DEFENSIVE PUBLICATION — TECHNICAL DISCLOSURE (TDCOMMONS DEPOSIT COPY) · 2026-06-25

Keywords: defensive-publication, prior-art, ai-agents, rag, llm-reranking, fail-open, retrieval, graceful-degradation

Hybrid RAG Ranking with Graceful LLM Fallback

A two-stage retrieval ranker whose deterministic stage owns the critical path and whose optional LLM re-ranking stage fails open without surfacing failure

Document type	Technical Defensive Publication (public prior art)
Publisher / Copyright holder	Gus IT LLC (Florida, USA)
Author	Gustavo Assuncao, PhD
Publication date	2026-06-25
Version	1.0
Classification	Public
License	AGPL-3.0-or-later (copyleft; commercial license available)
Deposit channel	To be assigned (IP.com / Zenodo / arXiv) — establishes a public, dated, citable prior-art record
Status	Published — prior art established 2026-06-25

Abstract

Retrieval-augmented generation (RAG) pipelines increasingly insert a language-model (LLM) re-ranking stage between candidate retrieval and answer synthesis, because LLM re-rankers materially improve the ordering of retrieved passages. But LLM re-rankers are slow, costly, rate-limited, and intermittently unavailable. The naive integration — retrieve, then *await* the LLM re-ranker, then return — promotes the slow, failure-prone stage onto the **critical path**. End-to-end retrieval latency, availability, and cost all become hostage to the LLM provider: one provider timeout or one tripped spend cap converts a sub-100 ms retrieval into a multi-second failure or an outage.

This publication describes a hybrid ranking architecture that inverts the dependency. A deterministic lexical stage (TF-IDF-style scoring with title and recency boosts) is the **critical path** and *always* returns a fully-ordered result within a tight latency budget. The LLM re-ranking stage is structurally a **non-critical-path enhancement**: it is invoked only on a bounded top-K subset of the already-ordered deterministic output; it is governed by a per-call time and budget bound; and — the central novelty — on **any** failure (bridge unavailability, timeout, malformed re-ranker output, or budget breach) the system **fails open**, returning the deterministic ranking unchanged and *without surfacing the failure to the caller*. The caller's contract is a valid, ordered result on every invocation. The LLM can only ever *improve* the ordering of the top-K window; it can never degrade availability, blow the latency budget, or propagate an error.

This document provides the system architecture, the fail-open state machine, the top-K and budget cost-bounding mechanics, a data model, a clean-room reference implementation that reduces the invention to

practice, a worked end-to-end example, a security and failure-mode analysis, framework mappings, an evaluation methodology, and an enumerated set of inventive claims (one independent and fourteen dependent). It is published defensively to keep the technique freely practiceable and to bar later patenting by others.

---

1. Executive Summary

1.1 Thesis

A retrieval ranker should never become *less available* or *less responsive* than its deterministic core merely because an optional LLM-quality enhancement was added. The correct architecture treats the LLM re-ranker as a **best-effort overlay** on a deterministic base — bounded in cost (top-K + budget), bounded in time (deadline), and **fail-open** (any error returns the base result silently). The base ranking is the contract; the LLM is an opportunistic improvement.

1.2 Contributions

#	Contribution	Where
C1	**Inverted criticality**: the deterministic lexical stage is the critical path; the LLM re-ranker is structurally non-critical	§6, §7
C2	**Fail-open guarantee**: any re-ranker failure (timeout, error, malformed output, budget breach) returns the deterministic ordering *without surfacing the failure*	§7.4, §11, §14
C3	**Cost bounding by construction**: the re-ranker sees at most a top-K subset, and a budget gate vetoes the call before it is made when spend would be exceeded	§7.3, §8
C4	**Time bounding**: the re-ranker call carries an explicit deadline; a deadline breach is a fail-open event, not an error	§7.4
C5	**Order-preserving merge**: re-ranker output re-orders only the top-K window and is validated (index-set equality) before it is allowed to replace the base ordering	§7.5
C6	**Clean-room reference implementation** demonstrating the full pattern with a pluggable, mockable re-ranker adapter	§9, Appendix C

1.3 Headline claim

> A retrieval system comprising a deterministic ranking stage and an optional re-ranking stage that invokes a language model on at most a top-K subset of the deterministic stage's output, characterized in that on **any** failure of the re-ranking stage the deterministic stage's output is returned **without that failure being surfaced to the caller**.

1.4 Scope

What this is. An architecture and method for composing a deterministic ranker with an optional LLM re-ranker such that the deterministic ranker bounds latency, availability, and cost, and the LLM re-ranker is a fail-open enhancement.

What this is NOT. It is not a new lexical scoring algorithm (TF-IDF, BM25, and recency boosting are well-known and used as the base). It is not a new LLM re-ranking *prompt* technique (listwise prompting à la RankGPT is prior art and is used as the overlay). It is not a vector index, an embedding model, or a fusion algorithm. The novelty is the **composition discipline**: criticality inversion + top-K/budget bounding + the silent fail-open contract.

---

2. Introduction & Motivation

2.1 The concrete problem

Modern RAG and agent systems retrieve candidate passages for a query and then re-order them before handing the top results to a generator. Two families of ranker exist:

- **Deterministic lexical rankers** (TF-IDF, BM25, with optional title/recency boosts). Fast (single-digit to low-tens of milliseconds), cheap (CPU only), perfectly available, but quality plateaus on semantically subtle queries.
- **LLM re-rankers** (a model scores or orders candidates by relevance). Higher ordering quality on hard queries, but each call costs tokens, adds hundreds of milliseconds to multiple seconds of latency, is subject to provider rate limits and outages, and can return malformed output.

2.2 The "tax" of the naive composition

The naive pipeline is `retrieve → await LLM rerank → return`. This places the LLM on the critical path. The taxes:

Tax	Mechanism	Symptom
Latency tax	Every query waits for the LLM round-trip	p99 retrieval latency tracks the LLM, not the index
Availability tax	An LLM outage = a retrieval outage	RAG goes down when the provider does
Cost tax	Every query pays for an LLM call	Spend scales linearly with traffic, even on easy queries
Blast-radius tax	A budget cap trip raises an error mid-request	A spend guardrail turns into a user-facing 500
Tail tax	A single slow re-rank inflates the tail	One stuck call holds a connection and a worker

A system that adds the LLM re-ranker "to improve quality" can, paradoxically, make the whole retrieval path *worse* on the dimensions users actually feel (availability and latency), because it converted an optional enhancement into a mandatory dependency.

