Package: ducksemantics (via r-universe)

July 21, 2026

Type Package

Title DuckDB-Native Semantic Graph Grounding

Version 0.1.0

Description DuckDB-native semantic graph grounding over arbitrary concept graphs. The package owns graph schema creation, alias indexing, lexical mention grounding, transitive graph closure, 'EmbeddingGemma' dense retrieval, 'ColBERT' late interaction, and 'BebeLM' judgment prompts. HPO, MONDO, ORPHANET, memory graphs, and run ledgers are graph sources, not special package APIs.

License MIT + file LICENSE

URL https://github.com/sounkou-bioinfo/ducksemantics, https://sounkou-bioinfo.github.io/ducksemantics/

BugReports https://github.com/sounkou-bioinfo/ducksemantics/issues

Encoding UTF-8

Depends R (>= 4.3.0)

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Imports DBI, S7, s7contract (>= 0.1.0), utils

Suggests duckdb, jsonlite, Rbebelm (>= 0.3.6-0.1.0), tinytest

Remotes sounkou-bioinfo/Rbebelm, sounkou-bioinfo/s7contract

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

Date/Publication 2026-07-21 13:18:18 UTC

RemoteUrl https://github.com/RGenomicsETL/ducksemantics

RemoteRef HEAD

RemoteSha a21153a9471880fc434f517fc954dea99c7ba0b6

Contents

`ducksemantics_annotate` *Annotate text against the semantic alias index*

Description

Annotate text against the semantic alias index

Usage

```
ducksemantics_annotate(
  conn,
  text,
  document_id = NULL,
  prefix = "semantic",
  longest_match = TRUE,
  record = FALSE
)
```

Arguments

conn	DBI connection.
text	Character scalar.
document_id	Optional document id.
prefix	Prefix used for semantic tables.
longest_match	Drop matches contained by a longer span.
record	Append returned mentions to the mentions table?

Value

A data frame of grounded mentions.

---

ducksemantics_bebel_judge

Judge mentions with a BebeLM/Rbebelm agent

Description

Judge mentions with a BebeLM/Rbebelm agent

Usage

```
ducksemantics_bebel_judge(
  agent,
  text,
  mentions,
  conn = NULL,
  prefix = "semantic",
  graph_context = NULL,
  instructions = ducksemantics_default_judgment_instructions(),
  parser = ducksemantics_json_judgment_parser(),
  on_event = NULL,
  max_retries = 1L,
  record = !is.null(conn),
  model = "Rbebelm"
)
```

Arguments

agent	A `Rbebelm` agent object.
text	Source text.
mentions	Mention data frame from `ducksemantics_annotate()`.
conn	Optional DBI connection. When supplied with `record = TRUE`, judgments are appended to the judgment table.

`prefix`	Prefix used for semantic tables.
`graph_context`	Optional data frame or list with nearby graph context.
`instructions`	Character scalar or vector containing the adjudication policy.
`parser`	Object implementing DucksemanticsJudgmentParser.
`on_event`	Optional Rbebelm event handler.
`max_retries`	Number of corrective turns after a response fails the strict `DucksemanticsJudgmentParser` contract. Each parse error is appended to the same BebeLM transcript as a user turn; it is never silently coerced.
`record`	Append judgments to the judgment table?
`model`	Model label recorded with judgments.

Value

A data frame of judgment rows. Its `responses` and `parse_errors` attributes retain every model response and corrective parse error.

`ducksemantics_bebel_runner`
Create a BebeLM prompt runner

Description

Create a BebeLM prompt runner

Usage

```
ducksemantics_bebel_runner(agent, on_event = NULL)
```

Arguments

`agent`	A `Rbebelm` agent object.
`on_event`	Optional Rbebelm event handler.

Value

An object implementing DucksemanticsPromptRunner.

ducksemantics_bebel_tool_judgment_parser

Create a BebeLM tool-call judgment parser

Description

Create a BebeLM tool-call judgment parser

Usage

```
ducksemantics_bebel_tool_judgment_parser(tool_name = NULL)
```

Arguments

tool_name	Optional accepted tool-call name or names.

Value

An object implementing DucksemanticsJudgmentParser.

ducksemantics_benchmark

Run a grounding benchmark

Description

Run a grounding benchmark

Usage

```
ducksemantics_benchmark(
  benchmark,
  conn,
  prefix = "semantic",
  annotator = ducksemantics_lexical_annotator(),
  longest_match = TRUE,
  record = FALSE,
  collect_index_stats = TRUE
)
```

Arguments

benchmark	Benchmark object from ducksemantics_benchmark_cases().
conn	DBI connection.
prefix	Prefix used for semantic tables.
annotator	Object implementing DucksemanticsAnnotator.
longest_match	Drop matches contained by a longer span.
record	Append predicted mentions to the mentions table?
collect_index_stats	Include index stats in the result?

Value

A list with predictions, timings, metrics, and optional index stats.

