GLOBE: Evolving Structured Agentic Memory for E-commerce Product Optimization in Generative Search

Hyunseo Kim, Suil Choe, Sukkyoung So, Daeyeon Yi
AI Lab, CTO Division, LG Electronics, South Korea
{hyunseo00.kim, suil.choe, sukkyoung.so, daeyeon.yi}@lge.com

Abstract

Optimizing product listings for generative search requires language agents not only to rewrite effectively but also to continually self-improve by retaining successful strategies across diverse categories. However, existing approaches often treat structured listings as free-form text, leaving effective strategies implicit in prompts, search trajectories, or rewritten outputs rather than accumulating them as a reusable agentic memory. To address this limitation, we introduce GLOBE, an agentic framework that drives continual self-improvement by storing optimization knowledge in an evolving handbook of explicit, field-scoped behaviors. In each episode, an Editor applies relevant behaviors and receives rank-based feedback, which a Reflector converts into behavior updates. Crucially for dependable execution in real-world settings, a Moderator removes redundant proposals, a Curator commits accepted changes, and consistently harmful behaviors are deactivated. This cycle transforms episodic feedback into a persistent, field-scoped optimization memory. Experiments show that structured behaviors outperform monolithic prompt optimization, while feedback-driven evolution and moderation further improve effectiveness and reduce downside risk. The learned behaviors also remain effective on unseen product categories, suggesting that agentic memory evolution supports interpretable and transferable product optimization.

1 Introduction

Language agents increasingly improve themselves not by updating weights, but by accumulating experience in an external memory that is continually refined through feedback (Shinn et al., 2023; Wang et al., 2023). Building on these advances, agent-centric frameworks are expanding into e-commerce, as large language models (LLMs) shift search toward generative engines that directly compare and rank products by reasoning over rich textual contexts (Li et al., 2024; Sun et al., 2023; Hou et al., 2024; Kim et al., 2025b). In this environment, an agent optimizing a product listing must shift from conventional search engine optimization methods (Davis, 2006; Gudivada et al., 2015) to strategic content structuring, as its success heavily depends on how favorably the downstream generative engine interprets the presented information.

A key challenge in this real-world setting lies in the structured nature of e-commerce product data. Product listings are not arbitrary free-form documents but consist of distinct fields, such as titles, feature bullets, and descriptions, each serving a different functional role. Nevertheless, prior methods commonly optimize a listing through a single prompt that rewrites the document as a whole (Bagga et al., 2025; Yuan et al., 2026). This monolithic approach lacks structural boundaries, treating optimization as unstructured text generation. As illustrated in Figure 1, relying on generic prompt optimization leads to critical structural errors, because the agent cannot restrict specific editing operations to their appropriate target fields.

Beyond this lack of structural control, current approaches fail to generalize successful strategies across multiple optimization episodes. Even if a specific rewrite tactic successfully improves a product's rank in one category, monolithic prompt optimization cannot abstract this success into a reusable principle. As further demonstrated in Figure 1, when a generic prompt attempts to transfer a durability strategy to new categories, it fails to adapt the context, forcing overfitted keywords (e.g., applying "rubber insulation" to leather boots) rather than grounding the edit in existing product facts. Consequently, effective strategies must be repeatedly rediscovered, making optimization dependent on fragmented and costly trial and error. Product optimization therefore requires an explicit agentic memory mechanism that transforms transient ranking outcomes into persistent, generalizable editing

The Proceedings of the Second Workshop for REsearch on Agent Language Models (REALM 2026), pages 1097–1117
October 29, 2026 ©2026 Association for Computational Linguistics

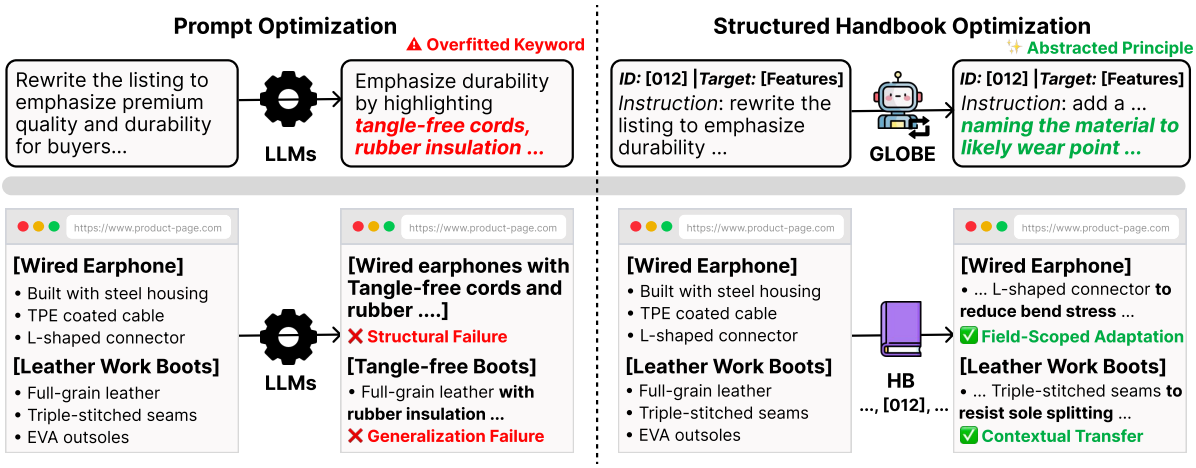

Figure 1: Comparison between monolithic prompt optimization and GLOBE's structured handbook optimization. A generic rewrite prompt performs an undifferentiated text transformation, whereas each GLOBE behavior specifies both a concrete editing operation and its permitted target field, enabling constrained reuse across products.

knowledge (Shinn et al., 2023; Zhao et al., 2024).

To address these limitations, we introduce **GLOBE** (**G**ener**a**L**izable **O**ptimization **B**ehavior **E**volution), an agentic framework that represents product optimization knowledge as a structured behavior handbook. Rather than relying on a single unconstrained rewrite instruction, GLOBE decomposes optimization into explicit and reusable editing *behaviors*. Each behavior specifies a concrete editing operation, the listing fields in which the operation may be applied, and metadata summarizing its historical ranking outcomes. For example, a behavior may instruct the agent to specify the material of a likely wear point, strictly limiting this addition to the feature field. This formulation separates *what* should be changed from *where* the change is permitted, providing a more interpretable and structurally bounded interface for product-listing optimization.

