BEYOND SEMANTIC MATCHING: EVIDENCE-GROUNDED MULTI-AGENT PRODUCT DEEP SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern product search excels at semantic matching, but realistic shopping queries often demand complex, multi-condition requirements—spanning appearance, specifications, compatibility, price, and user reviews—that holistic relevance scores alone cannot reliably guarantee. We introduce **Product Deep Search**, a benchmark and paradigm for evidence-grounded product retrieval. Candidate products are represented through structured product-page evidence, and models use tools to retrieve supporting or contradictory evidence across heterogeneous sources. To solve this problem, we propose **PDeepSearch**, a multi-agent framework that turns semantic search into evidence-grounded decision making. Starting from candidates produced by multimodal retrieval and reranking, PInspect independently examines each product with evidence-access tools and constructs condition-level profiles of support, uncertainty, and contradiction; PDecide then compares these profiles across candidates to make the final selection. The framework can be instantiated with both API-based and open-weight models, and consistently improves strong zero-shot backbones by 2.43–7.29 Hit@1 points. We further propose **PDeepSearch-9B**, a product search agent trained by synthesizing inspection and decision trajectories. PDeepSearch-9B achieves 69.43 Hit@1, outperforming larger open-weight models, including Qwen3.5-27B (63.71) and Gemma-3-27B-IT (62.86), while approaching competitive top-tier models like Gemini-3.5-Flash and Kimi-K2.6. Further analysis shows that evidence verification acts as an effective correction layer over semantic reranking, with the largest absolute gains observed on queries combining a reference image with visual, detail, and review-grounded conditions. These results highlight evidence-grounded verification as a key capability beyond semantic matching for complex product search, and show that it can be effectively learned by compact models.

1 INTRODUCTION

Modern product search relies on a two-stage paradigm—embedding retrieval followed by multimodal reranking—to identify semantically relevant candidates from large-scale catalogs (Reddy et al., 2022; Hou et al., 2026; Chen et al., 2026). Advances in dense representation learning and vision-language models have substantially improved query–product matching across textual and visual modalities (Radford et al., 2021; Lin et al., 2025; Li et al., 2026b). However, holistic semantic relevance does not guarantee that a product satisfies a user's complex shopping requirements. Realistic queries often combine multiple explicit conditions spanning product type, visual appearance, technical specifications, compatibility, price, and usage experience (Yao et al., 2022; Lyu et al., 2025; Wu et al., 2024). Standard rankers frequently retrieve globally similar candidates while overlooking specific, decisive constraints (Abdin et al., 2024). For example, a user searching for a DC converter with a 40V-to-12V requirement, easy wiring, and good thermal performance may receive candidates that match visually but violate precise voltage specs or thermal limits hidden in user reviews (Yang et al., 2022; Wang et al., 2022; Zou et al., 2024). Therefore, **semantic relevance is necessary, but not sufficient for complex product retrieval**: reliable selection additionally requires actively locating and verifying requirement-specific evidence.

To study this gap, we introduce **Product Deep Search**, a multimodal, multi-condition benchmark for evidence-grounded product retrieval. Rather than relying solely on representation matching, this benchmark requires models to explicitly verify user requirements. Queries contain multiple

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

textual conditions alongside an optional reference image, where the image expresses a soft visual preference secondary to explicit textual constraints. Candidate products are presented as rendered webpages containing images, descriptions, specifications, prices, and reviews. Instead of receiving pre-extracted snippets, models access webpage content through dedicated evidence-access tools to retrieve supporting or contradictory evidence before making a final decision.

Building on this setting, we propose **PDeepSearch**, a multi-agent framework that augments semantic search with explicit evidence verification. Following a coarse-to-fine paradigm, retrieval and reranking first establish a strong semantic prior. For each top candidate, **PInspect** independently examines the corresponding product webpage through evidence-access tools and constructs a structured condition-level evidence profile containing supporting, uncertain, and contradictory evidence. **PDecide** then compares these profiles across candidates to make the final selection. This decomposition separates efficient global matching from fine-grained verification while reducing the influence of candidate ranking priors during inspection. By making final decisions based on explicitly verified evidence rather than directly relying on reranker scores, PDeepSearch serves as an evidence-grounded refinement over semantic ranking.

Importantly, PDeepSearch can be instantiated with different model backbones. Across five zero-shot models of different scales and families, applying the PDeepSearch workflow consistently improves performance over their corresponding base settings, yielding gains of 2.43–7.29 Hit@1 points. These results demonstrate that the proposed inspection-and-decision framework generalizes across both API-based and open-weight models. We further investigate whether these evidence-seeking and decision-making capabilities can be learned by a compact model. To this end, we propose **PDeepSearch-9B**, a product search agent trained from synthesized inspection and decision trajectories. We generate structured multi-agent trajectories covering tool-based inspection, condition-level evidence collection, and cross-candidate comparison, and use them to fine-tune a Qwen3.5-9B backbone to perform the PInspect and PDecide roles. PDeepSearch-9B substantially improves over its zero-shot Qwen3.5-9B counterpart and outperforms larger open-weight models, including Qwen3.5-27B and Gemma-3-27B-IT, while approaching stronger models such as Gemini-3.5-Flash and Kimi-K2.6. Our subsequent analyses further examine candidate coverage, correction behavior, and performance across query compositions.

Our contributions are three-fold:

- We introduce **Product Deep Search**, a multimodal, multi-condition benchmark requiring models to verify complex shopping requirements using evidence from rendered product webpages.
- We propose **PDeepSearch**, a multi-agent framework that augments semantic retrieval and reranking with tool-based candidate inspection, condition-level verification, and evidence-grounded cross-candidate decision making. The framework consistently improves five zero-shot backbones across different model scales and families.
- We propose **PDeepSearch-9B**, a compact product search agent trained from synthesized inspection and decision trajectories. PDeepSearch-9B outperforms substantially larger open-weight models while approaching stronger API-based models on the Product Deep Search benchmark.

2 Related Work

Constraint-aware Product Retrieval and Multimodal Search. Product search has evolved from lexical matching to dense semantic retrieval and item-aware representation learning (Reddy et al., 2022; Hou et al., 2026). BLaIR studies complex product search from long natural-language contexts, while multimodal retrievers encode text, images, and interleaved inputs in shared spaces (Hou et al., 2026; Lin et al., 2025). Recent benchmarks further incorporate structured constraints: STaRK combines textual properties with relational constraints (Wu et al., 2024), MERIT studies multilingual interleaved multi-condition queries (Chow et al., 2026), and composed image retrieval combines reference images with textual modifications (Wu et al., 2021; Liu et al., 2021; Baldrati et al., 2023). These methods improve item-level relevance estimation but generally rely on holistic similarity or relevance scores. In contrast, we treat retrieval and reranking as semantic priors and perform explicit

condition-level verification over heterogeneous product evidence, with reference images serving only as soft visual preferences.

Multi-source Product Evidence and Verification. Prior work extracts product attributes from heterogeneous sources. MAVE models multi-source attribute extraction from product profiles (Yang et al., 2022), SMARTAVE jointly uses structured text and images (Wang et al., 2022), and ImplicitAVE studies attributes not explicitly stated in a single field (Zou et al., 2024). Explainable recommendation further uses attributes and reviews to generate rationales (Zhang & Chen, 2020; Xie et al., 2023). However, these methods primarily extract attributes or explain recommendations rather than verify each user requirement before selection. Our formulation is also related to evidence-based claim verification and fine-grained factuality evaluation, which retrieve evidence and assess atomic claims as supported, refuted, or unsupported (Thorne et al., 2018; Jiang et al., 2020; Min et al., 2023). We transfer this perspective to product retrieval: query conditions are the atomic units, heterogeneous product content provides evidence, and support, contradiction, and uncertainty directly inform candidate selection.

Agentic Shopping and Product Research. LLM-based shopping agents have been explored for multi-step product discovery and task completion (Yao et al., 2022; Lyu et al., 2025; Wang et al., 2026). Existing agents, however, mainly emphasize browsing and open-ended interaction rather than fine-grained evidence verification tied to ranking decisions. Our framework instead converts semantically plausible candidate sets into verifiable selections through tool-assisted, condition-level inspection and cross-candidate evidence comparison.

