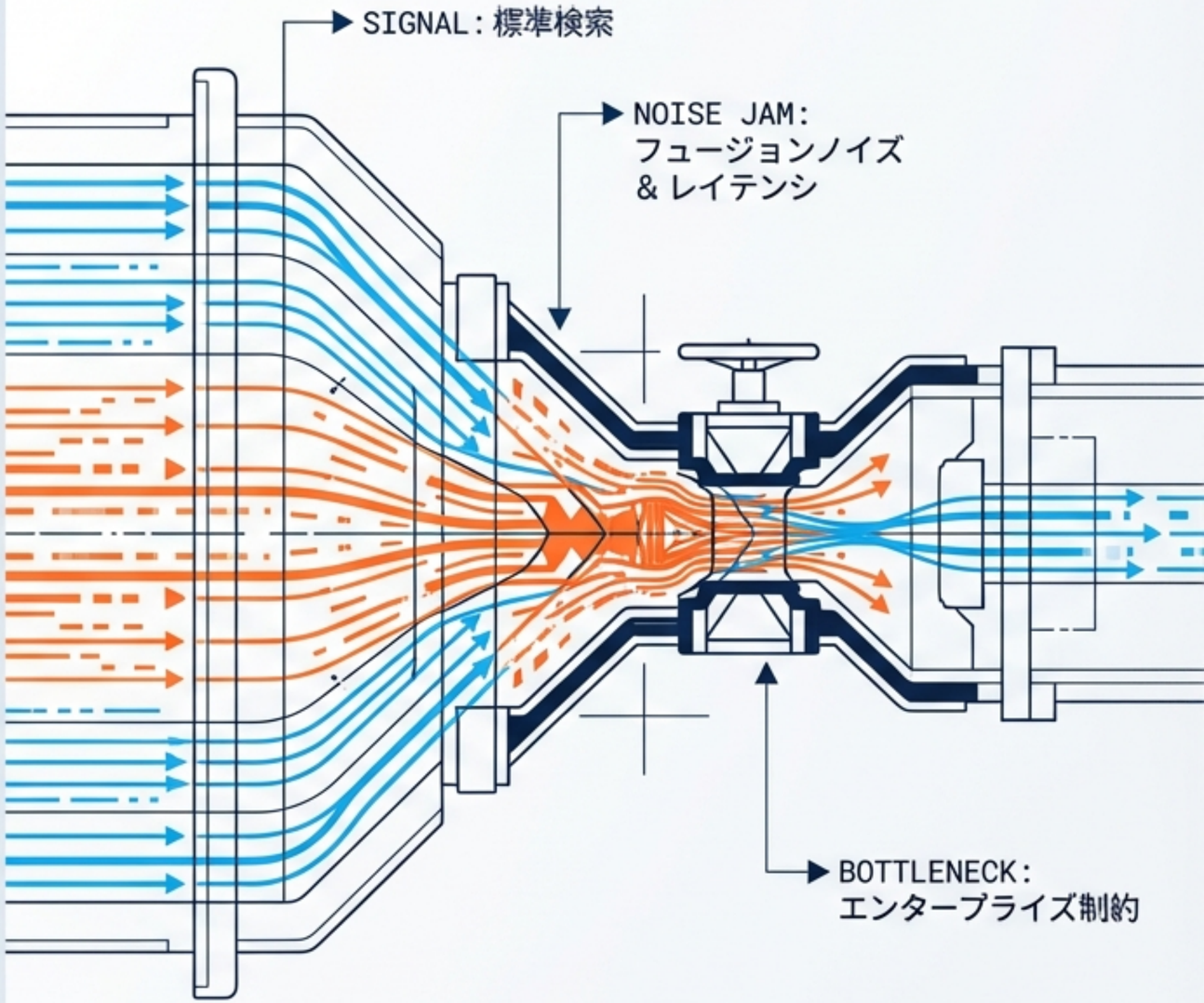
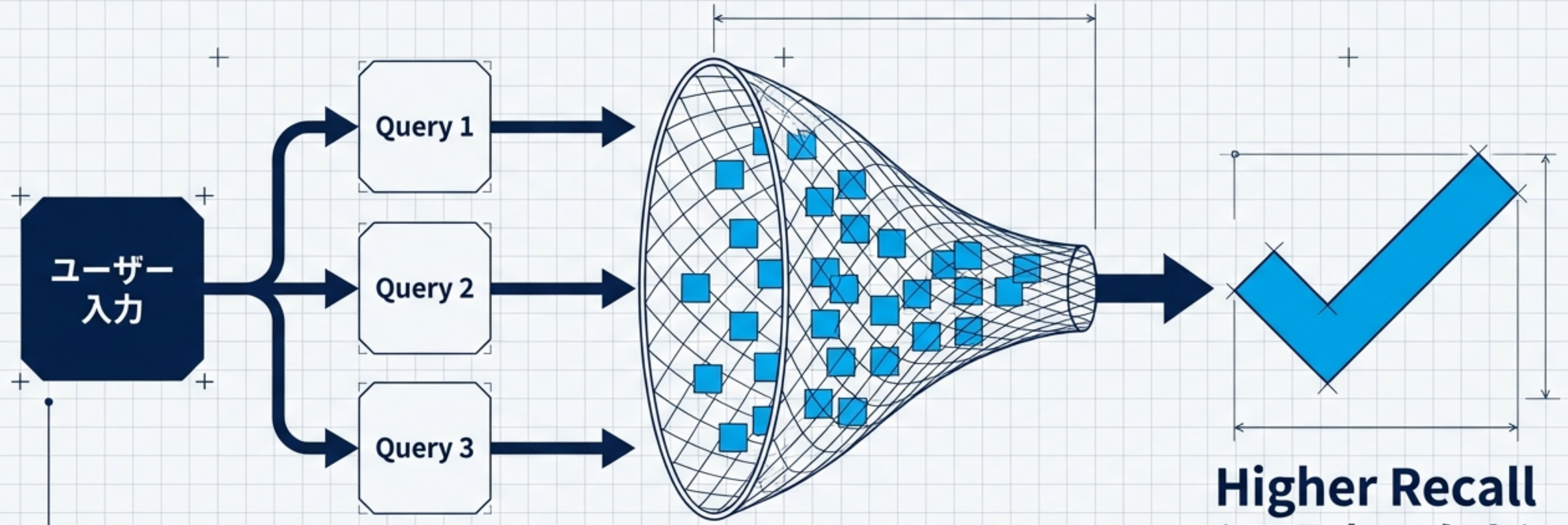
RAGフュージョンの現実：本番環境からの教訓