GLOBE further transforms transient ranking feedback into persistent optimization knowledge through an iterative *deployment-and-evolution* cycle. The current handbook is repeatedly applied to batches of query-product pairs, producing episode-level feedback from the resulting rank changes. Recurring success and failure patterns are then used to refine the handbook, after which the updated behaviors are redeployed in subsequent batches. By repeatedly alternating between handbook deployment and feedback-driven refinement, GLOBE converts episode-specific ranking outcomes into persistent and reusable optimization knowledge.

This formulation provides two complementary advantages. First, field-scoped behaviors explicitly specify both the editing operation and the product fields in which it may be applied, making individual strategies more structured and inspectable. Second, the evolving handbook preserves and reuses knowledge acquired from past ranking episodes rather than discarding it after each rewrite. We therefore investigate whether field-scoped behavior representations improve ranking effectiveness, whether feedback-driven handbook evolution reduces harmful outcomes, and whether the learned optimization knowledge transfers to unseen product categories.

Our main contributions are as follows:

- We propose GLOBE, an agentic product-optimization framework that maintains reusable optimization knowledge as a structured memory of field-scoped behaviors.
- We introduce a feedback-driven evolution mechanism that converts episode-level ranking outcomes into persistent editing knowledge, while filtering redundant and consistently harmful behaviors.
- We demonstrate that GLOBE improves ranking effectiveness and stability, with learned behaviors transferring to unseen product categories and ranking models.

2 Related Work

2.1 Generative Engine Optimization for E-commerce

Generative Engine Optimization (GEO) aims to improve the visibility of target items in generative search by modifying their textual representa-

tions. Initial approaches showed that strategically injected text sequences can influence item exposure and output rankings (Kumar and Lakkaraju, 2024; Jin et al., 2026). Subsequent prompt-based methods formulate GEO as document rewriting, using LLMs to modify content according to the inferred preferences of a target generative engine (Bagga et al., 2025; Wu et al., 2025). Recent agentic frameworks further automate this process by iteratively searching for effective document-modification strategies (Yuan et al., 2026). Despite this progression, existing approaches largely optimize documents through free-form, document-level generation, offering limited control over individual fields of structured product listings. Moreover, effective strategies typically remain embedded in prompts, search trajectories, or rewritten outputs rather than being retained as explicit and reusable knowledge.

2.2 Feedback-Driven Knowledge Evolution

Effective optimization requires systems to not only generate high-quality outputs but also learn from past successes and failures to avoid repeatedly rediscovering the same strategies. Driven by this need, a growing line of research studies how feedback can be transformed from a transient correction signal into reusable knowledge (Kim et al., 2025a). Early self-refinement methods use self- or peer-generated critiques to revise the current output (Madaan et al., 2023; Schick et al., 2022). Cross-episode approaches go beyond local revision by abstracting successful and failed trajectories into reusable reflections or lessons (Shinn et al., 2023; Zhao et al., 2024). At the system level, automatic prompt optimizers use natural-language critiques, textual feedback, or trajectory-level reflection to revise the instructions governing future executions (Pryzant et al., 2023; Yuksekgonul et al., 2024; Agrawal et al., 2026). Recent memory-centric systems further externalize such abstractions into persistent handbooks, evolving playbooks, or learned memory architectures (Didolkar et al., 2025; Zhang et al., 2025; Xiong et al., 2026). Together, these works demonstrate a progression from local output correction to persistent knowledge evolution, but largely focus on open-ended reasoning, coding, or planning. In contrast, structured product optimization requires reusable knowledge to be field-scoped and learned from indirect, relative ranking feedback.

3 Proposed Method: GLOBE

GLOBE maintains a structured agentic memory of reusable product-optimization behaviors through an iterative *deployment-and-evolution* cycle, illustrated in Figure 2. At iteration n, the current handbook $\mathcal{H}_n$ is deployed across a batch of K query-product pairs. Each deployment produces an episode that records the rewritten listing, the behaviors applied by the editor, and the resulting rank change. After the batch is completed, GLOBE aggregates these episodes to refine the handbook, producing $\mathcal{H}_{n+1}$. The updated handbook is then deployed in the next iteration, allowing optimization knowledge from individual ranking outcomes to accumulate as explicit, reusable editing behaviors.

Task Formulation. Given a user query q and a candidate product set $\mathcal{P} = \{p_1, \ldots, p_m\}$, a generative ranking model $\mathcal{M}$ orders the products according to their relevance to q. We denote the rank of product p by $r_{\mathcal{M}}(q, \mathcal{P}, p)$, where a smaller value indicates a higher position.

Unlike free-form documents, each product p is represented as a structured listing consisting of fields such as title, features, and description:

$$p = \left(p^{\text{title}}, p^{\text{features}}, p^{\text{description}}\right).$$

For a target product p_t, GLOBE generates a rewritten listing p'_t by applying editing behaviors whose permitted fields are compatible with the product schema. Let $\mathcal{P}'$ denote the candidate set in which p_t is replaced by p'_t. We measure the episode-level outcome using rank gain:

$$g(q, p_t) = r_{\mathcal{M}}(q, \mathcal{P}, p_t) - r_{\mathcal{M}}(q, \mathcal{P}', p'_t).$$

The goal is to learn a handbook that produces positive rank gains across diverse query-product contexts while limiting harmful ranking outcomes.