3 BENCHMARK: EVIDENCE-GROUNDED PRODUCT DEEP SEARCH

We introduce an evidence-grounded product search benchmark in which relevance is determined by the joint satisfaction of multiple shopping requirements. Many existing product and multimodal retrieval benchmarks emphasize query–product relevance or semantic and visual similarity (Reddy et al., 2022). In contrast, our benchmark grounds query-specific requirements in different combinations of product images, details, prices, and user reviews. This setting captures cases where a product appears semantically or visually relevant but fails one or more user-specified requirements. A central challenge is label ambiguity: a query generated from one product may also describe other valid products. We therefore validate each generated query against highly ranked competing products and discard ambiguous query instances.

3.1 TASK DEFINITION

Let $\mathcal{P} = \{p_i\}_{i=1}^{N}$ denote the product corpus, where each product is associated with product images, details, price, and user reviews. Given a query $q = (q_t, q_r)$, where q_t is a natural-language shopping request and q_r is an optional reference image, the task is to retrieve a product $\hat{p} \in \mathcal{P}$ whose available evidence supports all query-specified requirements. Each query contains a set of conditions $\mathcal{C}(q) = \{c_1, \ldots, c_M\}$. These conditions may be supported by different combinations of visual, detail, price, and review evidence; not every query involves all evidence sources. A product is considered valid only when its available evidence supports every condition specified by the query. The benchmark therefore evaluates *joint requirement satisfaction* rather than holistic semantic similarity alone.

3.2 BENCHMARK AND TRAINING DATA CONSTRUCTION

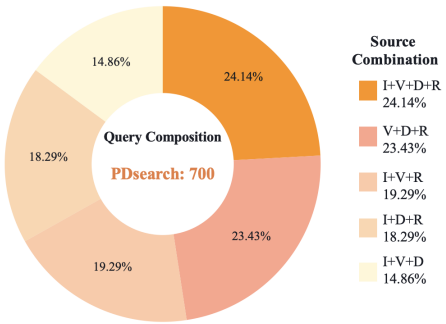
Product corpora. We construct our data from Amazon Reviews 2023 (Hou et al., 2024), using product records with metadata, images, prices, and user reviews. Every retained product is required to contain at least one valid image, a parseable price, non-empty product-detail information, and more than 20 reviews. We additionally perform cross-ASIN duplicate removal to eliminate duplicate listings of the same underlying product. For benchmark evaluation, we construct an approximately 80K-product corpus through stratified random sampling over fine-grained product categories using a fixed random seed. Separately, after excluding all benchmark products and their cross-ASIN duplicates, we randomly sample an additional approximately 120K products for training. The benchmark and training corpora are therefore product-disjoint.

Multi-source query generation. For each candidate positive product p^+, we use Gemini-3.1-pro to generate a natural-language shopping request from its product images, details, price, and reviews. The resulting conditions capture visual appearance, product details, price constraints, and review-grounded usage properties. During construction, we retain the evidence provenance of each condition for verification, while these source labels are hidden from evaluated models. To reduce trivial source-product reconstruction, the generation prompt excludes ASINs, exact product titles, model identifiers, and other identifying catalog strings. We retain five high-specificity query compositions based on four core components: a reference image (**I**), an explicit visual condition (**V**), a detail condition (**D**), and a review-grounded condition (**R**). Price constraints may additionally appear in a query but are not used to define these composition groups. The same query-construction procedure is applied independently to the benchmark and training corpora. The resulting training queries form the source data for the training procedure described in Section 4.3.

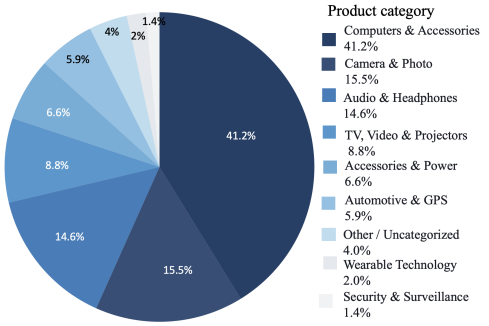
Reference image construction. For queries containing a reference-image component, we use a Gemini multimodal embedding model to retrieve products visually similar to p^+. This embedding model is used only for dataset construction and is not included among the retrieval models evaluated in our experiments. All images associated with the positive ASIN are excluded, and cross-ASIN duplicates have already been removed during corpus construction. The highest-ranked remaining image is selected as the reference image q_r, ensuring that it comes from a distinct product rather than another gallery image or duplicate listing of the positive. The reference image represents a *soft visual preference* over style, shape, structure, and form factor rather than an exact target image. When visual cues conflict with explicit textual requirements, textual requirements take precedence.

3.3 Query Validation and Uniqueness Control

Generating a query from p^+ does not guarantee that it is the only valid answer, which can make single-positive evaluation unreliable. To separate benchmark curation from the retrievers evaluated later, we use the Gemini multimodal embedding model exclusively for dataset construction. Given the complete multimodal query, we retrieve the top-30 most relevant products from the corresponding corpus. We then use GPT-5.6 Sol to evaluate p^+ and all retrieved candidates against every condition specified in the query. A query is retained only when the positive satisfies all required conditions and every competing candidate fails at least one. If another retrieved product also satisfies the complete query, we discard the corresponding *query instance*; all products remain in the fixed corpus. We refer to this procedure as *retrieval-based uniqueness verification*. It reduces single-positive ambiguity by explicitly validating highly ranked query-conditioned alternatives. Since verification is restricted to the retrieved neighborhood rather than all products in the corpus, it should be viewed as an ambiguity-reduction procedure rather than a formal guarantee of corpus-wide uniqueness. The same validation procedure is applied to both benchmark and training queries. We additionally conduct a stratified human audit of the final benchmark to assess annotation quality and residual ambiguity, with the detailed protocol and results reported in Appendix A.2.



(a) Query composition of the 700-query benchmark. **I**: reference image; **V**: visual condition; **D**: detail; **R**: review-grounded condition.



(b) Product category distribution of the approximately 80K-product benchmark corpus.

Figure 1: Characteristics of the Product Deep Search benchmark. The benchmark covers diverse query compositions and product categories.

3.4 Benchmark Statistics and Characteristics

After corpus filtering, multi-source query generation, and uniqueness verification, the final benchmark contains 700 verified queries over an approximately 80K-product corpus. The retained queries span diverse query compositions and cover a broad range of product categories. Figure 1 summarizes these characteristics. The left panel shows the distribution of query compositions, while the right panel presents the category distribution of the benchmark corpus. Together, multi-source query construction and ambiguity-aware validation provide a controlled benchmark for evidence-grounded product search. Evaluation protocols and metrics are described in Section 5.1.

4 Multi-Agent Product Deep Search

We propose **PDeepSearch**, a multi-agent framework that combines efficient semantic retrieval with evidence-grounded product verification. Our method follows a coarse-to-fine design: embedding retrieval and multimodal reranking first narrow the search space to a small set of plausible candidates, after which independent Inspector Agents examine product-level evidence and a Decision Agent performs the final cross-candidate comparison. This design avoids applying expensive evidence reasoning over the full product corpus while retaining a strong semantic prior from the retrieval stage. The overall framework is illustrated in Fig. 2. Given a textual shopping request and an optional reference image, the system first retrieves a Top-K candidate pool from the product corpus. Each candidate is then inspected independently against the query requirements using available evidence from product images, descriptions, specifications, prices, and reviews. The resulting structured evidence profile records condition-level support, uncertainty, and contradiction, providing a more explicit basis for comparison than a single holistic relevance score. The Decision Agent subsequently compares these profiles across candidates and selects the product that is best supported by the available evidence under the user's requirements. In this workflow, the components serve complementary roles: retrieval provides corpus-level coverage, reranking provides a strong semantic prior, Inspector Agents perform localized evidence verification, and the Decision Agent resolves the final choice through cross-candidate reasoning over the collected evidence.