検索の多様性が直面する「エンタープライズ制約」という物理的な壁



業界の神話：「検索の多様性を広げれば、回答精度は必ず向上する」



理論上のRAGフュージョン

ユーザーの質問をLLMで複数パターンに書き換え（パラフレーズ）、多角的に検索を行う。

期待される効果

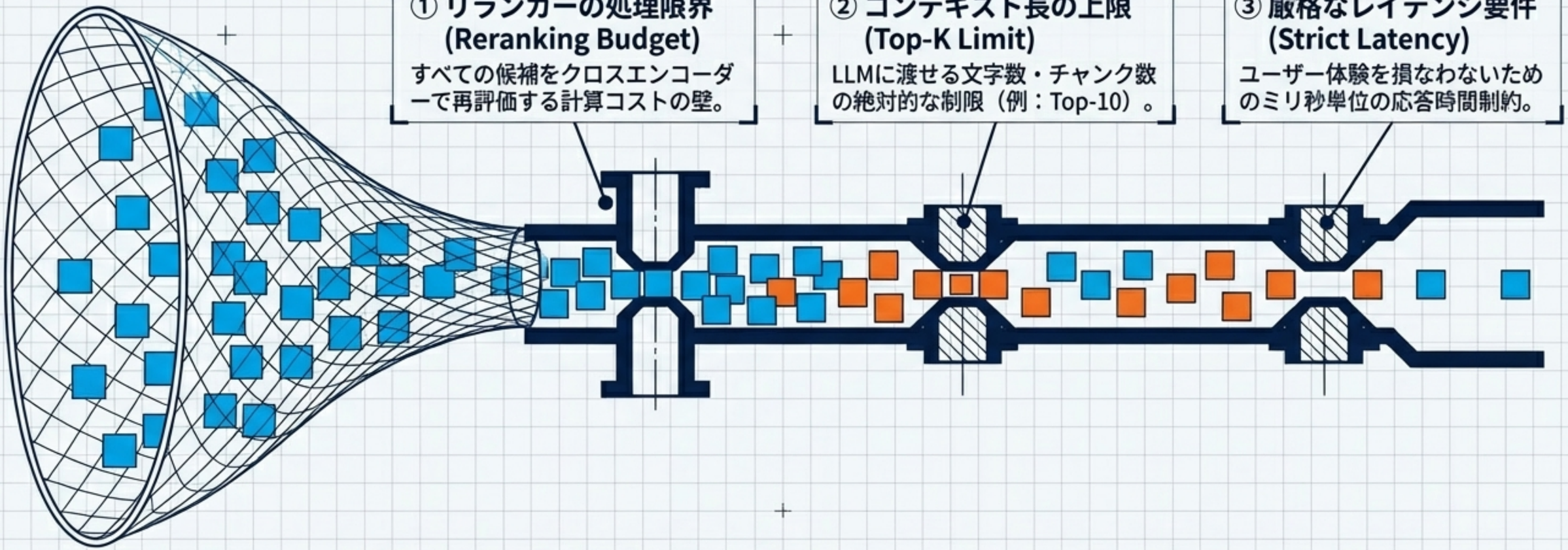
検索漏れ（Missed Evidence）を防ぎ、LLMにより豊かなコンテキストを提供できるという前提。

