
  visualize_documents_2d(model = topic_model,
    texts = texts_cleaned,
    reduced_embeddings = embeddings,
    custom_labels = FALSE,
    hide_annotation = TRUE,
    filename = "plot",
    auto_open = TRUE)
} else {
  message("Missing topic_model/texts_cleaned/embeddings. Train a model first.")
}
```

`visualize_documents_3d` *Visualize Documents in 3D Space using BERTopic*

Description

This function generates a 3D visualization of documents using a pre-trained BERTopic model and UMAP dimensionality reduction. It uses Plotly for interactive visualizations and saves the output as an HTML file.

Usage

```
visualize_documents_3d(
  model,
  texts,
  reduced_embeddings,
  custom_labels = FALSE,
  hide_annotation = TRUE,
  tooltips = c("Topic", "Name", "Probability", "Text"),
  filename = "visualize_documents_3d",
  auto_open = FALSE
)
```

Arguments

`model`	A BERTopic model object. Default is 'topic_model'.
`texts`	A character vector or list of cleaned text documents to visualize.
`reduced_embeddings`	A matrix or data frame of reduced-dimensionality embeddings (3D). Typically generated using UMAP.
`custom_labels`	Logical. If TRUE, custom topic labels are used. Default is FALSE.
`hide_annotation`	Logical. If TRUE, hides annotations on the plot. Default is TRUE.
`tooltips`	A character vector of tooltips for hover information. Default is c("Topic", "Name", "Probability", "Text").
`filename`	A character string specifying the name of the HTML file to save the visualization. Default is "visualize_documents_3d". The `.html` extension is automatically added if not provided.
`auto_open`	Logical. If TRUE, opens the HTML file in the browser after saving. Default is FALSE.

Value

The function does not return a value but saves an HTML file containing the visualization and displays it in the current R environment.

Examples

```
if (exists("topic_model") && exists("texts_cleaned") && exists("embeddings")) {
  visualize_documents_3d(model = topic_model,
    texts = texts_cleaned,
    reduced_embeddings = embeddings,
    custom_labels = FALSE,
    hide_annotation = TRUE,
    filename = "plot",
    auto_open = TRUE)
} else {
  message("Missing topic_model/texts_cleaned/embeddings. Train a model first.")
}
```

`visualize_heatmap` *Visualize Topic Similarity Heatmap using BERTopic*

Description

This function visualizes the topic similarity heatmap of topics from a BERTopic model using Python's Plotly library. The visualization is saved as an interactive HTML file, which can be opened and viewed in a web browser.

Usage

```
visualize_heatmap(
  model,
  filename = "topics_similarity_heatmap",
  auto_open = FALSE
)
```

Arguments

`model`	A BERTopic model object. The model must have the method `visualize_heatmap`.
`filename`	A character string specifying the name of the HTML file to save the visualization. The default value is "topics_similarity_heatmap". The filename should not contain illegal characters. The `.html` extension is added automatically if not provided.
`auto_open`	Logical. If TRUE, opens the HTML file after saving. Default is FALSE.

Value

The function does not return a value but saves an HTML file containing the visualization and displays it in the current R environment.

Examples

```
if (exists("topic_model")) {
 visualize_heatmap(model = topic_model, filename = "topics_similarity_heatmap", auto_open = FALSE)
} else {
  message("No topic_model found. Please train or load a model first.")
}
```

`visualize_hierarchy` *Visualize Topic Hierarchy Nodes using BERTopic*

Description

This function visualizes the hierarchical clustering of topics from a BERTopic model. If a hierarchical topics DataFrame is provided, it uses this for visualization; otherwise, it visualizes directly from the model. The visualization is saved as an interactive HTML file, which can be opened and viewed in a web browser.

Usage

```
visualize_hierarchy(
  model,
  hierarchical_topics = NULL,
  filename = "topic_hierarchy",
  auto_open = TRUE
)
```

Arguments

`model`	A BERTopic model object. The model must have the method `visualize_hierarchy`.
`hierarchical_topics`	Optional. A hierarchical topics DataFrame created using the BERTopic model's `hierarchical_topics` method. If provided, this object is used to generate the hierarchy visualization.
`filename`	A character string specifying the name of the HTML file to save the visualization. The default value is "topic_hierarchy". The filename should not contain illegal characters.
`auto_open`	Logical. If `TRUE`, the HTML file will be opened automatically after being saved. Default is `TRUE`.

Value

The function does not return a value but saves an HTML file containing the visualization and displays it in the current R environment.

Examples