2.3 Why existing approaches fall short

Two-stage retrieve-then-rerank is well-known (Cohere Rerank, ColBERT, RankGPT, Reciprocal Rank Fusion). But the published designs treat the re-ranker as a *mandatory* stage whose output the pipeline *depends on*. Robustness, where addressed, is treated as an operational concern (retries, circuit breakers around the provider call) rather than as an architectural *contract* baked into the ranker: that the ranker **always** returns a valid ordering, and the LLM can only ever improve it. The fail-open-by-construction contract — re-ranker failure is a *non-event* for the caller — is the gap this disclosure fills.

2.4 Why now

LLM re-rankers became good enough to deploy in production RAG in 2023–2025, which made the latency/availability/cost taxes acute and widespread. Simultaneously, spend governance (per-call budgets, org spend caps) became standard, which introduced a *new* failure mode — the budget breach — that the naive composition surfaces as an error. The pattern here is the disciplined answer.

---

3. Problem Statement

3.1 Formal framing

Let a query `q` and a candidate set `D = {d_1, ..., d_n}` be given. A ranker is a function `R(q, D) → ordered sequence over D`. We require a ranker `R*` composed of:

- a deterministic base ranker `B(q, D)` with bounded latency `L_B` and perfect availability, producing a total order `O_B`; and
- an optional LLM re-ranker `M(q, D_K)` operating on `D_K = top-K(O_B)`, producing an order `O_M` over `D_K`.

`R*` must satisfy:

1. **Totality** — `R*(q, D)` returns a total order over `D` for every input.
2. **Availability independence** — `R*` returns even if `M` is unavailable.
3. **Latency bound** — when `M` is skipped or fails, `R*` latency ≈ `L_B`.
4. **Cost bound** — `M` is invoked on at most `K` items, and only when within budget.
5. **Silent fallback** — failure of `M` does not raise to the caller of `R*`.
6. **Safe merge** — `O_M` replaces the top-K of `O_B` only if `O_M` is a valid permutation of `D_K`; otherwise `O_B` stands.
7. **Order preservation** — items beyond the top-K retain their `O_B` order.

3.2 Derived requirements

Req	Requirement	Satisfied in
R1	Deterministic base ranker always returns a total order in bounded time	§6, §7.1
R2	Base ranking is the critical path; re-ranker is non-critical	§6.2, §7
R3	Re-ranker input is bounded to a configurable top-K	§7.3
R4	Re-ranker invocation is gated by a budget check made *before* the call	§7.3, §8
R5	Re-ranker call carries an explicit deadline	§7.4
R6	Any re-ranker failure (error/timeout/budget/malformed) returns base result	§7.4, §11
R7	Re-ranker failure is NOT surfaced to the caller	§7.4, §14
R8	Re-ranker output is validated as a permutation of the top-K before use	§7.5
R9	Items beyond top-K keep their base order	§7.5
R10	The enhancement is a pure overlay: disabling it changes only quality, not the contract	§6.2, §7.1
R11	Failure events are observable (metrics/telemetry) without being thrown	§7.6, §11
R12	Configuration (enable, K, budget, deadline, model) is externalized	§7.7, §9.3

4. Related Work & Prior Art

This work **builds on** established retrieval and robustness techniques rather than inventing from scratch. The relevant prior art:

- **TF-IDF / BM25 lexical ranking** (Spärck Jones 1972; Robertson & Walker 1994). Deterministic, fast, well-understood term-weighting. **Adopted** as the base ranker `B`.
- **Cohere Rerank API** (Cohere, 2023+). A managed cross-encoder re-ranking service that re-orders candidate passages for a query. Representative of the *mandatory LLM/cross-encoder re-rank* stage. **Used conceptually** as the overlay `M`; the present work differs in the *composition contract*.
- **RankGPT** (Sun et al., *Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents*, EMNLP 2023). Listwise re-ranking by prompting an LLM to order candidate indices. **Adopted** as the re-ranker prompting style (`[best, ..., worst]` index ordering).
- **ColBERT / ColBERTv2** (Khattab & Zaharia 2020; Santhanam et al. 2022). Late-interaction neural re-ranking. Representative of the neural re-rank family; cited as prior art for two-stage retrieve-then-rerank.
- **Reciprocal Rank Fusion (RRF)** (Cormack et al., 2009). Combines multiple ranked lists by reciprocal-rank weighting. Cited because it is a well-known *combination* of rankers; it is *not* fail-open and treats all input lists as always-present.
- **Maximal Marginal Relevance (MMR)** (Carbonell & Goldstein, 1998). Diversity re-ranking. Optionally composable with the base ranker; cited for completeness.
- **Circuit-breaker / bulkhead patterns** (Nygard, *Release It!*, 2007). Operational resilience around remote calls. Related to the fail-open posture but framed as ops middleware, not as a ranker *contract*.
- **Graceful degradation / fail-open security & availability postures.** General systems principle. Applied here specifically to a two-stage ranker.

The combination — deterministic base as critical path + bounded top-K + budget gate + *silent* fail-open with permutation validation — is the contribution.

5. Prior-Art Delta

Per novel feature: what the closest prior source has, what it lacks, what this adds.

Prior source	What it has	What it LACKS	What THIS adds
Cohere Rerank API	High-quality managed cross-encoder re-ranking of candidates	The re-rank is a *mandatory dependency*; an outage/timeout/cap fails the request	LLM re-rank as a *non-critical overlay*; outage/timeout/cap returns the deterministic base **silently**
RankGPT (Sun 2023)	Listwise LLM re-ranking via index-order prompting	No fail-open contract; malformed/failed LLM output is an error; no budget/top-K *gate* tied to fallback	Same prompting, but wrapped in a permutation-validated, budget-gated, deadline-bounded **fail-open** merge
ColBERT / ColBERTv2	Neural late-interaction re-ranking; two-stage retrieve-then-rerank	Re-rank stage on the critical path; availability tied to model service	Criticality inversion: deterministic stage owns availability/latency; neural/LLM stage is best-effort
Reciprocal Rank Fusion	Combines multiple ranked lists robustly	Assumes all input lists are present; no notion of a stage *failing open* to another	A specific *fallback* relationship (base ← rerank) with silent degradation, not a symmetric fusion
Circuit breaker (Nygard)	Trips on repeated remote failure; sheds load	Generic ops middleware; not integrated as a ranker contract; tripping is often surfaced	Fail-open is part of the ranker's *functional contract*; the caller never observes the trip
Org/per-call budget caps	Enforce spend ceilings on LLM calls	A breached cap is typically raised as an error mid-request	A budget breach is a *pre-call veto* that routes to the base result as a non-event
Naive retrieve→await-rerank→return	Simple, gets quality boost	Slow path is critical path; couples availability, latency, cost to the LLM	Decouples all three; LLM can only improve, never degrade