しかし、本番環境のパイプラインには「3つの物理的限界」が存在する

① リランカーの処理限界 (Reranking Budget)
すべての候補をクロスエンコーダーで再評価する計算コストの壁。

② コンテキスト長の上限 (Top-K Limit)
LLMに渡せる文字数・チャンク数の絶対的な制限（例：Top-10）。

③ 厳格なレイテンシ要件 (Strict Latency)
ユーザー体験を損なわないためのミリ秒単位の応答時間制約。



無限のRecallは存在しない。すべての検索結果は、最終的にこの狭い「ボトルネック」を通過しなければならない。

問い：フュージョンによる「Recallの向上」は、下流工程のボトルネックを生き残かれるか？

データセット

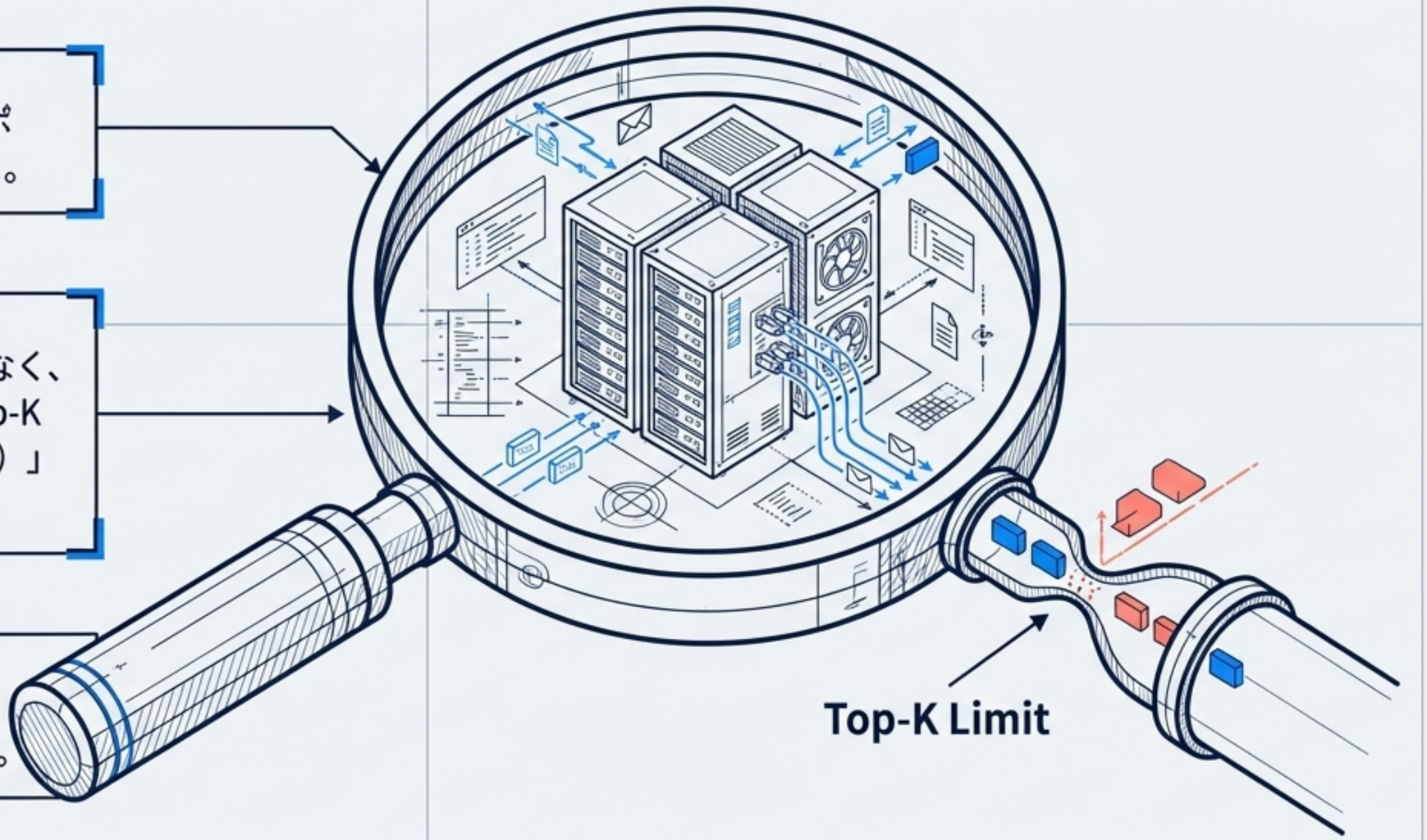
実運用に近いエンタープライズサポートクエリ（N=115, RAGAS生成）。

評価基準

