Package: openalexVectorComp (via r-universe)

November 18, 2026

Type Package

Title Embedding Vectorization and Distance-Based Scoring Workflows

Version 0.3.6

Description R-first orchestration for text vectorization (embeddings), embedding distance computation, and distance-based scoring workflows. Supports backend-neutral embedding providers (HF, OpenAI, TEI), prototype cosine-distance scoring, reference-area distance scoring, and threshold calibration utilities.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports cli, httr2, dplyr, arrow, digest, yaml, uwot, stats, utils, ggplot2, jsonlite

Suggests knitr, quarto, roxygen2, rmarkdown, keyring, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr, quarto

URL https://github.com/openalexPro/openalexVectorComp, https://openalexpro.github.io/openalexVectorComp/, https://doi.org/10.5281/zenodo.19607514

BugReports https://github.com/openalexPro/openalexVectorComp/issues

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libxml2-dev libssl-dev

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

Date/Publication 2026-11-18 16:06:37 UTC

RemoteUrl https://github.com/openalexPro/openalexVectorComp

RemoteRef main

RemoteSha ca5f168e132e87d90065a959b754a118b99bdbf2

Contents

backend_config . . . 2
backend_embed_texts . . . 3
backend_info . . . 4
backend_read . . . 4
backend_save . . . 5
backend_specter2_tei . . . 5
batch_collect_openai . . . 6
batch_status_openai . . . 7
batch_submit_openai . . . 7
calibrate_threshold . . . 8
clean_abstract_for_embedding . . . 10
demo_finalize_openai_batch . . . 11
distance_cosine . . . 11
distance_reference_cosine . . . 12
distance_ridge . . . 13
distances . . . 14
embed_corpus . . . 15
embed_texts . . . 16
fit_ridge . . . 16
plot_embeddings_pca . . . 17
plot_embeddings_umap . . . 18
run_demo_openai . . . 19
run_demo_openalex . . . 20
score_reference_cosine . . . 21
score_ridge . . . 22
similarity_cosine . . . 22

Index **23**

`backend_config` *Build embedding backend configuration*

Description

Creates a configuration object used by the embedding backend adapter.

Usage

```
backend_config(
  provider = c("hf", "openai", "tei"),
  base_url = NULL,
  model = NULL,
  max_batch_size = NULL,
  timeout = 60,
  retries = 3,
  tei_url = NULL
)
```

Arguments

provider	Backend provider: "hf", "openai", or "tei".
base_url	Provider base URL. If NULL, provider defaults are used.
model	Optional model id. If NULL, provider defaults are used.
max_batch_size	Optional max texts per HTTP request.
timeout	Request timeout (seconds) used by backends that support it.
retries	Number of retry attempts for transient failures.
tei_url	Optional TEI compatibility argument. If provided, it is treated as the full embedding endpoint URL and overrides base_url.

Value

A named list with backend configuration.

---

backend_embed_texts *Embed texts via configured backend*

Description

Uses the configured backend adapter to embed a character vector of texts. For authenticated providers, set OVC_API_TOKEN in the environment. The adapter sends it as a bearer token.

Usage

```
backend_embed_texts(texts, backend = backend_config())
```

Arguments

texts	Character vector of input texts.
backend	Backend configuration from backend_config().

Value

Numeric matrix with one row per text and columns V1..Vd.

backend_info *Get embedding backend model/service information*

Description

Returns normalized backend metadata used by the pipeline.

Usage

```
backend_info(backend = backend_config())
```

Arguments

backend	Backend configuration from backend_config().

Value

A list with fields provider, model_id, dim, max_batch_size, and raw.

backend_read *Read backend configuration from YAML*

Description

Reads backend configuration from a YAML file and returns a normalized object in the same format as backend_config().

Usage

```
backend_read(fn = "embed_model.yaml")
```

Arguments

fn	Path to YAML file. Defaults to "embed_model.yaml".

Details

Supports both the current flat format and legacy nested metadata format.

Value

A backend configuration list compatible with backend_config().

backend_save *Save backend configuration to YAML*

Description

Writes a backend configuration (same shape as returned by `backend_config()`) to YAML.