6. System Architecture

6.1 Components

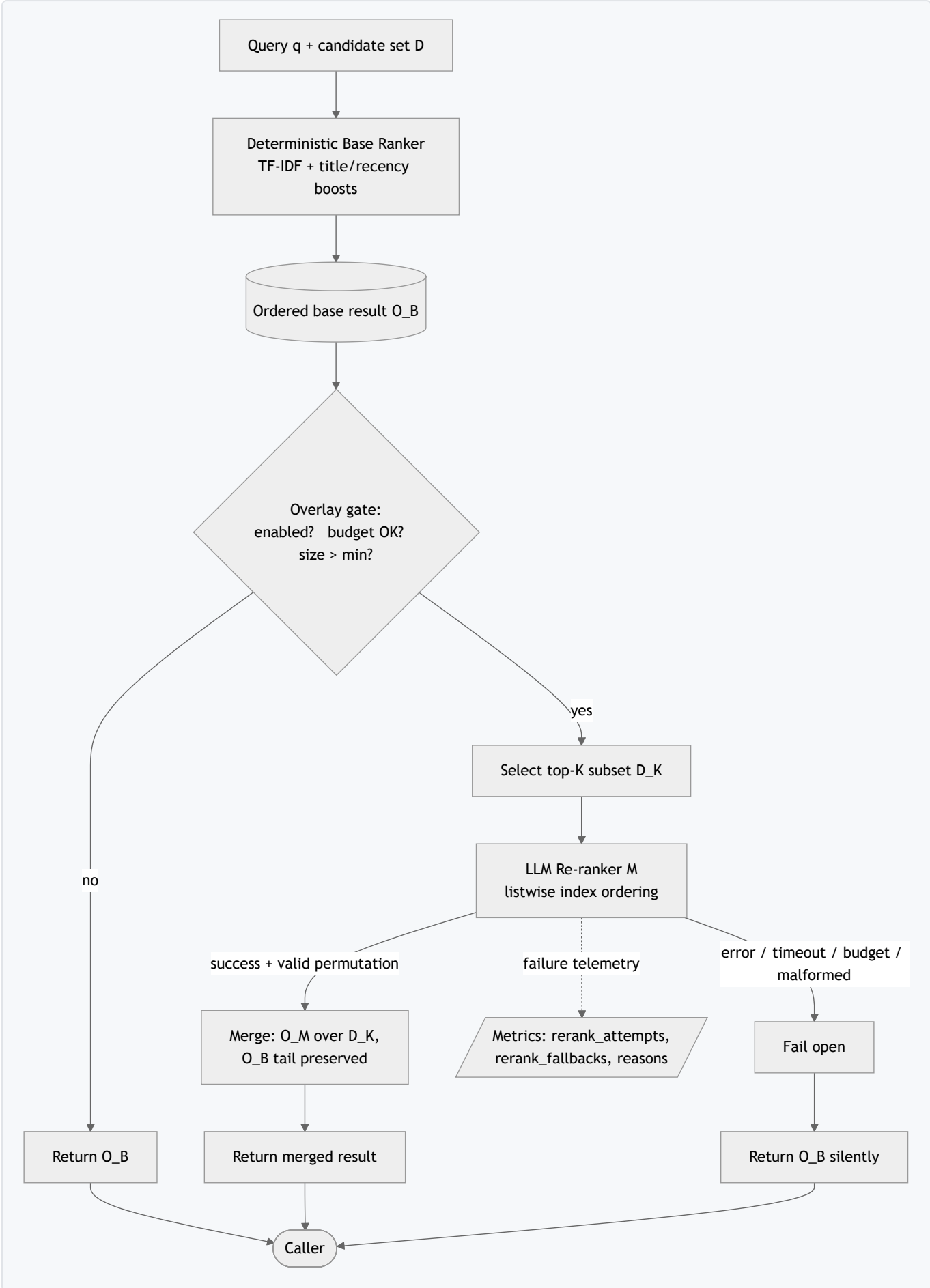
Query q + candidate set D
Deterministic Base Ranker
TF-IDF + title/recency
boosts
Ordered base result O_B
Overlay gate:
enabled? budget OK?
size > min?
no
yes
Select top-K subset D_K
LLM Re-ranker M
listwise index ordering
success + valid permutation
failure telemetry
error / timeout / budget /
malformed
Merge: O_M over D_K,
O_B tail preserved
Metrics: rerank_attempts,
rerank_fallbacks, reasons
Fail open
Return O_B
Return merged result
Return O_B silently
Caller

Figure 1 — System architecture. The base ranker is the only path that the caller strictly depends on; everything to the right of the gate is best-effort.

Component	Role	Criticality
Base ranker `B`	Deterministic TF-IDF-style scoring + title/recency boosts; produces total order `O_B`	**Critical**
Overlay gate	Decides whether to attempt the LLM re-rank (enabled flag, budget check, minimum-size check)	Non-critical
Top-K selector	Bounds the re-ranker input to `D_K = top-K(O_B)`	Non-critical
Re-ranker `M`	LLM listwise ordering of `D_K` via a pluggable adapter	Non-critical
Merge/validate	Validates `O_M` is a permutation of `D_K`; merges over `O_B`	Non-critical
Metrics sink	Records attempts, fallbacks, and reasons without throwing	Observability

6.2 Cross-cutting properties

- **Purity of the overlay.** Disabling the overlay (config flag off) yields exactly `O_B`. The overlay never changes the *contract*, only the *quality* of the top-K.
- **No shared failure domain.** The base ranker computes in-process on CPU with no network dependency. The overlay's failure domain (network, provider, budget) is fully contained behind the gate and the fail-open handler.
- **Determinism of the base.** Given `(q, D)` and a fixed clock for recency, `O_B` is reproducible — essential for testing the fallback path deterministically.
- **Idempotent fallback.** Falling open is a no-op transform on `O_B`; repeated fallback yields identical output.