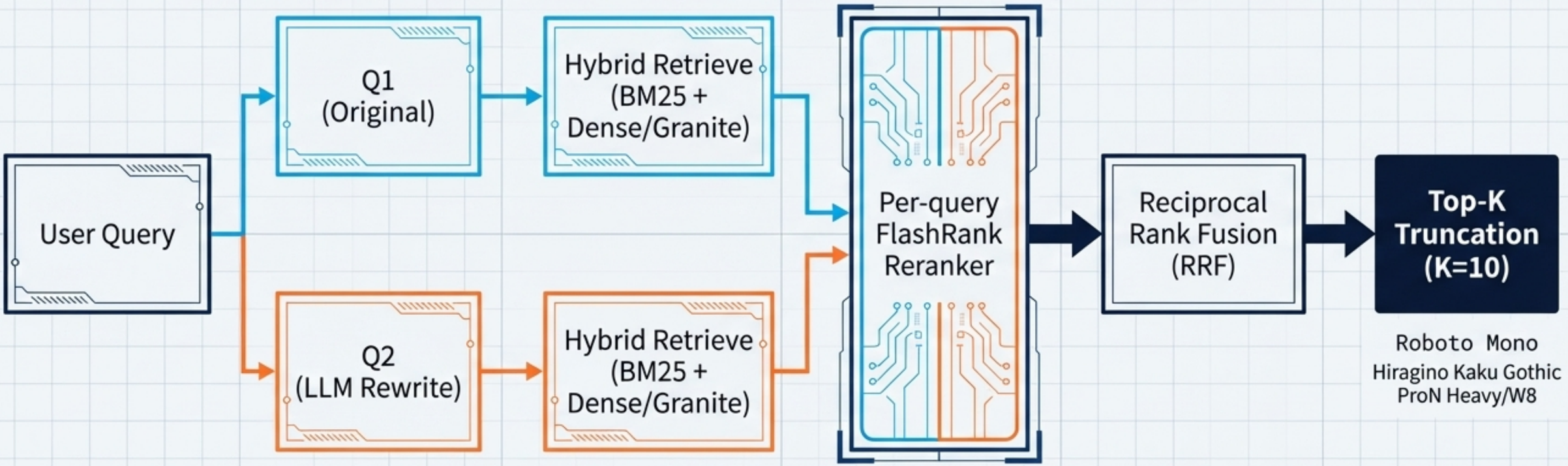
生成された文章の主観的評価ではなく、「正解のナレッジベース記事がTop-K（K=10）に生き残ったか（Hit@10）」という客観的証拠の有無。

焦点

検索層でのスコア向上ではなく、End-to-Endでのシステム提供価値。



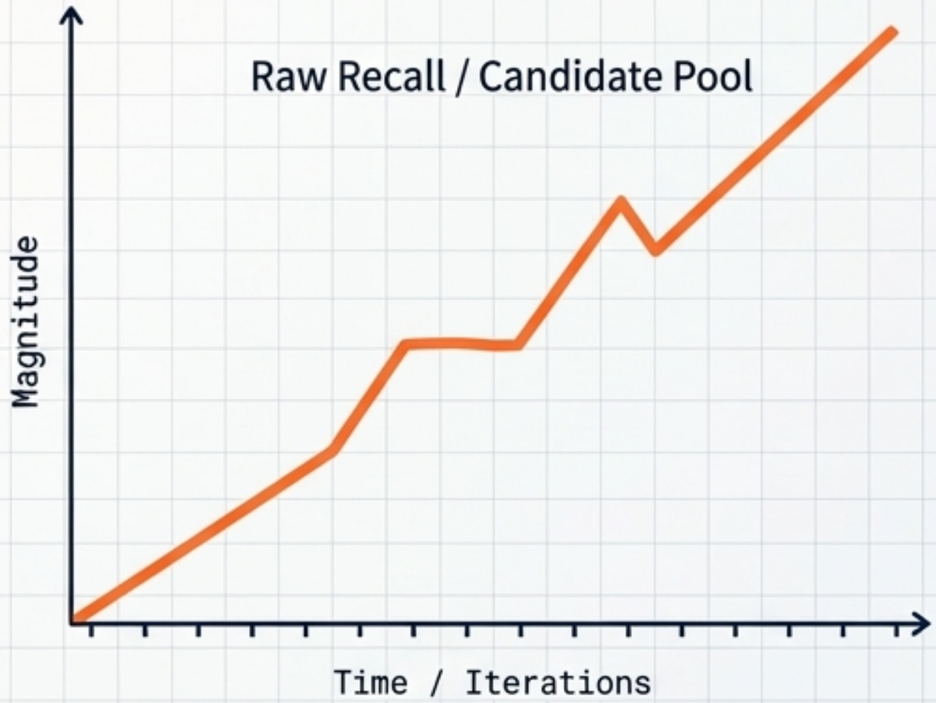
検証アプローチ：エンタープライズRAGの再現



この固定された「10枠」の予算内で、ベースライン（Q1単独）と
フュージョン（Q1+Q2）の勝敗を客観的に測定する。

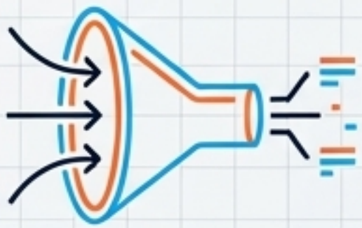
結論：検索層での恩恵は、下流の「Top-K制約」によって無効化される

Upstream/Retrieval Layer

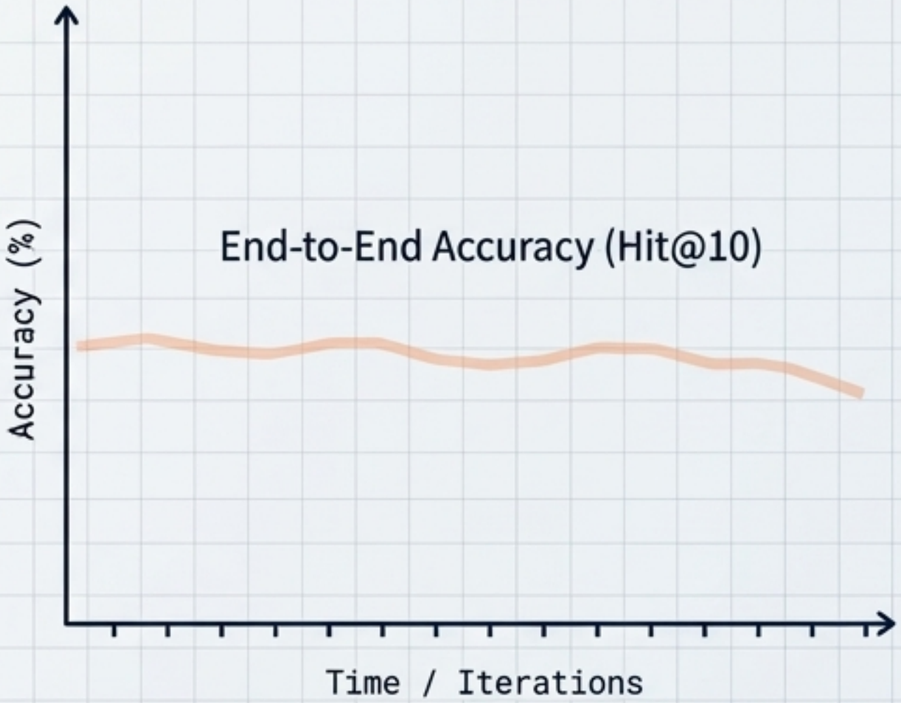


候補数は確実に増加する (+)

