FUSE: Failure-aware Usage of Subagent Evidence for MultiModal Search and Recommendation

Tushar Vatsa*, Vibha Belavadi*, Priya Shanmugasundaram*, Suhas Suresha*, Dewang Sultania
Adobe Inc.
345 Park Avenue, San Jose, CA 95110

***Abstract*—Multimodal creative assistants decompose user goals and route tasks to subagents for layout, styling, retrieval, and generation. Retrieval quality is pivotal, yet failures can arise at several stages: understanding user intent, choosing content types, finding candidates (recall), or ranking results. Meanwhile, sending and processing images is costly, making naive multimodal approaches impractical. We present FUSE: Failure-aware Usage of Subagent Evidence for multimodal search and recommendation. FUSE replaces most raw-image prompting with a compact Grounded Design Representation (GDR): a selection-aware JSON of canvas elements (image, text, shape, icon, video, logo), structure, styles, salient colors, and user selection. Beyond GDR, FUSE implements seven context budgeting strategies: comprehensive baseline prompting, context compression, chain-of-thought reasoning, mini-shot optimization, retrieval-augmented context, two-stage processing, and zero-shot minimalism. Finally, a pipeline attribution layer monitors system performance by converting subagent signals into simple checks: intent alignment, content-type/routing sanity, recall health (e.g., zero-hit and top-match strength), and ranking displacement analysis. We evaluate the seven context budgeting variants across 788 production queries from diverse users and design templates (refer Figure 3). Our systematic evaluation reveals that Context Compression achieves optimal performance across all pipeline stages, with 93.3% intent accuracy, 86.8% routing success(with fallbacks), 99.4% recall, and 88.5% NDCG@5. This approach demonstrates that strategic context summarization outperforms both comprehensive and minimal contextualization strategies. The narrow intent performance variance (89.5-93.3%) across variants validates that all context budgeting approaches provide substantial benefits over zero-shot baselines. We also measure p50/p95/p99 latency and normalized token cost: Context Compression delivers the best latency–cost trade-off (45% lower p95 vs. Baseline; 8 times fewer input tokens vs. Chain-of-Thought) while Chain-of-Thought provides maximal reasoning at the highest cost.**

***Index Terms*—creative assistants, pipeline attribution, subagent evidence, graphical design representation (GDR), context budgeting**

I. Introduction

Traditional multimodal search and recommendation in creative platforms is largely non-assistive: users condense intent into broad keywords to fetch assets, skim through presented results to select the asset appropriate for their intent, decide the modality or tool to be used, all while understanding different platform specific features. This burdens both novices and experts often leading to missed retrievals despite the platform possessing assets that might have been able to satisfy the user intent if the user experiences were more assistive. The advent of large language models has enabled customers to converse in natural language with various systems on the web, increasing their expectations from traditional search and recommendation systems to scale to natural language inputs and improve ease of use through real-time discovery assistants.

Modern multimodal assistants address this by adopting agentic orchestration: a neural planner (superagent) decomposes user queries and routes tasks to specialized subagents for layout, styling, retrieval, and generation [1]–[3]. Within this architecture, the search subagent is a linchpin; downstream actions cannot redeem poor retrieval. Failures can originate anywhere in the pipeline (intent, routing, recall, ranking), and naïvely sending rich multimodal context—especially images not only degrades the output quality but also significantly increases latency and token cost [4], [5].

Vision-language models present significant overhead in terms of latency and cost, as each image must be tiled into patches (e.g., 512×512) and contributes hundreds of tokens in addition to a fixed per-image overhead [6]. Commercial APIs price vision input at roughly \$10 per million input tokens and \$40 per million output tokens, with high-quality image generation reaching \$0.17 per image [7]. Latency is also substantial: benchmarks of GPT-4o (Vision) report over 5 seconds per 800×800 image, compared to hundreds of milliseconds for task-specific detectors [8]. Optimized vision-language models like FastVLM reduce but do not eliminate this gap, achieving up to $85\times$ faster time-to-first-token than LLaVA-OneVision while retaining accuracy [9]. These drawbacks make naive raw-image prompting impractical for production pipelines that must meet strict real-time SLAs and cost budgets.

FUSE: Failure-aware Usage of Subagent Evidence. We propose FUSE, a production-oriented recipe for multimodal search and recommendation that unifies (i) a compact *Graphical Design Representation* (GDR) for conditioning, (ii) *budgeted context policies* that right-size prompts under cost caps, and (iii) a *pipeline attribution* layer that detects failure modes and selects/escalates policies accordingly.

We operate three practical modes that reflect how real users engage the system (shown in Figure 1):

1) *Raw query* $\rightarrow$ direct search with no additional context.
2) *Superagent-enriched prompt* $\rightarrow$ the neural planner expands and clarifies the user request.
3) *GDR + superagent-enriched prompt* $\rightarrow$ the planner's prompt is augmented with a compact, selection-aware JSON of the canvas (elements such as image, text, shape,

*These authors contributed equally to this work.