---

ducksemantics_benchmark_cases *Define benchmark cases*

Description

Define benchmark cases

Usage

```
ducksemantics_benchmark_cases(
  cases,
  gold,
  suite = "semantic",
  task = "grounding",
  source = NULL,
  version = NULL,
  metadata = list()
)
```

Arguments

cases	Data frame with case_id and text.
gold	Data frame with case_id and node_id. Optional span columns are span, start_offset, and end_offset.
suite	Benchmark suite label.
task	Benchmark task label.
source	Optional source dataset label.
version	Optional source dataset version.
metadata	Optional named list of benchmark metadata.

Value

A benchmark object.

`ducksemantics_benchmark_metrics`
Compute benchmark precision and recall

Description

Compute benchmark precision and recall

Usage

```
ducksemantics_benchmark_metrics(predictions, gold, by = c("node", "span"))
```

Arguments

`predictions`	Prediction data frame from `ducksemantics_benchmark()` or `ducksemantics_annotate()`.
`gold`	Gold data frame with `case_id` and `node_id`.
`by`	Match on node only, or on exact span offsets when available.

Value

A one-row data frame with `tp`, `fp`, `fn`, `precision`, `recall`, and `f1`.

`ducksemantics_cache_file`
Cache a source file

Description

Cache a source file

Usage

```
ducksemantics_cache_file(
  url,
  filename = basename(url),
  cache_dir = tools::R_user_dir("ducksemantics", "cache"),
  refresh = FALSE
)
```

Arguments

`url`	Source URL.
`filename`	Cache filename.
`cache_dir`	Cache directory.
`refresh`	Download even when the cache file already exists?

Value

Normalized path to the cached file.

`ducksemantics_cache_rds` *Cache an R value on disk*

Description

Cache an R value on disk

Usage

```
ducksemantics_cache_rds(path, compute, refresh = FALSE)
```

Arguments

`path`	RDS cache path.
`compute`	Function called with no arguments when the cache is missing or refreshed.
`refresh`	Recompute even when the cache file already exists?

Value

The cached R object.

`ducksemantics_closure_sql` *Materialize transitive edge closure*

Description

Returns DuckDB SQL that computes `target_table(from_id, predicate, to_id)` as the transitive closure of `source_table` for the supplied predicates. This is the same graph-as-SQL primitive used for ontology ancestors, partonomy, memory walks, and arbitrary declared transitive graph relations.

Usage

```
ducksemantics_closure_sql(
  transitive_predicates,
  source_table = "semantic_edges",
  target_table = "semantic_entailed_edges"
)
```

Arguments

transitive_predicates	Character vector of predicates to close.
source_table	Source edge table.
target_table	Target closure table.

Value

A DuckDB SQL script that replaces the closure table and restores its graph indexes.

---

ducksemantics_cluster_embeddings *Cluster embedding rows*

Description

Clustering writes assignments and centroids back to DuckDB. It is intended as a first measurement surface for whether a provider's embeddings recover ontology structure before adding graph-aware or late-interaction scoring.

Usage

```
ducksemantics_cluster_embeddings(
  spec,
  conn,
  prefix = "semantic",
  replace = TRUE
)
```

Arguments

spec	A `DucksemanticsEmbeddingClusterSpec` object from `ducksemantics_embedding_cluster_spec()`.
conn	DBI connection.
prefix	Prefix used for semantic tables.
replace	Delete rows for `spec$run_id` before writing?

Value

A list with assignments, centroids, summary, and the `kmeans` object.

ducksemantics_colbert_provider

Create a native ColBERT token-vector provider

Description

`role = "document"` is for ontology labels, definitions, and candidate passages stored in DuckDB. `role = "query"` is for text being searched. The native encoder owns the distinct prefixes, token limits, projection, and L2 normalization required by the model; no causal BebeLM hidden states are used.

Usage

```
ducksemantics_colbert_provider(
  model,
  role = c("document", "query"),
  label = "Rbebelm ColBERT"
)
```

Arguments

`model`	A `Rbebelm` `ColbertModel` object.
`role`	Whether this provider encodes retrieval `"query"` or `"document"` text.
`label`	Provider label. Document rows and their query must share it.

Value

An object implementing DucksemanticsTokenEmbeddingProvider.

ducksemantics_colbert_query

Construct a ColBERT late-interaction query

Description

Encodes text with the profile's query contract and returns a query object directly consumable by `ducksemantics_late_interaction_search()`. Candidate blocks must have been stored from a ColBERT document provider with the same `label`.

Usage

```
ducksemantics_colbert_query(
  model,
  text,
  provider = "Rbebelm ColBERT",
  subject_kind = NULL,
  top_k = 10L,
  table = NULL,
  candidate_subject_id = NULL
)
```

Arguments

`model`	A `Rbebelm` `ColbertModel` object.
`text`	Non-empty query text.
`provider`	Stored document provider label.
`subject_kind`	Optional candidate subject-kind filter.
`top_k`	Number of candidate blocks to return.
`table`	Optional token-embedding table.
`candidate_subject_id`	Optional candidate subject identifiers.

Value

A DucksemanticsTokenEmbeddingQuery.

