Route, Don't Guess: Adaptive Interaction Strategy Selection for Agentic Product Search

Vinesh Gudla[2,†], Xiao Xiao[1], Ji Xin[1] and Tejaswi Tenneti[2,†]

[1]*Instacart*
[2]*Ambience Healthcare*

Abstract

Agentic product search systems must decide when to act autonomously and when to engage the user—yet no principled framework exists for making this choice. We formalize the problem as *interaction strategy selection* and evaluate five agentic strategies on the Amazon ESCI dataset. Our experiments on 1,500 queries reveal that agentic intervention is *asymmetric*: on the hardest quartile of queries, result-aware strategies improve nDCG@10 from 0.069 to 0.271 (+0.202), while on easy queries they actively hurt (−0.060). This asymmetry motivates adaptive routing. We design a router using 25 observable features along with a novel agentic probe that measures result-set divergence after a cheap trial reformulation. This setup achieves comparable quality to the best fixed strategy while reducing token cost by 30%. We further show that LLM capability gates strategy effectiveness: strategies that fail with GPT-4o-mini succeed with GPT-4o and GPT-5.4. Our results establish that *not all queries should be treated equally*, and that lightweight routing can target intervention where it provides large gains while avoiding queries where it hurts. The central insight is that the effectiveness of agentic search lies not in always intervening, but in knowing when not to.

Keywords

agentic search, product retrieval, query reformulation, conversational commerce, interaction strategy selection

1. Introduction

Agentic interfaces are reshaping product discovery. Rather than returning static results, modern shopping assistants powered by large language models (LLMs) can reformulate vague queries, ask clarifying questions, and autonomously refine result sets before presenting them to the user [1, 2]. This shift raises a fundamental UX question: *when should the agent act autonomously, and when should it engage the user?*

The effects are asymmetric. On hard queries, i.e. misspellings, ambiguous terms, underspecified intents, agentic intervention can improve retrieval quality from near-zero to usable (+0.202 nDCG@10 in our experiments). On easy queries that are already well-served by dense retrieval, the same intervention *hurts* (−0.060) by introducing semantic drift. This asymmetry means that a fixed policy, always reformulate, or always ask is fundamentally suboptimal. Yet existing benchmarks for product search, including ESCI [3], WebShop [4], and ShoppingBench [5], evaluate task completion without addressing this interaction strategy question.

In this paper, we formalize the problem of *interaction strategy selection* for agentic product search. We define five strategies that span a spectrum from fully autonomous to user-engaging, and from result-blind to result-aware: (1) direct retrieval with no LLM involvement, (2) traditional reformulation, (3) clarification via a simulated user, (4) result-aware reformulation that conditions on observed retrieval failures, and (5) result-aware clarification that targets specific ambiguities surfaced by initial results. We investigate three research questions:

RQ1 How do different agentic interaction strategies compare in retrieval quality across query difficulty levels?

ECOM'26: SIGIR Workshop on eCommerce, Jul 24, 2026, Melbourne, Australia

[†]Work done while at Instacart.

vinesh.gudla@ambiencehealthcare.com (V. Gudla); xiao.xiao@instacart.com (X. Xiao); ji.xin@instacart.com (J. Xin); tejaswi.tenneti@ambiencehealthcare.com (T. Tenneti)

RQ2 Does LLM capability gate strategy effectiveness? Specifically, do strategies that fail with a smaller model (GPT-4o-mini) succeed with a more capable one (GPT-5.4)?

RQ3 Can observable retrieval signals including an agentic "probe" predict the best strategy per query, enabling adaptive routing?

Our contributions are threefold:

1. We show that agentic intervention is *asymmetric*: result-aware strategies provide large gains on hard queries but hurt easy ones, establishing that not all queries should be treated equally (Table 3).
2. We demonstrate that LLM capability gates strategy effectiveness across three models (GPT-4o-mini, GPT-4o, GPT-5.4), identifying a capability threshold below which agentic strategies degrade retrieval.
3. We design an adaptive router using 25+ features including a novel agentic probe that achieves comparable quality at 30% lower cost by intervening where it helps and avoiding it where it hurts.