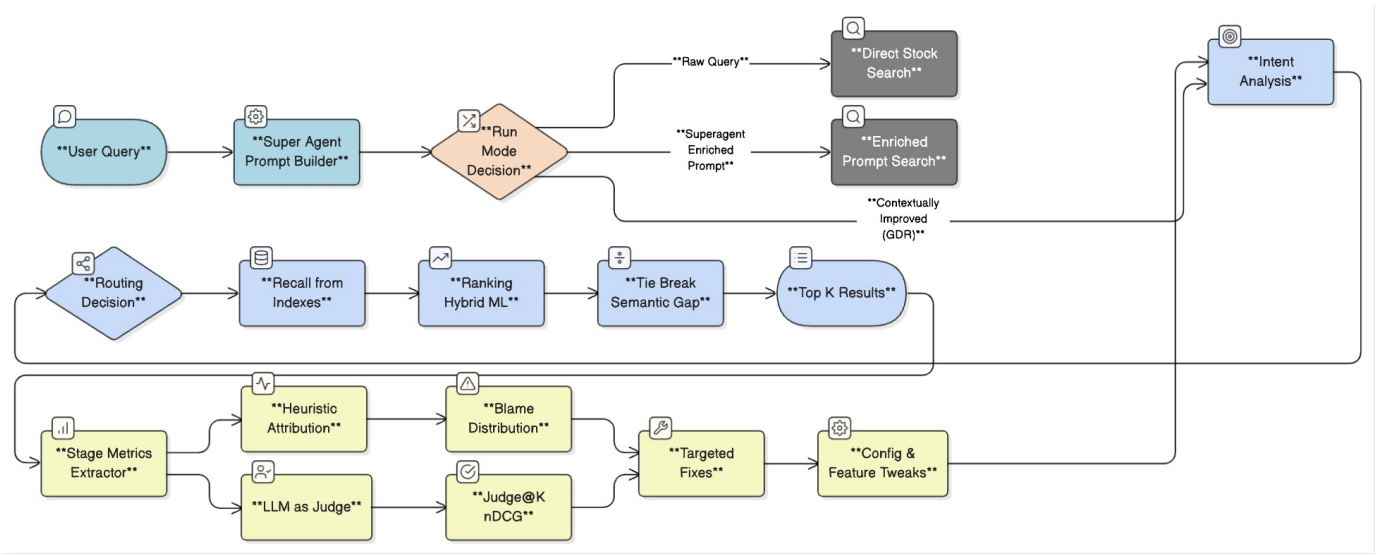

Fig. 1. Multi-stage search system flow with FUSE integration. The diagram shows three operation modes: (a) raw query, (b) superagent-enriched prompt, and (c) superagent prompt + compact GDR. The contextual subagent performs intent analysis, content-type routing, recall, ranking, and semantic tie-breaking. The evaluation and attribution layer extracts stage-wise metrics, applies heuristic and LLM-as-judge evaluation, localizes failures via blame distribution, and drives targeted fixes and configuration updates.

icon, video, logo; structure; styles; salient colors; and user selection), replacing costly raw pixels.

Over these modes, FUSE applies lightweight prompting variants chosen to meet latency and token constraints: adjustable examples (none, brief, or retrieved), GDR compression for large contexts, selective reasoning triggered only by ambiguity, and two-stage processing. Policies are chosen on demand rather than using fixed configurations.

A stage-wise attribution layer converts subagent evidence into operational checks: (1) *intent alignment* (semantic proximity between the planner subprompt and expected intent), (2) *routing sanity* (content-type confidence with fallbacks), (3) *recall health* (zero-hit and top-match strength), and (4) *ranking stability*. These checks serve two purposes: they enable real-time policy adjustments within budget constraints and inform offline system tuning (such as fallback mappings and re-ranking thresholds).

II. Related Work

Within the image-rich project creation like brochures (refer Figure 3), search and recommendation extend beyond generic multimodal matching to incorporate template structure, visual style, and layout cues. Design-aware retrieval systems highlight the role of structured conditioning for template retrieval, while user-interactive creative assistants increasingly incorporate context signals such as selection history and semantic grouping. Despite these advances, existing systems primarily emphasize improving embedding quality or context fusion rather than systematically diagnosing where failures occur within multi-stage pipelines. Our approach addresses this gap by combining a compact Graphical Design Representation (GDR) with stage-wise failure attribution. This enables both efficient multimodal representation and systematic error diagnosis in production creative workflows.

A. Agent-Orchestrated Search Systems

Agent-orchestrated architectures have evolved from monolithic systems toward modular, specialized components coordinated by superagents. Early frameworks such as ReAct [1] and Toolformer [2] showed how large language models (LLMs) can interleave reasoning with external actions or teach themselves to invoke tools. Cascades [3] extended orchestration by recursively summarizing long contexts, while surveys of augmented language models highlight general strategies for tool use and memory integration. More recent production-oriented frameworks such as EHR-Agent [4], D-PoT for dynamic GUI planning [5], and HM-RAG [10] demonstrate effective coordination among specialized agents for query analysis, retrieval, and synthesis. However, these systems emphasize agent coordination efficiency or component-level optimization rather than systematic pipeline diagnosis. In contrast, our framework introduces a production-grade attribution layer designed for low-latency, cost-aware multimodal environments. By operating under strict efficiency budgets, it provides systematic stage-wise diagnosis of failures across hierarchical agent interactions, enabling targeted optimization that is both actionable in practice and scalable beyond heuristic component tuning.

B. Context Budgeting and Multimodal Understanding

Context budgeting extends beyond simple prompt optimization to encompass broader information management in AI systems [11], [12]. This field involves dynamic context construction, memory management, and tool integration. Vision-language models and multimodal frameworks [13] show grow-

