

Package: ducksemantics (via r-universe)

October 4, 2026

Type Package

Title DuckDB-Native Semantic and Retrieval Relations

Version 0.2.1

Description DuckDB-native ontology graphs, validated Human Phenotype Ontology (HPO) observations, typed release-cataloged Monarch relations including read-only projections of attached dated packs, provider/snapshot-catalog-bound literature retrieval, and source-grounded semantic judgments. Bulk values are ordinary relations; optional S7 protocols are reserved for real prompt, embedding, parser, and annotator providers. The package does not store ClinVar or PubMed source data, download Monarch packs, rank cases, or evaluate benchmarks.

License MIT + file LICENSE

URL <https://github.com/RGenomicsETL/ducksemantics>,
<https://RGenomicsETL.github.io/ducksemantics/>

BugReports <https://github.com/RGenomicsETL/ducksemantics/issues>

Encoding UTF-8

Depends R (>= 4.3.0)

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Imports DBI, glue, 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 2026-08-05 15:18:52 UTC

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

RemoteRef HEAD

RemoteSha 6d5f799ab0a107149cc56ba70064dc812c525c8e

Contents

ducksemantics_annotate	3
ducksemantics_bebel_runner	4
ducksemantics_bebel_tool_judgment_parser	4
ducksemantics_closure_sql	5
ducksemantics_colbert_provider	5
ducksemantics_connect	6
ducksemantics_default_judgment_instructions	6
ducksemantics_embedding_batch	7
ducksemantics_embedding_provider	7
ducksemantics_embedding_search	8
ducksemantics_embeddinggemma_provider	9
ducksemantics_hpo_observation_contract	10
ducksemantics_hpo_observations	10
ducksemantics_index_aliases	11
ducksemantics_init	11
ducksemantics_json_judgment_parser	12
ducksemantics_judge	12
ducksemantics_judgment_prompt	13
ducksemantics_late_interaction_search	14
ducksemantics_lexical_annotator	14
ducksemantics_literature_retrieve	15
ducksemantics_literature_snapshot_catalog	16
ducksemantics_materialize_embedding_index	17
ducksemantics_monarch_gene_disease_holdout_audit	18
ducksemantics_monarch_import	18
ducksemantics_monarch_project_pack	19
ducksemantics_monarch_query	20
ducksemantics_monarch_release_catalog	21
ducksemantics_normalize	22
ducksemantics_projection_sql	22
ducksemantics_prompt_runner	23
ducksemantics_provider_generics	23
ducksemantics_read_obo	25
ducksemantics_record_judgments	25
ducksemantics_schema_sql	26
ducksemantics_tables	26
ducksemantics_token_embedding_batch	27
ducksemantics_token_embedding_batch_from_provider	28
ducksemantics_token_embedding_provider	29
ducksemantics_tokens	29
ducksemantics_write_embeddings	30
ducksemantics_write_graph	30
ducksemantics_write_obo	31
ducksemantics_write_token_embeddings	32
DucksemanticsAnnotator	32
DucksemanticsEmbeddingProvider	33

<i>ducksemantics_annotate</i>	3
DucksemanticsJudgmentParser	33
DucksemanticsPromptRunner	33
DucksemanticsTokenEmbeddingProvider	34
Index	35

ducksemantics_annotate
<i>Generate lexical HPO/ontology candidates</i>

Description

These candidates are not accepted observations. Use [ducksemantics_hpo_observations\(\)](#) to validate a separately accepted HPO observation relation against source documents.

Usage

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

Arguments

conn	DBI connection.
text	Source text.
document_id	Optional source-document identity.
prefix	Table prefix.
longest_match	Drop spans contained by a longer span.
record	Append candidate rows to the semantic mention table.

Value

Candidate data frame.

`ducksemantics_bebel_runner`*Create a BebeLM prompt provider*

Description

Create a BebeLM prompt provider

Usage

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

Arguments

<code>agent</code>	An Rbebelm agent.
<code>on_event</code>	Optional 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

<code>tool_name</code>	Accepted tool-call names, or NULL.
------------------------	------------------------------------

Value

An object implementing [DucksemanticsJudgmentParser](#).

