Package: okf (via r-universe)

June 23, 2026

Title Open Knowledge Format (OKF) Ingestion

Version 0.3.0

Description Read, validate, and load Open Knowledge Format (OKF) bundles (a directory of markdown files with YAML frontmatter) into a portable DuckDB catalog, build the concept graph, and optionally embed concept bodies for semantic search. Conformant and permissive per the OKF v0.1 specification.

License Apache License (>= 2)

Encoding UTF-8

Depends R (>= 4.1.0)

Imports yaml, DBI, duckdb, digest, jsonlite, utils

Suggests httr2, commonmark

RoxygenNote 7.3.2

Config/pak/sysreqs xz-utils

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

Date/Publication 2026-06-23 18:16:14 UTC

RemoteUrl https://github.com/travisjakel/okf-ingest

RemoteRef HEAD

RemoteSha 60031ffb523192e2b80e1213a150c3f5a7b64611

RemoteSubdir r/okf

Contents

`okf_chunk_body` *Split a concept body into chunks on paragraph boundaries.*

Description

Split a concept body into chunks on paragraph boundaries.

Usage

```
okf_chunk_body(body, target_chars = 600L)
```

Arguments

`body`	Concept body text.
`target_chars`	Approximate maximum chunk size in characters.

Value

Character vector of chunks.

`okf_context` *Assemble an index-first, link-following slice of a bundle as one markdown blob for direct LLM consumption.*

Description

This is the OKF / "LLM wiki" consume primitive (Karpathy): hand the agent ‘index.md‘ plus the relevant concept(s) and their link-neighborhood to read directly. It uses the concept graph – **no embeddings, no vector search**. With ‘start‘, it walks the (undirected) link graph from that concept to ‘depth‘; without ‘start‘, it packs all concepts. Output is capped to roughly ‘max_tokens‘ (estimated at ~4 chars/token).

Usage

```
okf_context(
  con,
  start = NULL,
  depth = 1L,
  max_tokens = 8000L,
  include_index = TRUE
)
```

Arguments

con	An open DuckDB connection to an okf catalog.
start	Optional concept path to center the neighborhood on.
depth	Link-graph radius around ‘start‘ (ignored when ‘start‘ is NULL).
max_tokens	Approximate output budget.
include_index	Prepend ‘index.md‘ (the map) when present.

Value

A list with ‘text‘ (the markdown blob), ‘included‘/‘omitted‘ concept paths, and ‘est_tokens‘.

---

okf_embed *Chunk and embed concept bodies into the catalog for semantic search.*

Description

Idempotent: replaces any existing chunks. Populates ‘okf_chunk‘ with one row per chunk plus its embedding vector.

Usage

```
okf_embed(con, embedder = NULL, target_chars = 600L)
```

Arguments

con	An open DuckDB connection to an okf catalog.
embedder	An embedder function; defaults to [okf_ollama_embedder()].
target_chars	Approximate chunk size in characters.

Value

The number of chunks written (invisibly usable as an integer).

`okf_extract_links` *Extract markdown link targets from a concept body (OKF cross-links, sec. 4).*

Description

Extract markdown link targets from a concept body (OKF cross-links, sec. 4).

Usage

```
okf_extract_links(body)
```

Arguments

`body`	Concept body text.

Value

Character vector of raw link targets (as written).

`okf_fetch` *Materialize an OKF bundle from a directory, git URL, or tar/zip archive.*

Description

Local directories are used in place. Git URLs (github/gitlab/bitbucket, '.git', or 'git@') are shallow-cloned. Tar/zip archives (local path or 'http(s)' URL) are downloaded if remote and extracted. The caller MUST invoke the returned 'cleanup()' when done to remove any temporary files.

Usage

```
okf_fetch(source, subdir = NULL, branch = NULL)
```

Arguments

`source`	A directory path, git URL, or tar/zip path/URL.
`subdir`	Optional bundle path within the cloned/extracted tree.
`branch`	Optional git branch or tag (git sources only).

Value

A list with 'dir' (the resolved bundle directory), 'source_kind' ('"dir"'/'"git"'/'"tar"'/'"zip"'), and 'cleanup' (a function).

okf_html *Render an ingested OKF catalog to HTML for viewing.*

Description

Two modes. As a navigable **site** (‘single = FALSE‘, the default), writes one self-contained ‘.html‘ per concept under ‘out/‘ (mirroring the bundle’s directory tree) plus an ‘index.html‘ landing page; internal ‘.md‘ links are rewritten to ‘.html‘. As a **single file** (‘single = TRUE‘), writes one self-contained ‘.html‘ at path ‘out‘, with each concept an anchored ‘<section>‘ and intra-bundle links rewritten to in-page anchors. No JavaScript; CSS is inlined so output is portable. Reserved concepts (‘index.md‘, ‘log.md‘) are rendered too. Bodies are rendered with the commonmark package; broken/orphan links are surfaced in a per-page footer badge from the validation findings.

Usage

```
okf_html(con, out, single = FALSE, site_title = NULL)
```

Arguments

con	An open DuckDB connection to an okf catalog (from [okf_ingest()]).
out	Output directory (site mode) or output ‘.html‘ file path (single).
single	Emit one self-contained file instead of a per-concept site.
site_title	Optional title for the landing page / single-file header; defaults to the bundle directory name.