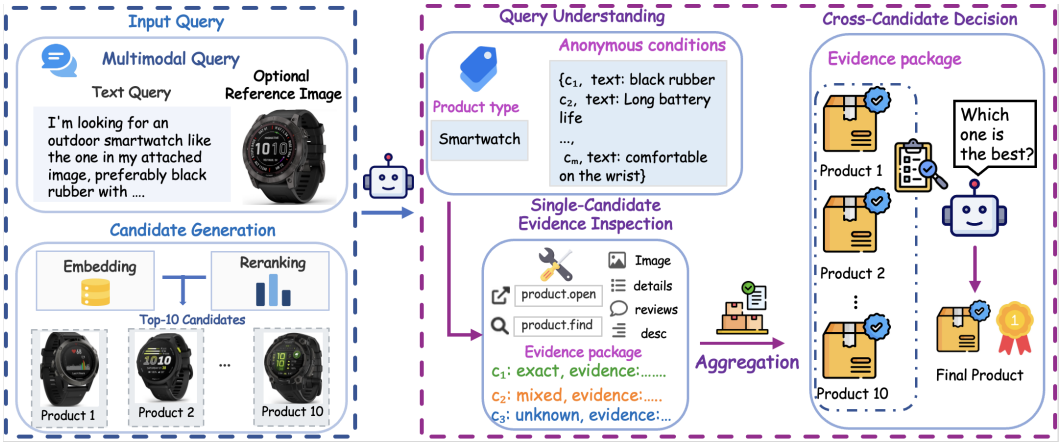


Figure 2: Overview of **PDeepSearch**. Given a textual shopping query and an optional reference image, embedding retrieval and multimodal reranking first construct a Top-K candidate pool. Independent Inspector Agents then verify each candidate against the query requirements using evidence from product images, descriptions, specifications, prices, and reviews. The resulting evidence profiles are jointly compared by a Decision Agent to select the final product.

4.1 Query Understanding and Candidate Generation

Query Understanding. We first transform the textual request into a target product type t and a set of atomic natural-language conditions $\mathcal{Q} = \{c_1, \ldots, c_M\}$. The exact structured format is provided in Appendix B.1. Query understanding is performed by the same backbone model used by each system in a zero-shot manner (e.g., Gemini-3.5-Flash for Gemini-based systems and Qwen3.5-9B

for PDeepSearch-9B), without access to benchmark construction metadata or evidence provenance annotations. A key design choice is that these conditions are source-anonymous: we do not indicate whether a condition should be verified from specifications, descriptions, reviews, prices, or visual content. The optional reference image is retained as a soft visual preference rather than converted into a hard textual condition, and explicit textual requirements take precedence when conflicts arise.

Candidate Generation. Applying evidence inspection to the entire product corpus is computationally expensive. We therefore use a coarse-to-fine procedure to identify a small set of plausible candidates. Embedding retrieval first returns $N = 30$ products, and a multimodal reranker selects the top $K_{\mathrm{I}} = 10$:

$$\mathcal{C}_R = \mathrm{Top}_{N,\, p\in\mathcal{P}}\, s_{\mathrm{emb}}(q,p), \qquad \mathcal{C}_I = \mathrm{Top}_{K_{\mathrm{I}},\, p\in\mathcal{C}_R}\, s_{\mathrm{rank}}(q,p). \tag{1}$$

Here, s_{emb} and s_{rank} denote the embedding and reranking scores, respectively. Retrieval provides corpus-level coverage, while reranking concentrates subsequent inspection on semantically relevant products. Inclusion in $\mathcal{C}_I$ does not establish that a candidate satisfies every query condition; this is assessed through the evidence inspection stage. Reranker scores and explicit ranking positions are not included in the downstream agents' inputs.

4.2 Candidate-Level Evidence Inspection

Semantically similar products may differ on a decisive specification, visual attribute, or review-grounded property. The inspection stage therefore examines each candidate against the individual query conditions. Each candidate in $\mathcal{C}_I$ is assigned an independent Inspector Agent, which receives the original query, decomposed textual conditions, optional reference image, candidate image, and basic product information. Two read-only tools support evidence acquisition: `open` retrieves product information, and `find` searches for evidence relevant to one or more conditions. Both operate only on the bound candidate within the fixed corpus. Because the conditions do not specify their evidence sources, the Inspector must determine what information to seek and whether it supports or contradicts each requirement. Inspectors cannot observe other candidates or their reranker scores and can run in parallel. This separation allows candidate-local verification to precede cross-candidate comparison.

For candidate p_i and condition c_j, the Inspector produces

$$e_{ij} = \left(s_{ij}, E_{ij}^{+}, E_{ij}^{-}\right), \tag{2}$$

where s_{ij} is the verification status, and E_{ij}^{+} and E_{ij}^{-} contain supporting and contradictory evidence, respectively. We use four statuses: `exact_match` for clear support, `mixed` for partial support or conflicting evidence, `unknown` for insufficient evidence, and `contradicted` for explicit conflict. Separating uncertainty from contradiction avoids treating missing information as evidence of a violation, while `mixed` preserves conflicts that require further consideration during decision making. Together, these records form the structured evidence profile E_i.

All ten candidates and their profiles are passed to the Decision Agent:

$$\mathcal{C}_D = \mathcal{C}_I, \qquad |\mathcal{C}_D| = 10. \tag{3}$$

Inspectors do not assign aggregate scores, rank candidates, or filter them. The Decision Agent jointly compares condition-level support, uncertainty, and contradictions to select the product best supported by the available evidence. Explicit textual requirements take precedence over the reference image, which remains a soft visual preference. This division separates evidence collection from final selection while preserving the condition-level information needed for comparison. Appendix B.3 provides the Inspector interfaces and evidence-profile format.

4.3 Trajectory Synthesis and Training

Learning the workflow requires supervision for both candidate-level evidence acquisition and cross-candidate decision making. Using the training queries and candidate sets constructed in Section 3.2, we execute the PDeepSearch pipeline with Gemini-3.5-Flash as the teacher. For each instance, the pipeline retrieves and reranks products, inspects the ten selected candidates, packages their evidence, and produces a final decision. After quality filtering, we retain approximately 5K PDeepSearch

trajectories for supervised fine-tuning of the Qwen3.5-9B multimodal backbone. The trajectories provide complementary supervision for the two agent roles. Inspector trajectories teach the model to interpret source-anonymous conditions, formulate evidence-access requests, and distinguish support, uncertainty, and contradiction in the returned content. Decision trajectories teach the model to compare the resulting profiles across candidates and select a product based on the complete set of requirements. Together, they supervise the intermediate evidence-processing steps as well as the final selection. We implement supervised fine-tuning using LLaMA-Factory. At inference time, the same fine-tuned model serves as both Inspector and Decision Agent through role-specific prompts, following the workflow described above.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmark and metrics. We evaluate all methods on the 700-query benchmark introduced in Section 3, using a fixed query set and product corpus. For retrieval and reranking, we report Hit@1, NDCG@10, NDCG@30, Recall@10, and Recall@30. For agentic methods, we report exact-match Hit@1 and additionally evaluate PDeepSearch with Judge@1 using Claude Opus 4.5, which assesses whether the selected product satisfies the query based on the available multimodal evidence.

Evaluation protocol. For retrieval, Qwen3-VL-Embedding-8B Li et al. (2026a), VLM2Vec-v2 Meng et al. (2025), LLaVE Lan et al. (2025), and CORAL Chow et al. (2026) independently search the full corpus and return the top-30 products. For reranking, we use the same top-30 pool retrieved by Qwen3-VL-Embedding-8B and compare Qwen3-VL-Reranker-8B Li et al. (2026a), MonoQwen2-VL-v0.1 Chaffin & Lac (2024), LamRA-Rank Liu et al. (2025), and Lychee-rerank-mm Zhang et al. (2026). Retrieval and reranking use the complete query and product content. For agentic evaluation, we fix the upstream pipeline to Qwen3-VL-Embedding-8B followed by Qwen3-VL-Reranker-8B. All agents receive the same reranked top-10 candidates and cannot access products outside this set.