`ducksemantics_closure_sql`*Materialize transitive closure SQL*

Description

Materialize transitive closure SQL

Usage

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

Arguments

<code>transitive_predicates</code>	Non-empty predicate names, or <code>character()</code> .
<code>source_table</code>	Source edge relation.
<code>target_table</code>	Target closure relation.

Value

SQL script.

`ducksemantics_colbert_provider`*Create a native ColBERT provider*

Description

Create a native ColBERT provider

Usage

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

Arguments

model	An Rbebelm ColbertModel.
role	Query or document encoding role.
label	Provider identity.

Value

An object implementing [DucksemanticsTokenEmbeddingProvider](#).

ducksemantics_connect	<i>Connect to a DuckDB semantic store</i>
-----------------------	---

Description

Connect to a DuckDB semantic store

Usage

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

Arguments

dbdir	DuckDB path or ":memory:".
read_only	Open read-only.
array	DuckDB array conversion mode.

Value

A DBI connection.

ducksemantics_default_judgment_instructions	<i>Default source-grounded judgment instructions</i>
---	--

Description

Default source-grounded judgment instructions

Usage

```
ducksemantics_default_judgment_instructions()
```

Value

A character scalar.

ducksemantics_embedding_batch
Construct embedding rows

Description

Bulk embedding values are an ordinary data frame with one `FLOAT[]`-ready numeric vector per row; they are not an `S7` value object.

Usage

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

Arguments

<code>embeddings</code>	Finite numeric matrix with one row per subject.
<code>subject_id</code>	Non-empty subject identifiers.
<code>subject_kind</code>	Subject type.
<code>provider</code>	Provider identity.
<code>text</code>	Optional source text per row.
<code>attrs</code>	Optional metadata text per row.

Value

A data frame suitable for `ducksemantics_write_embeddings()`.

ducksemantics_embedding_provider
Wrap an embedding function as an embedding provider

Description

Wrap an embedding function as an embedding provider

Usage

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

Arguments

fun	Function accepting a prompt and returning response text.
label	Provider identity.

Value

An object implementing [DucksemanticsEmbeddingProvider](#).

ducksemantics_embedding_search
Search dense embeddings

Description

Search dense embeddings

Usage

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

Arguments

conn	DBI connection.
embedding	Numeric query vector.
provider	Optional provider identity.
subject_kind	Optional subject-kind filter.
top_k	Number of results.
metric	"cosine", "cosine_distance", "l2", or "inner_product".
table	Caller-selected embedding table.

Value

Ranked embedding rows.

```
ducksemantics_embeddinggemma_provider
```

Create an EmbeddingGemma provider

Description

Create an EmbeddingGemma provider

Usage

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

Arguments

<code>model</code>	An Rbebelm EmbeddingGemmaModel.
<code>label</code>	Provider identity.
<code>task</code>	EmbeddingGemma task.
<code>title</code>	Optional document title.
<code>dimensions</code>	Matryoshka dimension.
<code>normalize</code>	L2-normalize output rows.
<code>truncate</code>	Truncate overly long input.
<code>check_interrupt</code>	Poll for R interrupts.

Value

An object implementing [DucksemanticsEmbeddingProvider](#).

ducksemantics_hpo_observation_contract
<i>HPO observation contract</i>

Description

The final observation relation has exactly these required semantic fields: document_id, hpo_id, start_offset, end_offset, source_text, context_status, method, provider_id, provider_version, confidence, and status. Offsets are zero-based, half-open R character offsets. Lexical candidates returned by `ducksemantics_annotate()` are not observations and cannot be substituted for this relation.

Usage

```
ducksemantics_hpo_observation_contract()
```

Value

Required HPO observation column names.

ducksemantics_hpo_observations
<i>Validate accepted HPO observations against source documents</i>

Description

Validate accepted HPO observations against source documents

Usage

```
ducksemantics_hpo_observations(documents, observations, conn = NULL)
```

Arguments

documents	Data frame (or a caller-owned DuckDB table name with conn) containing document_id and source_text.
observations	Data frame (or caller-owned DuckDB table name) following ducksemantics_hpo_observation_contract
conn	Optional DBI connection for a lazy caller-owned table name.