Top-K Constraint



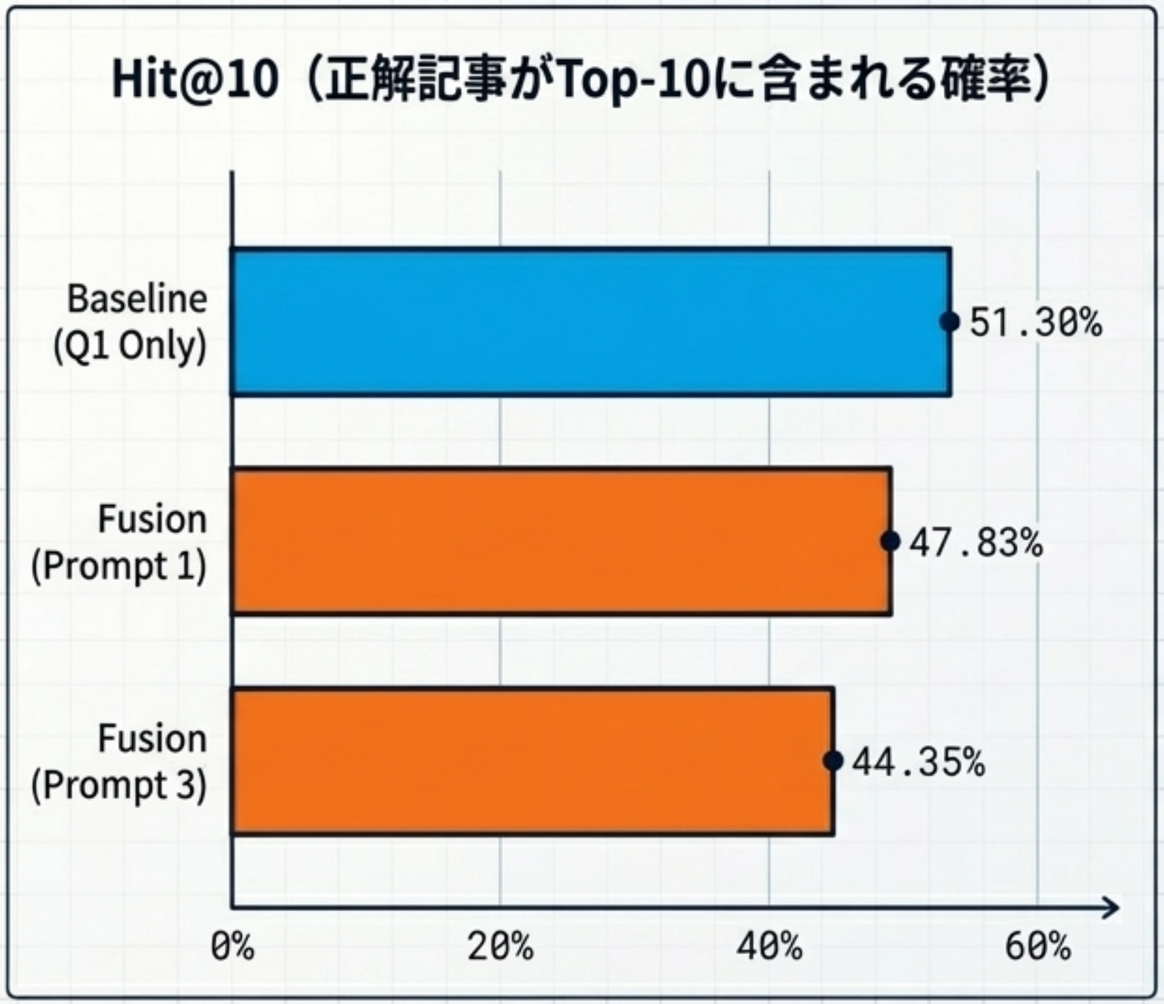
Downstream/Top-K Layer

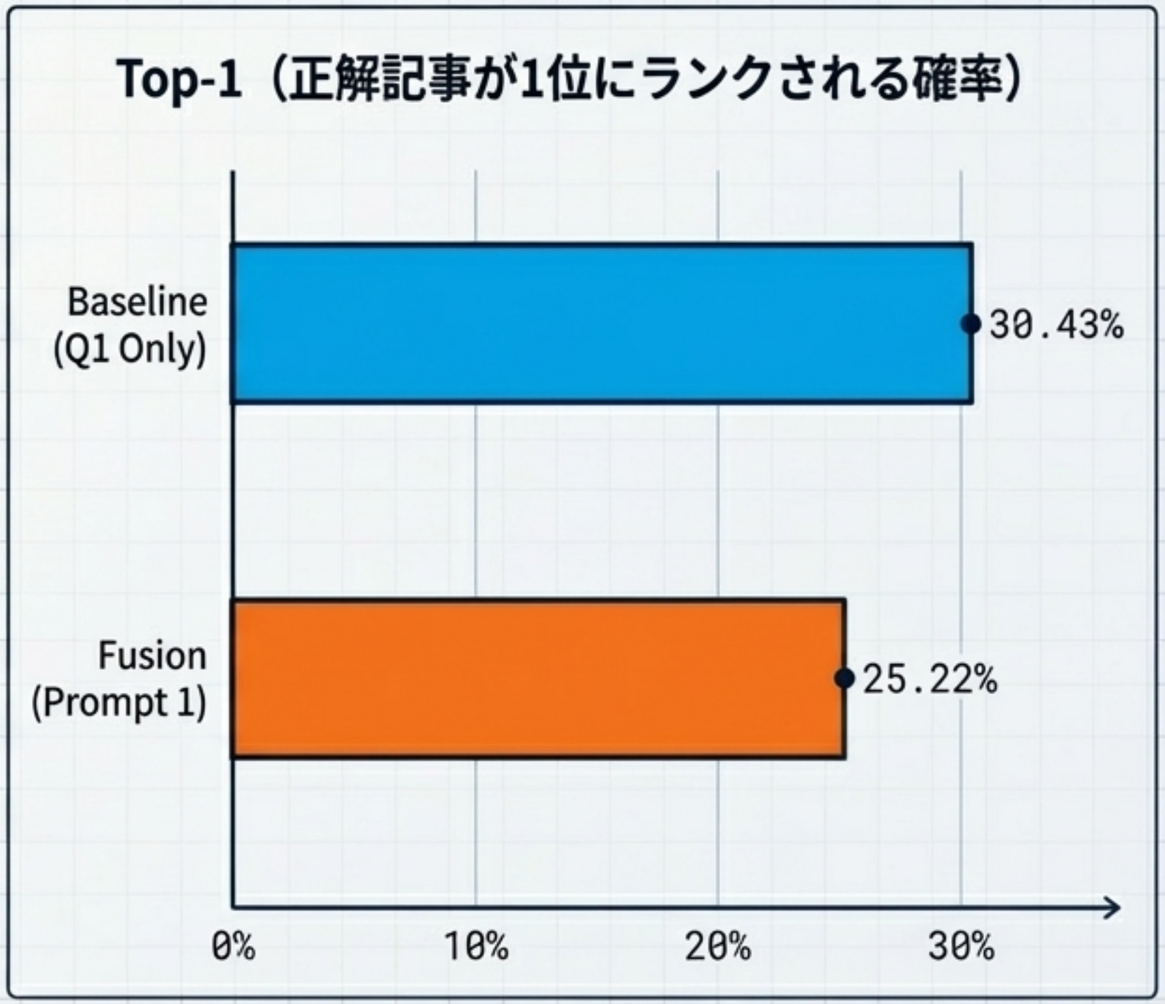


最終的な正解到達率は横ばい、または低下 (-)

フュージョンは「候補の網」を広げるが、リランカーとTop-Kの壁を越えた最終的なコンテキストには、その恩恵が反映されない。

データが示す現実：Top-K精度の低下





複数のプロンプト戦略でテストを実施したが、いずれのフュージョン構成も単一クエリのベースラインを下回る結果となった。

統計的有意性：微小な「Top-3」の向上はノイズに過ぎない

検証手法

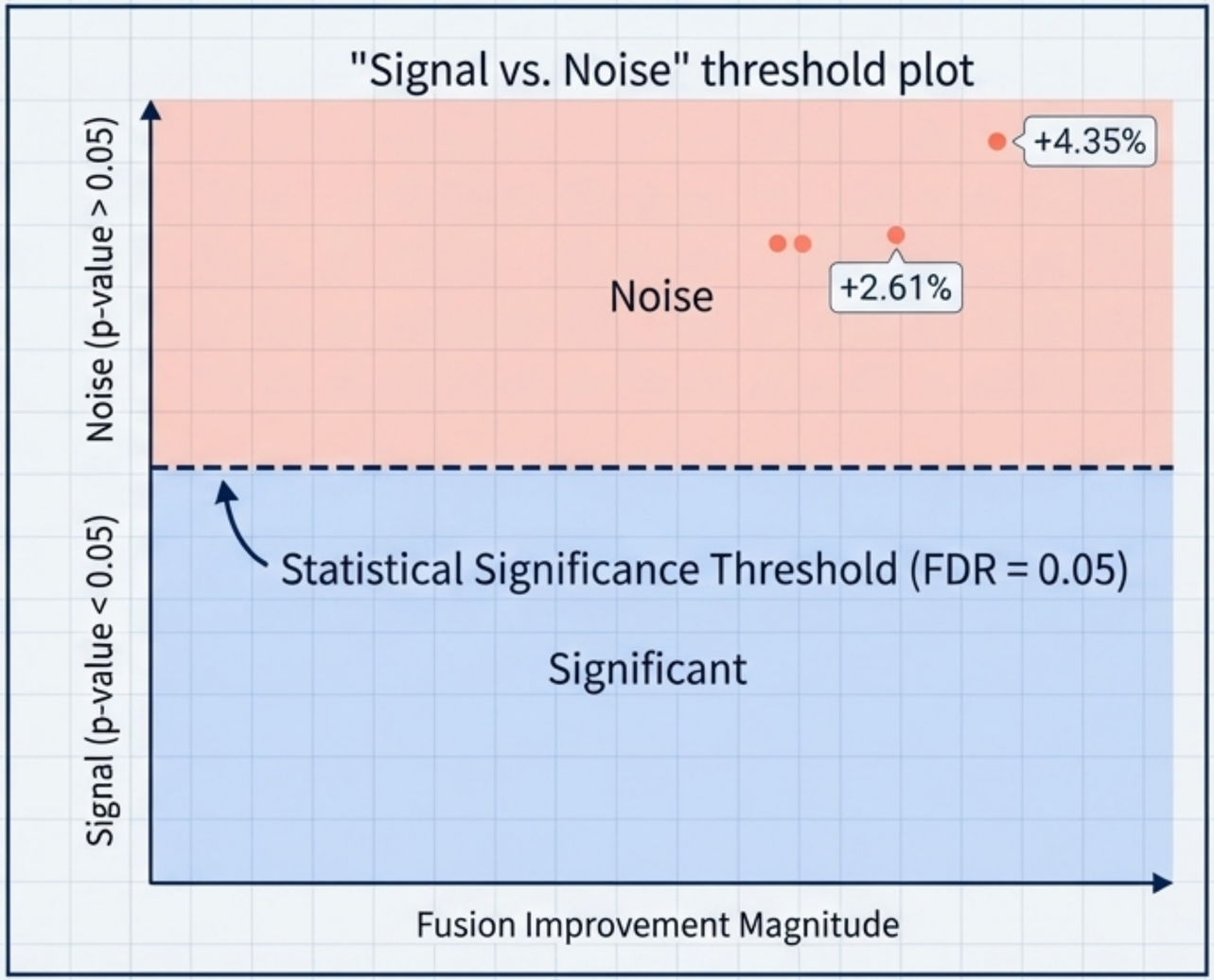
McNemar’s exact test (FDR=0.05補正).