`ducksemantics_connect` *Connect to a DuckDB semantic store*

Description

Connect to a DuckDB semantic store

Usage

```
ducksemantics_connect(dbdir = ":memory:", read_only = FALSE, array = "matrix")
```

Arguments

`dbdir`	DuckDB database path, or `":memory:"`.
`read_only`	Open read-only?
`array`	DuckDB array conversion mode. The default enables native vector columns to round-trip through R matrices.

Value

A DBI connection.

ducksemantics_default_judgment_instructions
Default judgment instructions

Description

These are only the default policy for the judgment prompt. Pass an explicit instruction string to `ducksemantics_judgment_prompt()` or `ducksemantics_judge()` when benchmarking a specific adjudication protocol.

Usage

```
ducksemantics_default_judgment_instructions()
```

Value

A character scalar.

ducksemantics_embed_cached
Cache provider embeddings in durable chunks

Description

This cache is used for large ontology passes. Each chunk is written after it finishes, so interrupted runs can resume without discarding completed native retrieval-encoder work.

Usage

```
ducksemantics_embed_cached(
  text,
  provider,
  cache_dir,
  chunk_size = 4096L,
  refresh = FALSE,
  cache_key = NULL,
  ...
)
```

Arguments

`text`	Character vector to embed.
`provider`	Object implementing DucksemanticsEmbeddingProvider.
`cache_dir`	Directory for chunk RDS files.

chunk_size	Number of texts per persisted chunk.
refresh	Recompute all chunks?
cache_key	Optional stable identifier for provider weights or other state that is not represented by the provider object. Changing it invalidates existing chunks.
...	Extra arguments passed to ducksemantics_embed(). These arguments are included in the cache identity.

Value

Numeric embedding matrix with one row per input text.

`ducksemantics_embedding_batch` *Construct an embedding batch*

Description

Construct an embedding batch

Usage

```
ducksemantics_embedding_batch(
  embeddings,
  subject_id,
  subject_kind = "node",
  provider = "embedding",
  text = NULL,
  attrs = NULL
)
```

Arguments

embeddings	Numeric matrix with one row per subject.
subject_id	Subject identifiers matching embedding rows.
subject_kind	Subject type, e.g. "node", "alias", "mention", or "document".
provider	Embedding provider label.
text	Optional source text for each embedding.
attrs	Optional JSON text or other metadata for each embedding.

ducksemantics_embedding_cluster_graph_agreement

Compare embedding clusters with graph edges

Description

This measures whether clustered node embeddings respect direct ontology relations such as `is_a` and `part_of`. It is a coarse diagnostic: high agreement does not prove semantic quality, but low agreement is actionable evidence for the embedding, text source, or clustering setup.

Usage

```
ducksemantics_embedding_cluster_graph_agreement(
  conn,
  cluster_run_id,
  predicates = c("is_a", "part_of"),
  prefix = "semantic"
)
```

Arguments

`conn`	DBI connection.
`cluster_run_id`	Cluster run id written by `ducksemantics_cluster_embeddings()`.
`predicates`	Edge predicates to evaluate.
`prefix`	Prefix used for semantic tables.

Value

One-row data frame with edge counts and same-cluster rate.

ducksemantics_embedding_cluster_spec

Construct an embedding clustering specification

Description

Construct an embedding clustering specification

Usage

```
ducksemantics_embedding_cluster_spec(
  k,
  provider = NULL,
  subject_kind = NULL,
  dimensions = NULL,
  table = NULL,
  run_id = NULL,
  seed = 1L,
  nstart = 10L,
  max_iter = 100L
)
```

Arguments

k	Number of clusters.
provider	Optional provider filter.
subject_kind	Optional subject-kind filter.
dimensions	Optional embedding dimension filter.
table	Optional embedding table. Defaults to semantic_embeddings.
run_id	Identifier written to cluster tables.
seed	Random seed used by stats::kmeans().
nstart	Number of starts used by stats::kmeans().
max_iter	Maximum k-means iterations.

---

ducksemantics_embedding_cluster_summary

Summarize stored embedding clusters

Description

Summarize stored embedding clusters

Usage

```
ducksemantics_embedding_cluster_summary(
  conn,
  cluster_run_id = NULL,
  prefix = "semantic"
)
```

Arguments

conn	DBI connection.
cluster_run_id	Optional cluster run filter.
prefix	Prefix used for semantic tables.

Value

Data frame with cluster sizes and distance summaries.

---

ducksemantics_embedding_index_spec *Construct an embedding index specification*

Description

Construct an embedding index specification

Usage

```
ducksemantics_embedding_index_spec(
  dimensions,
  provider = NULL,
  subject_kind = NULL,
  table = NULL,
  hnsw = TRUE,
  metric = "cosine",
  load_vss = TRUE
)
```

Arguments

dimensions	Embedding dimension.
provider	Optional provider filter.
subject_kind	Optional subject-kind filter.
table	Target table name. Defaults to semantic_embedding_index_<dimensions>.
hnsw	Create a DuckDB HNSW index on the materialized table?
metric	HNSW metric, usually "cosine", "l2sq", or "ip".
load_vss	Load DuckDB's vss extension before creating the HNSW index?