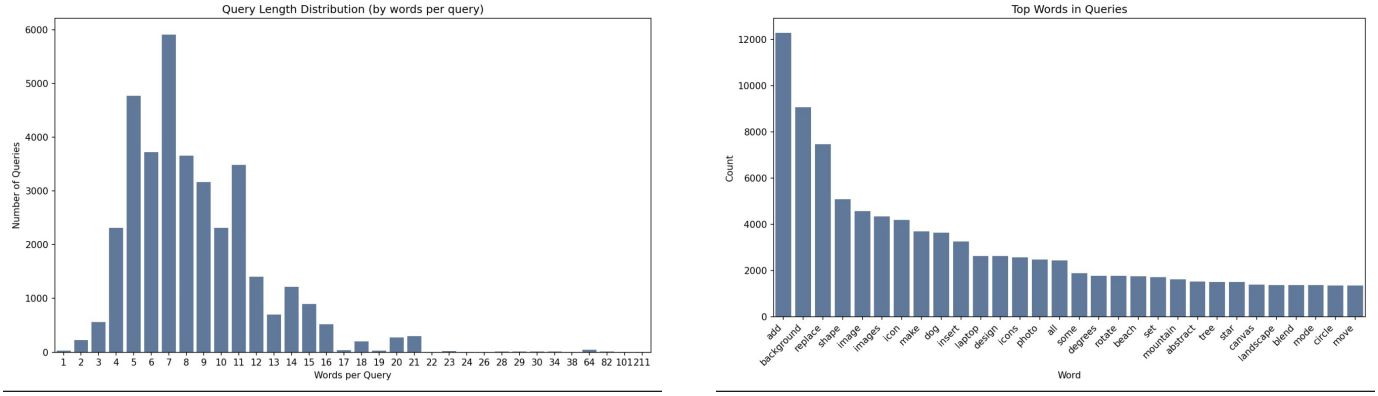

Fig. 2. Production Dataset Query Analysis. The left image shows query length distribution with the average length of seven words per query(after stemming). The right image shows the most frequently occurring words(after stemming) in the query, those being *add*, *background*, *replace*, *shape* and *image*.

ing capability in handling complex contexts. While existing context budgeting work focuses on general information architecture and context synthesis, our approach specifically targets context management in production multi-agent search systems, emphasizing design template constraints and reliable fallback handling.

C. Pipeline Attribution and Error Analysis

Traditional pipeline optimization relies on individual component tuning without systematic failure diagnosis [14], [15]. Recent evaluation frameworks employ LLM-as-a-Judge methods and established metrics like nDCG and MRR [16], [17] for assessing search quality. However, existing attribution techniques focus on model-level explanations rather than stage-specific failure detection in complex multi-agent systems. Our framework advances beyond current approaches by introducing quantitative thresholds for systematic attribution across five pipeline stages, enabling precise identification of failure sources within agent-orchestrated architectures where traditional methods fail to distinguish between coordination failures and component-specific issues.

III. System Architecture and Pipeline Attribution Framework

We analyzed approximately 36,000 production queries (shown in Figure 2) and found that most are short commands for asset operations—adding, replacing, or finding images, icons, or text styles. Crucially, many requests are context-dependent: creators frequently issue location-based or selection-dependent queries (e.g., "recommend a background" while a region is selected; "replace this icon" referring to a highlighted layer; "find beans" on a coffee poster). When executed as raw text, these queries omit crucial context—the target, canvas constraints (layout, aspect ratio, palette/typography, brand requirements)—leading to systematic failures: wrong content types, semantic mismatches (e.g., "beans" returning legumes instead of coffee beans), and layout incompatibilities. We manually evaluated a 788-case sample covering add/replace/find tasks from this production dataset. Each case was judged for action correctness, asset relevance and layout/style coherence. Empirically, the raw query mode (no additional context) performs worst (63% pass rate). These findings strengthen our design choice: infer the user's intent (add/replace/find), resolve the target scope from the selection and object graph, and rewrite the raw query into a contextualized request that injects canvas semantics and feasibility constraints before retrieval and ranking. In short, relying on raw queries is disastrous in real creative workflows because users naturally ask context-laden questions; robust systems must make that context explicit before they search.

The superagent model (responsible for routing to different subagents) was trained to infer context directly from the raw user query, using only a light canvas hint, and then forward it as a free-form superagent prompt. In production this was brittle: resulting in long, redundant rewrites. Beyond quality issues, the approach was operationally expensive and slow: we needed to embed and ingest an enormous corpus ($\approx$ 0.5B assets), which made index construction and refresh cycles cost-heavy and time-consuming. Although this method outperformed raw queries, these robustness and infrastructure constraints led us not to proceed; we instead moved to a compact, typed contextual schema (intent + canvas constraints) that drives retrieval/ranking without prompt bloat and with explicit contrast and feasibility checks. An example of context-aware search is shown in figure 3.

A. Problem Formulation

Canvas: a design state C with objects $O = \{o_k\}$ (text, images, shapes), styles S (palette, typography), layout metadata (bounding boxes, z-order), and domain cues (e.g., "coffee roasting spectrum").
User query: an imperative text q (e.g., "find beans")
Intent: $i \in \{\text{add}, \text{replace}, \text{find}, \text{edit}, \text{style}\}$ with target scope $t \in \{\text{selection}, \text{region}, \text{global}\}$.
Rewriting: $f(q, C, i, t) \mapsto q'$ injects semantic and geometric constraints (topic, aspect ratio, composition, licensing).
Retrieval: $R(q')$ over dual indices (semantic vectors and lexical/metadata), returning K candidates.