Value

Validated accepted-observation data frame.

ducksemantics_index_aliases	<i>Build the lexical alias index</i>
-----------------------------	--------------------------------------

Description

Build the lexical alias index

Usage

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

Arguments

conn	DBI connection.
prefix	Table prefix.

Value

Invisibly, index table name.

ducksemantics_init	<i>Initialize semantic graph tables</i>
--------------------	---

Description

Initialize semantic graph tables

Usage

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

Arguments

conn	DBI connection.
prefix	Table prefix.

Value

Invisibly, conn.

ducksemantics_json_judgment_parser
Create a JSON judgment parser

Description

Create a JSON judgment parser

Usage

```
ducksemantics_json_judgment_parser()
```

Value

An object implementing [DucksemanticsJudgmentParser](#).

ducksemantics_judge *Judge lexical candidates with a provider*

Description

Judge lexical candidates with a provider

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	Candidate relation.
runner	Prompt provider.
conn	Optional DBI connection to record judgments.
prefix	Table prefix.
graph_context	Optional graph context.
instructions	Explicit judgment instructions.
prompt_builder	Function building the prompt.
parser	Judgment parser provider.
record	Record result when conn is supplied.
model	Provider/model identity.
...	Arguments passed to prompt_builder.

Value

Source-grounded judgment relation.

ducksemantics_judgment_prompt
Build a source-grounded judgment prompt

Description

Build a source-grounded judgment prompt

Usage

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

Arguments

text	Source text.
mentions	Candidate data frame.
graph_context	Optional graph context relation.
instructions	Explicit judgment policy.

Value

Prompt text.

 ducksemantics_late_interaction_search

Exact token-level late-interaction search

Description

Exact token-level late-interaction search

Usage

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

Arguments

conn	DBI connection.
embeddings	Numeric query-token matrix.
provider	Optional provider identity.
subject_kind	Optional subject-kind filter.
top_k	Number of blocks to return.
table	Caller-selected token embedding table.
candidate_subject_id	Optional candidate restriction.

Value

Ranked exact MaxSim rows.

 ducksemantics_lexical_annotator

Create the default lexical grounding provider

Description

Create the default lexical grounding provider

Usage

ducksemantics_lexical_annotator()

Value

An object implementing [DucksemanticsAnnotator](#).

ducksemantics_literature_retrieve
<i>Retrieve exact literature spans from append-only caller-owned relations</i>

Description

RClinVarbitration owns the article and section relations. Retrieval binds a provider and exact cataloged source-order cutoff, selects the maximum article source_ordinal at or below that cutoff per provider/article, drops a latest deletion event, and joins only sections from that exact version. It never imports, writes, or fetches literature.

Usage

```
ducksemantics_literature_retrieve(  
  articles,  
  sections,  
  snapshots,  
  query,  
  provider_id,  
  snapshot_id,  
  section_names = NULL,  
  conn = NULL  
)
```

Arguments

articles	Caller-owned relation with provider_id, article_id, pmid, version_id, finite integer source_ordinal, and logical is_deleted.
sections	Caller-owned relation with provider_id, article_id, pmid, version_id, finite integer source_ordinal, section, and exact non-empty text.
snapshots	Source snapshot catalog from ducksemantics_literature_snapshot_catalog() .
query	Non-empty lexical query text.
provider_id	Provider identity.
snapshot_id	Exact provider snapshot identity.
section_names	Optional non-empty section names to search; NULL searches every version-bound section.
conn	Optional DBI connection for table relations.

Details

A read-only RClinVarbitration projection joins pubmed_sources to pubmed_articles and pubmed_abstracts: use source provider as provider_id, source id as version_id/snapshot_id, source ordinal for both ordinal fields, and source_applied_at as effective_at. A section must retain the exact article event’s version id and source ordinal.