```
if (exists("topic_model")) {
  visualize_hierarchy(model = topic_model, filename = "topic_hierarchy",
                      auto_open = TRUE)
  if (exists("hierarchical_topics")) {
    visualize_hierarchy(model = topic_model,
                        hierarchical_topics = hierarchical_topics,
                        filename = "topic_hierarchy",
                        auto_open = TRUE)
  }
} else {
  message("No topic_model found. Please train or load a model first.")
}
```

`visualize_topics` *Visualize Topics using BERTopic*

Description

This function visualizes the intertopic distance map of topics from a BERTopic model using Python's Plotly library. The visualization is saved as an interactive HTML file, which can be opened and viewed in a web browser.

Usage

```
visualize_topics(
  model,
  filename = "intertopic_distance_map",
  auto_open = FALSE
)
```

Arguments

`model`	A BERTopic model object. The model must have the method `visualize_topics`.
`filename`	A character string specifying the name of the HTML file to save the visualization. The default value is "intertopic_distance_map". The filename should not contain illegal characters. The `.html` extension is added automatically if not provided.
`auto_open`	Logical. If TRUE, opens the HTML file after saving. Default is FALSE.

Value

The function does not return a value but saves an HTML file containing the visualization and displays it in the current R environment.

Examples

```
if (exists("topic_model")) {
  visualize_topics(model = topic_model, filename = "plot", auto_open = TRUE)
} else {
  message("No topic_model found. Please train or load a model first.")
}
```

visualize_topics_over_time
Visualize Topics Over Time using BERTopic

Description

This function visualizes topics over time from a BERTopic model using Python's Plotly library. The visualization is saved as an interactive HTML file, which can be opened and viewed in a web browser.

Usage

```
visualize_topics_over_time(
  model,
  topics_over_time_model,
  top_n_topics = 20,
  filename = "topics_over_time"
)
```

Arguments

`model`	A BERTopic model object. The model must have the method `visualize_topics_over_time`.
`topics_over_time_model`	A topics-over-time model object created using the BERTopic model.
`top_n_topics`	An integer specifying the number of top topics to display in the visualization. Default is 20. Must be a positive integer.
`filename`	A character string specifying the name of the HTML file to save the visualization. The default value is "topics_over_time". The filename should not contain illegal characters.

Value

The function does not return a value but saves an HTML file containing the visualization and displays it in the current R environment.

Examples

```
if (exists("topic_model") && exists("topics_over_time")) {
  visualize_topics_over_time(model = topic_model,
                             topics_over_time_model = topics_over_time,
                             top_n_topics = 5,
                             filename = "plot")
} else {
  message("No topic_model/topics_over_time found. Train a model first.")
}
```

visualize_topics_per_class

Visualize Topics per Class

Description

This function visualizes the distribution of topics per class using a pre-trained BERTopic model. The visualization is generated using the Plotly Python package and displayed within an R environment.

Usage

```
visualize_topics_per_class(
  model = topic_model,
  topics_per_class = topics_per_class,
  start = 0,
  end = 10,
  filename = "topics_per_class",
  auto_open = TRUE
)
```

Arguments

`model`	A BERTopic model object. Default is 'topic_model'.
`topics_per_class`	A data frame or list containing the topics per class data. Default is 'topics_per_class'.
`start`	An integer specifying the starting index of the topics to visualize. Default is 0.
`end`	An integer specifying the ending index of the topics to visualize. Default is 10.
`filename`	A string specifying the name of the HTML file to save the visualization. Default is "topics_per_class".
`auto_open`	A logical value indicating whether to automatically open the HTML file after saving. Default is TRUE.

Value

A Plotly visualization of the topics per class, displayed as an HTML file within the R environment.

Examples

```
if (exists("topic_model") && exists("topics_per_class")) {
  visualize_topics_per_class(model = topic_model,
                             topics_per_class = topics_per_class,
                             start = 0, end = 7,
                             filename = "plot",
                             auto_open = TRUE)
} else {
  message("No topic_model/topics_per_class found. Train a model first.")
}
```

Index