---

ducksemantics_embedding_provider *Wrap an embedding function as a typed embedding provider*

Description

Wrap an embedding function as a typed embedding provider

Usage

```
ducksemantics_embedding_provider(fun, label = "function")
```

Arguments

`fun`	Function accepting a character vector and returning a numeric matrix with one row per input text.
`label`	Provider label for reports.

Value

An object implementing DucksemanticsEmbeddingProvider.

`ducksemantics_embedding_query` *Construct an embedding search query*

Description

Construct an embedding search query

Usage

```
ducksemantics_embedding_query(
  embedding,
  provider = NULL,
  subject_kind = NULL,
  top_k = 10L,
  metric = c("cosine", "cosine_distance", "l2", "inner_product"),
  table = NULL
)
```

Arguments

`embedding`	Numeric query embedding.
`provider`	Optional provider filter.
`subject_kind`	Optional subject-kind filter.
`top_k`	Number of nearest rows to return.
`metric`	One of `"cosine"`, `"cosine_distance"`, `"l2"`, or `"inner_product"`.
`table`	Optional table to search. Defaults to `semantic_embeddings`. Pass a table created by `ducksemantics_materialize_embedding_index()` to use a dimensioned, HNSW-indexable table.

ducksemantics_embedding_search

Search embeddings with DuckDB vector functions

Description

Search embeddings with DuckDB vector functions

Usage

```
ducksemantics_embedding_search(query, conn, prefix = "semantic")
```

Arguments

query	A DucksemanticsEmbeddingQuery object from ducksemantics_embedding_query().
conn	DBI connection.
prefix	Prefix used for semantic tables.

Value

Data frame ordered by best match.

ducksemantics_embeddinggemma_provider

Create an EmbeddingGemma dense retrieval provider

Description

Create an EmbeddingGemma dense retrieval provider

Usage

```
ducksemantics_embeddinggemma_provider(
  model,
  label = "Rbebelm EmbeddingGemma",
  task = "semantic_similarity",
  title = NULL,
  dimensions = 768L,
  normalize = TRUE,
  truncate = TRUE,
  check_interrupt = TRUE
)
```

Arguments

model	A Rbebelm EmbeddingGemmaModel object.
label	Provider label for stored dense vectors.
task	EmbeddingGemma task prompt. Use the same task for vectors that will be compared; use the dedicated query/document tasks only as a matched retrieval pair.
title	Optional document title, valid only for retrieval_document.
dimensions	Matryoshka dimension: 768, 512, 256, or 128.
normalize	L2-normalize output rows.
truncate	Truncate inputs longer than EmbeddingGemma's context.
check_interrupt	Poll for R interrupts between bounded native batches.

Value

An object implementing DucksemanticsEmbeddingProvider.

---

ducksemantics_index_aliases *Build the lexical alias index*

Description

Build the lexical alias index

Usage

```
ducksemantics_index_aliases(conn, prefix = "semantic")
```

Arguments

conn	DBI connection.
prefix	Prefix used for semantic tables.

Value

Invisibly, the alias index table name.

ducksemantics_index_stats *Summarize semantic index size*

Description

Summarize semantic index size

Usage

```
ducksemantics_index_stats(conn, prefix = "semantic")
```

Arguments

conn	DBI connection.
prefix	Prefix used for semantic tables.

Value

A list with table row counts and alias-index pressure metrics.

ducksemantics_init *Initialize semantic graph tables*

Description

Initialize semantic graph tables

Usage

```
ducksemantics_init(conn, prefix = "semantic")
```

Arguments

conn	DBI connection.
prefix	Prefix used for semantic tables.

Value

Invisibly, conn.

ducksemantics_json_judgment_parser *Create the default JSON judgment parser*

Description

Create the default JSON judgment parser

Usage

```
ducksemantics_json_judgment_parser()
```

Value

An object implementing DucksemanticsJudgmentParser.

ducksemantics_judge *Judge mentions with a model runner*

Description

Judge mentions with a model runner

Usage

```
ducksemantics_judge(
  text,
  mentions,
  runner,
  conn = NULL,
  prefix = "semantic",
  graph_context = NULL,
  instructions = ducksemantics_default_judgment_instructions(),
  prompt_builder = ducksemantics_judgment_prompt,
  parser = ducksemantics_json_judgment_parser(),
  record = !is.null(conn),
  model = "semantic-runner",
  ...
)
```

Arguments

text	Source text.
mentions	Mention data frame from ducksemantics_annotate().
runner	Object implementing DucksemanticsPromptRunner.
conn	Optional DBI connection. When supplied with record = TRUE, judgments are appended to the judgment table.
prefix	Prefix used for semantic tables.
graph_context	Optional data frame or list with nearby graph context.
instructions	Character scalar or vector containing the adjudication policy.
prompt_builder	Function that builds the prompt.
parser	Object implementing DucksemanticsJudgmentParser.
record	Append judgments to the judgment table?
model	Model label recorded with judgments.
...	Extra arguments passed to prompt_builder.

Value

A data frame of judgment rows. The prompt and raw response are stored as prompt and response attributes for audit and benchmarking.