---

7. Detailed Mechanics

7.1 Base ranking (critical path)

For each document `d`, the base score is:

```
score(q, d) = Σ_{t ∈ tokens(q)} tf_d(t)          # term-frequency overlap
            + 0.5 · [∃ t ∈ tokens(q): t ∈ title(d)]   # title boost
            + recency_boost(d)                    # 0.3 if <7d, 0.1 if <30d, else 0
score is clamped to [0, 1]
```

`tokens(·)` lowercases, strips punctuation, splits on whitespace, drops empties. `tf_d(t)` is the within-document term frequency. The base ranker sorts documents by descending score; ties break by original input index (stable). This stage is pure, CPU-only, and bounded by `O(|D| · avg_doc_len)`.

> The specific lexical formula is illustrative and replaceable (BM25 works identically here). The novelty does not reside in the formula; it resides in the formula's *role* as the always-available critical path.

7.2 The overlay gate

Before any LLM call, three conditions are checked in order; failing any one routes to the base result:

1. **Enabled** — the overlay feature flag is on (`RERANK_ENABLED`).
2. **Size** — `|D| > MIN_DOCS_FOR_RERANK` (re-ranking 1–3 docs is not worth a call).
3. **Budget** — the projected cost of a top-K re-rank call is within the remaining per-call/period budget. This check is made *before* the call, so a budget breach is a **pre-call veto**, never a mid-flight error.

7.3 Cost bounding (top-K + budget)

`D_K = O_B[0 : K]`, where `K = RERANK_TOP_K` (default 10). The re-ranker prompt includes only `D_K`, with each document's title and a truncated content snippet. Because `K` is fixed and snippets are length-capped, the per-call token cost has a hard ceiling independent of `|D|`. The budget gate (§7.2) multiplies the estimated tokens by a price factor and compares against the remaining budget; if over, the overlay is skipped.

7.4 Time bounding & the fail-open handler

The re-ranker adapter call is wrapped with:

- an explicit **deadline** (`RERANK_DEADLINE_MS`), enforced by racing the call against a timer; and
- a **try/fallback** envelope catching *every* throwable.

Any of the following is treated identically — as a **fail-open event** that returns `O_B` and records a reason, never re-throwing:

Failure	Source
Bridge/provider unavailable	network error, 5xx, connection refused
Timeout / deadline exceeded	deadline race wins
Rate limited	429 from provider
Budget breach (pre-call)	gate veto in §7.2
Malformed output	re-ranker text has no parseable index array
Invalid permutation	parsed indices are not a permutation of `0..K-1` (§7.5)
Empty/null response	adapter returned nothing

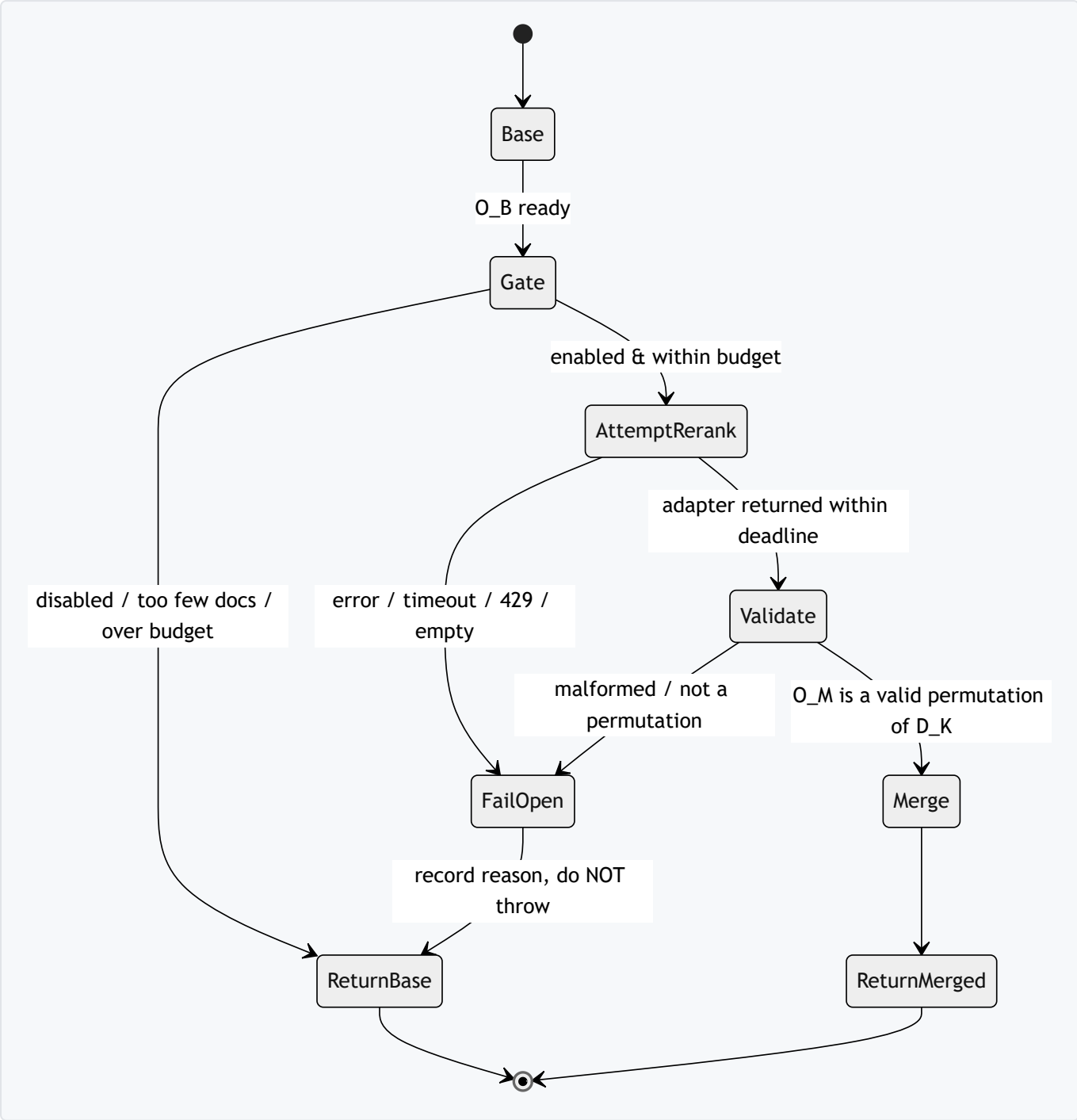

Figure 2 — Fail-open state machine. Every edge that is not a clean success funnels to `ReturnBase`, and `ReturnBase` never throws.