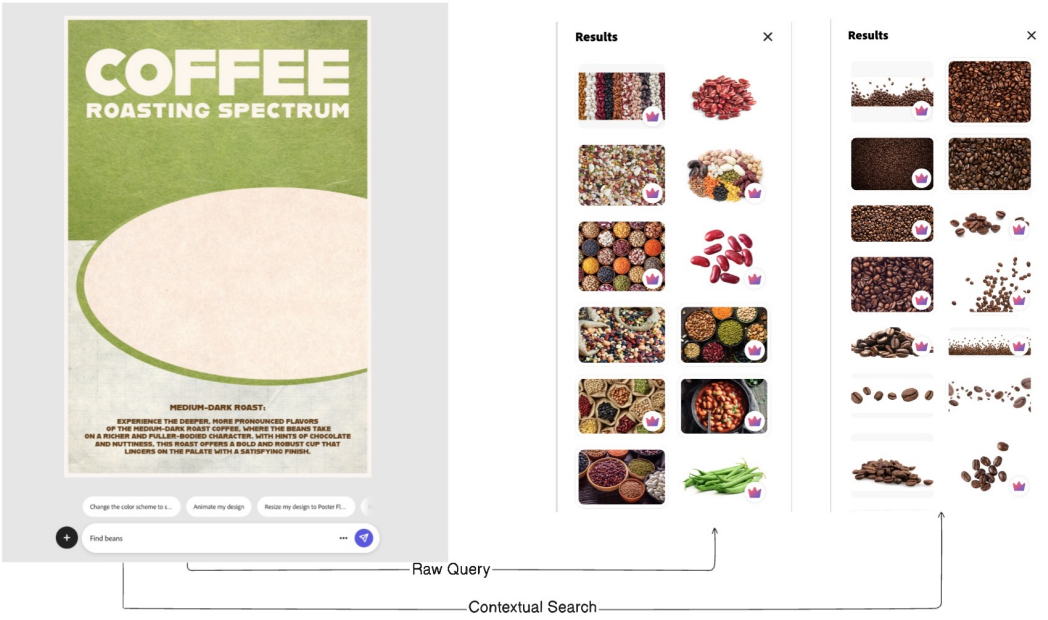

Fig. 3. Example of context-aware search in a creative design workflow. A user working on a coffee roasting poster issues the query *"find beans"*. The raw query returns generic beans (left result panel), while the contextual search: informed by the template content and design context prioritizes coffee beans (right result panel), demonstrating how contextual grounding improves retrieval relevance.

Re-ranking: $g\big(C, i, t, \{a_j\}_{j=1}^{K}\big) \mapsto$ ranked list via layout fit, brand/style similarity, and action feasibility.
Execution: the action planner selects top candidate(s) and applies a transformation consistent with i, t.

B. Canvas grounding and Intent inference

As shown in Figure 1, we use a lightweight GPT-4o-mini model for intent understanding and LLaVA models to caption the live canvas. The canvas is serialized as a *Graphical Design Representation (GDR)*: a typed JSON that records each page and element with geometry and appearance (opacity, zIndex), placement (tx, ty, rotation, scales), precise frame coordinates (topLeft, topRight, bottomRight, bottomLeft), color roles and palettes, and LLaVA captions plus detected regions. Example of GDR is presented below:

Listing 1. Example of Graphical Design Representation

```
{
  "gdrVersion": "assistant-100",
  "pages": [{
    "id": "page-0",
    "elements": [{
      "id": "image-0", "type": "image",
          "opacity": 1, "zIndex": 1,
      "frame": { "topLeft": [x0,y0],
          "bottomRight": [x1,y1] },
      "colors":
          [{"color":"#040404","weight":0.71},
          ...],
      "content": ["A blue background with
          vegetables ..."],
      "regions": [{ "label":"octopus (animal)",
          "score":0.79, ... }]}]}]
}
```

At query time, GPT-4o-mini takes the GDR and superagent prompt to create a detailed subprompt that makes hidden design requirements explicit. This process extracts six key elements: what action to take (add/replace/find), where to apply it (selected item, region, or whole canvas), size and shape needs (aspect ratio, dimensions), color compatibility (matching palettes, ensuring contrast), style consistency (textures, brand alignment), and content rules (licensing, appropriateness).

This approach fixes common problems with basic contextual search. For instance, if someone searches for "find similar cars" while working on a design with a green background, a regular contextual search might return green cars that disappear into the background. FUSE's subprompt instead specifies "find cars with sufficient contrast against green" and checks that results will actually be visible when placed on the canvas.

Our focus centers on the search subagent ecosystem that implements a comprehensive five-stage pipeline: pre-processing, intent understanding, routing, recall, and ranking. Unlike distributed architectures employing multiple search subagents, our system utilizes a unified search subagent capable of processing queries across multiple content surfaces including stock photos, templates, enterprise assets, and multimedia content. The search subagent connects to specialized Entity Search Handlers that interface with domain-specific Elastic-Search indices (e.g. for photos, icons, backgrounds, video clips, audio) enabling comprehensive multimodal searches while maintaining optimized retrieval performance through distributed indexing

While GDR-grounding with a GPT-4o-mini intent model eliminated most raw-query failures, end-to-end latency on ed-

itors still hovered at 2.8 s (p95), driven largely by an oversized system prompt and heavy few-shot conditioning. Much of the injected context did not apply uniformly across asset classes; conflicting hints in the prompt confused retrieval and ranking (e.g., adding shape-specific cues when our index had sparse shape coverage frequently yielded zero hits). To resolve this, we redesigned context handling to express canvas constraints explicitly *without* long prompts and to move expensive steps off the critical path.

C. Context Budgeting

We conducted comprehensive experiments across seven context budgeting variants, each testing different information density and processing strategies:

- **Baseline Configuration**: employs comprehensive system prompts with full few-shot examples across all content types. While maximizing available information, this "context-rich" approach often overwhelms the model with irrelevant details and incurs substantial token costs, particularly when processing complex canvas states.
- **Chain-of-Thought Contextualization** augments standard context with intermediate reasoning traces through an assistant "scratchpad." The model explicitly reasons through intent inference and constraint extraction before generating subprompts, improving complex query handling at increased token cost.
- **Context Compression** sends the summarized version to compress GDR representations and prompts while maintaining semantic richness, testing whether information density improvements translate to better performance-cost ratios.
- **Mini-Shot Optimization** provides one strategically curated example per content type, testing whether targeted context selection outperforms volume-based approaches.
- **Retrieval-Augmented Context** dynamically selects few-shot examples by computing embedding similarity between current queries and historical examples. This ensures contextual relevance while maintaining computational efficiency through asset-specific example banks.
- **Two-Stage Hierarchical Processing** first predicts content type, then applies asset-specific context conditioning. Once the system predicts "shapes", it skips contextual examples entirely given limited inventory coverage, while photo queries receive rich conditioning. This reduces context noise while preserving semantic richness where beneficial.
- **Zero-Shot Minimalism** strips context to bare system prompts without guidance examples, establishing performance floors while revealing inherent model capabilities. This "context-lean" baseline processes only essential GDR elements and superagent prompt.