2. Related Work

2.1. Product Search Benchmarks

The Amazon ESCI dataset [3] provides graded relevance judgments (Exact, Substitute, Complement, Irrelevant) for real shopping queries, making it a standard benchmark for product retrieval. WebShop [4] frames online shopping as a sequential decision-making task, while ShoppingBench [5] and WebMall [6] extend evaluation to multi-step agentic workflows. These benchmarks assess task completion specifically whether the agent finds the right product, but do not evaluate the *interaction strategy* the agent uses to get there. Our work addresses this gap by treating strategy selection itself as the object of study.

2.2. Agentic Search and Conversational Commerce

Production systems such as Amazon Rufus [1] and Google Shopping AI demonstrate the commercial relevance of agentic product search. On the research side, ProductAgent and PROCLARE [7] model multi-turn shopping dialogues with LLM-simulated users, while AgenticShop [2] proposes an end-to-end agentic shopping framework. However, these systems adopt fixed interaction patterns e.g., always reformulate, or always clarify without a principled mechanism for choosing among strategies. Our work provides such a mechanism via an adaptive router trained on observable retrieval signals.

2.3. Query Reformulation in eCommerce

Query rewriting is a core technique in product search. MAAQR [8] employs a multi-agent architecture for query rewriting, while MiniELM [9] distills LLM rewriting capability into compact models. These approaches treat reformulation as a monolithic operation: every query is rewritten regardless of whether it would benefit from clarification and none condition on the actual retrieval results. We show that result-aware reformulation substantially outperforms blind reformulation, and that clarification is preferable for certain query types (Section 5).

2.4. LLM Cascading and Routing

FrugalGPT [10] and RouteLLM [11] demonstrate that routing queries to different LLMs based on predicted difficulty can reduce cost without sacrificing quality. Our adaptive router applies a similar principle but operates at the *interaction strategy* level rather than the model level, and targets graded retrieval relevance (nDCG@10) rather than classification accuracy. To our best knowledge, this is the first application of routing to interaction strategy selection in product search.

3. Methodology

3.1. Problem Formulation

Given a user query q, our goal is to select an interaction strategy $s \in \mathcal{S}$ that maximizes retrieval quality as measured by nDCG@10. The strategy set $\mathcal{S} = \{s_1, \ldots, s_5\}$ spans five options described below. A *router* $r : \mathbb{R}^d \to \mathcal{S}$ maps a feature vector $\mathbf{x}(q) \in \mathbb{R}^d$, computed from observable signals of an initial retrieval pass, to the predicted best strategy. The router is trained to pick for each query, the strategy that achieves the highest nDCG@10.

3.2. Interaction Strategies

All strategies share a common retrieval backbone: queries (original or reformulated) are encoded with E5-large-v2 [12] and retrieved from a FAISS index over product embeddings. The five strategies differ in whether and how they involve the LLM and the user.

Strategy 1: Direct Retrieval (Direct). The query q is encoded and used to retrieve the top-k results directly. No LLM is invoked, making this the lowest-cost and lowest-latency strategy. It serves as the baseline and is optimal when the query is already precise and well-matched to the product vocabulary.

Strategy 2: Blind Reformulation (BlindReform). An LLM generates three reformulations of q without access to any retrieval results. e.g. "bat light" is reformulated to "bat-shaped lights", "bat-themed lights", "bat-shaped outdoor lights". Each reformulation is used to retrieve a separate result set, and the final ranking is produced by reciprocal rank fusion (RRF) [13] over the original and reformulated result lists. This strategy addresses vocabulary mismatch but cannot correct for systematic retrieval failures.