---

ducksemantics_judgment_prompt

Build a semantic judgment prompt

Description

Build a semantic judgment prompt

Usage

```
ducksemantics_judgment_prompt(
  text,
  mentions,
  graph_context = NULL,
  instructions = ducksemantics_default_judgment_instructions()
)
```

Arguments

text	Source text.
mentions	Mention data frame from ducksemantics_annotate().
graph_context	Optional data frame or list with nearby graph context.
instructions	Character scalar or vector containing the adjudication policy. This is deliberately explicit so benchmark runs can vary the policy without changing candidate generation.

Value

Prompt text.

`ducksemantics_late_interaction_search`
Search token embeddings with exact late interaction

Description

Scores stored native ColBERT document blocks with exact MaxSim. For each query token, the scorer finds the best matching document token and sums those maxima, matching `Rbebelm::colbert_maxsim()` once both matrices have been materialized. Use dense EmbeddingGemma/HNSW, aliases, FTS, or graph context to reduce large corpora before this reranker.

Usage

```
ducksemantics_late_interaction_search(query, conn, prefix = "semantic")
```

Arguments

`query`	A `DucksemanticsTokenEmbeddingQuery` object from `ducksemantics_token_embedding_query()`.
`conn`	DBI connection.
`prefix`	Prefix used for semantic tables.

Value

Data frame ordered by descending MaxSim score.

`ducksemantics_lexical_annotator`
Create the default DuckDB lexical annotator

Description

Create the default DuckDB lexical annotator

Usage

```
ducksemantics_lexical_annotator()
```

Value

An object implementing DucksemanticsAnnotator.

ducksemantics_materialize_embedding_index

Materialize a fixed-dimension embedding table

Description

DuckDB's HNSW index requires a fixed-size vector type such as `FLOAT[384]`. This function projects rows from `semantic_embeddings` into a dimensioned table and can create a native HNSW index on that table.

Usage

```
ducksemantics_materialize_embedding_index(spec, conn, prefix = "semantic")
```

Arguments

`spec`	A DucksemanticsEmbeddingIndexSpec object from `ducksemantics_embedding_index_spec()`.
`conn`	DBI connection.
`prefix`	Prefix used for semantic tables.

Value

Target table name.

ducksemantics_normalize

Normalize text for semantic grounding

Description

Normalize text for semantic grounding

Usage

```
ducksemantics_normalize(text)
```

Arguments

`text`	Character vector.

Value

Normalized character vector.

ducksemantics_projection_sql

Project any edge-shaped source relation into graph shape

Description

This mirrors the graph projection profile used in pi-bio-agent: a source table with caller-named columns becomes a stable graph edge table with `from_id`, `predicate`, `to_id`, `attrs`, and `trust`.

Usage

```
ducksemantics_projection_sql(
  source_table,
  from,
  predicate,
  to,
  target_table = "semantic_edges",
  attrs = NULL,
  trust = NULL
)
```

Arguments

`source_table`	Source table or view name.
`from`, `predicate`, `to`	Source column names.
`target_table`	Target table name.
`attrs`, `trust`	Optional source columns containing JSON text.

Value

A DuckDB SQL script that replaces the target table and restores its graph indexes.

ducksemantics_prompt_runner

Wrap a prompt function as a typed prompt runner

Description

Wrap a prompt function as a typed prompt runner

Usage

```
ducksemantics_prompt_runner(fun, label = "function")
```

Arguments

fun	Function accepting prompt and returning response text.
label	Provider label for reports.

Value

An object implementing DucksemanticsPromptRunner.

---

ducksemantics_provider_generics

Provider interface generics

Description

These S7 generics are the behavior required by the structural interfaces. Provider packages should define concrete S7 classes and methods for these generics, then consuming code can assert the corresponding Ducksemantics* interface.

Usage

```
ducksemantics_run(provider, prompt, ...)

ducksemantics_embed(provider, text, ...)

ducksemantics_token_embed(provider, text, ...)

ducksemantics_parse(parser, response, ...)

ducksemantics_ground(
  annotator,
  conn,
  text,
  document_id = NULL,
  prefix = "semantic",
  longest_match = TRUE,
  record = FALSE,
  ...
)
```

Arguments

provider	Prompt or embedding provider.
prompt	Prompt text.
...	Provider-specific arguments.
text	Source text.

parser	Judgment parser.
response	Raw model response text.
annotator	Text-grounding provider.
conn	DBI connection.
document_id	Optional document id.
prefix	Semantic table prefix.
longest_match	Drop matches contained by a longer span.
record	Append returned rows to the semantic store?

Value

Provider-specific output: response text, embedding matrix, parsed judgment data frame, or grounded mention data frame.

`ducksemantics_read_obo` *Read an OBO ontology into semantic graph rows*

Description

Read an OBO ontology into semantic graph rows

Usage

```
ducksemantics_read_obo(
  path,
  family,
  source = basename(path),
  include_obsolete = FALSE
)
```

Arguments

path	OBO file path.
family	Graph family label, for example "HPO" or "MONDO".
source	Source label stored on alias rows.
include_obsolete	Include terms marked is_obsolete: true?