7.5 Order-preserving, validated merge

When the adapter returns, its output is parsed into an index order π over `0..K-1`. The merge is permitted **only if** π is a valid permutation of the top-K index set (same cardinality, no duplicates, no out-of-range indices). If valid:

```
merged = [ D_K[π[0]], D_K[π[1]], ..., D_K[π[K-1]] ]  ++  O_B[K:]
```

i.e., the top-K is re-ordered per the LLM, and the tail (`O_B[K:]`) is appended unchanged, preserving R9. If π is not a valid permutation, the system fails open (R8). This validation defends against an LLM that drops,

duplicates, or invents indices — a common failure mode of listwise prompting.

7.6 Observability without throwing

Each invocation increments counters: `rerank_attempts`, `rerank_success`, `rerank_fallbacks{reason}`. The fallback reason is one of the failure rows in §7.4. These are emitted to a metrics sink / telemetry channel. Crucially, telemetry is the *only* externally-visible signal of a fallback; the caller's return value is always a valid ordering.

7.7 Configuration points

Key	Default	Meaning
`RERANK_ENABLED`	`false`	Master switch for the overlay
`RERANK_TOP_K`	`10`	Max documents sent to the re-ranker
`MIN_DOCS_FOR_RERANK`	`3`	Skip overlay below this candidate count
`RERANK_DEADLINE_MS`	`1500`	Per-call deadline; breach → fail open
`RERANK_BUDGET_TOKENS`	`4000`	Pre-call token budget ceiling
`RERANK_MODEL`	(config)	Model id resolved from config, never hard-coded
`RECENCY_BOOST_7D / _30D`	`0.3 / 0.1`	Base recency boosts

All values are externalized (env/config); none are hard-coded in the algorithm.

---

8. Data Model

The mechanism is largely stateless, but three logical structures are involved: the candidate document, the scored/ordered result, and the per-invocation telemetry record.

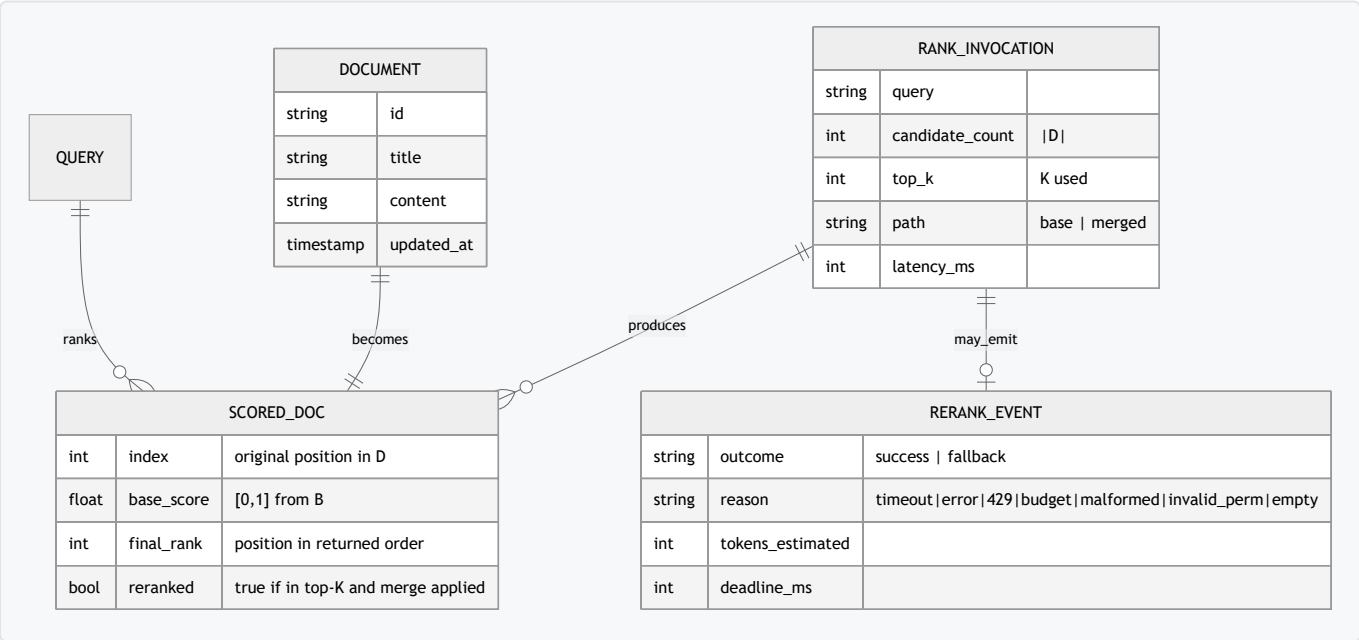

Figure 3 — Logical data model. `RERANK_EVENT` is telemetry-only; it never affects the returned `SCORED_DOC` list beyond setting `reranked` flags.

Field notes:

- `SCORED_DOC.base_score` is always populated (base ranker always runs).
- `SCORED_DOC.reranked` is `true` only for documents whose top-K position changed via a successful, validated merge.
- `RANK_INVOCATION.path` is `merged` only when a re-rank succeeded and validated; otherwise `base`.
- `RERANK_EVENT` is optional per invocation and carries the fallback reason for observability; a clean `base`-only path (overlay disabled) may emit none.

---

9. Reference Implementation & Enablement

9.1 What `src/` demonstrates

`src/hybrid-ranker.js` is an original, clean-room, dependency-light Node.js module that implements the full pattern:

- `tokenize`, `termFrequency`, `baseScore`, `baseRank` — the deterministic critical path (R1).
- `rank(query, documents, options, adapter)` — the composed ranker that runs the base, applies the overlay gate (R3, R4), enforces a deadline (R5), validates the permutation (R8), merges order-preservingly (R9), and **fails open silently** (R6, R7) while recording telemetry (R11).
- A pluggable `adapter` interface so any re-ranker (real LLM bridge or a mock) can be injected. The included `mockAdapters` demonstrate success, timeout, error, malformed-output, and invalid-permutation behaviors.

`src/selfcheck.js` exercises every fallback branch and asserts the invariants (totality, silent fallback, order preservation, budget veto).

9.2 How it reduces the invention to practice

Running `node src/selfcheck.js` shows that for *every* adapter failure mode the `rank` function returns the same ordering it would return with the overlay disabled, never throws, and records the correct fallback reason — i.e., the fail-open contract (Claim 1) holds in executable form. The success adapter shows a validated permutation merge that re-orders only the top-K. This is a complete, runnable enablement of the claims.