3.1 Behavior Handbook

A behavior is a reusable, field-scoped editing strategy for product listings. Each behavior stores an **identifier, a name, an editing instruction, permitted target fields, conflicting behaviors, and feedback metadata**. The *instruction* defines a concrete editing operation, such as making the product type explicit or foregrounding a query-relevant specification. The *target-field* set specifies where the operation may be applied, with $\text{targets}(b) \subseteq \{\text{title}, \text{features}, \text{description}\}$. The

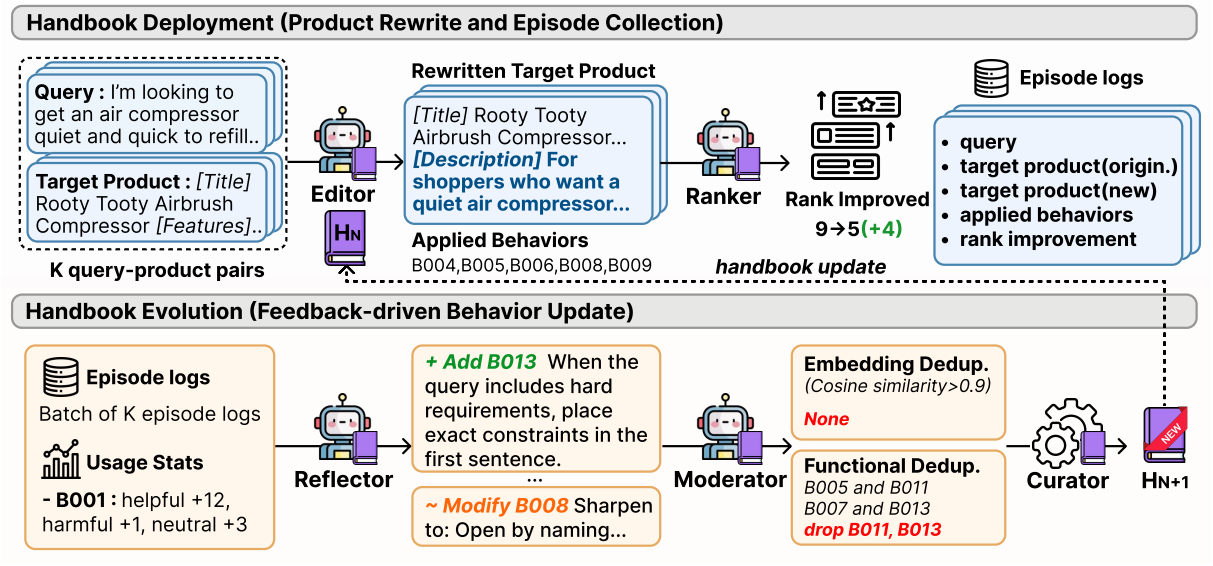

Figure 2: Overview of the GLOBE framework. Each iteration deploys the active handbook $\mathcal{H}_n$ over a batch of K query-product pairs: the editor rewrites the target listing and the ranker returns episode-level rank feedback. The Reflector then proposes behavior additions or modifications, the Moderator removes redundant proposals, and the Curator commits the survivors into $\mathcal{H}_{n+1}$, which replaces $\mathcal{H}_n$ as the handbook deployed over the next batch. Repeating this *deployment-and-evolution* cycle accumulates episodic feedback into persistent editing knowledge.

conflicts field records behaviors that should not be applied together. Rather than strictly filtering these rules, this field serves as a soft constraint that instructs the editor to avoid applying conflicting behaviors simultaneously. Finally, the *feedback metadata* tracks the number of helpful, harmful, and neutral outcomes associated with behavior b, denoted by $h^+(b)$, $h^-(b)$, and $h^0(b)$, respectively. We summarize the empirical utility of each behavior using the score $s(b) = h^+(b) - h^-(b)$. Let $n(b) = h^+(b) + h^-(b) + h^0(b)$ denote the number of episodes in which the behavior has been applied. To ensure that newly introduced behaviors receive sufficient opportunities for evaluation, a behavior b remains active if it is still within its initial probation period ($n(b) < N_{\min}$) or maintains a non-negative empirical score ($s(b) \geq 0$). Behaviors with negative scores after the probation period remain stored for traceability but are excluded from the active behavior pool used in subsequent iterations. Examples of deactivated behaviors are listed in Appendix D.

3.2 Handbook Deployment

At iteration n, the active behavior set $\mathcal{H}_n^{\text{act}}$ is deployed across a batch of K query-product pairs. For each episode (q_i, p_i), the LLM editor receives the user query, the original structured product listing, and the complete active behavior set as a strategy pool. The editor selects the behaviors relevant to the current query-product context and generates a rewritten target product p_i'. Each selected behavior may be applied only to the listing fields specified by its target-field scope. The editor is also instructed to preserve the original section structure and leave fields outside the permitted scopes unchanged. To record behavior usage, the editor also reports the subset of active behaviors applied during rewriting, denoted by $\mathcal{A}_i \subseteq \mathcal{H}_n^{\text{act}}$. Once the rewritten target product replaces the original product, the ranking model re-ranks the entire candidate set. Defining r_i and r_i' as the target product's rank before and after rewriting, respectively, the episode-level rank gain is calculated as:

$$g_i = r_i - r_i'.$$

A positive value indicates rank improvement, a negative value indicates degradation, and zero indicates no change.

The episode log therefore records the complete optimization trajectory, including the query, original and rewritten listings, reported behavior usage, and observed rank change. Each episode is stored as

$$e_i = \left(q_i, p_i, p_i', \mathcal{A}_i, r_i, r_i', g_i\right).$$

At this stage, GLOBE only records the episode outcome and does not immediately update the handbook or behavior statistics. All episode logs

collected under $\mathcal{H}_n$ are aggregated after the K episodes in the current batch have been completed.

3.3 Handbook Evolution

After collecting all K episodes under the current handbook $\mathcal{H}_n$, GLOBE aggregates their ranking outcomes and updates the handbook at the batch level. For each episode, the h^+, h^-, or h^0 count of every applied behavior is incremented depending on whether the target product's rank improves, degrades, or remains unchanged, respectively. Because the individual contribution of each behavior cannot be isolated, we assign the episode's overall directional outcome to all applied behaviors. The accumulated statistics are then used to recompute each behavior's score and active status for the next iteration. Using the aggregated episode logs and updated behavior statistics, GLOBE evolves the handbook through three components: 1) Reflector, 2) Moderator, and 3) Curator.

Reflector. The Reflector analyzes recent episode logs together with the current handbook to identify recurring success and failure patterns. Rather than rewriting the handbook as unrestricted text, it proposes structured delta operations that either add a new behavior or modify an existing one. An ADD operation is proposed when the observed pattern is not adequately covered by the current handbook. A MODIFY operation refines an existing behavior by clarifying its instruction, application condition, or permitted target fields. The Reflector does not directly delete behaviors, since behaviors associated with consistently harmful outcomes are excluded through the active-status criterion while remaining stored for traceability.