Value

A list with nodes, aliases, and edges data frames.

ducksemantics_record_judgments *Record semantic judgments*

Description

Record semantic judgments

Usage

```
ducksemantics_record_judgments(conn, judgments, prefix = "semantic")
```

Arguments

conn	DBI connection.
judgments	Judgment data frame.
prefix	Prefix used for semantic tables.

Value

Invisibly, judgments.

ducksemantics_schema_sql *DuckDB semantic graph schema*

Description

Returns the core SQL DDL for the generic semantic graph and grounding contract. The schema is intentionally not HPO-specific: ontology terms, local concept graphs, memory nodes, and pi-bio-agent graph projections can all use the same node, alias, edge, mention, and judgment tables.

Usage

```
ducksemantics_schema_sql(prefix = "semantic")
```

Arguments

prefix	Prefix used for the generated table names.

Value

A character vector of SQL statements.

ducksemantics_tables *Semantic graph table names*

Description

Semantic graph table names

Usage

```
ducksemantics_tables(prefix = "semantic")
```

Arguments

prefix	Prefix used for generated table names.

Value

A named character vector.

ducksemantics_token_embedding_batch *Construct a token embedding batch*

Description

Construct a token embedding batch

Usage

```
ducksemantics_token_embedding_batch(
  embeddings,
  subject_id,
  subject_kind = "node",
  provider = "embedding",
  token_index = NULL,
  block_id = NULL,
  token = NULL,
  start_offset = NULL,
  end_offset = NULL,
  attrs = NULL
)
```

Arguments

embeddings	Numeric matrix with one row per token.
subject_id	Subject identifier for each token row.
subject_kind	Subject type, e.g. "node", "alias", "mention", or "document".
provider	Embedding provider label.
token_index	Token index within each subject block. Defaults to zero-based order within subject_id.
block_id	Matrix/block identifier. Defaults to one block per provider, subject_kind, and subject_id.
token	Optional token text for each row.
start_offset, end_offset	Optional zero-based source offsets.
attrs	Optional JSON text or other metadata for each token row.

---

ducksemantics_token_embedding_batch_from_provider

Construct a token embedding batch from a provider

Description

Construct a token embedding batch from a provider

Usage

```
ducksemantics_token_embedding_batch_from_provider(
  text,
  provider,
  subject_id = text,
  subject_kind = "node",
  provider_label = NULL,
  block_id = NULL,
  attrs = NULL,
  ...
)
```

Arguments

text	Character vector to embed.
provider	Object implementing DucksemanticsTokenEmbeddingProvider.
subject_id	Subject identifiers for input texts. Defaults to text.
subject_kind	Subject type for stored token rows.
provider_label	Stored provider label. Defaults to the provider label when available.
block_id	Optional block id per input text.
attrs	Optional attrs value per input text.
...	Extra arguments passed to ducksemantics_token_embed().

Value

A DucksemanticsTokenEmbeddingBatch object.

`ducksemantics_token_embedding_provider` *Wrap a token embedding function as a typed token provider*

Description

Wrap a token embedding function as a typed token provider

Usage

```
ducksemantics_token_embedding_provider(fun, label = "function-token")
```

Arguments

fun	Function accepting a character vector and returning one token-embedding object per input text.
label	Provider label for stored token rows.

Value

An object implementing DucksemanticsTokenEmbeddingProvider.

`ducksemantics_token_embedding_query` *Construct a token embedding late-interaction query*

Description

Construct a token embedding late-interaction query

Usage

```
ducksemantics_token_embedding_query(
  embeddings,
  provider = NULL,
  subject_kind = NULL,
  top_k = 10L,
  table = NULL,
  candidate_subject_id = NULL
)
```

Arguments

embeddings	Numeric query-token matrix.
provider	Optional provider filter.
subject_kind	Optional subject-kind filter.
top_k	Number of scored blocks to return.
table	Optional table to search. Defaults to semantic_token_embeddings.
candidate_subject_id	Optional candidate subject identifiers.

---

ducksemantics_tokens *Tokenize text for semantic grounding*

Description

Tokenize text for semantic grounding

Usage

```
ducksemantics_tokens(text)
```

Arguments

text	Character scalar.

Value

A data frame with token text, normalized token text, zero-based start offset, end offset, and token index.

---

ducksemantics_write_embeddings *Store an embedding batch in DuckDB*

Description

Embeddings are stored in DuckDB as native FLOAT[] vectors. Similarity search casts those vectors to fixed-size FLOAT[N] arrays so DuckDB's vector functions and optional HNSW index can be used directly.

Usage

```
ducksemantics_write_embeddings(
  batch,
  conn,
  prefix = "semantic",
  replace = FALSE
)
```

Arguments

batch	A DucksemanticsEmbeddingBatch object from ducksemantics_embedding_batch().
conn	DBI connection.
prefix	Prefix used for semantic tables.
replace	Delete existing embeddings for the same subjects and provider before inserting?

Value

Invisibly, the written embedding rows.