Strategy 3: Clarification (Clarify). The LLM examines metadata from the initial top-k results (titles, brands, categories) and generates a clarifying question targeting the most salient ambiguity. A simulated user (Section 3.3) responds with a preference grounded in the target product. The LLM then constructs a refined query incorporating the user's response, which is used for a second retrieval pass.

Strategy 4: Result-Aware Reformulation (ResultReform). The LLM receives the actual top-10 results from the initial retrieval and analyzes patterns of confusion—e.g., results spanning multiple product categories, or matching the query literally but missing user intent. It generates three targeted reformulations that address the observed failure modes. The final ranking is produced by RRF over the original and reformulated result lists.

Strategy 5: Result-Aware Clarification (ResultClarify). This strategy combines result awareness with user engagement. The LLM inspects the top-10 results, identifies a specific ambiguity (e.g., two plausible product categories), and asks a targeted clarifying question. After receiving the simulated user's response, the LLM generates result-aware reformulations conditioned on both the user's answer and the observed retrieval failures.

3.3. User Simulator

Strategies that involve clarification require a user to respond. Following prior work on simulated evaluation of conversational search [7], we implement a user simulator grounded in the ESCI relevance labels. For each query, the simulator has access to the Exact-match product (the gold-standard relevant item) and responds to clarifying questions by describing preferences that distinguish the target from alternatives—without naming the product directly. To ensure coherent and realistic responses, the simulator uses GPT-5.4, a more capable model than the agent LLM in the weaker configuration.

3.4. Routing Features

The router operates on a d-dimensional feature vector $\mathbf{x}(q)$ extracted from the initial direct retrieval pass. Features fall into three tiers of increasing novelty:

Standard retrieval signals capture score distribution (top-1 score, top-10 standard deviation, kurtosis), result-set diversity (brand count, brand entropy, title diversity), query surface properties (length, brand/color/number indicators), and query–result lexical overlap.

Semantic embedding signals measure the geometry of top-10 result embeddings: pairwise cosine coherence, embedding spread etc. Low coherence indicates the result set lacks a consistent topic.

Agentic probe signals are the key novelty. We issue a single cheap reformulation (one GPT-4o-mini call) and measure how much the result set shifts: Jaccard distance, result set overlap, mean rank displacement, and score shift. High divergence indicates the query is sensitive to reformulation and likely to benefit from deeper intervention.

3.5. Adaptive Router

We train the router as a multiclass classifier that predicts the best strategy for each query i.e. the strategy achieving the highest nDCG@10 in our offline evaluation. We evaluate two model families: logistic regression (with L2 regularization) and xgboost, selected for interpretability and low inference cost. Training and evaluation use 5-fold stratified cross-validation to prevent leakage. At inference time, the router's nDCG@10 for a query is the actual nDCG@10 achieved by the *predicted* strategy for that query, not a modeled estimate.

4. Experimental Setup

4.1. Dataset

We evaluate on the Amazon ESCI dataset [3], a large-scale product search benchmark containing 130K queries and 2.6M human relevance judgments across four graded labels: *Exact* (E=3), *Substitute* (S=2), *Complement* (C=1), and *Irrelevant* (I=0). We filter to the US-English large split and use the official test partition of 22,458 queries.

To ensure balanced coverage of query difficulty, we stratify queries by *label entropy* H computed over the judgment distribution for each query. We define three tiers: **clear** ($H < 0.5$), **moderate** ($0.5 \leq H < 1.0$), and **ambiguous** ($H \geq 1.0$), and sample 500 queries per tier, yielding a 1,500-query *development set* for strategy comparison and router training. We additionally sample a disjoint 501-query *held-out test set* (167 per tier) from the remaining queries for router generalization evaluation.

4.2. Models and Infrastructure

We encode all product titles and descriptions with `intfloat/e5-large-v2` (1024-dimensional embeddings) and index 360,873 products in a FAISS `IndexFlatIP` store for exact inner-product search.