Usage

```
backend_save(backend = backend_config(), fn = "embed_model.yaml")
```

Arguments

backend	Backend configuration from `backend_config()`.
fn	Output YAML file path. Defaults to `"embed_model.yaml"`.

Value

Invisibly returns `fn`.

backend_specter2_tei *Backend preset for a local TEI server serving the merged SPECTER2 proximity model*

Description

Convenience wrapper around `backend_config()` for a SPECTER2 setup served by a local TEI (text-embeddings-inference) server. The model itself must be prepared and started externally (see `inst/scripts/prepare_specter2_merged.py` and `inst/scripts/start_tei_specter2.sh`, and the `specter2-setup` vignette).

Usage

```
backend_specter2_tei(
  port = 8080L,
  host = "localhost",
  model = "allenai/specter2_proximity_merged"
)
```

Arguments

port	Port that TEI is listening on. Defaults to `8080`.
host	Host for TEI. Defaults to `"localhost"`.
model	Provenance label for the served model. Defaults to `"allenai/specter2_proximity_merged"`.

Details

The `model` argument is metadata only and is recorded in `embed_model.yaml` and parquet partition paths; TEI itself loads whichever model it was started with.

Value

A backend configuration list compatible with `backend_config()`.

`batch_collect_openai` *Collect completed OpenAI batch embedding jobs*

Description

Collect completed OpenAI batch embedding jobs

Usage

```
batch_collect_openai(
  project_dir,
  backend = backend_config(provider = "openai"),
  label = "corpus",
  verbose = TRUE
)
```

Arguments

`project_dir`	Project root directory.
`backend`	Backend configuration from `backend_config()`. Must use `provider = "openai"`.
`label`	Embedding label partition to collect into.
`verbose`	Logical; print progress messages.

Value

Invisibly returns a list with collection summary.

batch_status_openai *Inspect OpenAI batch state for a label*

Description

Inspect OpenAI batch state for a label

Usage

```
batch_status_openai(project_dir, label = "corpus", refresh_remote = TRUE)
```

Arguments

project_dir	Project root directory.
label	Embedding label.
refresh_remote	Logical; if TRUE, refresh non-terminal job statuses from OpenAI before returning.

Value

A data frame with one row per tracked job.

batch_submit_openai *Submit OpenAI Batch jobs for corpus embeddings (asynchronous)*

Description

Preprocesses corpus text, performs preflight request-size checks, splits work into compliant OpenAI batch jobs, submits them, and returns immediately.

Usage

```
batch_submit_openai(
  project_dir,
  backend = backend_config(provider = "openai"),
  corpus_name = "corpus",
  label = corpus_name,
  batch_size = 5000,
  delete_existing = FALSE,
  text_preprocessor = clean_abstract_for_embedding,
  cleaner_args = list(),
  save_text = TRUE,
  max_requests_per_job = 20000L,
  max_job_bytes = 150 * 1024^2,
  completion_window = "24h",
  verbose = TRUE
)
```

Arguments

project_dir	Project root directory.
backend	Backend configuration from backend_config(). Must use provider = "openai".
corpus_name	Folder name under project_dir containing the corpus dataset. Defaults to "corpus".
label	Embedding label partition. Defaults to corpus_name.
batch_size	Number of corpus rows per Arrow scan batch while preparing requests.
delete_existing	If TRUE, remove existing embeddings for label and existing OpenAI batch state before submitting new jobs.
text_preprocessor	Text-preparation function returning id, text, text_hash.
cleaner_args	Additional named arguments passed to text_preprocessor.
save_text	Logical; whether to keep cleaned text for downstream parquet output.
max_requests_per_job	Max requests per submitted OpenAI job. Must be <= 50000.
max_job_bytes	Max JSONL bytes per submitted OpenAI job. Must be <= 200 MB.
completion_window	OpenAI batch completion window. Defaults to "24h".
verbose	Logical; print progress messages.

Value

Invisibly returns a list with state path and submission summary.

---

calibrate_threshold *Calibrate threshold from Parquet scores by streaming batches*

Description

Sweeps candidate thresholds over scores stored in a Parquet dataset without loading all rows into memory. Uses two passes: first to determine the score range on the labeled subset; second to accumulate confusion counts across a fixed grid of thresholds. Returns the best threshold per the chosen metric.