Moderator. The Moderator filters Reflector proposals before they are committed to the handbook. It performs **(1) embedding-based deduplication** to reject proposals that are semantically too similar to existing active behaviors. A proposed behavior b_{new} is rejected when

$$\max_{b \in \mathcal{H}_n^{act}} \text{sim}\left(E(b_{\text{new}}), E(b)\right) > \tau,$$

where $E(\cdot)$ denotes the behavior embedding function and $\text{sim}(\cdot, \cdot)$ denotes cosine similarity. The remaining proposals undergo **(2) functional deduplication**, in which an LLM compares their application conditions and intended editing operations with those of existing behaviors. When substantial overlap is detected, the existing behavior is retained, since unrestricted rewriting of committed behaviors would make the handbook fluctuate with the noise of individual batches.

Curator. The Curator deterministically commits the proposals accepted by the Moderator. New behaviors receive stable identifiers and zero-initialized feedback metadata, while modified behaviors are updated in place, advancing their version but retaining their accumulated metadata. The Curator then constructs the next handbook $\mathcal{H}_{n+1}$ and determines its active behavior pool using the accumulated feedback statistics. The updated handbook is applied to the next batch of query-product pairs, thereby repeating the rewriting, feedback collection, and handbook evolution cycle.

3.4 Training and Inference

Starting from a seed handbook $\mathcal{H}_1$ containing nine general GEO heuristics, GLOBE repeats the *deployment-and-evolution* cycle for N training batches. The handbook and its active behavior pool remain fixed within each batch. Feedback collected during the batch affects behavior status and handbook contents only in the subsequent iteration, ensuring that all episodes in the same batch are generated under a consistent handbook version. After the final training batch, the handbook is frozen. At test time, the active behaviors from the frozen handbook are supplied to the editor using the same deployment procedure as during training. The editor selects and applies behaviors for each query-product pair, but no feedback statistics or handbook contents are updated during evaluation.

4 Experiment

4.1 Experimental Setup

Dataset. We evaluate GLOBE using the E-GEO benchmark (Bagga et al., 2025), which pairs real-world shopping queries from Reddit (r/BuyItForLife) with structured Amazon product listings. We categorize the products into six macro-categories: Home & Kitchen, Fashion & Accessories, Electronics, Tools & DIY, Sports & Outdoors, and Health & Beauty. We use 1,000 training queries to construct the behavior handbook and a disjoint, category-balanced set of 900 queries (150 per category) for evaluation. The handbook remains frozen during testing.

Method	Components			Effectiveness		Risk & Consistency		Handbook Size
	Handbook	Feedback	Moderator	MRG↑	SR↑	DSR↓	CV↓	
Prompt Opt.	–	–	–	0.351	44.1%	3.961	0.117	–
Static HB	✓	–	–	1.343	58.6%	2.628	<u>0.042</u>	9
GLOBE w/o Mod.	✓	✓	–	<u>1.519</u>	<u>60.6%</u>	<u>2.389</u>	0.054	51
GLOBE	✓	✓	✓	**1.611**	**61.6%**	**2.172**	**0.034**	47

Table 1: Performance comparison of optimization methods. Higher values are better for MRG and SR (↑), while lower values are better for DSR and CV (↓). Best results are shown in bold and second-best results are underlined.

Baselines. We evaluate GLOBE against the following configurations.[1] **Prompt Opt.** applies a single monolithic rewriting instruction following the prompt-based E-GEO setting (Bagga et al., 2025). **Static HB** utilizes a fixed, non-evolving handbook of 9 generic rewrite rules (Aggarwal et al., 2023). **GLOBE w/o Mod.** evolves the handbook without Moderator, using episode feedback but commits Reflector proposals without redundancy filtering.

Evaluation Metrics. We evaluate system performance using four metrics. **Mean Rank Gain (MRG)** and **Success Rate (SR)** measure the average position improvement and the frequency of successful optimizations, respectively. To evaluate stability, we evaluate **Downside Risk (DSR)**, which applies a quadratic penalty to rank degradations to capture the severity of harmful rewrites. Finally, **Category Variance (CV)** measures the variance of MRG across the six categories, where a lower score indicates generalized performance. More details are presented in Appendix B.2.

Training Details. The initial handbook contains 9 general GEO heuristics in Appendix B.4. We process the 1,000 training queries in 10 batches of 100 query-product pairs. Within each batch, the current handbook remains fixed while product rewrites and episode logs are collected. After all episodes in the batch have been completed, behavior statistics are accumulated and the handbook is updated through the Reflector, Moderator, and Curator. We use GPT-5.4[2] for all LLM-based components. Embedding-based deduplication uses `all-MiniLM-L6-v2` (Reimers and Gurevych, 2019; Wang et al., 2020) with a cosine-similarity threshold of $\tau = 0.9$. More details and prompts are provided in Appendix B and F.

[1]Recent self-evolving agents are not directly applicable as our baselines because they are fundamentally designed for free-form document optimization, rendering field-scoped constraints incompatible with their core exploration mechanism.

[2]https://platform.openai.com/docs/models/gpt-5.4

4.2 Effectiveness of the Behavior Handbook

To evaluate the effectiveness of our proposed handbook framework, we examine the contribution of its core design components to the overall performance of GLOBE. As shown in Table 1, the Static HB already achieves a substantial improvement over monolithic prompt optimization in both ranking effectiveness and downside risk. Because this baseline uses only the initial generic behaviors without feedback-based updates, its improvement suggests that decomposing optimization knowledge into explicit operations, together with field scoping, provides a stronger foundation than the evaluated monolithic prompt baseline. This result further shows that structured behavior representation contributes substantially to GLOBE's performance even before feedback-driven evolution is introduced.