Each variant is systematically evaluated through our Pipeline Attribution Framework, which tracks failure distribution across the five stage pipeline: pre-processing, intent understanding, routing, recall, ranking.

D. Pipeline Attribution Framework

To evaluate pipeline performance, we developed Pipeline Attribution Framework that classifies failures into four distinct attribution categories a) Intent b) Routing c) Recall and d) Ranking by comparing predicted outputs against expected ground truth at each pipeline stage respectively.

Intent Attribution captures failures in extracting user intent from natural language queries. Example: For the query "add photo of beans" we expect subprompt to be "coffee beans" due to the GDR context (template has coffee elements in it) but the system predicted "black beans" as the main search subprompt, indicating complete semantic misunderstanding between user intent and system interpretation.

Routing Attribution identifies incorrect agent selection decisions despite successful intent extraction. Example: A system correctly identifies "fireworks" intent but incorrectly routes it to video content type search instead of the expected photo content type search, requiring fallback routing mechanisms. Having incorrect routing can potentially result in change in the asset modality and degrade user experience.

Recall Attribution indicates content coverage gaps where the system fails to retrieve any relevant results due to a lack of assets. Example: The query "create a deer pattern background" returns zero hits, suggesting missing content in the underlying database or indexing failures.

Ranking Attribution captures cases where relevant content is retrieved but incorrectly ordered by relevance. Example: For the query "stone pattern backgrounds" if the most semantically similar result (similarity 0.674) appears at rank 3 and a less relevant result (similarity 0.474) occupied rank 1 in the search results list.

E. Evaluation Metrics

We employ stage-specific metrics aligned with our Performance Attribution Framework to evaluate pipeline performance across the seven context budgeting variants.

Intent Evaluation counts semantic mismatches (below a certain threshold) between ground truth subprompts and system-predicted subprompts. For example, the query "replace image with a modern salon" expects the subprompt "modern salon" - successful intent extraction increments the match count, while mismatched predictions like "dandelions" contribute to failure counts.

Routing Evaluation counts exact matches between expected content type and predicted content type. Queries expecting "photo" content type that correctly predict "photo" increment success counts, while routing failures like predicting "icon" instead of "shape" increment failure counts.

Recall Evaluation counts zero-hit queries that return no search results. Queries like "create a deer pattern background" returning empty result sets increment the zero-hit count, indicating recall failures and content coverage gaps.

Ranking Evaluation employs NDCG@5 to assess result ordering quality within the top 5 positions, reflecting user focus on initial search results. NDCG@5 uses semantic similarity scores as relevance judgments, rewarding systems that place

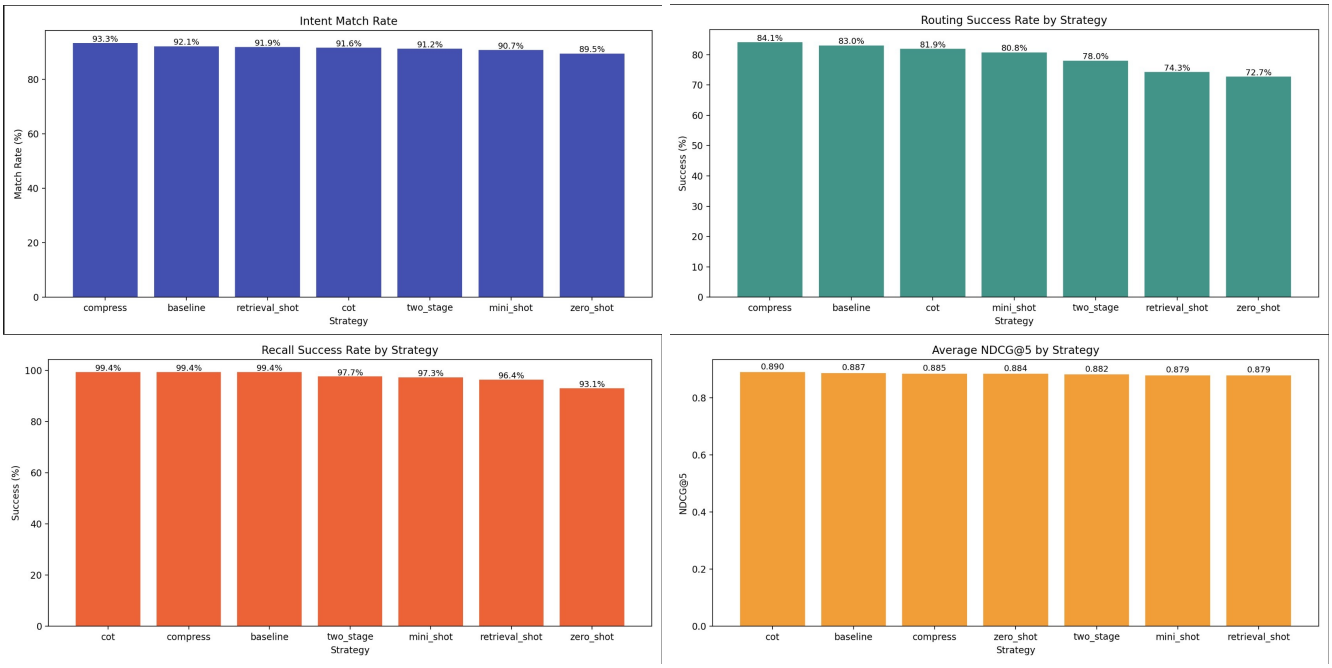

Fig. 4. We analyze Pipeline Performance for *Intent Match Rate*, *Routing Success Rate*, *Recall Success Rate* and *Average NDCG@5* across Context Budgeting strategies

higher-similarity results at top ranks. For instance, a query where the best semantic match appears at rank 5 instead of rank 1 receives lower NDCG@5 scores than optimal ordering.

