BetterQuery Intention Bridge: *Architectural Foundations, Operational Guarantees, and Validation Plan*

CDLI Lab
research@cdli.ai

Abstract

BetterQuery began in late 2023 as an intent-to-enhancement experiment. This document formalizes the production architecture that has emerged: intent-aware routing before generation, explicit fallback behavior, quota telemetry, and distribution across API, MCP, and browser surfaces.

The design goal is simple: improve reliability and usefulness without increasing unpredictability. The Intention Bridge makes routing explicit, applies policy controls under uncertainty, and exposes the signals needed for iterative calibration.

1 Problem Definition

Production prompt enhancement pipelines commonly fail in four modes:

1. wrong intent assignment;
2. unsafe or low-value generation;
3. uncontrollable latency;
4. opaque failure behavior.

BetterQuery addresses these with a staged architecture:

- decide intent before generation,
- apply quality gates around output,
- enforce bounded rollout behavior under uncertainty,
- persist structured telemetry for tuning.

2 End-to-End Flow

Request flow in production is:

$$\text{sanitize} \rightarrow \text{route} \rightarrow \text{cache} \rightarrow \text{generate} \rightarrow \text{gate} \rightarrow \text{persist/emit} \rightarrow \text{response}$$

Subsystem	**Role**
Input boundary	Auth (JWT/API key), request validation, rate limits, quota checks
Intention router	Rule checks, semantic retrieval, optional LLM arbitration, fallback
Session layer	Encrypted context and tiered retention windows
Enhancement engine	Architect + refiner prompt stages with circuit breakers
Caching layer	Exact and semantic cache, similarity thresholding
Safety/controls	Output guards, fallback and degraded-path signaling
Observability	Per-call events, skill usage, latency, ratings, admin stats

3 Intention Bridge Formalization

Let request embedding be $q \in \mathbb{R}^d$, skill set $\mathcal{S}$, and e_s the embedding for skill $s \in \mathcal{S}$.

$$\text{sim}(q, s) = \frac{q \cdot e_s}{\|q\| \, \|e_s\|}.$$

Let $C_k(q)$ be the top-k candidates by sim, and $a(s \mid q)$ be arbitration confidence.

$$\pi(q) = \begin{cases} \text{passthrough}, & q \in \mathcal{R}_{\text{rule}} \\ \arg\max_{s \in C_k(q)} \text{sim}(q, s), & \max_{s \in C_k(q)} \text{sim}(q, s) \geq \tau_{\text{sim}} \\ \arg\max_{s \in C_k(q)} a(s \mid q), & \max_{s \in C_k(q)} a(s \mid q) \geq \tau_{\text{llm}} \\ \text{fallback}, & \text{otherwise} \end{cases}$$

This policy prefers cheap deterministic routing when confidence is high and shifts to controlled arbitration only on ambiguous inputs.

4 Quota Window Semantics

Each account has rolling reset horizon of D days (default $D = 30$):

$$\text{Reset}(t) = t + D, \quad \text{Remaining} = \max(0, L - u_t),$$

where L is plan quota and u_t is usage within the current window.

If stored horizon values are malformed, stale, or legacy boundary values, the system normalizes them to a fresh rolling window. This guarantees consistent quota behavior and avoids legacy month-based resets.

5 Caching and Latency

For request i with cache hit H_i, cache latency L_c, generation latency L_g:

$$\mathbb{E}[L_i] = P(H_i = 1)L_c + (1 - P(H_i = 1))L_g.$$

Since $L_c \ll L_g$, even moderate hit rates reduce p95 significantly.

6 Quality Control

Enhancement is generated by a two-step pipeline:

- architecture pass (intent framing and structure),
- refine pass (clarity, actionability, tone),
- output gate (length, identity preservation, fallback detection),
- circuit breaker for repeated failures.

When a gate rejects, behavior remains safe and explicit rather than silently degrading into ambiguous output.

7 Observability

Admin analytics currently include:

- top skills by usage,
- fallback rate,
- average latency,
- daily active enhancers,
- total enhancements and active users.

User pages also persist ratings, forming explicit preference signals for future calibration.

8 Distribution Surfaces

The same intention-aware policy engine serves:

1. REST integration,
2. MCP clients (Claude Code, Cursor, Gemini CLI, etc.),
3. Browser extension overlay in chat and agent UIs.

The design reduces duplicated logic while increasing accessibility.

9 Evaluation Plan

Benchmarking is organized as:

1. **Offline**: intent stratification, date-split leakage controls, stress subsets for ambiguous prompts.
2. **Ablation**: retrieval-only, arbitration-only, no-cache, one-stage baseline, full pipeline.
3. **Online**: staged threshold experiments with fallback, latency, and preference outcomes.

Primary metrics: intent precision/recall, fallback ratio, p50/p95 latency, cache hit ratio, and user preference agreement.

10 Threats and Limits

Current risks are short-input ambiguity, label sparsity, and threshold drift. Operational risk is mitigated through periodic calibration and conservative policy defaults.

11 Conclusion

Intention-first architecture is the central control layer in BetterQuery. It improves behavior under uncertainty, supports safe product scaling, and enables continuous improvement from production telemetry.

References

References

[1] Devlin, J. et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” NAACL, 2019.

[2] Reimers, N. and Gurevych, I. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.” ACL, 2019.

[3] Lewis, P. et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” NeurIPS, 2020.

[4] Wei, J. et al. “Chain-of-Thought Prompting Elicits Reasoning in LLMs.” NeurIPS, 2022.

[5] Yao, S. et al. “ReAct: Synergizing Reasoning and Acting in Language Models.” 2022.