---

ducksemantics_write_graph

Write graph rows into the semantic store

Description

Write graph rows into the semantic store

Usage

```
ducksemantics_write_graph(
  conn,
  nodes = NULL,
  aliases = NULL,
  edges = NULL,
  prefix = "semantic",
  replace = FALSE,
  index = TRUE
)
```

Arguments

conn	DBI connection.
nodes	Data frame with node_id, family, and optional label, description, attrs, trust.
aliases	Data frame with node_id, alias, and optional alias_kind, source, weight, attrs.

edges	Data frame with from_id, predicate, to_id, and optional attrs, trust.
prefix	Prefix used for semantic tables.
replace	Delete existing rows from populated target tables before writing?
index	Rebuild the alias index after writing aliases?

Value

Invisibly, the semantic table names.

---

ducksemantics_write_obo

Write an OBO ontology into the semantic store

Description

Write an OBO ontology into the semantic store

Usage

```
ducksemantics_write_obo(
  conn,
  path,
  family,
  source = basename(path),
  prefix = "semantic",
  replace = FALSE,
  index = TRUE,
  include_obsolete = FALSE
)
```

Arguments

conn	DBI connection.
path	OBO file path.
family	Graph family label, for example "HPO" or "MONDO".
source	Source label stored on alias rows.
prefix	Prefix used for semantic tables.
replace	Delete existing graph rows before writing?
index	Rebuild the alias index?
include_obsolete	Include terms marked is_obsolete: true?

Value

The parsed graph rows, invisibly.

ducksemantics_write_token_embeddings

Store token embeddings for late-interaction scoring

Description

Native ColBERT document vectors are grouped by `block_id`, so exact MaxSim can compare a query-token matrix to a stored candidate matrix without changing the graph schema. Dense vectors in `semantic_embeddings` remain the inexpensive broad-retrieval layer.

Usage

```
ducksemantics_write_token_embeddings(
  batch,
  conn,
  prefix = "semantic",
  replace = FALSE
)
```

Arguments

`batch`	A `DucksemanticsTokenEmbeddingBatch` object from `ducksemantics_token_embedding_batch()`.
`conn`	DBI connection.
`prefix`	Prefix used for semantic tables.
`replace`	Delete existing token rows for the same subjects and provider before inserting?

Value

Invisibly, the written token embedding rows.

DucksemanticsAnnotator

Structural interface for text annotators

Description

An annotator grounds text against the semantic store and returns mention rows. The default implementation is the DuckDB lexical alias index.

Usage

```
DucksemanticsAnnotator
```

DucksemanticsDbConnection *DBI connection reference*

Description

S7 value object for a valid DBI connection.

Usage

```
DucksemanticsDbConnection(value = NULL)
```

Arguments

value	DBI connection.

Value

A DucksemanticsDbConnection object.

DucksemanticsEmbeddingBatch *Embedding batch for the semantic store*

Description

Embedding batch for the semantic store

Usage

```
DucksemanticsEmbeddingBatch(
  embeddings = integer(0),
  subject_id = character(0),
  subject_kind = character(0),
  provider = character(0),
  text = NULL,
  attrs = NULL
)
```

Arguments

embeddings	Numeric matrix with one row per subject.
subject_id	Subject identifiers matching embedding rows.
subject_kind	Subject type, e.g. "node", "alias", "mention", or "document".
provider	Embedding provider label.
text	Optional source text for each embedding.
attrs	Optional JSON text or other metadata for each embedding.

Value

A DucksemanticsEmbeddingBatch object.

DucksemanticsEmbeddingClusterSpec
Embedding clustering specification

Description

Embedding clustering specification

Usage

```
DucksemanticsEmbeddingClusterSpec(
  k = integer(0),
  provider = NULL,
  subject_kind = NULL,
  dimensions = NULL,
  table = NULL,
  run_id = character(0),
  seed = integer(0),
  nstart = integer(0),
  max_iter = integer(0)
)
```

Arguments

k	Number of clusters.
provider	Optional provider filter.
subject_kind	Optional subject-kind filter.
dimensions	Optional embedding dimension filter.
table	Optional embedding table. Defaults to semantic_embeddings.
run_id	Identifier written to cluster tables.
seed	Random seed used by stats::kmeans().
nstart	Number of starts used by stats::kmeans().
max_iter	Maximum k-means iterations.

Details

Clustering always materializes an ordinary finite R matrix from the DuckDB FLOAT[] rows.

Value

A DucksemanticsEmbeddingClusterSpec object.

DucksemanticsEmbeddingIndexSpec *Embedding index specification*

Description

Embedding index specification

Usage

```
DucksemanticsEmbeddingIndexSpec(
  dimensions = integer(0),
  provider = NULL,
  subject_kind = NULL,
  table = NULL,
  hnsw = logical(0),
  metric = character(0),
  load_vss = logical(0)
)
```

Arguments

dimensions	Embedding dimension.
provider	Optional provider filter.
subject_kind	Optional subject-kind filter.
table	Target table name. Defaults to semantic_embedding_index_<dimensions>.
hnsw	Create a DuckDB HNSW index on the materialized table?
metric	HNSW metric, usually "cosine", "l2sq", or "ip".
load_vss	Load DuckDB's vss extension before creating the HNSW index?

Value

A DucksemanticsEmbeddingIndexSpec object.