Usage

```
calibrate_threshold(
  scores_parquet,
  score_col,
  labels_parquet,
  metric = c("f1", "precision_at_recall"),
  recall_min = 0.8,
```

```
  thresholds = NULL,
  n_thresholds = 1001,
  batch_size = 1e+05,
  verbose = TRUE
)
```

Arguments

scores_parquet	Path to a Parquet dataset (file or directory) with at least columns id and the score column.
score_col	Name of the score column to calibrate (e.g., "ensemble", "relevance_score", or "margin").
labels_parquet	Parquet dataset path with columns id and label (0/1) used for calibration labels.
metric	Optimisation target: "f1" (default) or "precision_at_recall".
recall_min	Minimum recall required when metric = "precision_at_recall".
thresholds	Optional numeric vector of thresholds to evaluate. If NULL, a regular grid between observed min/max is used (see n_thresholds).
n_thresholds	Number of thresholds to generate when thresholds is NULL (default 1001).
batch_size	Approximate Arrow scan batch size.
verbose	Logical; print progress messages.

Value

List containing the selected threshold (th) and the associated precision, recall, and f1 values.

Examples

```
## Not run:
best <- calibrate_threshold(
  scores_parquet = "output/scores/",
  score_col = "ensemble",
  labels_parquet = "output/labels/",
  batch_size = 200000
)
best$th

## End(Not run)
```

clean_abstract_for_embedding *Clean title/abstract rows into embedding-ready text*

Description

Applies lightweight rule-based cleaning to `title`/`abstract` rows and returns embedding-ready text plus a deterministic `text_hash`.

Usage

```
clean_abstract_for_embedding(
  df,
  mode = c("lenient", "balanced", "strict"),
  no_abstract_policy = c("keep_title_only", "discard", "conditional"),
  min_chars = NULL,
  min_alpha_ratio = NULL,
  placeholder_patterns = NULL,
  boilerplate_patterns = NULL,
  html_patterns = NULL,
  return_flags = TRUE
)
```

Arguments

`df`	Data frame with columns `id`, `title`, and `abstract`.
`mode`	Cleaning intensity: `"lenient"`, `"balanced"` (default), or `"strict"`.
`no_abstract_policy`	Policy when abstract is missing/invalid: `"keep_title_only"` (default), `"discard"`, or `"conditional"`.
`min_chars`	Optional minimum abstract length in characters after cleaning. If `NULL`, mode-specific defaults are used.
`min_alpha_ratio`	Optional minimum ratio of alphabetic characters in the cleaned abstract. If `NULL`, mode-specific defaults are used.
`placeholder_patterns`	Optional regex vector for placeholder abstract detection.
`boilerplate_patterns`	Optional regex vector for publisher boilerplate detection.
`html_patterns`	Optional regex vector for HTML/XML artifact detection.
`return_flags`	If `TRUE`, include provenance/quality columns.

Value

A data frame with at least columns `id`, `text`, `text_hash`. When `return_flags = TRUE`, also includes `text_quality`, `abstract_raw_present`, `abstract_kept`, `discard_reason`, and `cleaning_mode`.

demo_finalize_openai_batch

Finalize OpenAI demo batch jobs and compare direct vs batch embeddings

Description

Runs OpenAI batch status/collect for a prepared demo workspace and, when batch embeddings are available, computes direct-vs-batch vector comparison. Comparison artifacts are written to: `project/openai_batch_comparison/label=<label>/`.

Usage

```
demo_finalize_openai_batch(
  demo_dir,
  api_key = NULL,
  label = "corpus_batch",
  refresh_remote = TRUE,
  verbose = TRUE
)
```

Arguments

`demo_dir`	Demo workspace directory created by `run_demo_openai()` or `run_demo_openalex()`.
`api_key`	Optional OpenAI API key. If provided, it is set in `OVC_API_TOKEN` for the duration of this call.
`label`	Batch embedding label to finalize. Defaults to `"corpus_batch"`.
`refresh_remote`	Logical; forwarded to `batch_status_openai()`.
`verbose`	Logical; print progress messages.