Agentic settings. We evaluate Qwen3.5-9B and Qwen3.5-27B Team (2026), Gemma-3-27B-IT Team (2025), Gemini-3.5-Flash Google DeepMind (2026), Kimi-K2.6 Moonshot AI (2026), and PDeepSearch-9B, our trajectory-trained Qwen3.5-9B model. For each zero-shot backbone, we compare two inference procedures. *Base* uses a single unstructured agent that decides which candidates to inspect and which product to select. *PDeepSearch* independently inspects all ten candidates and passes their condition-level evidence profiles to a Decision Agent. Inspectors do not assign aggregate scores, rank candidates, or remove candidates. Both procedures use the same query, candidate pool, product evidence, and `open`/`find` tools. Tool calls and inference tokens are not budget-matched; comparisons therefore measure end-to-end performance under a shared candidate and evidence interface. Implementation and evaluation details are provided in Appendices B and C.

5.2 RETRIEVAL AND RERANKING

Table 1: Retrieval and reranking results. All rerankers operate on the same top-30 pool from Qwen3-VL-Embedding-8B. Best results within each stage are bolded, except for Recall@30, which is identical across rerankers.

Stage	Backbone	Hit@1	NDCG@10	NDCG@30	Recall@10	Recall@30
Retrieval	VLM2Vec-v2	6.57	13.57	17.24	22.14	37.57
	LLaVE	18.43	30.52	33.33	43.00	54.71
	CORAL	20.14	34.61	38.92	51.86	69.86
	Qwen3-VL-Emb-8B	**24.86**	**44.09**	**49.13**	**66.14**	**87.14**
Reranking	MonoQwen2-VL-v0.1	35.43	51.03	53.70	76.00	87.14
	LamRA-Rank	51.00	67.05	67.73	84.29	87.14
	Lychee-rerank-mm	53.86	69.46	69.73	86.00	87.14
	Qwen3-VL-Reranker-8B	**61.43**	**72.41**	**72.55**	**86.57**	87.14

Table 2: Agentic results on the same reranked top-10 candidates. Base and PDeepSearch columns report Hit@1; Judge@1 evaluates PDeepSearch predictions. Inference budgets are not matched.

Backbone	Base	PDeepSearch	Judge@1
Qwen3.5-9B	39.57	42.00	42.86
Qwen3.5-27B	57.86	63.71	63.14
Gemma-3-27B-IT	55.57	62.86	62.57
Gemini-3.5-Flash	69.71	**75.71**	**75.57**
Kimi-K2.6	64.57	71.00	70.43
PDeepSearch-9B	–	69.43	69.14

Table 1 compares the upstream methods. Qwen3-VL-Embedding-8B achieves the strongest retrieval results, with 24.86 Hit@1 and 87.14 Recall@30. All rerankers reorder its fixed top-30 pool, so their Recall@30 remains unchanged. Qwen3-VL-Reranker-8B achieves the highest Hit@1 of 61.43 and Recall@10 of 86.57. We use this retriever–reranker pair for all agentic experiments. Its top-10 recall sets an exact-match Hit@1 upper bound of 86.57 for the downstream agents, leaving substantial room to improve selection over the reranker's top-1 prediction.

5.3 Main Results

Table 2 reports the agentic results. PDeepSearch improves over Base across all five zero-shot backbones by 2.43–7.29 Hit@1 points. Gemini-3.5-Flash achieves the best result at 75.71. These gains support the effectiveness of the complete procedure across model families, but do not isolate the effects of workflow design from additional inference computation. Trajectory training improves the Qwen3.5-9B backbone from 42.00 to 69.43 Hit@1 under the same PDeepSearch workflow. The trained model also exceeds zero-shot Qwen3.5-27B (63.71) and Gemma-3-27B-IT (62.86). These results demonstrate the value of task-specific supervision on this benchmark, rather than general superiority over larger models.

5.4 Analysis

We examine three aspects of PDeepSearch beyond aggregate accuracy: whether it corrects reranker errors while preserving correct predictions, what limits further improvement, and which query compositions benefit most. Here, Gemini refers to Gemini-3.5-Flash using PDeepSearch.

Selective correction. An effective refinement stage should recover missed targets without frequently overturning correct predictions. Figure 3 examines this trade-off relative to Qwen3-VL-Reranker-8B. PDeepSearch-9B corrects 86 of the reranker's 270 errors while changing 30 correct predictions to incorrect ones, preserving 93.02% of its 430 correct predictions. Gemini corrects 120 errors and introduces 20 incorrect overrides, preserving 95.35%. Successful corrections therefore outnumber harmful overrides by approximately 2.9:1 and 6.0:1, respectively, yielding net gains of 56 and 100 correct predictions. These results indicate that PDeepSearch improves selection while retaining most of the reranker's correct outcomes. Gemini achieves both more corrections and fewer harmful overrides, suggesting that stronger reasoning within the same workflow improves the balance between revising incorrect choices and preserving correct ones. The remaining overrides also show that additional inspection does not guarantee a better decision: evidence may be incomplete, misinterpreted, or incorrectly weighed during comparison. Distinguishing these causes requires trajectory-level analysis.

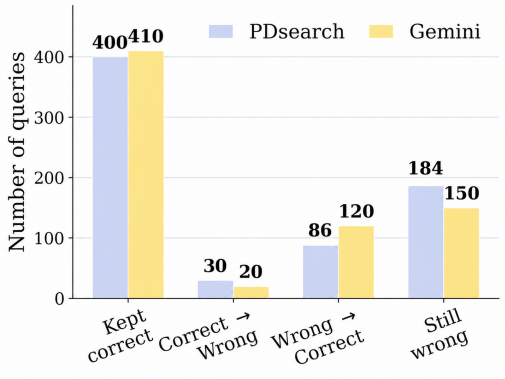


Figure 3: Changes in prediction correctness relative to Qwen3-VL-Reranker-8B.

Residual errors and candidate coverage. To separate candidate availability from downstream selection, we decompose errors according to whether the annotated target appears in the reranked top-10 pool. The target is available for 606 of 700 queries (86.57%), whereas the reranker selects it at rank one for 430. Thus, 176 of its 270 errors are recoverable within the fixed candidate pool, while 94 cannot be corrected without changing candidate generation. All ten candidates pass from inspection to decision, so no additional targets are lost through intermediate filtering. PDeepSearch-9B answers 486 queries correctly, leaving 94 candidate misses and 120 errors with available targets. Gemini answers 530 correctly, leaving the same 94 misses and 76 errors with available targets. Conditioned on target availability, selection accuracy rises from 70.96% for the reranker to 80.20% for PDeepSearch-9B and 87.46% for Gemini. After accounting for harmful overrides, their net gains close 31.8% and 56.8% of the gap between reranker accuracy and the candidate-coverage upper bound. This decomposition reveals different priorities for further improvement. For PDeepSearch-9B, errors with available targets remain the larger category, leaving substantial room to improve evidence acquisition and cross-candidate comparison. For Gemini, candidate misses account for 55.3% of the remaining 170 errors, making upstream coverage the larger limitation. Improving selection alone cannot resolve these misses; broader or more effective candidate generation is also needed. Errors with available targets, however, cannot be attributed solely to the Decision Agent, since incorrect Inspector profiles may also lead to a wrong final choice.

Performance by query composition. Table 3 shows that both systems improve over reranking across all five query compositions, with the largest absolute gains on I+V+D+R. PDeepSearch-9B and Gemini improve by 13.02 and 21.89 points on this group, compared with 1.92 and 2.88 points on I+V+D, where reranking already achieves 93.27 Hit@1. The smaller absolute gains on I+V+D should therefore be interpreted in light of its limited remaining headroom. The differences also extend beyond component count. Among three-component groups, reranker accuracy ranges from 53.33 on I+V+R to 93.27 on I+V+D. Moreover, Gemini gains 15.24 points on V+D+R, which contains no reference image. The benefit of the workflow therefore extends to queries whose requirements are expressed entirely in text. All four review-containing groups show larger absolute gains than I+V+D, a pattern consistent with the value of inspecting usage-related evidence beyond general product similarity. However, these groups differ in query content and difficulty, so the comparison does not isolate the causal contribution of review evidence. The performance gap between the two agent backbones is also larger on some difficult compositions. Gemini exceeds PDeepSearch-9B by 11.85 points on I+V+R, but by less than one point on I+D+R and I+V+D. This suggests that the remaining difference between the compact model and the stronger backbone is concentrated in particular requirement combinations. Together, these results motivate evaluating evidence-based selection by query composition as well as overall accuracy.