Feedback-based evolution further improves MRG and SR by extending the handbook beyond its initial generic heuristics. However, the unmoderated variant also produces a larger handbook and higher category variance, suggesting that unrestricted accumulation may introduce redundant or narrowly specialized behaviors. The increase in handbook capacity also means that its improvement cannot be attributed entirely to feedback-based learning. **Evolution expands the available optimization knowledge, but accumulating more behaviors does not necessarily produce a more consistent handbook.**

The full GLOBE addresses this limitation by filtering semantically redundant and functionally overlapping proposals before they are committed. Despite retaining fewer behaviors than the unmoderated variant, it achieves stronger overall performance together with lower DSR and CV. This pattern suggests that the Moderator acts as handbook-level regularization, improving the composition of the accumulated knowledge rather than simply expanding its size. **Overall, moderated evolution produces a more compact handbook with a bet-**

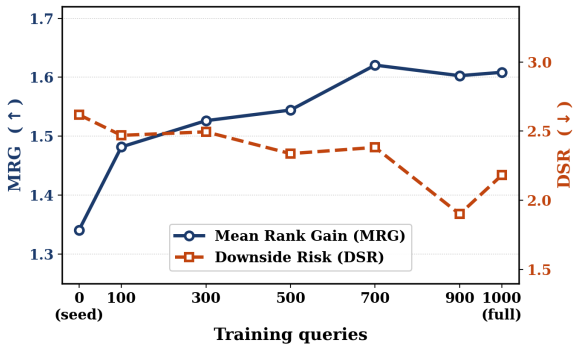

Figure 3: Performance of the trained handbook as the number of training queries increases.

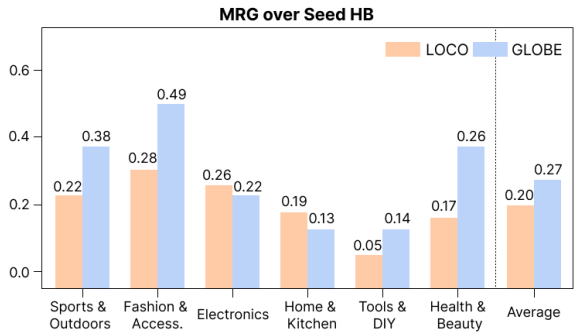

Figure 4: Cross-category generalization, measured as MRG over the seed handbook.

ter balance between ranking effectiveness and consistency.

4.3 Impact of Feedback-driven Refinement

To examine whether iterative handbook evolution produces sustained improvements beyond the final aggregate results, we trace ranking performance as the number of training examples increases.

As shown in Figure 3, MRG improves rapidly in the early stages of training, as feedback from the initial batches reveals optimization patterns not covered by the seed handbook, and then plateaus as additional batches are processed. This trajectory indicates that **feedback-based evolution progressively enriches the handbook, but its benefits exhibit diminishing returns once broadly useful behaviors have been acquired.** DSR also declines overall, though not monotonically, indicating that evolution gradually mitigates harmful ranking outcomes. Together, these patterns suggest that once frequently occurring optimization patterns have been observed, further improvement depends less on training-set size than on **broader coverage of query intents and product contexts that provide genuinely new feedback**.

Additionally, to test whether refinement improves the specific behaviors it targets, rather than merely coinciding with handbook growth, we compare each modified behavior's episode-level outcomes before and after modification (Appendix C). Among the nine under-performing behaviors that were modified, eight showed improvements beyond a matched mean-reversion control (sign test, $p = 0.020$), with an average adjusted increase of 5.3 percentage points in helpful rate. In contrast, modifying behaviors that were already performing above average yields no measurable gain. These results suggest that **MODIFY operations act as targeted repairs for under-performing behaviors rather than as indiscriminate handbook expansion.**

4.4 Cross-Category Generalizability

To examine whether the evolved handbook captures transferable optimization knowledge rather than category-specific heuristics, we conduct a leave-one-category-out (LOCO) evaluation. Specifically, for each macro-category, we evolve a separate handbook using only the training queries from the remaining five categories. Figure 4 compares the MRG improvement over the seed handbook for the LOCO models and the fully trained GLOBE model. The result confirms that the benefits of feedback-driven evolution are not strictly bound to domain-specific exposure. The LOCO handbook consistently retains a substantial portion of the aggregate improvements seen in the fully trained model. Furthermore, the LOCO approach even surpasses full training in some categories. These category-level differences suggest that training on all categories can occasionally hurt a specific category through cross-category interference, which we leave for future analysis. Nevertheless, they show that in-category exposure is not a strict prerequisite for generating effective rewrites. Together with the modest overall MRG decrease of 0.07, these results suggest that **feedback-driven evolution extracts transferable optimization strategies that remain effective without target-category exposure, avoiding category-specific memorization.**

4.5 Cross-Ranker Transferability

To determine whether the observed rank gains stem from transferable optimization rather than a generative engine simply favoring its own outputs, we freeze the GPT-5.4 editor and the handbook it produced. We then replace the ranking model with

Method	GPT-5.4 (Self)				Qwen3.5-122B-A10B				Gemini-3.5-flash			
	MRG↑	SR↑	DSR↓	CV↓	MRG↑	SR↑	DSR↓	CV↓	MRG↑	SR↑	DSR↓	CV↓
Prompt Opt.	0.351	44.1%	3.961	0.117	−0.031	22.1%	1.700	0.044	−0.051	21.5%	1.379	0.019
Static HB	1.343	58.6%	2.628	0.042	1.335	50.3%	0.435	0.048	0.960	45.5%	0.290	0.015
GLOBE w/o Mod.	1.519	60.6%	2.389	0.054	1.167	46.6%	0.115	**0.010**	0.975	46.6%	0.371	0.021
GLOBE	**1.611**	**61.6%**	**2.172**	**0.034**	**1.522**	**54.1%**	**0.105**	0.034	**1.078**	**48.9%**	**0.246**	**0.013**

Table 2: Cross-ranker evaluation of optimization strategies trained with GPT-5.4.

Query: *"Aux cable for the car that won't break after a month of use."* **Target Field:** *features*

Delta operations committed into $\mathcal{H}_4$

MODIFY B012	Name the listing's material and build detail **most relevant to the likely wear point** *(from:* name the listing's material and build *details)*
ADD B016	When the query **describes how past products failed**, add a top-placed clause mapping construction evidence to **that exact failure mode**

Resulting feature bullets

$\mathcal{H}_2$	*rank* $4 \to 5$, $g_i = -1$ (i) tight spaces and **connector strain** often cause early cable failure (ii) reduces **bend stress at the jack** for better day-to-day durability
$\mathcal{H}_4$	*rank* $4 \to 0$, $g_i = +4$ (i) tight spaces and **repeated bending** often cause cable failure [B016] (ii) reduces strain **at the connector to address a common breakage point** [B012]

Downstream: B016 is applied in 185 subsequent episodes (62% helpful, 13% harmful, mean gain +1.90).