Value

Exact non-empty source spans from non-deleted article versions visible at the cataloged source-order cutoff.

ducksemantics_literature_snapshot_catalog
<i>Validate an append-only literature snapshot catalog</i>

Description

RClinVarbitration owns article and section storage. This catalog identifies a provider snapshot by its source-order cutoff; it creates no shadow literature table.

Usage

ducksemantics_literature_snapshot_catalog(snapshots, conn = NULL)

Arguments

snapshots	Data frame or caller-owned DuckDB table name containing provider_id, snapshot_id, and finite integer high_water_ordinal; effective_at is an optional typed POSIXct timestamp.
conn	Optional DBI connection for a table relation.

Value

Validated source snapshot catalog.

ducksemantics_materialize_embedding_index
<i>Materialize a fixed-dimension embedding table</i>

Description

Materialize a fixed-dimension embedding table

Usage

```
ducksemantics_materialize_embedding_index(  
  conn,  
  dimensions,  
  provider = NULL,  
  subject_kind = NULL,  
  table = NULL,  
  source_table = "semantic_embeddings",  
  hnswn = FALSE,  
  metric = "cosine"  
)
```

Arguments

conn	DBI connection.
dimensions	Embedding width.
provider	Optional provider filter.
subject_kind	Optional subject kind filter.
table	Target table name.
source_table	Source embedding table.
hnswn	Create an optional DuckDB HNSW index.
metric	HNSW metric.

Value

Target table name.

 ducksemantics_monarch_gene_disease_holdout_audit

Audit a historical Monarch gene-disease holdout

Description

The selected training and holdout releases must be exact rows of one typed provider catalog. The result is a provenance-preserving anti-join.

Usage

```
ducksemantics_monarch_gene_disease_holdout_audit(
  facts,
  releases,
  provider_id,
  train_release_id,
  holdout_release_id,
  conn = NULL
)
```

Arguments

facts	Caller-supplied gene-disease relation.
releases	Typed Monarch release catalog.
provider_id	Provider identity.
train_release_id	Exact training release identity.
holdout_release_id	Exact holdout release identity.
conn	Optional DBI connection for table relations.

Value

Holdout rows absent from the historical relation.

 ducksemantics_monarch_import

Import caller-supplied Monarch relations

Description

This function never downloads Monarch data. A typed release catalog is required to prove that every fact has a known provider/release identity.

Usage

```

ducksemantics_monarch_import(
  facts,
  releases,
  relation = c("gene_phenotype", "gene_disease", "disease_phenotype"),
  conn = NULL,
  table = NULL,
  replace = FALSE
)

```

Arguments

facts	Gene-phenotype, gene-disease, or disease-phenotype data frame, or a caller-owned DuckDB table name with conn.
releases	Typed catalog from ducksemantics_monarch_release_catalog() .
relation	One of "gene_phenotype", "gene_disease", or "disease_phenotype".
conn	Optional DBI connection.
table	Optional destination table when conn is supplied.
replace	Replace destination rows.

Value

Normalized Monarch relation.

ducksemantics_monarch_project_pack

Project one attached, dated Monarch annotation pack

Description

Projects an already attached read-only official Monarch DuckDB pack into connection-local temporary views. It never attaches, downloads, collects, copies, rewrites, or merges pack rows. `source_catalog` must name an attached DuckDB catalog whose main schema contains the official `denormalized_edges`, `nodes`, `information_content`, and `node_has_phenotype` tables. The edge table is checked before any view is made, including its scalar role categories, provider-serialized text negation, and array provenance fields. The official required nodes, `node_has_phenotype`, and `information_content` projection columns/types are also validated.

Usage

```

ducksemantics_monarch_project_pack(
  conn,
  source_catalog,
  releases,
  provider_id,

```

```
    release_id,  
    prefix = "semantic_monarch"  
)
```

Arguments

conn	Valid DBI connection with the pack already attached.
source_catalog	Attached DuckDB catalog name, not the current catalog.
releases	Typed Monarch release catalog or caller-owned catalog table. This binds provider/release semantics; the caller must verify the immutable file receipt before attaching the pack.
provider_id	Exact provider identity in releases.
release_id	Exact, dated release identity in releases; "latest" is rejected.
prefix	Safe unqualified prefix for temporary view names.