Table 3: Hit@1 by query composition. I/V/D/R denote reference image, visual, detail, and review components. PDS-9B denotes PDeepSearch-9B; Gemini uses PDeepSearch. Deltas are gains over reranking.

Comp.	Rerank	PDS-9B	Δ_P	Gemini	Δ_G
I+V+D+R	40.83	53.85	+13.02	62.72	+21.89
V+D+R	54.88	63.41	+8.53	70.12	+15.24
I+V+R	53.33	62.22	+8.89	74.07	+20.74
I+D+R	79.69	84.38	+4.69	85.16	+5.47
I+V+D	93.27	95.19	+1.92	96.15	+2.88

6 CONCLUSION

In this work, we introduce Product Deep Search, a benchmark for complex multi-condition product retrieval with evidence verification, and PDeepSearch, a multi-agent framework that augments semantic reranking with condition-level inspection and cross-candidate reasoning. By separating candidate-level evidence verification (PInspect) from cross-candidate decision making (PDecide), PDeepSearch constructs structured support–contradiction profiles and uses them to refine semantically plausible candidates. Trajectory supervision enables our 9B model to achieve 69.43% Hit@1, outperforming zero-shot 27B open-weight baselines. Our analyses show that evidence verification selectively corrects semantic ranking errors, particularly when plausible candidates differ on fine-grained query requirements. As selection improves, candidate coverage becomes an increasingly important constraint for stronger systems. These findings highlight structured evidence verification as a promising direction for reliable, evidence-grounded product search.

ETHICS STATEMENT

This work studies evidence-grounded product retrieval using publicly available Amazon Reviews 2023 data and synthetic shopping queries. We mitigate potential inaccuracies through automatic validation and a stratified human audit. We encourage responsible use, respect for user privacy, and verification of product claims in downstream applications.

REPRODUCIBILITY STATEMENT

We will release the Product Deep Search benchmark and code for data preprocessing, model training, and evaluation. Detailed training configurations, including hyperparameters and hardware specifications, will be provided with the code release to facilitate reproducibility.

REFERENCES

Marah I Abdin, Suriya Gunasekar, Varun Chandrasekaran, Jerry Li, Mert Yuksekgonul, Rahee Peshawaria, Ranjita Naik, and Besmira Nushi. Kitab: Evaluating llms on constraint satisfaction for information retrieval. In *International Conference on Learning Representations*, volume 2024, pp. 30664–30686, 2024.

Alberto Baldrati, Lorenzo Agnolucci, Marco Bertini, and Alberto Del Bimbo. Zero-shot composed image retrieval with textual inversion. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15292–15301. IEEE, 2023.

Antoine Chaffin and Aurélien Lac. Monoqwen: Visual document reranking, 2024. URL `https://huggingface.co/lightonai/MonoQwen2-VL-v0.1`.

Lei Chen, Chen Ju, Xu Chen, Zhicheng Wang, Yuheng Jiao, Hongfeng Zhan, Zhaoyang Li, Shihao Xu, Zhixiang Zhao, Tong Jia, et al. Pailitao-vl: Unified embedding and reranker for real-time multi-modal industrial search. *arXiv preprint arXiv:2602.13704*, 2026.

Wei Chow, Yuan Gao, Linfeng Li, Xian Wang, Qi Xu, Hang Song, Lingdong Kong, Ran Zhou, Yi Zeng, Yidong Cai, et al. Merit: Multilingual semantic retrieval with interleaved multi-condition query. *Advances in Neural Information Processing Systems*, 38:74806–74867, 2026.

Google DeepMind. Gemini 3.5 flash model card. `https://deepmind.google/models/model-cards/gemini-3-5-flash/`, 2026. Accessed: 2026-09-10.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation: Benchmarking llms as semantic encoders. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3251–3265, 2026.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. Hover: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3441–3460, 2020.

Zhibin Lan, Liqiang Niu, Fandong Meng, Jie Zhou, and Jinsong Su. Llave: Large language and vision embedding models with hardness-weighted contrastive learning. In *EMNLP (Findings)*, pp. 13721–13735, 2025.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv*, 2026a.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, et al. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv preprint arXiv:2601.04720*, 2026b.

Sheng-Chieh Lin, Chankyu Lee, Mohammad Shoeybi, Jimmy Lin, Bryan Catanzaro, and Wei Ping. Mm-embed: Universal multimodal retrieval with multimodal llms. In *International Conference on Learning Representations*, volume 2025, pp. 44215–44234, 2025.

Yikun Liu, Yajie Zhang, Jiayin Cai, Xiaolong Jiang, Yao Hu, Jiangchao Yao, Yanfeng Wang, and Weidi Xie. Lamra: Large multimodal model as your advanced retrieval assistant. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4015–4025. IEEE, 2025.

Zheyuan Liu, Cristian Rodriguez-Opazo, Damien Teney, and Stephen Gould. Image retrieval on real-life images with pre-trained vision-and-language models. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2105–2114. IEEE, 2021.

Yougang Lyu, Xiaoyu Zhang, Lingyong Yan, Maarten de Rijke, Zhaochun Ren, and Xiuying Chen. Deepshop: A benchmark for deep research shopping agents. *arXiv preprint arXiv:2506.02839*, 2025.

Rui Meng, Ziyan Jiang, Ye Liu, Mingyi Su, Xinyi Yang, Yuepeng Fu, Can Qin, Zeyuan Chen, Ran Xu, Caiming Xiong, et al. Vlm2vec-v2: Advancing multimodal embedding for videos, images, and visual documents. *arXiv preprint arXiv:2507.04590*, 2025.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 12076–12100, 2023.

Moonshot AI. Kimi k2.6: Advancing open-source coding. `https://www.kimi.ai/blog/kimi-k2-6`, 2026. Accessed: 2026-09-10.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Chandan K Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. Shopping queries dataset: A large-scale esci benchmark for improving product search. *arXiv preprint arXiv:2206.06588*, 2022.

Gemma Team. Gemma 3. 2025. URL `https://goo.gle/Gemma3Report`.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819, 2018.

Jiangyuan Wang, Kejun Xiao, Qi Sun, Huaipeng Zhao, Tao Luo, Jian Dong Zhang, and Xiaoyi Zeng. Shoppingbench: A real-world intent-grounded shopping benchmark for llm-based agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33521–33529, 2026.

Qifan Wang, Li Yang, Jingang Wang, Jitin Krishnan, Bo Dai, Sinong Wang, Zenglin Xu, Madian Khabsa, and Hao Ma. Smartave: Structured multimodal transformer for product attribute value extraction. In *Findings of the association for computational linguistics: EMNLP 2022*, pp. 263–276, 2022.

Hui Wu, Yupeng Gao, Xiaoxiao Guo, Ziad Al-Halah, Steven Rennie, Kristen Grauman, and Rogerio Feris. Fashion iq: A new dataset towards retrieving images by natural language feedback. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11302–11312. IEEE, 2021.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. *Advances in Neural Information Processing Systems*, 37:127129–127153, 2024.

Zhouhang Xie, Sameer Singh, Julian McAuley, and Bodhisattwa Prasad Majumder. Factual and informative review generation for explainable recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 13816–13824, 2023.