We compare three models as agentic backbones: **GPT-4o-mini** (low cost), **GPT-4o** (mid-tier), and **GPT-5.4** (high capability). For user simulation in clarification strategies, we use GPT-5.4 to generate responses grounded in the set of *Exact*-match products for each query.

We additionally evaluate a cross-encoder baseline (`ms-marco-MiniLM-L-12-v2`) that re-scores the top-100 dense retrieval candidates without any LLM involvement.

4.3. Metrics

Our primary metric is **nDCG@10**; we also report nDCG@20 and Precision@10 (counting only *Exact* matches as relevant). We measure **token cost** (total prompt + completion tokens per query) and **wall-clock latency** (seconds). All pairwise strategy comparisons use the Wilcoxon signed-rank test; we report significance at $p < 0.05$ ($*$).

Table 1
Strategy comparison with GPT-5.4 (1,500 queries). Significance vs. Direct: $*$ p<0.05; ns– not significant.

Strategy	nDCG@10	Δ	Tokens	Latency (s)
Direct	0.492	—	0	0.04
HyDE-RRF	0.505*	+0.013	∼280	5.06
Blind Reform	0.508*	+0.016	96	1.39
Cross-Encoder Rerank	0.514*	+0.023	0†	0.20
Clarification	0.502ns	+0.010	500	3.06
Result-Aware Reform	0.523*	+0.031	377	1.42
Result-Aware Clarif	**0.541***	**+0.050**	1,148	4.92

4.4. Baselines and Strategies

We evaluate five agentic strategies alongside *Direct* retrieval as described in Section 3.2, plus HyDE-RRF [14] and cross-encoder reranking as non-agentic baselines.

5. Results

5.1. Strategy Comparison (RQ1)

Table 1 presents the main results with GPT-5.4 on 1,500 queries. All result-aware strategies significantly outperform Direct retrieval, whereas Clarification alone does not reach significance.

We also compare against HyDE [14] (hypothetical document retrieval; 0.505 for the RRF variant) and cross-encoder reranking († no LLM tokens; runs locally) which re-scores the top-100 dense retrieval candidates. The cross-encoder (0.514) outperforms Blind Reformulation (0.508), confirming that better ranking of existing candidates is a strong baseline. However, Result-Aware Reformulation (0.523) and Result-Aware Clarification (0.541) both exceed the reranker, demonstrating that LLM-driven strategies solve a complementary problem: finding *different* products via vocabulary expansion, not just reordering the same candidates.

Result-Aware Reformulation yields +0.031 nDCG@10 over Direct ($p < 0.05$) at only 1.4 s additional latency, making it the best cost–quality trade-off among LLM-based strategies. Result-Aware Clarification achieves the highest absolute quality (0.541) but at higher cost.

Breakdown by tier and query type: Result-Aware Clarification is the best strategy across all entropy tiers: clear (0.547), moderate (0.537), and ambiguous (0.539). However, entropy tiers measure intent ambiguity rather than retrieval difficulty. A more actionable stratification by query type reveals sharper contrasts: short generic queries (1–2 words, no brand, e.g., "gift for dad") benefit most from clarification (+0.050), while brand-generic queries (e.g., "nike shoes") are the only type where strategies hurt (−0.010) because the brand already informs retrieval. This contrast provides a clear routing signal: suppress intervention when query specificity is high.

5.2. Model Capability as Gating Factor (RQ2)

A striking finding is that LLM capability *gates* strategy effectiveness, with a clear threshold between non-functional and effective models. We test three LLMs as agentic backbones: **GPT-4o-mini** yields $\Delta = -0.005$ for Reformulation (ns); **GPT-4o** yields $\Delta = +0.015$ ($p = 0.016$); **GPT-5.4** yields $\Delta = +0.016$ ($p < 0.001$). The jump from mini to 4o is the critical threshold. Once the model can preserve query intent and recognize entities, further capability provides diminishing returns for blind reformulation.