DucksemanticsEmbeddingMatrix *Embedding matrix contract*

Description

S7 value object for a finite numeric embedding matrix with an expected row count.

Usage

```
DucksemanticsEmbeddingMatrix(embeddings = integer(0), rows = integer(0))
```

Arguments

embeddings	Numeric matrix.
rows	Expected number of matrix rows.

Value

A DucksemanticsEmbeddingMatrix object.

DucksemanticsEmbeddingProvider

Structural interface for embedding providers

Description

An embedding provider accepts a character vector and returns a numeric matrix with one row per input text.

Usage

```
DucksemanticsEmbeddingProvider
```

DucksemanticsEmbeddingQuery

Embedding search query

Description

Embedding search query

Usage

```
DucksemanticsEmbeddingQuery(
  embedding = integer(0),
  provider = NULL,
  subject_kind = NULL,
  top_k = integer(0),
  metric = character(0),
  table = NULL
)
```

Arguments

`embedding`	Numeric query embedding.
`provider`	Optional provider filter.
`subject_kind`	Optional subject-kind filter.
`top_k`	Number of nearest rows to return.
`metric`	One of `"cosine"`, `"cosine_distance"`, `"l2"`, or `"inner_product"`.
`table`	Optional table to search. Defaults to `semantic_embeddings`. Pass a table created by `ducksemantics_materialize_embedding_index()` to use a dimensioned, HNSW-indexable table.

Value

A `DucksemanticsEmbeddingQuery` object.

`DucksemanticsJudgmentParser` *Structural interface for judgment parsers*

Description

A judgment parser turns raw model text into a data frame that includes `mention_id` and `decision`.

Usage

```
DucksemanticsJudgmentParser
```

`DucksemanticsPromptRunner` *Structural interface for prompt runners*

Description

A prompt runner accepts a prompt string and returns response text. BebeLM is one implementation; cloud LLMs, test fixtures, and other local models should implement the same generic instead of changing downstream judgment code.

Usage

```
DucksemanticsPromptRunner
```

DucksemanticsScalarText

Non-empty scalar text

Description

S7 value object for a required non-empty character scalar.

Usage

```
DucksemanticsScalarText(value = character(0))
```

Arguments

value	Character scalar.

Value

A DucksemanticsScalarText object.

DucksemanticsSqlIdentifier

SQL identifier

Description

S7 value object for table and column identifiers used when constructing DuckDB SQL.

Usage

```
DucksemanticsSqlIdentifier(value = character(0), qualified = logical(0))
```

Arguments

value	Identifier text.
qualified	Whether value may contain schema/table qualification.

Value

A DucksemanticsSqlIdentifier object.

DucksemanticsTable *Data frame contract*

Description

S7 value object for a data frame with required columns.

Arguments

value	Data frame.
required	Required column names.
allow_empty	Whether zero-row input is allowed.

Value

A DucksemanticsTable object.

DucksemanticsTokenEmbeddingBatch *Token embedding batch for late-interaction storage*

Description

Token embedding batch for late-interaction storage

Usage

```
DucksemanticsTokenEmbeddingBatch(
  embeddings = integer(0),
  subject_id = character(0),
  subject_kind = character(0),
  provider = character(0),
  token_index = integer(0),
  block_id = character(0),
  token = NULL,
  start_offset = NULL,
  end_offset = NULL,
  attrs = NULL
)
```

Arguments

`embeddings`	Numeric matrix with one row per token.
`subject_id`	Subject identifier for each token row.
`subject_kind`	Subject type, e.g. `"node"`, `"alias"`, `"mention"`, or `"document"`.
`provider`	Embedding provider label.
`token_index`	Token index within each subject block. Defaults to zero-based order within `subject_id`.
`block_id`	Matrix/block identifier. Defaults to one block per `provider`, `subject_kind`, and `subject_id`.
`token`	Optional token text for each row.
`start_offset, end_offset`	Optional zero-based source offsets.
`attrs`	Optional JSON text or other metadata for each token row.

Details

Token vectors are always persisted as DuckDB `FLOAT[]` rows. This makes the index durable and queryable without an external allocator.

Value

A `DucksemanticsTokenEmbeddingBatch` object.

`DucksemanticsTokenEmbeddingProvider` *Structural interface for token embedding providers*

Description

A token embedding provider accepts a character vector and returns one token-embedding object per input text. Each object contains an `embeddings` matrix and token metadata.

Usage

```
DucksemanticsTokenEmbeddingProvider
```

DucksemanticsTokenEmbeddingQuery
Token embedding late-interaction query

Description

Token embedding late-interaction query

Usage

```
DucksemanticsTokenEmbeddingQuery(
  embeddings = integer(0),
  provider = NULL,
  subject_kind = NULL,
  top_k = integer(0),
  table = NULL,
  candidate_subject_id = NULL
)
```

Arguments

embeddings	Numeric query-token matrix.
provider	Optional provider filter.
subject_kind	Optional subject-kind filter.
top_k	Number of scored blocks to return.
table	Optional table to search. Defaults to semantic_token_embeddings.
candidate_subject_id	Optional candidate subject identifiers.

Value

A DucksemanticsTokenEmbeddingQuery object.

Index