9.3 Configuration points

The reference implementation reads all knobs from an `options` object with the defaults of §7.7, demonstrating R12 (externalized configuration). No model id, budget, K, or deadline is hard-coded in the algorithm body.

> The reference implementation is **illustrative and clean-room**. It is not the publisher's production code and contains no proprietary identifiers, endpoints, or credentials.

---

10. Worked Example / Scenario

Setup. Query `q = "renewable energy storage"`. Candidate set `D` of 12 passages retrieved from an index. `RERANK_ENABLED=true`, `K=5`, `RERANK_DEADLINE_MS=1500`, budget ample.

Base stage. TF-IDF + title/recency yields `O_B`. The lexically-strongest five (`D_K`) are, in base order: `[d3, d7, d1, d9, d5]`. One passage, `d9`, scores high lexically because it repeats "energy storage" but is actually about *thermal* storage, not the *batteries* the query implies — a semantic miss the base cannot catch.

Overlay gate. Enabled ✓; `|D|=12 > 3` ✓; projected tokens for `K=5` within budget ✓ → attempt re-rank.

Re-ranker. The LLM, given the five snippets, returns the index order `[2, 0, 1, 4, 3]` (meaning: `d1` best, then `d3`, `d7`, `d5`, `d9` last). This is a valid permutation of `0..4`.

Merge. `merged_topK = [d1, d3, d7, d5, d9]`; the tail `O_B[5:]` is appended unchanged. `d9` is correctly demoted within the window; nothing outside the window moved.

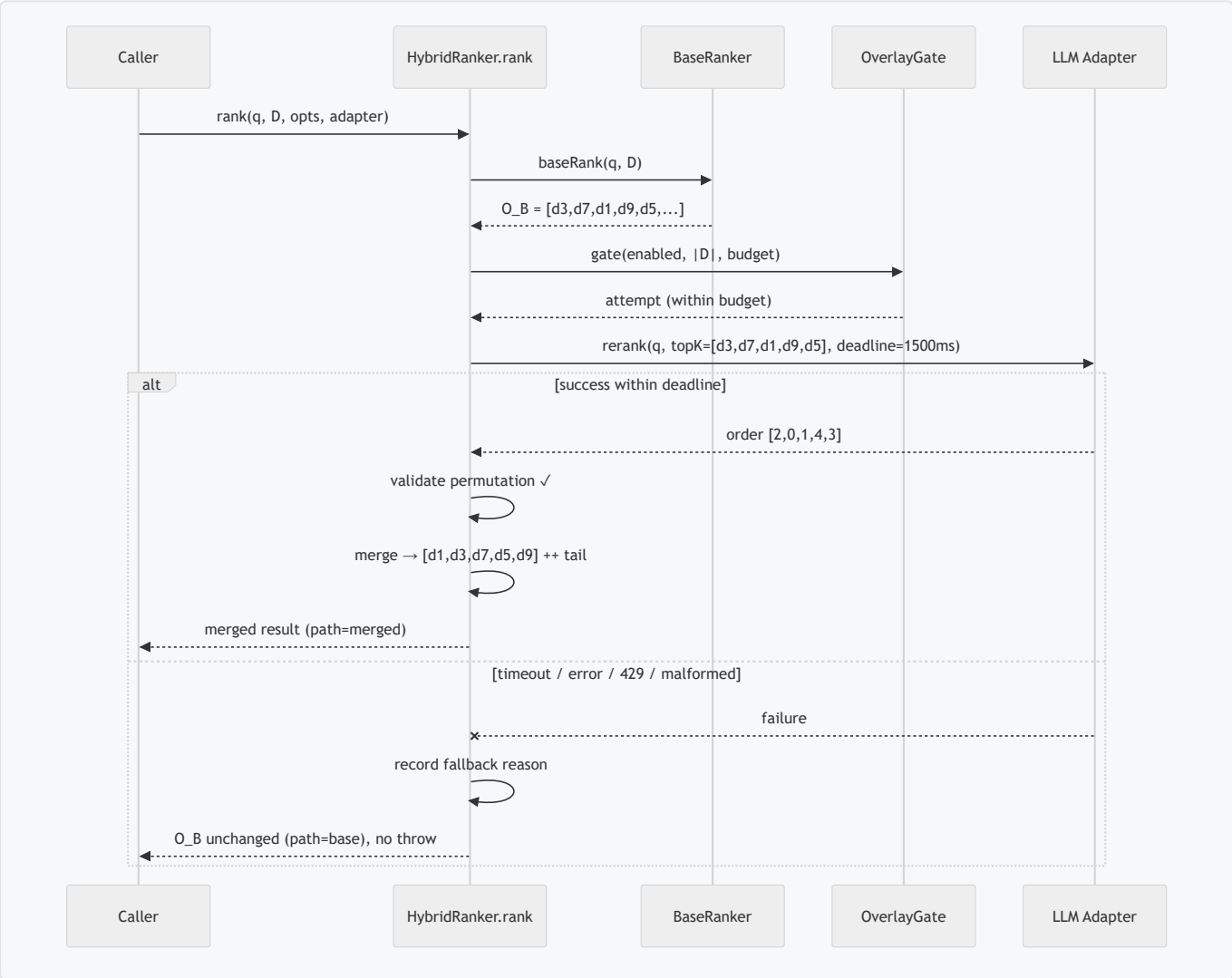

Figure 4 — Sequence for the worked example, showing both the success merge and the fail-open branch returning `O_B` silently.

Counterfactual (fail-open). Had the LLM provider timed out at 1.5 s, the deadline race wins, a `timeout` fallback is recorded, and the caller receives `O_B` exactly — `[d3, d7, d1, d9, d5, ...]` — with no exception

and within the base latency budget. The user still gets a perfectly serviceable ranking; only quality (the `d9` demotion) is lost, not availability.

---

11. Security, Safety & Failure Modes

11.1 Fail-open vs fail-closed posture

This mechanism is deliberately **fail-open on availability**: when the optional enhancement fails, the system continues with the deterministic base. This is the correct posture *for a quality-enhancing overlay* — losing the enhancement must not lose the service. It is explicitly **not** a security gate; no authorization decision is delegated to the re-ranker, so failing open carries no security downgrade. (For security-critical decisions a fail-*closed* posture would be required; this ranker is not such a component.)