From a qualitative viewpoint, GPT-4o-mini retains only 54% of original query terms in its reformulations (vs. 70% for GPT-5.4), frequently hallucinating incorrect brand names. GPT-5.4 recognizes brands (e.g., "eon" → "ENO"), corrects misspellings ("lippling" → "Kipling"), and generates structured clarifying questions. These results underscore that agentic search requires models above a capability threshold.

Table 2
Router performance on dev (1,500 queries, 5-fold CV) and held-out test (501 queries). The router achieves comparable quality to the best fixed strategy while reducing token cost by 30%.

	nDCG@10			
Configuration	**Dev**	**Holdout**	**Tok/q**	**Lat (s)**
Always Direct	0.492	0.496	0	0.04
Always Aware Ref	0.523	0.514	377	1.42
Always Aware Clar	0.541	0.547	1,148	4.92
Router (LR)	**0.547**	**0.550**	807	3.45
Oracle	0.618	0.619	—	—

5.3. Adaptive Router (RQ3)

We collapse the five strategies to the three Pareto-optimal ones: Direct, Result-Aware Reformulation, and Result-Aware Clarification since Blind Reform and plain Clarification are dominated on both quality and cost. We evaluated XGBoost, and MLP routers alongside logistic regression; all performed comparably at this sample size, so we report logistic regression for its interpretability. The router is trained on the 1,500-query development set and evaluated both via 5-fold cross-validation and on a held-out test set of 501 unseen queries (Table 2).

The router achieves comparable quality to the best fixed strategy on both dev (0.547) and held-out test (0.550 vs. 0.547) while reducing average token cost by 30% by routing 27% of queries to Direct (zero LLM cost). While a simple heuristic thresholding on top-1 retrieval score achieves similar routing performance, uniformly applying a cheaper strategy (Always Result-Aware Reform) yields substantially lower quality (0.514 vs. 0.550). This highlights that the key benefit of routing is not model complexity but selective allocation of expensive interventions: clarification is high-cost but high-reward on hard queries, and the router correctly avoids it on easy ones.

Probe feature ablation. Removing the five agentic probe features does not degrade holdout performance (0.550 with vs. without), indicating that score-based retrieval features already capture the core routing signal at this scale. We retain the probe as a conceptual contribution. It provides a principled, interpretable mechanism for estimating query sensitivity to intervention, complementing score-based confidence signals. Additionally its value may increase in harder routing settings (e.g., multi-step pipelines or heterogeneous retrieval backends) where score distributions are less informative.

Tail-query analysis. The overall +0.031 improvement in nDCG@10 masks a dramatic interaction with query difficulty. Stratifying by Direct retrieval quality reveals that agentic strategies matter most where they are most necessary:

On the hardest quartile i.e. queries where Direct nDCG@10 < 0.1, including misspellings (“longgines,” “brine coastas”), truncated queries (“cat front door ma”), and unusual phrasing (“iphone charger without electricity”). Result-Aware Clarification improves nDCG@10 to 0.271, a +0.202 absolute gain. On the easiest quartile, strategies hurt by −0.060 due to semantic drift. This asymmetry is the core motivation for routing: the router’s value lies not in improving average nDCG but in correctly identifying the 25% of queries where intervention provides large gains while avoiding the 25% where it hurts.

Cost–quality trade-off. Figure 1 plots each strategy as a point in (token cost, nDCG@10) space. Direct sits at the origin; Blind Reformulation offers moderate gains at minimal cost (~100 tokens); Result-Aware Reformulation provides the best Pareto trade-off; and Result-Aware Clarification achieves peak quality at the highest cost. The router traces a path along the Pareto frontier by mixing strategies per query.