結果

すべてのフュージョン構成において、p_adj ⩾ 0.125 となり、統計的に有意な改善は認められなかった。

結論

本番環境において、追加のシステムコストを正当化できるほどの確実な精度向上は存在しない。



失敗のメカニズム①：コンテキストの重複（Contextual Redundancy）

Jaccard係数

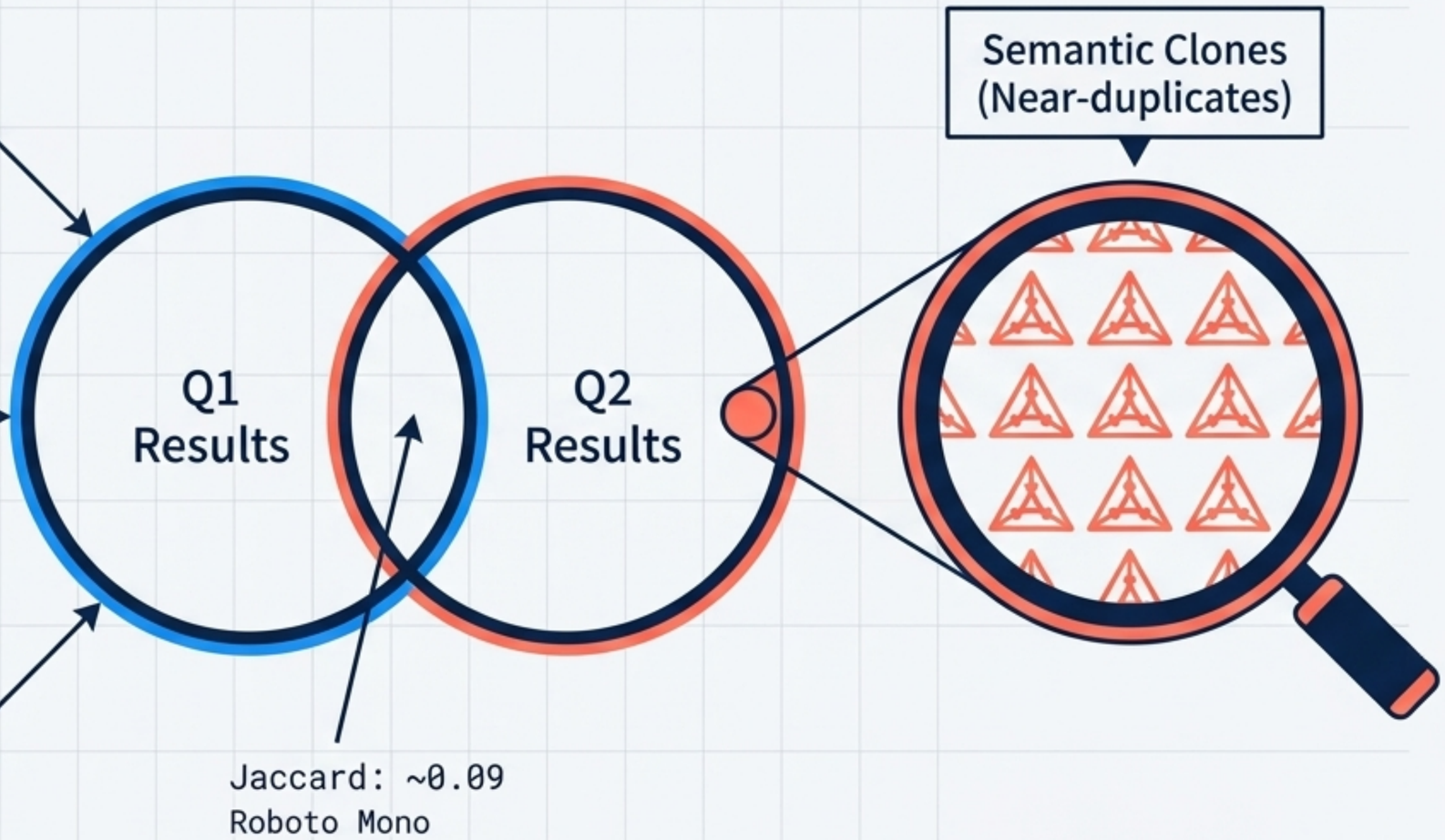
～0.09（非常に低い完全一致率）.
Roboto Mono/Noto Sans JP