Li Yang, Qifan Wang, Zac Yu, Anand Kulkarni, Sumit Sanghai, Bin Shu, Jon Elsas, and Bhargav Kanagal. Mave: A product dataset for multi-source attribute value extraction. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pp. 1256–1265, 2022.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Xin Zhang, Ziqi Dai, Mingxin Li, Dingkun Long, Pengjun Xie, Meishan Zhang, Wenjie Li, Min Zhang, et al. Supervised fine-tuning or contrastive learning? towards better multimodal llm reranking. In *International Conference on Learning Representations*, volume 2026, pp. 65766–65789, 2026.

Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *Foundations and trends® in information retrieval*, 14(1):1–101, 2020.

Henry Peng Zou, Vinay Samuel, Yue Zhou, Weizhi Zhang, Liancheng Fang, Zihe Song, Philip S Yu, and Cornelia Caragea. Implicitave: An open-source dataset and multimodal llms benchmark for implicit attribute value extraction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 338–354, 2024.

A Benchmark Details

This appendix provides additional details of the benchmark construction pipeline, including query generation, retrieval-based uniqueness verification, and a manual quality audit. We additionally provide a representative benchmark case in Appendix A.3.

A.1 Query Generation Prompt

We use Gemini-3.1-pro to generate each benchmark query from the available evidence associated with a candidate positive product, including product images, details, price, and user reviews. The generated query is formulated as a natural shopping request containing multiple requirements grounded in the positive product.

During benchmark construction, we retain the evidence provenance of the requirements used in each generated query for subsequent validation. These provenance annotations are not exposed to evaluated models. To prevent trivial source-product reconstruction, the generation process excludes ASINs, exact product titles, model identifiers, and other catalog-specific strings that directly reveal the source product.

The system instruction is shown in Figure 4, while Figure 5 presents the structured user-prompt template and output schema.

```
You generate short and natural product-search queries from product
    evidence.

Requirements:
- Write a concise and conversational shopping request.
- Combine multiple user requirements when applicable.
- State requirements directly without mentioning their evidence source
    .
- Do not use verification or meta-language such as
  "reviews should confirm", "according to the product page",
  or "verify that".
- Do not include ASINs, exact product titles, model identifiers,
  or other catalog-specific strings that directly reveal the product.
- Do not invent requirements that are unsupported by the input
    evidence.

If a reference image is provided:
- Treat it only as a soft visual preference.
- Do not require an exact visual match.
- Explicit textual requirements take precedence over visual similarity
    .

Return strict JSON only.
```

Figure 4: System instruction used for benchmark query generation.

A.2 Human Quality Audit

We manually inspect 250 queries via stratified random sampling from the final 700-query benchmark. The annotation is conducted independently by two computer science Ph.D. students. For each query instance, annotators verify: (1) whether the designated positive product satisfies all explicit requirements expressed in the query, and (2) whether there exists obvious residual ambiguity among retrieved competing products. The annotators compare the query requirements against the available product evidence, including product images, descriptions, specifications, prices, and reviews. Any disagreement between the two annotators is resolved through discussion and additional evidence inspection. Initially, 98.8% of the inspected instances pass the manual quality check. The remaining failed cases are further reviewed manually. For these cases, we identify better-matched products that satisfy the query requirements and replace the original designated positives when necessary. After replacing a positive, we rerun the uniqueness verification procedure. Following this correction process, all audited benchmark instances satisfy the manual quality criteria.

A.3 Benchmark Case Study

We provide a representative benchmark instance to illustrate how product evidence is presented to the evaluated models. The example query is:

> *"I'm excited to find a tablet hand strap holder like the look in the reference image, with a dark green leather texture, corner mounting, and a swivel handle that feels soft and snug in my hand, all for no more than $17.99."*

The corresponding positive product is represented as a rendered webpage with separate *Product*, *Reviews*, and *Details* views. Figures 6–8 show these three views. Together, they expose heterogeneous evidence that can be used to verify different query requirements, including visual appearance, product features, price, specifications, and user experience.

```
{
  "task":
    "Generate one short, natural product-search query using only
     the provided product evidence.",

  "product_type": "<product type>",
  "price": "<price>",
  "image_attached": <true/false>,

  "evidence": {
    "visual": [
      "<visual attribute>",
      "..."
    ],

    "details": [
      "<product-detail requirement>",
      "..."
    ],

    "reviews": [
      "<review-grounded requirement>",
      "..."
    ]
  },

  "rules": [
    "Use only the provided evidence.",
    "Keep the query concise, natural, and multi-condition.",
    "Do not include identifying catalog information.",
    "If a reference image is provided, use it only as a
     soft visual preference."
  ],

  "output_schema": {
    "query_text": "<generated query>",

    "used": {
      "visual_requirements": [],
      "description_requirements": [],
      "review_requirements": [],
      "price_requirements": []
    }
  }
}
```

Figure 5: Structured user-prompt template and output schema for benchmark query generation. When available, the reference image is provided as a separate multimodal input. The `used` fields retain condition-level evidence provenance for benchmark construction and validation and are hidden from evaluated models.

B PDeepSearch Implementation Details

This appendix describes the query representation, evidence-access tools, Inspector profiles, evidence packaging, and Decision interface. PDeepSearch independently inspects the reranked top-10 candidates and passes all ten evidence profiles to the Decision Agent. The optional reference image remains separate from the textual conditions and serves as a soft visual preference throughout.

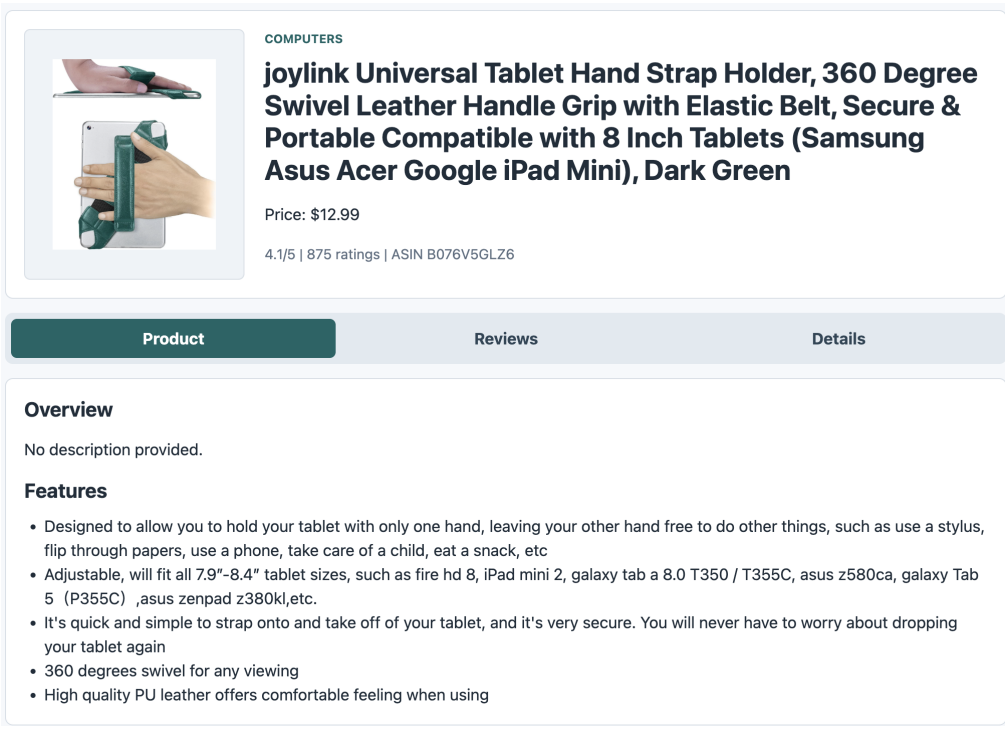


Figure 6: **Product view** of the positive product in the representative benchmark case. The rendered page exposes the product image, title, price, and feature descriptions. In this example, the page provides evidence such as the dark-green appearance, tablet hand-strap form, 360-degree swivel mechanism, supported tablet size, and product price.

B.1 Query Understanding Format

For each PDeepSearch system, the corresponding backbone converts the original shopping request into a target product type and atomic textual conditions. Query understanding is performed zero-shot, without benchmark construction annotations or evidence provenance labels. The conditions are generated online and do not specify which evidence sources should be consulted. The optional reference image is retained separately.