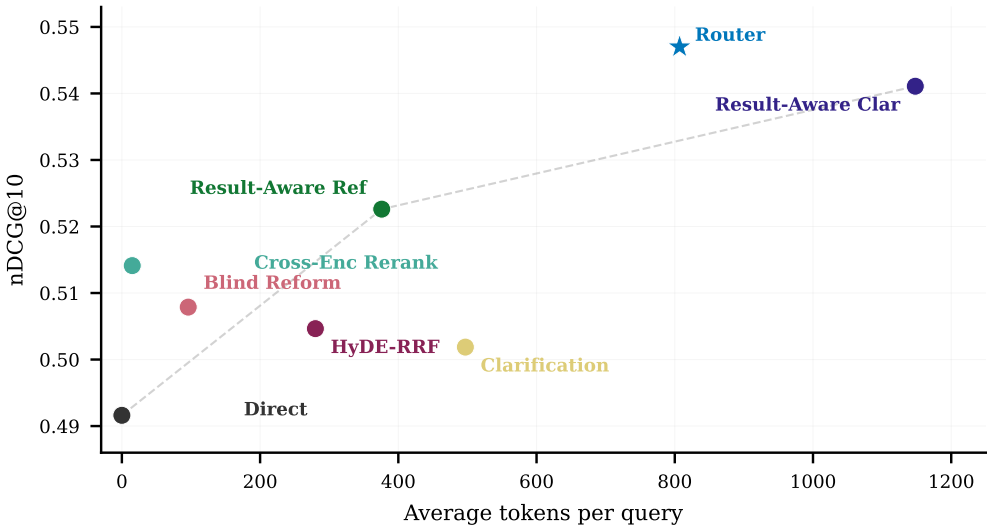

Figure 1: Cost–quality trade-off across strategies (GPT-5.4). Each point represents a fixed strategy; the dashed line traces the Pareto frontier.

Simulator sensitivity. A key validity concern is that the user simulator has access to the Exact-match product, making it unrealistically informative. To quantify this, we run Result-Aware Clarification on a 510-query subset with three simulator fidelity levels: grounded (full product access; current setup), vague (sees the product but responds with only a single general preference), and ungrounded (no product access; responds based on the query alone). All three levels significantly outperform Direct retrieval ($p < 0.001$): ungrounded achieves 0.507, vague 0.516, and grounded 0.534. The grounded simulator inflates the clarification advantage by +0.027 nDCG over the ungrounded setting. Crucially, Result-Aware Reformulation which requires no simulator achieves 0.512, within this range. The true production value of clarification strategies likely falls between the ungrounded and grounded bounds; closing this gap with real user studies is critical future work.

6. Discussion

Result-awareness is the key insight. Across all conditions, showing the LLM what the retriever returned before asking it to reformulate or clarify yields the largest quality improvement. This "look before you leap" pattern is essentially free in practice. Initial retrieval results are already in memory, yet it transforms the LLM from a blind query rewriter into an informed search assistant that can diagnose retrieval failures and target its intervention.

Model capability gates everything. Our GPT-4o-mini results serve as a cautionary finding: deploying agentic search with an insufficiently capable LLM can *degrade* performance. Practitioners should validate that their backbone model can faithfully preserve query intent, recognize entities, and generate well-structured reformulations before investing in agentic infrastructure.

Routing signals. The agentic probe features (rank displacement and score shift after a cheap trial reformulation) rank among the top routing signals, suggesting a practical two-stage architecture: probe cheaply, then decide whether to invest. However, categorical query-type features do not improve routing—continuous retrieval signals (score kurtosis, semantic coherence) already capture query broadness more precisely.

Implications for agentic UX. Our tail-query analysis (Table 3) has direct UX implications. Result-Aware Reformulation achieves 95% of clarification quality on moderate queries without interrupting

Table 3
Strategy gains by Direct nDCG@10 quartile. Strategies provide massive gains on hard queries (+0.202) but hurt easy ones (−0.060).