結合後の候補数

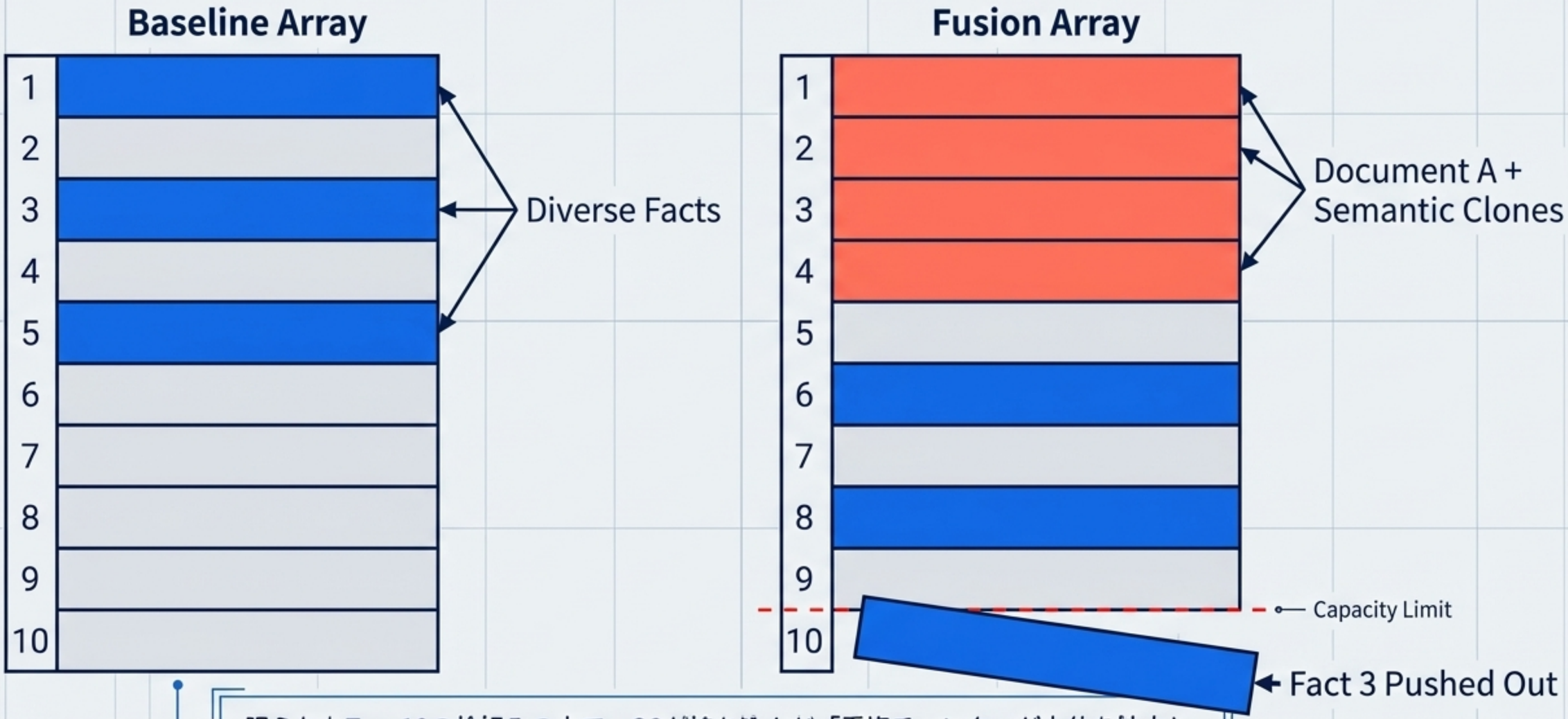
～15件に増加.
Roboto Mono/Noto Sans JP

実態

Q2は「新しい情報」ではなく、隣接するセクションや言い回しが異なるだけの「意味的重複チャンク（Near-duplicates）」を大量に引き込む。



「ノイズ」が「真の正解」を押し出す（Crowding Out Effect）



限られたTop-10の枠組みの中で、Q2が持ち込んだ「重複チャンク」が上位を独占し、本来LLMが必要としていた「多様な正解データ」を圏外へ押し出してしまう。

失敗のメカニズム②：リランカーの過剰適合（Anchor Skew）

現象

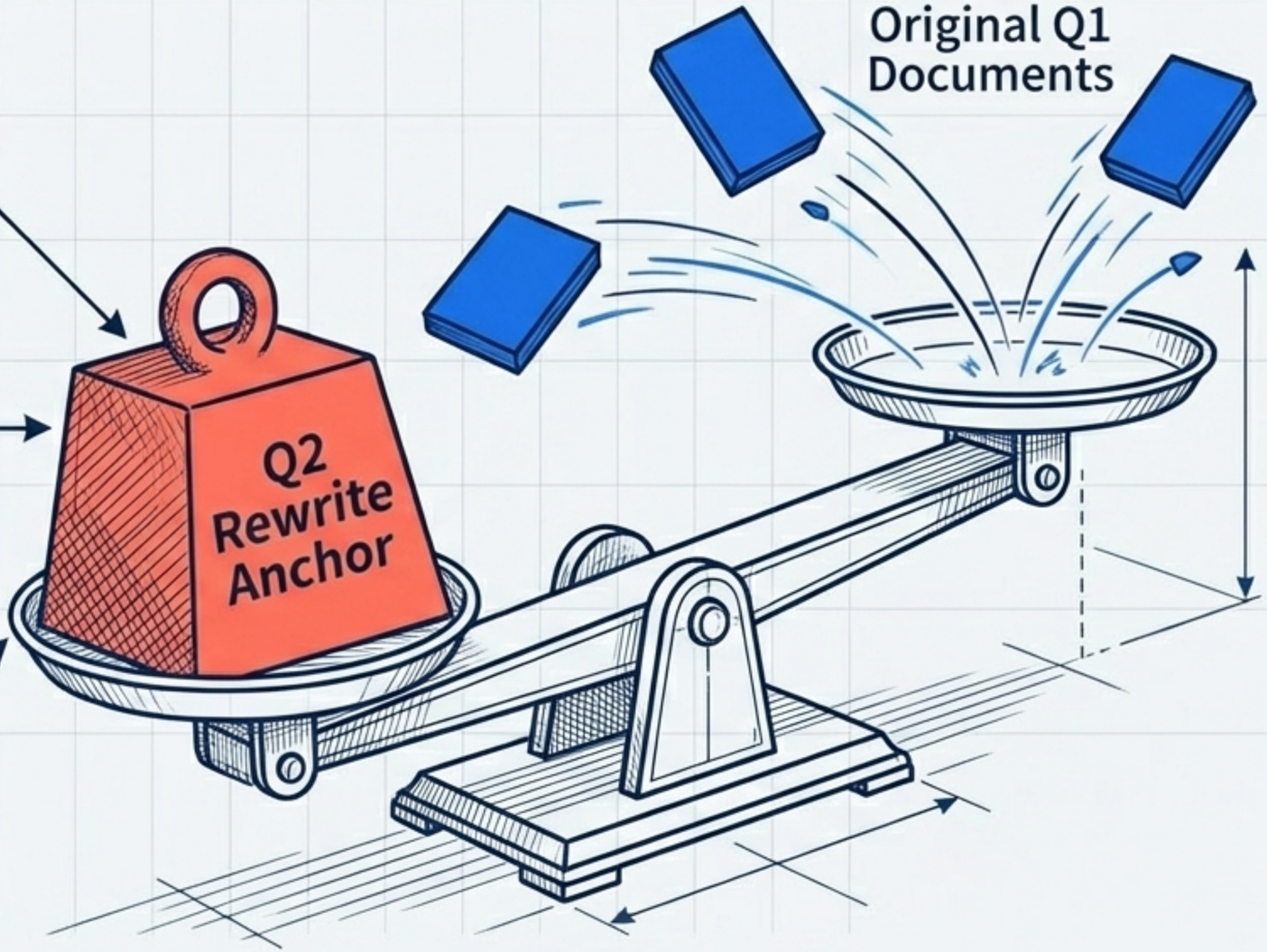
統合されたリストをQ2（LLMの書き換え）を基準に再評価すると、Q2特有の言い回しに過剰に適合（Over-align）してしまう。

致命的なスコア低下

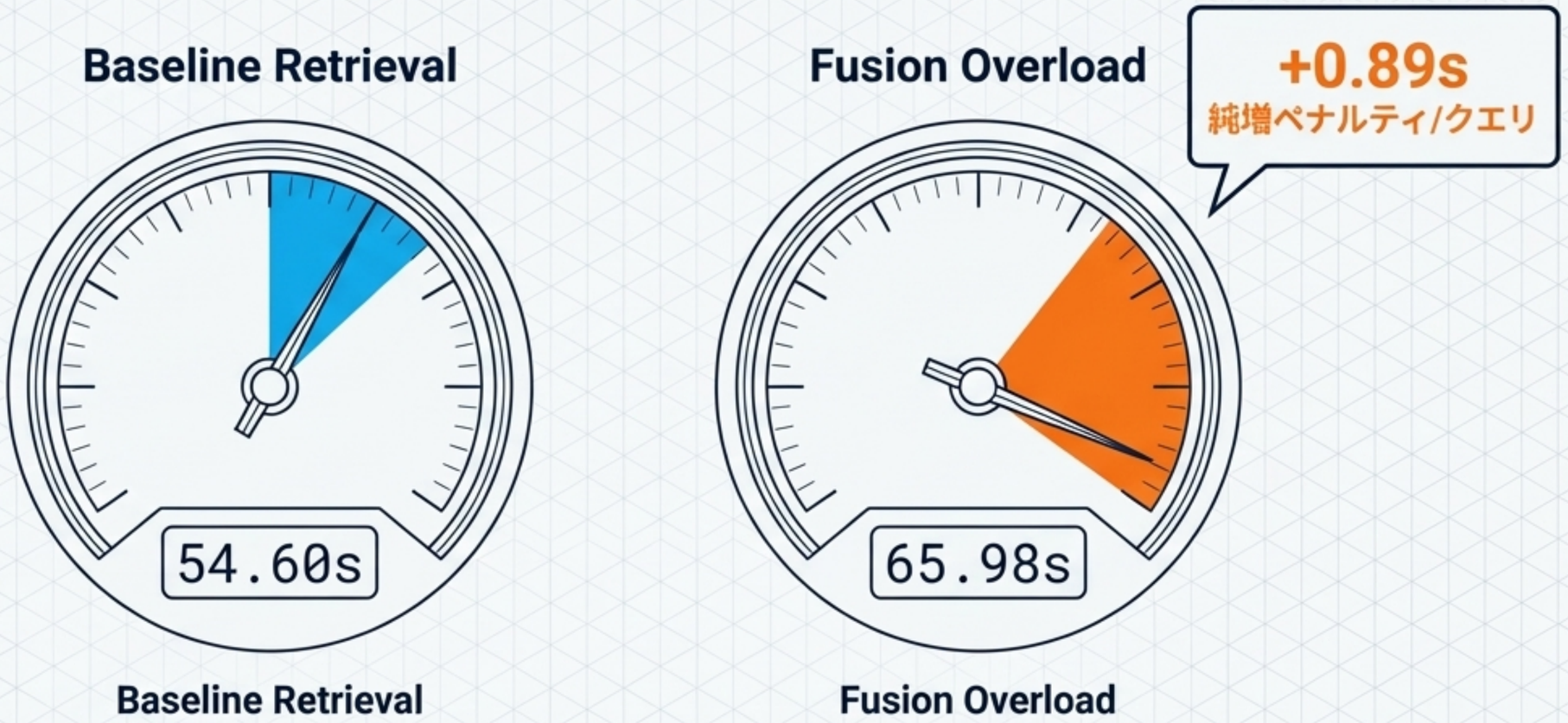
この設定（rerank_on_rrf_q2）では、Top-1精度がベースラインのRoboto Mono: 29.57%からRoboto Mono: 7.83%へと激減する。