Figure 9 illustrates the structured representation shared by the Inspectors and Decision Agent.

B.2 Product Evidence Tools

Each Inspector uses two read-only tools, `product.open` and `product.find`. Both operate only on its bound candidate within the fixed corpus. An Inspector cannot inspect another candidate, search the catalog, or access external information.

B.2.1 `PRODUCT.OPEN`

`product.open` returns product metadata and structured evidence blocks with stable `block_ids`. Each Inspector must open the product page at least once before producing its evidence profile.

B.2.2 `PRODUCT.FIND`

`product.find` searches the bound candidate's product, detail, or review evidence using one or more attribute-level queries. Retrieval combines BM25 and embedding similarity, deduplicates results by `block_id`, and returns at most seven evidence blocks. The requested `topk_per_query` is restricted to 1–5.

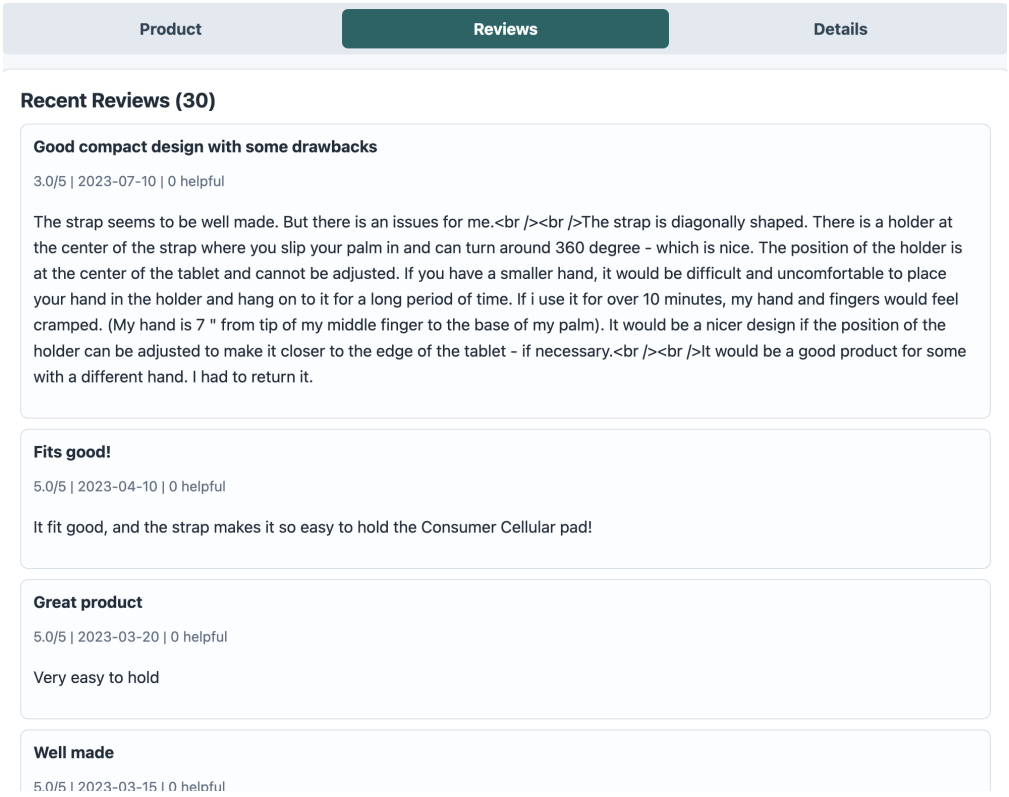


Figure 7: **Reviews view** of the same positive product. User reviews provide additional usage-oriented evidence, including observations about ease of holding, fit, comfort, and practical handling experience. Such review evidence complements information available from the product description and specifications.

B.3 Inspector Input and Evidence Profile

Each of the reranked top-10 candidates is assigned an independent Inspector. It receives the original query, canonical query representation, bound candidate information, candidate image, and optional reference image. It collects condition-relevant evidence using `product.open` and `product.find`, without access to other candidates, explicit reranking positions, or reranker scores.

For each textual condition, the Inspector assigns one of four verification statuses:

- `exact_match`: the evidence clearly supports the condition;
- `mixed`: the evidence provides partial support or conflicting signals;
- `unknown`: the evidence is insufficient to verify the condition;
- `contradicted`: the evidence explicitly conflicts with the condition.

These statuses describe individual conditions and are not aggregated into a candidate score. In particular, `unknown` distinguishes missing support from explicit contradiction.

B.3.1 Inspector Input

Figure 12 shows the input to one Inspector.

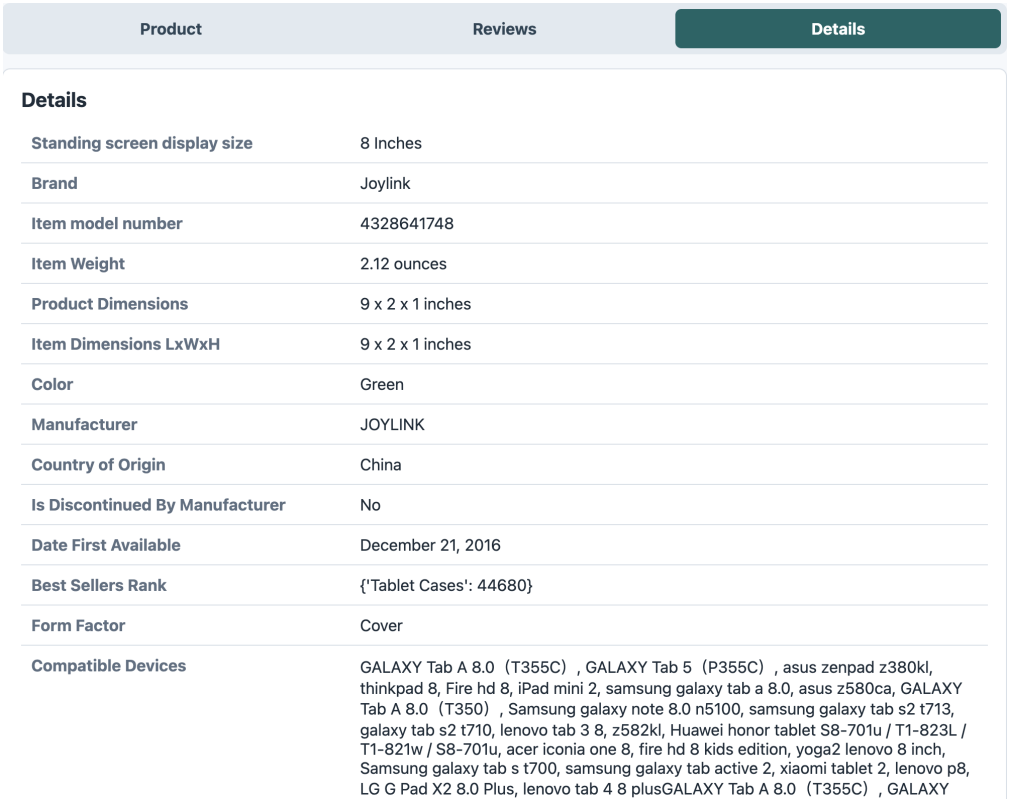


Figure 8: **Details view** of the same positive product. Structured metadata provides additional specification-level evidence, including display-size compatibility, color, dimensions, form factor, and compatible devices.

```
{
  "product_type": "tablet hand strap holder",

  "conditions": [
    {"id": "c1", "text": "dark green leather texture"},
    {"id": "c2", "text": "corner mounting"},
    {"id": "c3", "text": "a swivel handle"},
    {"id": "c4", "text": "feels soft and snug in my hand"},
    {"id": "c5", "text": "no more than $17.99"}
  ],

  "reference_image_attached": true
}
```

Figure 9: Structured query representation shared by the Inspector and Decision stages. The reference image is provided separately as a soft visual preference.