IV. Results and Analysis

TABLE I
Comprehensive Performance Analysis Across Context Budgeting Strategies

Variant	Performance Metrics (%)				
	Intent Match	Routing Success	Recall Success	NDCG@5 Success	Overall
Baseline	92.1	83	**99.4**	88.7	80.8
Chain-of-Thought	91.6	81.9	**99.4**	**89.0**	81.7
Context Compression	**93.3**	**84.1**	**99.4**	88.5	**82.9**
Mini-Shot	90.7	80.8	97.3	87.9	78.6
Retrieval-Augmented	91.9	74.3	96.4	87.9	79.4
Two-Stage	91.2	78	97.7	88.2	79.9
Zero-Shot	89.5	72.7	93.1	88.4	74.5

Our systematic evaluation of 788 production (templates + queries) across seven context budgeting variants reveals distinct performance patterns at each pipeline stage. Of these we only consider scenarios where there is no failure on the superagent side and the requests reach the search subagent. The error attribution analysis demonstrates that different context strategies produce markedly different failure distributions, validating our hypothesis that context engineering impacts vary significantly across pipeline components.

A. Intent Extraction Performance

Intent extraction demonstrates consistent performance across context budgeting variants, ranging from 89.5% to 93.3% accuracy. Context Compression achieves optimal performance at 93.3%, followed by Baseline (92.1%) and Retrieval-Augmented (91.9%). Chain-of-Thought (91.6%), Two-Stage (91.2%), and Mini-Shot (90.7%) maintain competitive accuracy, while Zero-Shot (89.5%) establishes a minimal context baseline(Table I)

The narrow 3.8-percentage-point variance indicates that all context budgeting strategies provide substantial semantic understanding benefits, with even minimal approaches achieving production-viable accuracy. This validates our hypothesis that strategic context engineering enhances intent extraction while creating opportunities for computational optimization.

B. Routing Performance and Fallback Mechanisms

Routing performance exhibits the highest variance among pipeline stages, ranging from 72.7% to 84.12% success rates. Context Compression achieves optimal routing performance at 84.12%, followed by Baseline (83.04%).

Retrieval-based strategies demonstrate routing challenges, with Zero-Shot (72.73%) and Retrieval-Augmented (74.25%) exhibiting the lowest success rates. This 12-percentage-point variance confirms that content type prediction requires stable contextual guidance, with curated compression approaches outperforming dynamic retrieval methods for routing decisions.

C. Fallback Mechanism Enhancement

Fallback routing mechanisms provide consistent performance improvements across all variants as demonstrated in Table II and Figure 5, with gains ranging from 2.22 to 7.83

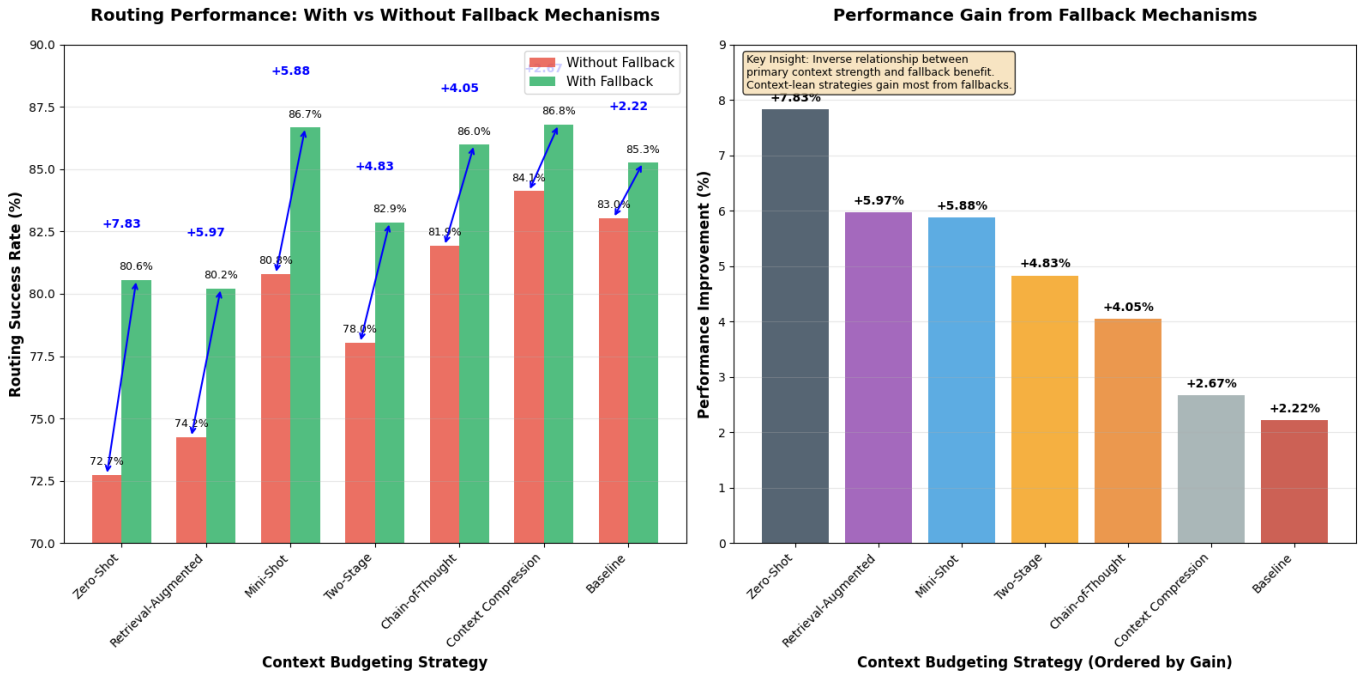

Fig. 5. Improvement in routing success rate with fallback across Context Budgeting strategies. The strategies are ordered by most gain in performance.