11.2 Failure-mode table

Failure mode	Effect if unhandled	Mitigation here	Residual risk
Provider timeout	Request hangs / slow	Deadline race → fail open	Lost quality on that query
Provider 5xx / outage	Request errors	Catch-all fail open	Lost quality
Rate limit (429)	Error or retry storm	Treated as fail open; no inline retry on critical path	Lost quality
Budget breach	Mid-request error	Pre-call veto → base path	Lost quality (by design)
Malformed LLM output	Crash on parse / bad order	Parse-guard → fail open	Lost quality
Invalid permutation (dropped/dup/invented index)	Items lost or duplicated in results	Permutation validation → fail open	Lost quality
Prompt-injected passage tries to reorder maliciously	Skewed ordering	Re-rank only re-orders an *existing* top-K permutation; cannot inject new docs or exfiltrate; validated as permutation	Within-window reordering only (low impact)
Excessively long snippet	Token blow-up / cost	Snippet length cap + top-K bound + budget gate	Bounded

11.3 STRIDE-style note

Threat	Relevance	Posture
Tampering (with results)	A malicious passage cannot add/remove docs — only a within-window reorder is possible, and only if it produces a valid permutation	Low
Denial of service (via the LLM)	Deadline + top-K + budget + fail-open prevent the LLM from holding the critical path	Mitigated
Information disclosure	Only top-K snippets (already retrieved for this query) are sent to the re-ranker; no broadening of data exposure	Bounded
Elevation of privilege	None — re-ranker makes no authz decision	N/A

12. Standards & Framework Mapping

These are **semantic alignments**, not certifications. No compliance is claimed.

Framework / standard	Relevant principle	How this aligns
ISO/IEC 25010 (software quality)	Reliability → *fault tolerance*, *recoverability*; Performance efficiency → *time behaviour*	The fail-open overlay is fault tolerance by construction; base latency bound is time-behaviour
ISO/IEC 23894 (AI risk management)	Treat AI component failure as a managed risk	LLM failure is an explicitly designed-for, telemetered, non-fatal event
NIST AI RMF (Manage / Measure)	Manage residual AI risk; measure failure rates	Fallback reasons are measured; residual risk is bounded to "lost quality on that query"
**Reliability engineering (Nygard, Release It!)*	Bulkhead, timeout, fail fast/safe	Deadline = timeout; overlay isolation = bulkhead; fail-open = fail safe
The Twelve-Factor App (Config)	Strict separation of config from code	All knobs (K, budget, deadline, model) externalized
Graceful degradation (web/UX resilience)	Core function survives feature failure	Core ranking survives re-ranker failure

13. Evaluation Methodology

The following is a **methodology**; any specific numbers below are explicitly labeled illustrative.

13.1 Dimensions

Dimension	Metric	How measured
Availability of ranking	% of calls returning a valid order	Should be 100% by construction; verify under injected LLM faults
Base latency	p50/p95/p99 of base-only path	Benchmark with overlay disabled
End-to-end latency	p50/p95/p99 with overlay enabled	Includes successful re-ranks and fallbacks
Fallback rate	`rerank_fallbacks / rerank_attempts` per reason	From telemetry
Quality lift	ΔnDCG@K (merged vs base) on a labeled set	Offline relevance judgments
Cost	mean tokens/query, % queries hitting the LLM	From budget gate + telemetry

13.2 Fault-injection protocol

Run the same query workload with the adapter forced into each failure mode (timeout, error, 429, malformed, invalid permutation, budget breach) and assert: (a) 100% of calls return a valid total order; (b) zero exceptions reach the caller; (c) returned order equals the base order; (d) the correct fallback reason is recorded. The reference `selfcheck.js` automates (a)–(d).

13.3 Interpretation table (illustrative)

Observation (illustrative)	Interpretation
Base p99 = 18 ms; e2e p99 (overlay on) = 1510 ms	Re-rank dominates e2e tail; deadline correctly caps it
Fallback rate spikes to 40% with reason=429	Provider rate-limited; quality lift drops but availability holds at 100%
ΔnDCG@5 = +0.06 on hard queries, +0.00 on easy	Overlay earns its cost mainly on semantically hard queries — supports gating

> All numbers in §13.3 are illustrative placeholders to show how results would be read, not measurements of any deployed system.

---

14. Novelty & Inventive Claims

Presented as prose. One independent claim, fourteen dependent.

Claim 1 (independent). A retrieval system comprising a deterministic ranking stage that produces a total ordering of a candidate document set for a query, and an optional re-ranking stage that invokes a language model on at most a top-K subset of the deterministic stage's output to produce a re-ordering of that subset, characterized in that on any failure of the re-ranking stage the deterministic stage's output is returned to the caller without said failure being surfaced to the caller, such that the deterministic stage is the critical path and the re-ranking stage is a non-critical enhancement.

Claim 2. The system of claim 1, wherein the deterministic ranking stage computes term-frequency overlap between the query and each document and is executed without any network dependency, so as to guarantee a bounded-latency, always- available total ordering.

Claim 3. The system of claim 1, wherein invocation of the re-ranking stage is gated by a budget check performed *before* the language-model call, such that a projected cost exceeding a configured budget vetoes the call and routes to the deterministic output as a non-error event.

Claim 4. The system of claim 1, wherein the re-ranking stage operates on a configurable top-K subset of the deterministic ordering, whereby the per-call cost of the language-model invocation is bounded independently of the total candidate count.

Claim 5. The system of claim 1, wherein the re-ranking-stage invocation carries an explicit deadline, and a deadline breach is treated as a failure that returns the deterministic output without raising.

Claim 6. The system of claim 1, wherein "any failure" comprises at least: provider unavailability, a timeout, a rate-limit response, a malformed re-ranker output, an invalid-permutation re-ranker output, and a budget breach, all of which are handled identically by returning the deterministic output.

Claim 7. The system of claim 1, wherein the re-ranking-stage output is parsed into an index ordering and is permitted to replace the corresponding subset of the deterministic ordering only if said index ordering is a valid permutation of the top-K index set; otherwise the deterministic output is returned.

Claim 8. The system of claim 7, wherein documents outside the top-K subset retain their relative order from the deterministic stage in the returned result.

Claim 9. The system of claim 1, wherein the re-ranking stage is skipped when the candidate count is below a configured minimum, returning the deterministic output.