Details

associations retains every source edge and all source columns exactly, then adds the selected provider and release. Each role view is likewise a total projection: it retains raw source columns and adds normalized role columns, source_subject, source_object, source_direction, and association_status. Only supported_subject_to_object and supported_object_to_subject rows have non-missing normalized roles. missing_endpoint, missing_predicate, malformed_negation, missing_role_category, malformed_role_category, negated, and unsupported_orientation are not support. negation_status separately distinguishes not_provided, not_negated, negated, and malformed: Monarch omits the optional qualifier on many positive assertion rows, so omission is retained but does not erase the assertion. In particular, a missing predicate never becomes a causal predicate.

Value

A data frame naming the seven connection-local temporary views, with relation, view_name, provider_id, and release_id columns. It does not return pack contents.

ducksemantics_monarch_query
<i>Select one cataloged Monarch release</i>

Description

Select one cataloged Monarch release

Usage

```

ducksemantics_monarch_query(
  facts,
  releases,
  relation = c("gene_phenotype", "gene_disease", "disease_phenotype"),
  provider_id,
  release_id = NULL,
  as_of = NULL,
  as_of_ordinal = NULL,
  conn = NULL
)

```

Arguments

facts	Monarch fact relation or a caller-owned DuckDB table name.
releases	Typed Monarch release catalog.
relation	Relation type.
provider_id	Provider identity to bind.
release_id	Exact catalog release identity.
as_of	Typed Date or POSIXct cutoff.
as_of_ordinal	Numeric source-order cutoff.
conn	Optional DBI connection for table relations.

Value

Facts from one selected provider/release. Selection metadata is held in `release_selected` and `provider_selected` attributes.

```
ducksemantics_monarch_release_catalog
```

Validate a typed Monarch release catalog

Description

A catalog row identifies one provider release. It must carry either a typed `effective_date` (Date or POSIXct) or a finite numeric `source_ordinal`; both may be supplied when the provider declares both.

Usage

```
ducksemantics_monarch_release_catalog(releases, conn = NULL)
```

Arguments

releases	Data frame or caller-owned DuckDB table name containing release_id, provider_id, and one or both ordering columns.
conn	Optional DBI connection for a table relation.

Value

Validated release catalog.

ducksemantics_normalize
<i>Normalize semantic text</i>

Description

Normalize semantic text

Usage

```
ducksemantics_normalize(text)
```

Arguments

text	Character vector.
------	-------------------

Value

Normalized character vector.

ducksemantics_projection_sql
<i>Project edge-shaped data to graph SQL</i>

Description

Project edge-shaped data to graph SQL

Usage

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

Arguments

source_table	Source relation name.
from, predicate, to	Source columns.
target_table	Target table.
attrs, trust	Optional source columns.

Value

SQL script.

ducksemantics_prompt_runner
<i>Wrap a prompt function as a prompt provider</i>

Description

Wrap a prompt function as a prompt provider

Usage

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

Arguments

fun	Function accepting a prompt and returning response text.
label	Provider identity.

Value

An object implementing [DucksemanticsPromptRunner](#).

ducksemantics_provider_generics
<i>Provider protocol generics</i>

Description

These S7 generics are reserved for real pluggable providers. All bulk input and output remains an ordinary data frame or matrix.

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 response text.
annotator	Grounding provider.
conn	DBI connection.
document_id	Optional document identifier.
prefix	Semantic-table prefix.
longest_match	Drop nested lexical candidates.
record	Persist candidates.

Value

Provider-specific text, matrix, or data frame.

`ducksemantics_read_obo`*Read an OBO ontology into graph relations*

Description

Read an OBO ontology into graph relations

Usage

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

Arguments

<code>path</code>	OBO file path.
<code>family</code>	Ontology family.
<code>source</code>	Alias source identity.
<code>include_obsolete</code>	Include obsolete terms.