Quartile	n	Direct	Aware Ref	Aware Clarif
Q1 (hardest)	385	0.069	0.145	**0.271**
Q2	365	0.368	0.406	**0.443**
Q3	375	**0.636**	0.649	0.614
Q4 (easiest)	375	**0.901**	0.897	0.841

the user suggesting agentic systems should default to silent, result-informed action.

Clarification should be reserved for the hardest queries (Q1), where the +0.202 gain justifies the interaction cost. For easy queries (Q3–Q4), the agent should not intervene at all since routing strategies actively hurt. This "intervene only when necessary" principle directly supports the workshop theme of designing agentic experiences that build user trust.

Not all hard queries need the same fix. Our error analysis reveals distinct failure modes with different cost profiles. Misspellings ("longgines," "brine coastas") are recoverable by reformulation alone: a single LLM call at ~100 tokens. Genuine intent ambiguity ("gift for dad," "big glasses") benefits from clarification at ~1,100 tokens. A cost-aware router should distinguish these: cheap reformulation for correctable queries, expensive clarification only for truly ambiguous ones.

Reranking and reformulation are complementary. Cross-encoder reranking (0.514) improves over raw dense retrieval (0.492) by better ordering existing candidates, but result-aware strategies go further by surfacing *new* candidates through vocabulary expansion. On the hardest queries, the reranker and Result-Aware Reformulation achieve similar gains (~0.15), but Result-Aware Clarification (0.271) nearly doubles both—suggesting that the hardest queries require not just better ranking but fundamentally different retrieval paths. In production, combining a reranker with selective LLM intervention is likely optimal; we leave this integration to future work.

Limitations. Our simulator sensitivity analysis bounds the information advantage at +0.027 nDCG, but real user behavior may differ from even the ungrounded simulator in ways we cannot capture offline. ESCI's sparse judgment pool (~19 products per query) penalizes strategies that surface genuinely relevant but unjudged products, and our findings are conditioned on specific model versions that will evolve.

7. Conclusion

We presented a framework for systematically evaluating and routing agentic interaction strategies in product search. Our experiments on 1,500 development queries and 501 held-out test queries reveal three key findings: (1) result-aware strategies significantly outperform blind reformulation, with Result-Aware Reformulation—which requires no user interaction—achieving +0.031 nDCG@10 over direct retrieval ($p < 0.001$); (2) LLM capability is a hard prerequisite—agentic strategies degrade results when backed by weaker models; and (3) an adaptive router using 25 observable retrieval signals, including a novel agentic probe based on reformulation divergence, achieves comparable quality on both dev and held-out test while reducing token cost by 30%.

Future work will validate these findings with real user studies, explore richer probe signals (e.g., product-level features from probe results to capture *what* shifted, not just *how much*), and investigate integration with cross-encoder reranking and session-level personalization. Code and evaluation scripts are available for reproducibility.[1]

[1]https://github.com/vgudla/agentic-product-search

Declaration on Generative AI. During the preparation of this work, the authors used Claude Code (Anthropic) in order to correct grammar and refine the language of select sections of the manuscript text. It was also used to help generate the pareto frontier chart. The authors reviewed and edited all content and code as needed and accept full responsibility for the publication content.

References

[1] Amazon, Amazon rufus: A new generative AI-powered conversational shopping experience, https://www.aboutamazon.com/news/retail/amazon-rufus, 2024.

[2] S. Kim, R. Heo, Y. Seo, J. Yeo, D. Lee, AgenticShop: Benchmarking agentic product curation, in: Proceedings of the ACM Web Conference 2026, ACM, 2026.

[3] C. K. Reddy, L. Màrquez, F. Valero, N. Rao, H. Zaragoza, S. Bandyopadhyay, A. Biswas, A. Xing, K. Subbian, Shopping queries dataset: A large-scale ESCI benchmark for improving product search, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, ACM, 2022, pp. 4129–4139.

[4] S. Yao, H. Chen, J. Yang, K. Narasimhan, Webshop: Towards scalable real-world web shopping with grounded language agents, in: Advances in Neural Information Processing Systems, volume 35, 2022, pp. 20527–20541.