Claim 10. The system of claim 1, further comprising a telemetry sink that records each re-ranking attempt and, for each failure, a categorized fallback reason, wherein said recording does not raise to the caller and is the sole externally-observable signal that a fallback occurred.

Claim 11. The system of claim 1, wherein the language-model re-ranking is expressed as a listwise prompt that requests an ordering of candidate indices, and the re-ranker adapter is pluggable such that the language model may be replaced without altering the fail-open contract.

Claim 12. The system of claim 1, wherein disabling the re-ranking stage via configuration yields exactly the deterministic stage's output, such that the re-ranking stage alters only ordering quality of the top-K and never the availability, latency bound, or return contract of the system.

Claim 13. The system of claim 1, wherein the deterministic ranking stage further applies a title-match boost and a recency boost to the term-frequency score, and the resulting score is clamped to a bounded interval.

Claim 14. The system of claim 1, wherein the failure-handling is idempotent in that returning the deterministic output on failure is a no-op transform on said output, yielding identical results across repeated fallbacks for the same input.

Claim 15. A method corresponding to the system of claim 1, comprising: producing a deterministic total ordering of candidate documents for a query; conditionally selecting a top-K subset; gating a language-model re-ranking invocation on an enable flag, a minimum-size check, and a pre-call budget check; invoking the re-ranking under a deadline; validating the re-ranker output as a permutation; merging a valid re-ranking over the top-K while preserving the tail order; and, upon any failure of said invocation, returning the deterministic ordering without surfacing the failure to the caller.

---

15. Limitations & Threats to Validity

- **The base formula is not novel.** TF-IDF/BM25 + boosts are prior art; the contribution is compositional, not in the lexical scorer.
- **The LLM prompting style is not novel.** Listwise index ordering is RankGPT prior art; it is used as-is as the overlay.
- **Prior art is close on parts.** Two-stage retrieve-then-rerank and circuit breakers each overlap with components here. The defensible delta is the *silent, permutation-validated, budget-gated fail-open contract as part of the ranker's functional definition*, which is why this is published **defensively** rather than filed.
- **Fail-open is a deliberate trade.** On a query where the LLM would have fixed a bad base ordering, a fallback means the user gets the worse ordering. The design accepts lost quality to preserve availability and latency — appropriate for a quality overlay, inappropriate for a correctness/security gate.

- **Quality numbers are workload-dependent.** The lift from the overlay depends on query hardness and corpus; §13 numbers are illustrative.
- **Intentionally withheld.** Production parameter values, internal endpoints, model identifiers, and infrastructure details are intentionally omitted; only the generic, enabling mechanism is disclosed.

---

16. Future Work & Open-Source Reference App

Planned: a small open-source service (`docs/OPEN-SOURCE-APP.md`) packaging the hybrid ranker behind an HTTP API, with a pluggable re-ranker adapter (mock + real LLM bridge), Prometheus-style fallback metrics, and a generic Kubernetes/AKS deployment sketch. Roadmap:

1. v0 — library + self-check (this repo's `src/`).
2. v1 — HTTP service + adapter interface + metrics endpoint.
3. v2 — BM25 base option, hybrid dense/lexical base, configurable budget governor.
4. v3 — A/B harness for quality-lift measurement and a labeled-set evaluation kit.

---

17. Conclusion

Adding an LLM re-ranker to a RAG pipeline should be a strict improvement, never a new way to fail. By making the deterministic lexical ranker the critical path and the LLM re-ranker a top-K-bounded, budget-gated, deadline-bounded, permutation- validated, **silently fail-open** overlay, this architecture guarantees a valid, ordered result on every call while still capturing the LLM's quality lift when it is available and affordable. We publish the complete, enabling description as prior art so the technique stays free for everyone to use.

Appendix A — Prior-Art Landscape (well-trodden vs candidate-novel)

Well-trodden (treated as prior art, adopted):

- TF-IDF / BM25 lexical ranking; title/recency boosting.
- Two-stage retrieve-then-rerank (Cohere Rerank, ColBERT).
- Listwise LLM re-ranking via index prompting (RankGPT).
- Rank fusion (RRF) and diversity re-ranking (MMR).
- Circuit-breaker / bulkhead / timeout resilience patterns.
- Per-call and org-level LLM budget caps.

Candidate-novel (this disclosure's contribution):

- The LLM re-ranker as a *non-critical overlay* on a deterministic critical-path base, with a **silent** fail-open contract (failure is a non-event for the caller).
- A *pre-call* budget veto that routes to the base result as a non-error.
- Permutation validation of the re-ranker output as a precondition for the merge, with tail-order preservation.

Honesty attestation. The prior-art searches behind this document are **directional, not exhaustive**. They reflect a good-faith review of widely-known products and literature as of June 2026. No claim is made that every relevant reference has been found. This publication's purpose is to *establish* prior art, not to adjudicate the novelty of any third party's filing.

Appendix B — Glossary

Term	Definition
Base ranker	The deterministic lexical scoring stage; the critical path.
Overlay / re-ranker	The optional LLM stage that re-orders the top-K.
Critical path	The computation the caller's contract strictly depends on.
Fail open	On failure, continue with the base result instead of erroring.
Top-K	The K highest-base-scored documents sent to the re-ranker.
Budget gate / veto	A pre-call check that skips the LLM when over budget.
Permutation validation	Verifying re-ranker output is a valid reordering of the top-K.
nDCG@K	Normalized discounted cumulative gain at rank K (quality metric).
TF-IDF	Term-frequency × inverse-document-frequency lexical weighting.
RRF	Reciprocal Rank Fusion.

Appendix C — Reference-Implementation Index

File	Purpose
`src/hybrid-ranker.js`	The hybrid ranker: base stage, overlay gate, deadline, validation, fail-open merge, telemetry.
`src/selfcheck.js`	Runnable self-check exercising every fallback branch and asserting invariants.
`src/README.md`	What the sample shows, how to run it, clean-room disclaimer.

Appendix D — Defensive-Publication Deposit & Timestamp

This document and repository constitute a **technical defensive publication**.

- **Publication date:** 2026-06-25.
- **Publisher / copyright holder:** Gus IT LLC (Florida, USA).
- **Author:** Gustavo Assuncao, PhD.
- **Deposit channel:** to be assigned (IP.com / Zenodo / arXiv) to obtain a public, dated, citable identifier. No DOI is asserted here.
- **Intent:** this disclosure is intentionally public to place the described technique in the public domain as prior art, thereby **barring later patenting of the same technique by others** and keeping it freely practiceable.