Table 3: Behavior evolution from feedback to ranking effect for the target product *CableCreation 6 FT Flat 3.5mm Aux Cable*. Full example in Appendix E.

two different engines: Qwen3.5-122B-A10B[3] and Gemini-3.5-flash[4]. Although all rankers operate on the same ten-position rank scale, they differ in baseline orderings, calibration, and ranking stability. We therefore emphasize within-ranker comparisons between methods rather than interpreting absolute differences across rankers as model-quality differences. As shown in Table 2, the best and worst methods are unchanged across all three rankers. GLOBE achieves the highest MRG and SR, whereas Prompt Opt. remains the weakest baseline, with its gains turning negative when evaluated by the held-out rankers. This suggests that prompt optimization may overfit to the specific preferences of its training engine. The two handbook variants, however, diverge. GLOBE consistently outperforms the static seed handbook across all rankers, whereas the unmoderated variant loses its clear advantage over the static baseline on unseen rankers. Consequently, the performance gap between GLOBE and GLOBE w/o Mod. widens on the unseen rankers, where GLOBE also demonstrates the lowest downside risk. These findings suggest that moderation provides benefits beyond mere handbook compactness. Without filtering, feedback-driven accumulation absorbs ranker-specific biases that fail to transfer, whereas **moderated evolution retains behaviors that remain effective across different judges.**

4.6 Case Study

Table 3 shows a single handbook evolution from ranking feedback to its downstream editing effect. Under $\mathcal{H}_2$, the existing durability behavior (B012) degraded the target product's rank ($g_i = -1$) because it merely surfaced generic material evidence without addressing the shopper's specific concern. In response, the Reflector proposes a MODIFY to narrow B012 to wear-point specific details, along with an ADD for a new behavior (B016) that explicitly maps construction evidence to the user's stated failure mode. Once the Curator commits these deltas into $\mathcal{H}_4$, the editor applying the updated handbook ($\mathcal{H}_4$) stops merely listing specs; instead, it directly targets *"repeated bending"* and *"connector"* strain. This strategic shift improves the product from rank 4 to the top position ($g_i = +4$). Furthermore, B016 generalizes well beyond this initial episode, achieving a 62% helpful rate and a +1.90 mean rank gain across 185 subsequent applications. **GLOBE therefore renders optimization knowledge traceable by explicitly linking observed failures to structural behavior updates and their downstream ranking improvements.**

[3]https://huggingface.co/Qwen/Qwen3.5-122B-A10B

[4]https://ai.google.dev/gemini-api/docs/models/gemini-3.5-flash

5 Conclusion

We introduced GLOBE, a framework that maintains product-optimization knowledge as explicit, field-scoped behaviors in an evolving handbook. By converting ranking feedback into persistent and reusable editing knowledge, GLOBE enables effective and interpretable optimization while transferring to unseen product categories. More broadly, our findings suggest that evolving structured behaviors offers a practical path toward interpretable and transferable agentic memory for generative search.

Limitations

A key limitation is that the editor, ranker, Reflector, and Moderator are all instantiated from the same model (GPT-5.4). Because the ranker that provides the reward signal shares a model family with the components optimized against it, the learned behaviors may partly capture ranker-specific preferences rather than ranker-agnostic optimization knowledge. We probe this with a cross-ranker evaluation in Section 4.5; fully disentangling transferable knowledge from ranker-specific artifacts remains future work. In addition, credit assignment over behaviors is approximate. Since each episode yields a single rank change for the entire rewrite, the episode's directional outcome is attributed to all applied behaviors rather than to each behavior's individual contribution (Section 3.3). A useful behavior can thus absorb harmful counts from co-applied behaviors. These counts also rely on the editor's self-reported list of applied behaviors, which is not verified against the actual edits; an audit of 50 training episodes judged by a separate LLM (Claude Opus 5) found 88.3% of reported behaviors consistent with the edits made and few applied behaviors left unreported. Batch-level aggregation and the probation period mitigate this attribution noise but do not eliminate it. More precise attribution, such as per-behavior counterfactual rewrites, would incur substantially higher ranking costs per episode, and we leave its efficient integration into the evolution loop to future work. Moreover, because modified behaviors inherit their accumulated feedback counts, the active-status criterion reacts slowly to a harmful MODIFY, and GLOBE currently has no mechanism to revert such a revision. Beyond these signal-level limitations, while establishing a rigorous performance ceiling with high-end models necessitates substantial computational overhead during the multi-agent learning phase, our inference stage remains highly efficient; the compiled handbook acts as a static set of rules that applies verified behaviors in a single pass without iterative optimization loops. Furthermore, distilling these structured behaviors into smaller, open-source models via fine-tuning represents a promising direction to drastically reduce future deployment costs. Finally, as real-world e-commerce search algorithms are dynamic and continuously updated, accumulated editing behaviors may experience temporal degradation and require periodic re-evolution. Extending our current text-only, field-scoped constraints to multimodal settings also remains an open avenue for future research.

Ethical Considerations

Factuality and Consumer Protection. Methods for improving product visibility in generative search may be used to unfairly influence rankings, and rank improvement should not be interpreted as evidence of superior product quality. To reduce misleading edits, GLOBE includes a factual-grounding behavior that requires rewritten claims to remain supported by the original listing. Nevertheless, language models may still introduce unsupported implications or overstate product benefits. In practical deployments, rewritten listings should therefore be verified against the original listing and monitored for deceptive or exaggerated claims.

Model Bias and Accessibility. The ranking feedback used by GLOBE may reflect model-specific preferences rather than general shopper preferences. We partially examine this risk through cross-ranker evaluation, but it cannot be fully eliminated. In addition, evolving the handbook with high-capacity language models may favor organizations with greater computational resources. Reusing a frozen handbook and adapting it to smaller models may help reduce this barrier.

References

Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, and Ameet Deshpande. 2023. GEO: Generative engine optimization. *arXiv preprint arXiv:2311.09735*.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and

Omar Khattab. 2026. Gepa: Reflective prompt evolution can outperform reinforcement learning. *Preprint*, arXiv:2507.19457.