TABLE II
PERFORMANCE IMPROVEMENT USING FALLBACKS ACROSS CONTEXT BUDGETING ACROSS

Variant	Performance Metrics (%)		
	Without Fallback	With Fallback	Performance Gain
Baseline	83.04	85.26	+2.22
Chain-of-Thought	81.93	85.98	+4.05
Context Compression	84.12	86.79	+2.67
Mini-Shot	80.79	86.67	+5.88
Retrieval-Augmented	74.25	80.22	**+5.97**
Two-Stage	78.04	82.87	+4.83
Zero-Shot	72.73	80.56	+7.83

percentage points. Context-lean strategies benefit most dramatically: Zero-Shot (+7.83 points), Retrieval-Augmented (+5.97 points), and Mini-Shot (+5.88 points) show that robust fallback logic compensates for limited primary contextualization.

Context-rich strategies demonstrate modest but consistent gains, with Compression (+2.67 points) and Baseline (+2.22 points) showing reduced fallback dependency. This inverse relationship indicates that strategies with stronger primary context achieve consistency through pathway optimization, while context-lean approaches rely more heavily on fallback recovery mechanisms.

D. Ranking Quality Assessment

Ranking performance measured via NDCG@5 demonstrates notable consistency across context budgeting variants, with scores ranging from 87.9% to 89% (Table I). This 0.011% variance represents stable performance across different contextualization approaches.

Chain-of-Thought Context leads ranking performance at 0.890 NDCG@5, followed closely by Baseline (88.7%) and Context Compression (88.5%). The competitive performance across different approaches suggests that ranking algorithms benefit from but are not critically dependent on upstream contextualization strategies.

The ranking consistency indicates that semantic embeddings capture sufficient information for effective result ordering regardless of upstream context processing. This finding suggests that ranking optimization efforts should focus on embedding quality and similarity calculation methods rather than extensive context engineering.

E. Recall Performance Analysis

Recall performance demonstrates varied effectiveness across context budgeting variants, ranging from 93.1% to 99.4% success rates. Context-rich strategies achieve optimal performance, with Compression, Baseline, and Chain-of-Thought all reaching 99.4% recall success, demonstrating comprehensive content coverage under well-contextualized conditions.

Moderate context strategies show manageable degradation: Two-Stage (97.7%) and Mini-Shot (97.3%) maintain acceptable production thresholds, while Retrieval-Augmented (96.4%) and Zero-Shot (93.1%) exhibit more significant challenges. The 6.3-percentage-point variance indicates that minimal context strategies require enhanced retrieval algorithms to maintain comprehensive coverage.

F. Overall Success Analysis

Context Compression emerges as the superior strategy across all pipeline stages, achieving 93.3% intent accuracy,

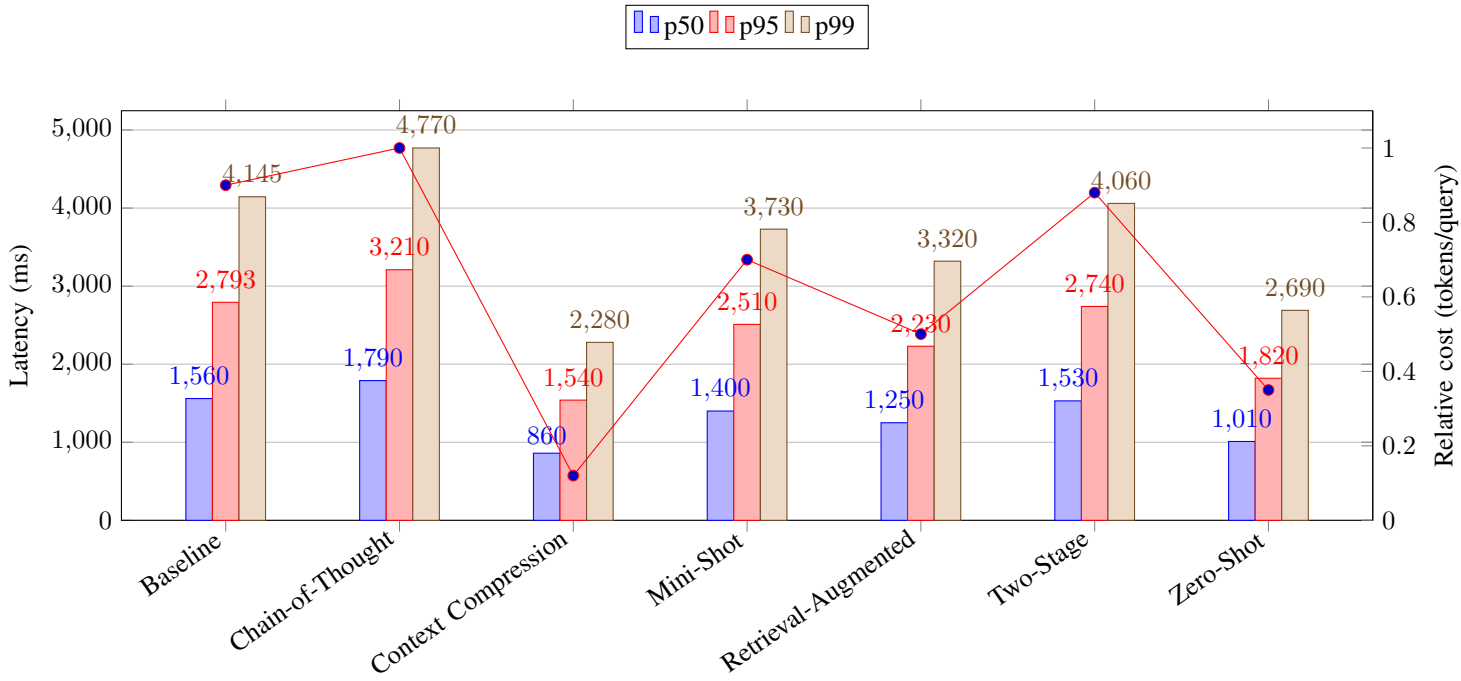

Fig. 6. Latency distribution (p50/p95/p99) with relative cost overlay across contextualization strategies.