B.3.2 Inspector Evidence Profile

The Inspector records each condition's status, supporting evidence, and contradictory evidence, together with a candidate-level summary. Figure 13 shows an abridged example. Inspectors do not rank or select candidates.

```
name: product.open

Input
{
  "product_id": "<bound candidate id>",
  "page_type": "product | detail"
}

Output (abridged)
{
  "product_id": "<candidate id>",
  "page_type": "product",
  "title": "<product title>",
  "price": 12.99,

  "blocks": [
    {
      "block_id": "<block id>",
      "type": "feature",
      "text": "<structured product evidence>"
    },
    "..."
  ]
}
```

Figure 10: Interface of `product.open`. Stable `block_id`s identify the returned evidence blocks.

```
name: product.find

Input
{
  "queries": ["<attribute query>", "..."],
  "product_id": "<bound candidate id>",
  "page_type": "product | review | detail",
  "topk_per_query": 3
}

Output (abridged)
{
  "evidence": [
    {
      "block_id": "<block id>",
      "matched_queries": ["soft", "comfortable"],
      "text": "<retrieved evidence>"
    },
    "..."
  ]
}
```

Figure 11: Interface of `product.find`. `matched_queries` records which queries retrieved each block; retrieval does not establish condition satisfaction.

B.4 Evidence-Package Construction

A deterministic Aggregator combines each candidate's metadata with its Inspector profile to form a compact evidence package. It restores the canonical condition order and removes redundant tool traces while preserving verification statuses and supporting and contradictory evidence.

```
{
  "original_query": "<original shopping query>",

  "query_understanding":
    "<structured query representation>",

  "bound_candidate": {
    "product_id": "<candidate id>",
    "title": "<candidate title>",
    "price": 12.99
  },

  "attachments": [
    "<optional reference image>",
    "<bound candidate image>"
  ]
}
```

Figure 12: Inspector input: the query, structured conditions, one bound candidate, and relevant images.

The Aggregator does not score, rank, or filter candidates. All ten packages are passed to the Decision Agent, so $\mathcal{C}_D = \mathcal{C}_I$ and $|\mathcal{C}_D| = 10$.

B.5 Decision Agent Input and Output

The Decision Agent receives the original query, canonical query representation, optional reference image, and evidence packages for all ten inspected candidates. Reranker scores and explicit ranking positions are not included as input fields.

The agent compares condition-level evidence across candidates and selects one product. Explicit requirements and contradictions take precedence over generic positive signals, while `unknown` remains unresolved rather than being treated as support. The reference image serves only as a soft visual preference.

The Decision Agent cannot call `product.open` or `product.find`. Its output includes the selected product, condition-level comparisons, a selection rationale, and remaining uncertainties.

C Experimental Protocol Details

This appendix provides the Base agent prompt and Judge@1 evaluation prompt used in Section 5.

C.1 Base Agent Prompt

The Base agent receives the original query, optional reference image, the same reranked top-10 candidates as PDeepSearch, basic candidate information, and access to `product.open` and `product.find`. It can inspect any candidate in this pool but cannot access products outside it.

The Base agent is not supplied with a canonical condition decomposition, candidate-specific Inspector roles, structured evidence profiles, or a separate Decision stage. It determines which candidates to inspect, how to use the tools, and which product to return. Figure 15 provides its system prompt.

C.2 Judge@1 Evaluation Prompt

Judge@1 complements exact-match Hit@1 by evaluating whether the selected product satisfies the shopping request. For each prediction, Claude Opus 4.5 receives the original multimodal query and available evidence for the selected product.

```
{
  "product_id": "<candidate id>",

  "product_type": {
    "status": "exact_match",
    "supporting_evidence": [
      "The product is identified as a tablet hand strap holder."
    ],
    "contradictory_evidence": []
  },

  "conditions": [
    {
      "id": "c1",
      "status": "exact_match",
      "supporting_evidence": [
        "Dark-green leather appearance is supported."
      ],
      "contradictory_evidence": []
    },
    {
      "id": "c4",
      "status": "mixed",
      "supporting_evidence": [
        "Several reviews support comfortable handling."
      ],
      "contradictory_evidence": [
        "One review reports discomfort during extended use."
      ]
    },
    {
      "id": "c5",
      "status": "exact_match",
      "supporting_evidence": [
        "The listed price is $12.99."
      ],
      "contradictory_evidence": []
    },
    "..."
  ],

  "overall_evidence_summary":
    "<candidate-level evidence summary>"
}
```

Figure 13: Example Inspector profile. Each condition follows the same status–support–contradiction schema. Only a subset of conditions is shown.

A prediction passes only when the evidence supports all explicit textual requirements without a decisive contradiction. The reference image is treated as a soft visual preference. Figure 16 provides the evaluation prompt.

D AI USAGE DISCLOSURE

This work uses large language models in several stages of the research pipeline. Gemini-3.1-pro was used for generating product search queries during benchmark construction, as described in Section 3.2. Gemini-3.5-Flash was used as a teacher model to synthesize training trajectories for supervised fine-tuning, as described in Section 4.3. GPT-5.6 Sol was used for retrieval-based uniqueness verification during benchmark construction. All generated data and model outputs were filtered and

```
// Decision input
{
  "original_query": "<original shopping query>",

  "query_understanding":
    "<structured query representation>",

  "reference_image":
    "<optional reference image>",

  "candidate_evidence_packages": [
    "<candidate evidence package>",
    "...",
    "<candidate evidence package>"
  ]
}

// Decision output
{
  "final_product_id": "<selected candidate>",

  "per_condition_comparison": [
    {
      "condition_id": "c1",
      "assessment": "<cross-candidate evidence comparison>"
    },
    {
      "condition_id": "c2",
      "assessment": "<cross-candidate evidence comparison>"
    },
    "..."
  ],

  "final_choice_reason":
    "<evidence-grounded reason for the final selection>",

  "risk_assessment":
    "<remaining uncertainty or conflicting evidence>"
}
```

Figure 14: Decision interface. The input contains evidence packages for all ten inspected candidates, with no intermediate candidate filtering.

validated according to the procedures described in the paper. The proposed method and experimental analysis were designed and conducted by the authors.

```
You are a product-search agent.

You are given:
- a user's shopping query;
- an optional reference image;
- a fixed set of candidate products;
- basic information for each candidate;
- read-only product.open and product.find tools.

Your task is to identify the candidate that best satisfies the user's
shopping request.

You may inspect any of the provided candidates using the available
tools. Use product evidence such as product images, descriptions,
details, prices, and reviews when needed.

Important:
- Evaluate the explicit requirements in the user's query.
- Treat the reference image as a soft visual preference rather than
  an exact-match requirement.
- Do not assume that a semantically similar product satisfies all
  requested constraints.
- Use available evidence to resolve uncertain attributes.
- Explicit contradictions to a requested requirement should weigh
  against a candidate.
- Do not access products outside the provided candidate set.

Return exactly one candidate product as the final answer.
```

Figure 15: Base agent system prompt. Base uses the same reranked top-10 candidate pool and evidence-access tools as PDeepSearch, without the Inspector–Decision decomposition.

```
You are evaluating whether a predicted product satisfies a user's
shopping request.

You are given:
- the original shopping query;
- an optional reference image;
- the predicted product;
- the available product evidence, including images, descriptions,
  details, price, and reviews.

Evaluate the predicted product against every explicit textual
requirement in the query.

Guidelines:
- Use only the provided product evidence.
- Treat the reference image as a soft visual preference, not as an
  exact-match requirement.
- Explicit textual requirements take precedence over visual similarity
    .
- A requirement should be treated as satisfied only when supported by
  the available evidence.
- Missing evidence should not be interpreted as positive support.
- Explicit contradictory evidence means the corresponding requirement
  is not satisfied.
- Generic product similarity, high ratings, or low price should not
  compensate for violation of an explicit requirement unless that
  property is itself requested.

Return JSON only:

{
  "judgment": "pass | fail",
  "reason": "<brief evidence-based explanation>"
}
```

Figure 16: Judge@1 prompt for evaluating whether the selected product satisfies the query using the available multimodal evidence.