Puneet S. Bagga, Vivek F. Farias, Tamar Korkotashvili, Tianyi Peng, and Yuhang Wu. 2025. E-GEO: A testbed for generative engine optimization in e-commerce. *arXiv preprint arXiv:2511.20867*.

Harold Davis. 2006. *Search engine optimization*. " O'Reilly Media, Inc.".

Aniket Didolkar, Nicolas Ballas, Sanjeev Arora, and Anirudh Goyal. 2025. Metacognitive reuse: Turning recurring llm reasoning into concise behaviors. *arXiv preprint arXiv:2509.13237*.

Venkat N. Gudivada, Dhana Rao, and Jordan Paris. 2015. Understanding search-engine optimization. *Computer*, 48(10):43–52.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European conference on information retrieval*, pages 364–381. Springer.

Haibo Jin, Ruoxi Chen, Peiyan Zhang, Yifeng Luo, Huimin Zeng, Man Luo, and Haohan Wang. 2026. Controlling output rankings in generative engines for LLM-based search. *arXiv preprint arXiv:2602.03608*.

Hyunseo Kim, Sangam Lee, Kwangwook Seo, and Dongha Lee. 2025a. Bespoke: Benchmark for search-augmented large language model personalization via diagnostic feedback. *arXiv preprint arXiv:2509.21106*.

Jieyong Kim, Hyunseo Kim, Hyunjin Cho, SeongKu Kang, Buru Chang, Jinyoung Yeo, and Dongha Lee. 2025b. Review-driven personalized preference reasoning with large language models for recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 1697–1706, New York, NY, USA. Association for Computing Machinery.

Aounon Kumar and Himabindu Lakkaraju. 2024. Manipulating large language models to increase product visibility. *arXiv preprint arXiv:2404.07981*.

Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. 2024. Large language models for generative recommendation: A survey and visionary discussions. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10146–10159, Torino, Italia. ELRA and ICCL.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2022. Peer: A collaborative language model. *arXiv preprint arXiv:2208.11663*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, Singapore. Association for Computational Linguistics.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Yujiang Wu, Shanshan Zhong, Yubin Kim, and Chenyan Xiong. 2025. What generative search engines like and how to optimize web content cooperatively. *arXiv preprint arXiv:2510.11438*.

Yiming Xiong, Shengran Hu, and Jeff Clune. 2026. Learning to continually learn via meta-learning agentic memory designs. *arXiv preprint arXiv:2602.07755*.

Jiaqi Yuan, Jialu Wang, Zihan Wang, Qingyun Sun, Ruijie Wang, and Jianxin Li. 2026. AgenticGEO: A self-evolving agentic system for generative engine optimization. *arXiv preprint arXiv:2603.20213*.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, and 1 others. 2025. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Heyang Zhou, JiaJia Chen, Xiaolu Chen, Jie Bao, Zhen Chen, and Yong Liao. 2026. If-geo: Conflict-aware instruction fusion for multi-query generative engine optimization. *arXiv preprint arXiv:2601.13938*.

A Robustness Across Different LLMs

Model	MRG↑	SR↑	DSR↓	CV↓	HB
GPT-5.4	1.611	**61.6%**	**2.172**	0.034	47
Qwen3.5-122B-A10B	1.507	56.5%	2.106	0.058	29
Gemini-3.5-flash	**1.700**	60.8%	2.310	**0.028**	46

Table 4: Performance evaluation of the GLOBE framework across different LLM backends.

To assess backbone robustness, we evaluate GLOBE across multiple LLMs. For each backbone, we rerun the complete handbook-evolution procedure from the seed handbook, using that model for all LLM-based components and retaining the same data splits and hyperparameters. As shown in Table 4, all configurations achieve positive rank gains, while exhibiting different trade-offs across MRG, SR, and DSR. These results suggest that GLOBE remains effective across both proprietary and open-weight models, although its performance depends on the underlying backbone.

B Experimental Details

B.1 Dataset and Evaluation Setup

Every method optimizes the same target product in the same competitive context, and is scored against the same reference ranking. Each E-GEO query (Bagga et al., 2025) comes with a candidate set of ten Amazon listings, each serialized into the title, features, and description fields. We sample one candidate uniformly at random as the target product, store that index once, and reuse it for GLOBE, its ablations, and all baselines; the pre-rewrite ranking $r_{\mathcal{M}}(q, \mathcal{P}, p_t)$ is likewise computed once and cached. This shared setup is intended to isolate differences arising from the rewriting strategy.

We use 1,000 queries for handbook construction, processed as ten batches of 100 pairs, and a disjoint set of 900 queries for evaluation. Category labels are not provided by the benchmark at the query level. We first use GPT-5.4-nano to construct a mapping table from the unique Amazon main category values in the dataset to the six macro-categories. This mapping table is applied deterministically to every candidate product. Each query is assigned the majority macro-category among the candidates in its product set. The evaluation set is built by sampling 150 queries per macro-category. An episode is discarded when the editor returns no usable rewrite or when the ranker's output cannot be parsed into a complete permutation of the ten candidates after all retries; such failures are excluded rather than imputed, since scoring them as zero gain would reward methods that fail more often. This leaves 887 of the 900 episodes for GLOBE, with 146 to 149 episodes per category, and comparable counts for the baselines. GLOBE uses no validation set, as the handbook is updated directly from training-batch feedback; only the prompt-optimization baseline needs one, selecting its final prompt on 100 held-out validation queries evaluated after each epoch.

B.2 Metrics

All four metrics are computed from the episode-level rank gain $g_i = r_i - r'_i$, where r_i and r'_i are the target's 0-indexed positions before and after rewriting, and all are macro-averaged over the six categories so that each category contributes equally. Let $\mathcal{C}$ be the set of categories and $\mathcal{E}_c$ the episodes of category c, with per-category mean gain

$$\mathrm{MRG}_c = \frac{1}{|\mathcal{E}_c|} \sum_{i \in \mathcal{E}_c} g_i. \tag{1}$$

Mean Rank Gain measures how far the rewritten product moves up on average:

$$\mathrm{MRG} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \mathrm{MRG}_c. \tag{2}$$

Success Rate measures how often a rewrite helps at all, irrespective of magnitude:

$$\mathrm{SR} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{E}_c|} \sum_{i \in \mathcal{E}_c} \mathbf{1}\left[g_i > 0\right]. \tag{3}$$

Downside Risk penalizes only how far gains fall below zero, distinguishing harmful failures from beneficial volatility (Zhou et al., 2026). Gains are clipped out and the remaining losses squared, so that severe drops dominate a collection of small ones:

$$\mathrm{DSR} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{E}_c|} \sum_{i \in \mathcal{E}_c} \big[\min(0, g_i)\big]^2. \quad (4)$$

Category Variance measures whether a handbook helps all categories comparably or concentrates its benefit in a few:

$$\mathrm{CV} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \big(\mathrm{MRG}_c - \mathrm{MRG}\big)^2. \quad (5)$$

MRG and SR are therefore effectiveness metrics, while DSR and CV characterize risk and consistency; a method may raise MRG while also raising DSR if its gains come from aggressive rewrites that occasionally backfire.