[5] Y. Jin, Z. Li, C. Zhang, T. Cao, Y. Gao, P. Jayarao, M. Li, et al., Shoppingbench: A massive multi-task online shopping benchmark for language models, arXiv preprint arXiv:2501.00745 (2025).

[6] R. Peeters, et al., Webmall - a multi-shop benchmark for evaluating web agents, arXiv preprint arXiv:2508.13024 (2025).

[7] J. Ye, Y. Jiang, X. Wang, Y. Li, et al., Productagent: Benchmarking conversational product search agent with asking clarification questions, arXiv preprint arXiv:2407.00942 (2025).

[8] Z. Qi, et al., MAAQR: An llm-based multi-agent framework for adaptive query rewriting in alipay search, in: Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2025.

[9] Y. Zhang, et al., MiniELM: Lightweight query rewriting with reinforcement learning, in: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics, 2025.

[10] L. Chen, M. Zaharia, J. Zou, FrugalGPT: How to use large language models while reducing cost and improving performance, arXiv preprint arXiv:2305.05176 (2023).

[11] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, I. Stoica, RouteLLM: Learning to route LLMs with preference data, arXiv preprint arXiv:2406.18665 (2024).

[12] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, F. Wei, Text embeddings by weakly-supervised contrastive pre-training, arXiv preprint arXiv:2212.03533 (2022).

[13] G. V. Cormack, C. L. A. Clarke, S. Buettcher, Reciprocal rank fusion outperforms Condorcet and individual rank learning methods, in: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2009, pp. 758–759.

[14] L. Gao, X. Ma, J. Lin, J. Callan, Precise zero-shot dense retrieval without relevance labels, arXiv preprint arXiv:2212.10496 (2023).

A. Prompt Templates

All prompts use the same structure: a system role setting the context, followed by the query and (where applicable) retrieval results. Variables in `{braces}` are filled at runtime.

Blind Reformulation (Strategy 2).

```
You are an ecommerce search agent. The user searched for: "{query}"
Generate {n} reformulated search queries that better capture the user's likely intent.
```

```
Consider specific product attributes, common interpretations, and related categories.
Output ONLY the {n} queries, one per line. No numbering or explanation.
```

Clarification Question (Strategy 3).

```
You are an ecommerce shopping assistant. The user searched for: "{query}"

The top results include products from these brands: {brands}
Product types include: {sample_titles}

Generate ONE clarifying question that would most help you find exactly what the user
wants. Be specific and offer 2-3 concrete options. Keep it conversational and brief.
```

Result-Aware Reformulation (Strategy 4).

```
You are an ecommerce search agent. The user searched for: "{query}"

Here are the top search results returned:
{result_list}

The results span these brands: {brands}

Based on what you see in the results, identify the most likely user intent and generate
{n} refined search queries that would surface the most relevant products. If the results
seem scattered across different product types, focus on the most probable interpretation.
Output ONLY the {n} queries, one per line. No numbering or explanation.
```

Result-Aware Clarification Question (Strategy 5).

```
You are an ecommerce shopping assistant. The user searched for: "{query}"

Here are the top search results returned:
{result_list}

Look at these results carefully. Identify the key ambiguity or confusion. Are the
results split across different product types, use cases, brands, or price tiers?

Based on the SPECIFIC confusion you see in the results, generate ONE clarifying question
that would most help you determine what the user actually wants. Offer 2-3 concrete
options drawn from what you see in the results. Keep it conversational and brief.
```

User Simulator (Grounded).

```
You are a shopper who searched for "{query}".
You are actually looking for a product like: "{target_title}"
(Description: {target_desc})

The shopping assistant asked you: "{clarifying_question}"

Respond naturally as a shopper briefly, 1-2 sentences.
Do NOT mention the exact product name. Describe your preferences in terms of features,
use case, or constraints.
```