Value

Invisibly returns a list containing status/collect summaries, comparison readiness, and output paths.

distance_cosine

Cosine distance between two numeric vectors

Description

Computes cosine distance as `1 - similarity_cosine(a, b)`.

Usage

```
distance_cosine(a, b)
```

Arguments

a	Numeric vector.
b	Numeric vector or numeric matrix with embeddings in rows.

Value

A single numeric distance value, or NA_real_ when undefined.

---

distance_reference_cosine

Pairwise cosine distances with centroid axis between label partitions

---

Description

Reads embeddings from a model-specific dataset and computes cosine distances between all vectors in corpus_label and all vectors in reference_label. A centroid row/column is added to the matrix:

- rows are corpus ids plus "centroid" (corpus centroid),
- columns are reference ids plus "centroid" (reference centroid).

Usage

```
distance_reference_cosine(
  project_dir,
  embeddings_dir = "model_id=BAAI_bge-small-en-v1.5",
  corpus_label = "corpus",
  reference_label = "reference",
  batch_size = 1e+05,
  max_cells = 5e+07,
  verbose = TRUE
)
```

Arguments

project_dir	Project root directory containing embeddings/.
embeddings_dir	Model subfolder under project_dir/embeddings, e.g. "model_id=BAAI_bge-small-en-v1.5".
corpus_label	Label partition used as corpus side. Defaults to "corpus".
reference_label	Label partition used as reference side. Defaults to "reference".
batch_size	Unused placeholder for compatibility with planned streaming extension.
max_cells	Maximum allowed matrix size ((n_corpus + 1) * (n_reference + 1)) to guard memory use.
verbose	Logical; print progress messages.

Details

Embeddings are expected under: project_dir/embeddings/model_id=<...>/label=<label>/batch=<n>/...

Output file:

- pairwise-cosine.parquet: wide table with first column id (corpus id or "centroid"), reference-id columns, and a final centroid column.

Value

Invisibly the output directory project_dir/distance_reference_cosine/model_id=<...>/corpus_label=<...>/refe

---

distance_ridge *Compute corpus distance to a reference embedding area*

Description

Fits (or loads) a reference-area model from reference_label embeddings and computes squared Mahalanobis distance for rows in corpus_label.

Usage

```
distance_ridge(
  project_dir,
  reference_label = "reference",
  corpus_label = "corpus",
  fit_path = NULL,
  batch_size = 1e+05,
  regularization = 1e-06,
  verbose = TRUE
)
```

Arguments

project_dir	Project root containing embeddings/.
reference_label	Label partition used to fit the reference area. Defaults to "reference".
corpus_label	Label partition to score. Defaults to "corpus".
fit_path	Optional path to an existing reference-area fit (.rds). If NULL, a new fit is created at project_dir/ridge_fit.rds.
batch_size	Approximate number of rows per Arrow scan batch.
regularization	Diagonal covariance regularization added before inversion.
verbose	Logical; print progress messages.

Value

Invisibly the model output directory under project_dir/distance_ridge/model_id=<...>/corpus_label=<...>/refer

`distances` *Join prototype and ridge distances lazily via Arrow*

Description

Opens two Parquet datasets (prototype margins and ridge scores) as Arrow datasets and performs a lazy inner join on the common key (typically `id`). The result is an Arrow-dplyr query that is not materialized until you call `dplyr::collect()` or write it with `arrow::write_dataset()`.

Usage

```
distances(prototype_distances, ridge_distance)
```

Arguments

`prototype_distances`	Path to a Parquet dataset (file or directory) containing prototype distances, e.g., columns `id` and `margin`.
`ridge_distance`	Path to a Parquet dataset (file or directory) containing ridge-based scores, e.g., columns `id` and `relevance_score`.

Value

A lazy Arrow dplyr query representing the joined datasets.