86.8% routing success, 99.4% recall, and 88.5% NDCG@5. This approach leverages superagent context summarization to generate compact, semantically-rich representations that preserve critical information while reducing prompt complexity.

Figure 6 shows p50/p95/p99 end-to-end latency across strategies. Context Compression is fastest (p95 1.54 s), followed by Zero-Shot (1.82 s), Retrieval-Augmented (2.23 s), Mini-Shot (2.51 s), Two-Stage (2.74 s), Baseline (2.79 s), and Chain-of-Thought (3.21 s). The ordering aligns with context size: shorter prompts (compression/zero-shot) reduce serialization and model time; adding few-shots or reasoning increases tail latency. Tails widen most under Chain-of-Thought, reflecting added reasoning tokens.

We compute relative cost per strategy as proportional to total tokens processed by the GPT-4o-mini intent model, normalized to Chain-of-Thought = 1.00: $\text{rel\_cost}(s) = \frac{\text{tokens\_in}(s)+\text{tokens\_out}(s)}{\text{tokens\_in}(\text{CoT})+\text{tokens\_out}(\text{CoT})}$. Token counts include: (i) the fixed system instructions in SystemPrompt, (ii) strategy-specific few-shots from GPTShots, and (iii) conext from superagent (superagent prompt + GDR JSON). Output tokens are small and included for completeness. For reference, a 7-word output (9 tokens) costs 5–6 micro-dollars at $0.60 per 1M output tokens.

Given these results, we recommend Context Compression as the production strategy, as it consistently delivers optimal performance across all pipeline components while providing a scalable framework for context management in large-scale deployment scenarios.

V. Conclusion

This work presents the first comprehensive study of context budgeting strategies in production-scale multimodal search and recommendation systems, addressing a critical gap between research-oriented models and deployment-oriented constraints. Through the systematic evaluation of seven distinct context budgeting variants across 788 real-world templates and user queries, we demonstrate that context engineering decisions exert a decisive influence on end-to-end system performance, with their impact varying significantly across pipeline stages. Empirical results reveal that context-rich strategies, while incurring higher computational cost, yield substantial gains in query understanding and user experience, exceeding 50-point performance improvements over minimal-context baselines, thus justifying their adoption in production-grade, user-facing applications. These findings underscore that context budgeting cannot be treated as a local optimization problem; rather, it requires a holistic, pipeline-aware evaluation using production-relevant metrics and real user data. The proposed Performance Attribution Framework, together with the experimental insights, offers actionable guidance for practitioners seeking to optimize multimodal retrieval systems under real-world constraints, enabling principled trade-offs between computational efficiency and user experience fidelity.

Operationally, the cost overlay in Figure 6 (right axis) confirms that Context Compression minimizes token spend while preserving quality, delivering the strongest latency–cost trade-off; Chain-of-Thought provides the most reasoning headroom at the highest cost.

REFERENCES

[1] S. Yao *et al.*, "React: Synergizing reasoning and acting in language models," in *ICLR*, 2023.

[2] T. Schick *et al.*, "Toolformer: Language models can teach themselves to use tools," *arXiv:2302.04761*, 2023.

[3] N. Liu *et al.*, "Language model cascades," *arXiv:2309.03409*, 2023.

[4] Y. Shi *et al.*, "A learnable agent collaboration network framework for personalized multimodal ai search engine," in *Proceedings of the 2nd International Workshop on Deep Multimodal Generation and Retrieval*, ser. MMGR '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 12–20. [Online]. Available: https://doi.org/10.1145/3689091.3690087

[5] A. Seabra *et al.*, "Dynamic multi-agent orchestration and retrieval for multi-source question-answer systems using large language models," *arXiv preprint arXiv:2412.17964*, 2024. [Online]. Available: https://arxiv.org/abs/2412.17964

[6] Spurnow, "Openai chatgpt api pricing calculator," 2024, accessed: 2025-08-23. [Online]. Available: https://www.spurnow.com/en/tools/openai-chatgpt-api-pricing-calculator

[7] OpenAI, "Openai api pricing," 2024, accessed: 2025-08-23. [Online]. Available: https://openai.com/api/pricing

[8] AI Multiple Research, "Large vision models: Benchmarks and latency analysis," 2024, accessed: 2025-08-23. [Online]. Available: https://research.aimultiple.com/large-vision-models/

[9] Y. Jiang, Y. Li, R. Zhang *et al.*, "Fastvlm: Efficient vision-language model with hybrid encoder," *arXiv preprint arXiv:2412.13303*, 2024. [Online]. Available: https://arxiv.org/abs/2412.13303

[10] P. L. Liu *et al.*, "Hm-rag: Hierarchical multi-agent multimodal retrieval augmented generation," 2025. [Online]. Available: https://arxiv.org/abs/2504.12330

[11] P. Schmid, "The new skill in ai is not prompting, it's context engineering," 2025. [Online]. Available: https://www.philschmid.de/context-engineering

[12] W. Yan, "Don't build multi-agents," 2025, referenced in multiple sources on context engineering evolution. [Online]. Available: https://cognition.ai/blog/dont-build-multi-agents

[13] D. Chen *et al.*, "Mllm-as-a-judge: assessing multimodal llm-as-a-judge with vision-language benchmark," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.

[14] R. Mazumder *et al.*, "Failure risk analysis of pipelines using data-driven machine learning algorithms," *Structural Safety*, vol. 89, p. 102047, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167473020301259

[15] K. T. Uçar, "A comprehensive guide to error analysis in machine learning," *Medium*, 2024.

[16] A. Thomas *et al.*, "Improving retrieval with llm-as-a-judge," *Vespa Blog*, 2024.

[17] H. Li *et al.*, "Llms-as-judges: A comprehensive survey on llm-based evaluation methods," 2024. [Online]. Available: https://arxiv.org/abs/2412.05579