Value

A list with ‘files‘ (paths written), ‘n_concepts‘, and ‘mode‘ (invisibly).

okf_ingest *Ingest an OKF bundle into a DuckDB catalog.*

Description

Reads, validates, and loads the bundle into the ‘okf_bundle‘, ‘okf_concept‘, ‘okf_link‘, and ‘okf_validation‘ tables of a (file or in-memory) DuckDB database.

Usage

```
okf_ingest(
  root,
  db_path = ":memory:",
  ingested_at = NULL,
  bundle_id = NULL,
  source_kind = "dir",
```

```
  subdir = NULL,
  branch = NULL
)
```

Arguments

root	A bundle directory path, a git URL, a tar/zip path or URL, or a bundle list from [okf_read()]. Non-directory sources are fetched via [okf_fetch()] and cleaned up afterwards.
db_path	DuckDB path; defaults to in-memory ‘":memory:"‘.
ingested_at	Optional ISO-8601 timestamp; defaults to the current time.
bundle_id	Optional stable bundle id.
source_kind	How the bundle was obtained (e.g. ‘"dir"‘); auto-set for fetched sources.
subdir	Optional bundle path within a cloned/extracted source.
branch	Optional git branch or tag (git sources only).

Value

A list with the open ‘con‘, the ‘bundle_id‘, and a ‘summary‘ (counts, conformance, link totals). The caller owns/closes ‘con‘.

---

okf_links *Build the concept graph (resolved and broken links) for a bundle.*

Description

Build the concept graph (resolved and broken links) for a bundle.

Usage

```
okf_links(rd)
```

Arguments

rd	A bundle as returned by [okf_read()].

Value

A data.frame with ‘src_path‘, ‘dst_raw‘, ‘dst_path‘, ‘resolved‘.

okf_ollama_embedder *Build an embedder backed by a local Ollama embeddings model.*

Description

An embedder is a function of ‘texts‘ returning a list of numeric vectors. Swap in any such function (e.g. an OpenAI client) for [okf_embed()] / [okf_rag()].

Usage

```
okf_ollama_embedder(
  model = "nomic-embed-text",
  url = Sys.getenv("OLLAMA_URL", "http://localhost:11434")
)
```

Arguments

model	Ollama embedding model name.
url	Ollama base URL (defaults to the ‘OLLAMA_URL‘ env var or localhost).

Value

A function ‘texts -> list(numeric)‘. Requires the httr2 package.

okf_parse_file *Parse the YAML frontmatter and body of a single OKF concept file.*

Description

Parse the YAML frontmatter and body of a single OKF concept file.

Usage

```
okf_parse_file(path)
```

Arguments

path	Path to a markdown file.

Value

A list with ‘meta‘ (parsed frontmatter, or ‘NULL‘), ‘body‘, and ‘err‘ (‘NA‘ on success, else ‘"no_frontmatter"‘, ‘"unclosed_frontmatter"‘, or ‘"yaml_parse_error"‘).

okf_query *Query helpers over an ingested OKF catalog.*

Description

Query helpers over an ingested OKF catalog.

Usage

```
okf_concepts(con)

okf_graph_df(con)

okf_findings(con)

okf_search(con, term)
```

Arguments

con	An open DuckDB connection to an okf catalog.
term	Search term for [okf_search()] (matched against concept bodies).

Value

A data.frame: concepts ([okf_concepts]), link edges ([okf_graph_df]), validation findings ([okf_findings]), or body matches ([okf_search]).

okf_rag *Semantic search over an embedded catalog.*

Description

Embeds ‘query‘ and returns the top-k most cosine-similar chunks (via DuckDB’s native ‘list_cosine_similarity‘). Run [okf_embed()] first.

Usage

```
okf_rag(con, query, embedder = NULL, k = 5L)
```

Arguments

con	An open DuckDB connection to an embedded okf catalog.
query	Query string.
embedder	An embedder function; defaults to [okf_ollama_embedder()].
k	Number of results to return.

Value

A data.frame with ‘path‘, ‘title‘, ‘chunk_id‘, ‘score‘, ‘text‘.

`okf_read` *Read an OKF bundle from a directory into an in-memory representation.*

Description

Read an OKF bundle from a directory into an in-memory representation.

Usage

```
okf_read(root, bundle_id = NULL, source_kind = "dir")
```

Arguments

`root`	Path to the bundle directory.
`bundle_id`	Optional stable id; defaults to a hash of the root path.
`source_kind`	How the bundle was obtained (e.g. ‘"dir"‘).

Value

A list with ‘bundle_id‘, ‘root‘, ‘okf_version‘, ‘source_kind‘, ‘concepts‘ (parsed per-file records), and ‘known‘ (all concept paths).

`okf_resolve_link` *Resolve a markdown link target to a bundle-relative concept path.*

Description

Resolve a markdown link target to a bundle-relative concept path.

Usage

```
okf_resolve_link(raw, src_rel, known)
```

Arguments

`raw`	Raw link target.
`src_rel`	Bundle-relative path of the linking concept.
`known`	Character vector of all known concept paths in the bundle.

Value

The resolved bundle-relative path, or ‘NA‘ if it does not resolve.

`okf_validate` *Validate a bundle against the OKF v0.1 conformance rules (permissively).*

Description

Hard rules (severity 'error'): parseable frontmatter, non-empty 'type'. Soft findings (severity 'warn'): missing recommended fields, non-ISO timestamps, broken links. Never rejects the bundle – returns findings.

Usage

```
okf_validate(rd)
```

Arguments

`rd`	A bundle as returned by [okf_read()].

Value

A data.frame with 'path', 'severity', 'rule', 'message'.

Index