Examples

```
## Not run:
joined <- distances(
  prototype_distances = "path/to/prototype_distances/",
  ridge_distance      = "path/to/ridge_scores/"
)

# Continue piping lazily and write without loading into memory
joined |>
  dplyr::mutate(ensemble = (margin + relevance_score) / 2) |>
  arrow::write_dataset(path = "path/to/output_scores/", format = "parquet")

# Or collect a small sample for inspection
head(dplyr::collect(joined))

## End(Not run)
```

embed_corpus *Stream a corpus dataset, embed in batches, and write Parquets*

Description

Processes a Parquet dataset without loading it fully in memory. Reads Arrow record batches, builds canonical text from `title` + `abstract`, calls the configured embedding backend, and writes Parquet batch files.

Usage

```
embed_corpus(
  project_dir = NULL,
  backend = backend_config(),
  corpus_name = "corpus",
  batch_size = 5000,
  delete_existing = FALSE,
  text_preprocessor = clean_abstract_for_embedding,
  cleaner_args = list(),
  save_text = TRUE,
  label = corpus_name,
  dry_run = FALSE,
  verbose = TRUE
)
```

Arguments

`project_dir`	Project root directory. Must contain `project_dir/<corpus_name>` with columns `id`, `title`, `abstract`.
`backend`	Backend configuration created with `backend_config()`.
`corpus_name`	Folder name under `project_dir` containing the corpus parquet dataset. Defaults to `"corpus"`.
`batch_size`	Number of corpus rows per Arrow scan batch.
`delete_existing`	If `TRUE`, old embeddings for the target model are deleted before processing. If `FALSE`, unchanged rows are skipped using `id` + `text_hash`.
`text_preprocessor`	Function that prepares embedding text from a batch data frame and returns at least columns `id`, `text`, `text_hash`. Defaults to `clean_abstract_for_embedding()`.
`cleaner_args`	Named list of additional arguments passed to `text_preprocessor`.
`save_text`	Logical; if `TRUE` (default), store the cleaned embedding text in output Parquet files as column `text`. If `FALSE`, only `text_hash` is stored.
`label`	Partition label written under `project_dir/embeddings/model_id=<...>/label=<label>/batch=<n>`. Defaults to `corpus_name`.

dry_run	Logical; if TRUE, run preprocessing and unchanged-row filtering without requesting embeddings or writing output files. In this mode, a Parquet preview file is written to project_dir/<corpus_name>_dryrun.parquet.
verbose	Logical; print progress and summary messages.

Value

Invisibly the model-specific embeddings directory under project_dir/embeddings/model_id=<...>/.

---

embed_texts *Embed texts through a configured backend*

Description

Sends a character vector to the configured backend and returns embeddings as a numeric matrix.

Usage

```
embed_texts(texts, backend = backend_config())
```

Arguments

texts	Character vector of texts to embed. Empty inputs return a 0-row matrix; missing values are not supported.
backend	Backend configuration created with backend_config().

Value

A numeric matrix with one row per input text and one column per embedding dimension.

---

fit_ridge *Fit a reference-area model from embeddings parquet*

Description

Fits a reference-area model (centroid + regularized covariance inverse) using rows from reference_label.

Usage

```
fit_ridge(
  embeddings,
  reference_label = "reference",
  output,
  regularization = 1e-06,
  verbose = TRUE
)
```

Arguments

embeddings	Path to a Parquet dataset (file or directory opened by Arrow) with columns id, label, and V1..Vd.
reference_label	Label partition used to define the reference area.
output	Name of the .rds file to save the fit object.
regularization	Positive numeric diagonal regularization added to covariance.
verbose	Logical; print progress messages.

Value

Invisibly returns output.

---

plot_embeddings_pca *Plot embeddings via PCA, colored by arbitrary labels*

Description

Reads an embeddings Parquet dataset (produced by embed_corpus()) with columns id and V1..Vd, computes a PCA on the embedding matrix, and returns a scatter plot of the first two principal components. Points are colored by labels provided via labels. Rows not found in labels are shown as "other".

Usage

```
plot_embeddings_pca(
  embeddings,
  labels,
  center = TRUE,
  scale. = FALSE,
  point_size = 2,
  alpha = 0.5
)
```

Arguments

embeddings	Path to a Parquet file or dataset directory containing columns id and V1..Vd.
labels	Label mapping for ids. Supported formats: 1. data frame with columns id and label, 2. path to CSV with columns id and label, 3. named character vector where names are ids and values are labels, 4. named list where each element is an id vector for that label.
center, scale.	Passed to stats::prcomp() for PCA. Defaults center = TRUE, scale. = FALSE.
point_size, alpha	Point size and transparency for points in the plot. Defaults point_size = 2, alpha = 0.5.