結論

LLMによる「意図のズレ（Drift）」が、本来の正解をランクダウンさせる。



隠れたコスト：本番環境で無視できない「レイテンシのペナルティ」



- **追加されるレイテンシ:** +0.89秒の純増オーバーヘッド（クエリ生成、追加検索、RRF処理、再リランク）。
- **スループットへの影響:** 検索時間（バッチ）も54.6秒から65.98秒へ増加。
- **ビジネスインパクト:** ユーザー体験が重視されるエンタープライズ環境において、精度向上が伴わない+0.89秒の遅延は致命的なトレードオフとなる。

フュージョンが有効な「限定的なエッジケース」

1 恩恵を受けるクエリ層 (Recall-scarce regimes) :

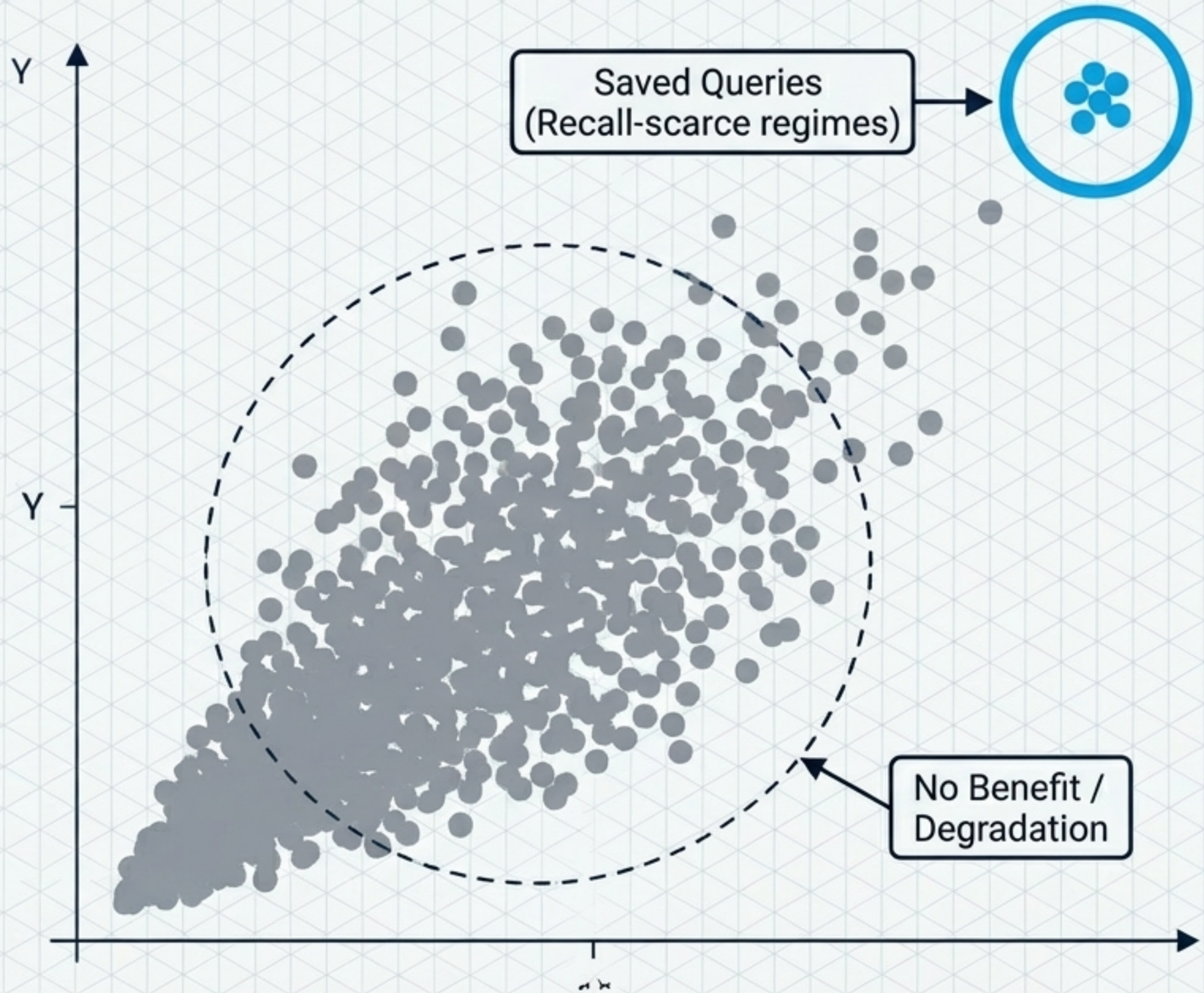
非常に短く曖昧なクエリで、ベースラインの検索が「完全に失敗」しているケース。

2 現実

現実 : このようなケースはワークロード全体の少数派である。

3 結論

結論 : ごく一部の救済のために、システム全体の複雑性とレイテンシを犠牲にするのは、アーキテクチャ設計として非効率である。



アーキテクチャ比較：シングルクエリ vs RAGフュージョン

	Baseline (シングルクエリ)	Fusion (RRF Q1+Q2)
Candidate Diversity (候補の多様性)	○	◎ (Win)
Top-K Context Quality (最終コンテキスト品質)	◎ (Win)	△ (Crowding Out)
System Latency (応答速度)	◎ (Win)	✕ (+0.89s Penalty)
ROI / Complexity (費用対効果)	◎ (Win)	✕ (High complexity)

検索層（候補の多様性）での勝利は、下流工程の圧倒的なデメリットによって相殺される。

エンタープライズRAGへの提言：シンプルさを保ち、基礎に投資せよ

① Recall ≠ Top-K Accuracy

候補を増やすことと、LLMに最適なコンテキストを渡すことは同義ではない。

2 Beware Redundancy

フュージョンが生み出す「意味的重複」は、真の正解をTop-Kから押し出すノイズとなる。

3 Optimize the Foundation

複雑な検索パイプラインを構築する前に、強力なハイブリッド検索とリランカーの組み合わせという「シンプルで堅牢なベースライン」を極めるべきである。