Value

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

`ducksemantics_record_judgments`*Record source-grounded judgments*

Description

Record source-grounded judgments

Usage

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

Arguments

<code>conn</code>	DBI connection.
<code>judgments</code>	Judgment relation.
<code>prefix</code>	Table prefix.

Value

Invisibly, validated judgment rows.

ducksemantics_schema_sql
<i>DuckDB semantic graph schema</i>

Description

DuckDB semantic graph schema

Usage

ducksemantics_schema_sql(prefix = "semantic")

Arguments

prefix Prefix for semantic tables.

Value

Character DDL statements.

ducksemantics_tables	<i>Semantic table names</i>
----------------------	-----------------------------

Description

Semantic table names

Usage

ducksemantics_tables(prefix = "semantic")

Arguments

prefix Table prefix.

Value

A named character vector.

ducksemantics_token_embedding_batch
Construct token embedding rows

Description

Construct token embedding rows

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	Finite numeric matrix with one row per subject.
subject_id	Non-empty subject identifiers.
subject_kind	Subject type.
provider	Provider identity.
token_index	Zero-based token index within a document block.
block_id	Optional block identifier.
token	Optional token text.
start_offset, end_offset	Optional zero-based half-open source offsets.
attrs	Optional metadata text per row.

Value

A data frame suitable for [ducksemantics_write_token_embeddings\(\)](#).

`ducksemantics_token_embedding_batch_from_provider`*Construct token embedding rows from a provider*

Description

Construct token embedding rows 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

<code>text</code>	Character vector to embed.
<code>provider</code>	Object implementing DucksemanticsTokenEmbeddingProvider .
<code>subject_id</code>	Subject identifiers for input texts.
<code>subject_kind</code>	Subject type for stored rows.
<code>provider_label</code>	Stored provider identity.
<code>block_id</code>	Optional block identifier per input text.
<code>attrs</code>	Optional metadata per input text.
<code>...</code>	Arguments forwarded to ducksemantics_token_embed() .

Value

A token-embedding data frame.

ducksemantics_token_embedding_provider
<i>Wrap a token embedding function as a token provider</i>

Description

Wrap a token embedding function as a token provider

Usage

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

Arguments

- | | |
|-------|--|
| fun | Function accepting a prompt and returning response text. |
| label | Provider identity. |

Value

An object implementing [DucksemanticsTokenEmbeddingProvider](#).

ducksemantics_tokens	<i>Tokenize source text</i>
----------------------	-----------------------------

Description

Tokenize source text

Usage

ducksemantics_tokens(text)

Arguments

- | | |
|------|-------------------|
| text | Character scalar. |
|------|-------------------|

Value

Token data frame with zero-based half-open character offsets.

ducksemantics_write_embeddings
Store embedding rows in DuckDB

Description

Store embedding rows in DuckDB

Usage

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

Arguments

conn	DBI connection.
embeddings	Data frame from ducksemantics_embedding_batch() .
prefix	Table prefix.
replace	Replace existing rows with the same subject and provider.

Value

Invisibly, written rows.

ducksemantics_write_graph
Write ontology graph rows

Description

Write ontology graph rows

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 and family.
aliases	Data frame with node_id and alias.
edges	Data frame with from_id, predicate, and to_id.
prefix	Table prefix.
replace	Replace supplied target relations.
index	Rebuild lexical alias index.

Value

Invisibly, semantic table names.

ducksemantics_write_obo

Write an OBO ontology into the graph

Description

Write an OBO ontology into the graph

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	Ontology family.
source	Alias source identity.
prefix	Table prefix.
replace	Replace graph data.
index	Rebuild alias index.
include_obsolete	Include obsolete terms.

Value

Parsed graph relations, invisibly.

ducksemantics_write_token_embeddings
<i>Store token embedding rows in DuckDB</i>

Description

Store token embedding rows in DuckDB

Usage

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

Arguments

conn	DBI connection.
embeddings	Data frame from ducksemantics_token_embedding_batch() .
prefix	Table prefix.
replace	Replace existing rows with the same subject and provider.

Value

Invisibly, written rows.