Value

A ggplot object with points mapped to PC1 vs PC2 and colored by group.

Examples

```
## Not run:
p <- plot_embeddings_pca(
  embeddings = "inst/examples/embedings/",
  labels = data.frame(
    id = c("W1", "W2", "W10"),
    label = c("reference", "reference", "corpus")
  )
)
print(p)

## End(Not run)
```

---

plot_embeddings_umap *Plot embeddings via UMAP, colored by arbitrary labels*

Description

Computes a 2D UMAP projection of V1..Vd and returns a scatter plot colored by labels membership. Uses cosine distance by default to align with common embedding similarity.

Usage

```
plot_embeddings_umap(
  embeddings,
  labels,
  n_neighbors = 15,
  min_dist = 0.1,
  metric = "cosine",
  n_epochs = 500,
  seed = 42,
  sample_n = NULL,
  point_size = 2,
  alpha = 0.5
)
```

Arguments

embeddings	Path to a Parquet file or dataset directory containing columns id and V1..Vd.
labels	Label mapping for ids. Supported formats: 1. data frame with columns id and label, 2. path to CSV with columns id and label,

3. named character vector where names are ids and values are labels,
4. named list where each element is an id vector for that label.

`n_neighbors`, `min_dist`, `metric`, `n_epochs`
: UMAP parameters passed to `uwot::umap()`. Defaults: `n_neighbors` = 15, `min_dist` = 0.1, `metric` = "cosine", `n_epochs` = 500.

`seed`
: Random seed for reproducibility (set to `NULL` to skip).

`sample_n`
: Optional maximum number of rows to sample for plotting (applied before UMAP). If `NULL`, uses all rows.

`point_size`, `alpha`
: Point size and transparency for points in the plot. Defaults `point_size` = 2, `alpha` = 0.5.

Value

A `ggplot` object of UMAP1 vs UMAP2 colored by group.

---

run_demo_openai *Create and optionally run an OpenAI-based demo project via Quarto*

Description

Uses the same demo structure as `run_demo_openalex()`, but configures an OpenAI backend and requires an explicit API key argument. The key is set in `OVC_API_TOKEN` for the duration of the call.

Usage

```
run_demo_openai(
  api_key,
  demo_dir = file.path(getwd(), "demos", "openai"),
  render = TRUE,
  model = "text-embedding-3-small",
  max_corpus = 100,
  max_reference = 10,
  overwrite = FALSE,
  quarto_file = "openai_demo_analysis.qmd",
  verbose = TRUE
)
```

Arguments

`api_key`
: OpenAI API key. Must be a non-empty string.

`demo_dir`
: Demo workspace directory. Defaults to `file.path(getwd(), "demos", "openai")`.

`render`
: Logical; if `TRUE` (default), run `quarto render` on the copied template.

`model`
: OpenAI embedding model id. Defaults to `"text-embedding-3-small"`.

`max_corpus`
: Maximum number of corpus fixture rows to copy.

max_reference	Maximum number of reference fixture rows to copy.
overwrite	Logical; if FALSE (default), stop when demo-managed files already exist. If TRUE, refresh demo-managed files.
quarto_file	Name of the analysis file created in demo_dir.
verbose	Logical; print progress messages.

Value

Invisibly returns a list with project paths and render status.

---

run_demo_openalex *Create and optionally run a self-contained demo project via Quarto*

Description

Sets up a demo workspace under demo_dir, creates a pipeline project under demo_dir/project, copies small corpus and reference fixtures and Quarto template from inst/ovc_demo, and optionally renders the analysis.