DucksemanticsAnnotator
<i>Grounding-provider protocol</i>

Description

Grounding-provider protocol

Usage

DucksemanticsAnnotator

DucksemanticsEmbeddingProvider
<i>Embedding-provider protocol</i>

Description

Embedding-provider protocol

Usage

DucksemanticsEmbeddingProvider

DucksemanticsJudgmentParser
<i>Judgment-parser provider protocol</i>

Description

Judgment-parser provider protocol

Usage

DucksemanticsJudgmentParser

DucksemanticsPromptRunner
<i>Prompt-runner provider protocol</i>

Description

Prompt-runner provider protocol

Usage

DucksemanticsPromptRunner

DucksemanticsTokenEmbeddingProvider

Token-embedding-provider protocol

Description

Token-embedding-provider protocol

Usage

DucksemanticsTokenEmbeddingProvider

Index

ducksemantics_annotate, 3
ducksemantics_annotate(), 10
ducksemantics_bebel_runner, 4
ducksemantics_bebel_tool_judgment_parser,
4
ducksemantics_closure_sql, 5
ducksemantics_colbert_provider, 5
ducksemantics_connect, 6
ducksemantics_default_judgment_instructions,
6
ducksemantics_embed
(ducksemantics_provider_generics),
23
ducksemantics_embedding_batch, 7
ducksemantics_embedding_batch(), 30
ducksemantics_embedding_provider, 7
ducksemantics_embedding_search, 8
ducksemantics_embeddinggemma_provider,
9
ducksemantics_ground
(ducksemantics_provider_generics),
23
ducksemantics_hpo_observation_contract,
10
ducksemantics_hpo_observation_contract(),
10
ducksemantics_hpo_observations, 10
ducksemantics_hpo_observations(), 3
ducksemantics_index_aliases, 11
ducksemantics_init, 11
ducksemantics_json_judgment_parser, 12
ducksemantics_judge, 12
ducksemantics_judgment_prompt, 13
ducksemantics_late_interaction_search,
14
ducksemantics_lexical_annotator, 14
ducksemantics_literature_retrieve, 15
ducksemantics_literature_snapshot_catalog,
16
ducksemantics_literature_snapshot_catalog(),
15
ducksemantics_materialize_embedding_index,
17
ducksemantics_monarch_gene_disease_holdout_audit,
18
ducksemantics_monarch_import, 18
ducksemantics_monarch_project_pack, 19
ducksemantics_monarch_query, 20
ducksemantics_monarch_release_catalog,
21
ducksemantics_monarch_release_catalog(),
19
ducksemantics_normalize, 22
ducksemantics_parse
(ducksemantics_provider_generics),
23
ducksemantics_projection_sql, 22
ducksemantics_prompt_runner, 23
ducksemantics_provider_generics, 23
ducksemantics_read_obo, 25
ducksemantics_record_judgments, 25
ducksemantics_run
(ducksemantics_provider_generics),
23
ducksemantics_schema_sql, 26
ducksemantics_tables, 26
ducksemantics_token_embed
(ducksemantics_provider_generics),
23
ducksemantics_token_embed(), 28
ducksemantics_token_embedding_batch,
27
ducksemantics_token_embedding_batch(),
32
ducksemantics_token_embedding_batch_from_provider,
28
ducksemantics_token_embedding_provider,
29

ducksemantics_tokens, [29](#)
ducksemantics_write_embeddings, [30](#)
ducksemantics_write_embeddings(), [7](#)
ducksemantics_write_graph, [30](#)
ducksemantics_write_obo, [31](#)
ducksemantics_write_token_embeddings,
 [32](#)
ducksemantics_write_token_embeddings(),
 [27](#)
DucksemanticsAnnotator, [15](#), [32](#)
DucksemanticsEmbeddingProvider, [8](#), [9](#), [33](#)
DucksemanticsJudgmentParser, [4](#), [12](#), [33](#)
DucksemanticsPromptRunner, [4](#), [23](#), [33](#)
DucksemanticsTokenEmbeddingProvider, [6](#),
 [28](#), [29](#), [34](#)