Usage

```
run_demo_openalex(
  demo_dir = file.path(getwd(), "demos", "openalex"),
  render = TRUE,
  backend = NULL,
  max_corpus = 100,
  max_reference = 10,
  overwrite = FALSE,
  quarto_file = "openalex_demo_analysis.qmd",
  verbose = TRUE
)
```

Arguments

demo_dir	Demo workspace directory. Defaults to file.path(getwd(), "demos", "openalex").
render	Logical; if TRUE (default), run quarto render on the copied template.
backend	Optional backend config from backend_config(). If NULL, defaults to Hugging Face (provider = "hf", model "BAAI/bge-small-en-v1.5").
max_corpus	Maximum number of corpus fixture rows to copy.
max_reference	Maximum number of reference fixture rows to copy.
overwrite	Logical; if FALSE (default), stop when demo-managed files already exist. If TRUE, refresh demo-managed files.
quarto_file	Name of the analysis file created in demo_dir.
verbose	Logical; print progress messages.

Details

The workspace keeps all generated directories and output artifacts. The Quarto file is created in `demo_dir`, while embedding pipeline data is stored in `demo_dir/project`.

Value

Invisibly returns a list with project paths and render status.

---

score_reference_cosine

Convert reference-cosine distances to scores

Description

Reads a wide reference-cosine distance matrix (as written by `distance_reference_cosine()`) and converts all numeric distance columns to scores.

Usage

```
score_reference_cosine(
  distance_parquet,
  output_dir = NULL,
  method = c("linear", "exponential"),
  alpha = 1,
  verbose = TRUE
)
```

Arguments

`distance_parquet`	Path to a Parquet dataset (file or directory) with first column `id` and one or more numeric distance columns.
`output_dir`	Optional output directory. If `NULL`, defaults to replacing `"distance_reference_cosine"` with `"score_reference_cosine"` in `distance_parquet`.
`method`	Scoring transform: `"linear"` (default, `1 - distance`) or `"exponential"` (`exp(-alpha * distance)`).
`alpha`	Positive numeric scaling factor used when `method = "exponential"`.
`verbose`	Logical; print progress messages.

Value

Invisibly returns output directory.

score_ridge *Convert ridge distances to ridge scores*

Description

Reads a distance dataset with columns `id` and `area_distance`, computes `relevance_score = exp(-alpha * area_distance)`, and writes a scored Parquet dataset.

Usage

```
score_ridge(distance_parquet, output_dir = NULL, alpha = 0.5, verbose = TRUE)
```

Arguments

`distance_parquet`	Path to a Parquet dataset (file or directory) with at least columns `id` and `area_distance`.
`output_dir`	Optional output directory. If `NULL`, defaults to replacing `"distance_ridge"` with `"score_ridge"` in `distance_parquet`.
`alpha`	Positive numeric scaling factor in `exp(-alpha * area_distance)`. Default `0.5` reproduces previous behavior.
`verbose`	Logical; print progress messages.

Value

Invisibly returns output directory.

similarity_cosine *Cosine similarity between two numeric vectors*

Description

Computes cosine similarity for two numeric vectors of equal length. Returns `NA_real_` if either vector has zero norm.

Usage

```
similarity_cosine(a, b)
```

Arguments

`a`	Numeric vector.
`b`	Numeric vector or numeric matrix with embeddings in rows.

Value

A single numeric similarity value in `[-1, 1]`, or `NA_real_` when undefined.

Index

backend_config, 2
backend_config(), *3–6*, *8*, *15, 16*, *20*
backend_embed_texts, 3
backend_info, 4
backend_read, 4
backend_save, 5
backend_specter2_tei, 5
batch_collect_openai, 6
batch_status_openai, 7
batch_status_openai(), *11*
batch_submit_openai, 7

calibrate_threshold, 8
clean_abstract_for_embedding, 10
clean_abstract_for_embedding(), *15*

demo_finalize_openai_batch, 11
distance_cosine, 11
distance_reference_cosine, 12
distance_reference_cosine(), *21*
distance_ridge, 13
distances, 14

embed_corpus, 15
embed_corpus(), *17*
embed_texts, 16

fit_ridge, 16

plot_embeddings_pca, 17
plot_embeddings_umap, 18

run_demo_openai, 19
run_demo_openai(), *11*
run_demo_openalex, 20
run_demo_openalex(), *11*, *19*

score_reference_cosine, 21
score_ridge, 22
similarity_cosine, 22
stats::prcomp(), *17*

uwot::umap(), *19*