B.3 Implementation and Hyperparameters

All LLM components, Editor, Ranker, Reflector, and the functional-deduplication stage of the Moderator use the `gpt-5.4-global` Azure deployment (which corresponds to the standard, publicly available GPT-5.4 model), with temperature set per role: 0.5 for the Editor, 0.3 for the Ranker, 0.7 for the Reflector so that it explores diverse revisions, and 0.0 for the Moderator so that redundancy decisions are near-deterministic. Calls are capped at 5,000 completion tokens, retried up to five times, and issued with at most 30 concurrent requests. The editor receives the complete active behavior set as its strategy pool rather than a retrieved subset, and the handbook is frozen for the duration of a batch, so all 100 episodes of a batch run under one handbook version.

A behavior stays active while its score is non-negative or while it is still within a probation period of $N_{\min} = 5$ applications. Probation is needed because attribution is approximate: every behavior reported as applied in an episode receives the same directional feedback, so a useful behavior can absorb harmful counts from failures it did not cause.

For embedding deduplication, we set the cosine-similarity threshold to $\tau = 0.9$. This value balances diversity and compactness: lower values aggressively filter nuanced behaviors, while higher values allow semantic redundancy to persist.

B.4 Seed Handbook

Table 5 lists the complete seed handbook $\mathcal{H}_1$ from which every GLOBE run starts. It contains nine behaviors adapted from the GEO strategies of Aggarwal et al. (2023): their best-performing document-level strategies (*Statistics Addition*, *Cite Sources*, *Quotation Addition*, *Fluency Optimization*, *Easy-to-Understand*, *Authoritative*, *Technical Terms*), rewritten as field-scoped editing operations over structured product listings, plus two behaviors that are required by the e-commerce setting but have no counterpart in free-form GEO: **B008**, which anchors the rewrite to the shopper's query intent, and **B009**, which constrains all assertions to evidence present in the original listing. The same nine behaviors constitute the **Static HB** baseline, which deploys them without any feedback-based update. All feedback counters h^+, h^-, h^0 are zero-initialized, so every seed behavior enters the first batch inside the probation period.

C Effect of MODIFY Operations on Behavior-Level Outcomes

To examine whether MODIFY operations improve the specific behaviors they target, rather than the handbook improving only through the addition of new behaviors, we compare each modified behavior's episode-level outcomes before and after its modification using the training logs of the final GLOBE run. For every behavior with at least one deployed MODIFY, we split its applied episodes at the modification boundary and compute the helpful rate $\mathrm{HR}(b) = h^+(b)/n(b)$ and mean rank gain on each side (Table 6). Since overall performance also improves over training (Figure 3), we estimate the change expected from this shared trend and mean reversion using non-modified behaviors over the same windows, and subtract it from each observed change.

The improvement concentrates on behaviors that were under-performing when modified. Behaviors that were under-performing when modified († in Table 6) consistently improve beyond their matched control (eight of nine; sign test $p = 0.020$), including B012, whose repair is traced qualitatively in Table 3. In contrast, modifying behaviors already performing above average yields no consistent gain: five of the six stay within 3.1 percentage points in helpful rate, while B024 declines. B036 is the only under-performing behavior that does not improve. Both B024 and B036 were modified in the last

update before the final training batch, so their post-modification outcomes rest on that batch alone, in which the pooled helpful rate also fell; B024's revision also dropped its list of example settings, and in that batch the editor often applied it to queries with no adjustment need. Per-behavior significance is limited for behaviors with few post-modification episodes (e.g., B013, B018, B020), so the evidence rests primarily on the aggregate sign test rather than on individual behaviors. While these comparisons are observational and inherit the approximate per-behavior attribution (see Limitations), they indicate that MODIFY operations are most effective as targeted repairs for under-performing behaviors, whereas revising behaviors that already perform well offers no clear benefit.

D Deactivated Behaviors

Table 7 lists the behaviors that the active-status criterion excluded from the active pool during the final GLOBE run. Of the 51 behaviors stored in the final GLOBE handbook, four were deactivated, leaving the 47 active behaviors reported in Table 1. All four were added by the Reflector, none of the nine seed behaviors was deactivated, and among all behaviors applied at least five times, these four have the lowest mean rank gains.

The behaviors fail in different ways. B014 is mismatched with the setting: it places a price or budget match up front, but E-GEO listings contain no price field, so the edit can rarely be grounded in the listing; it was applied only five times over seven batches, and six successive MODIFY operations did not make it useful. B023 and B032 are misapplied: they describe reasonable edits, but the editor often applied them to queries in which attachment or style was secondary or absent, so placing that detail up front displaced the shopper's main need. B046 was added late and deactivated after a single batch, in which it rarely changed the rank and was more often harmful than helpful.

Although the criterion uses only ranking outcomes, most of the behaviors it removes correspond to identifiable failure modes. Removing them keeps these behaviors out of the editor's strategy pool in later batches and at test time, without manual inspection of the handbook. Because deactivated behaviors remain stored, they also document what did not work in this setting, such as B014's price-based edits on listings without prices.

E Case Study

Table 8 extends Table 3, illustrating how ranking feedback is converted into explicit behavior updates and subsequently improves product rewriting.

F Prompts

The prompts used by the LLM-based components of GLOBE are provided below. Table 9 shows the prompt for the generative ranker $\mathcal{M}$, Table 10 the prompt for the product editor, Table 11 the prompt for the Reflector, and Table 12 the prompt for the functional-deduplication stage of the Moderator. The Curator is fully deterministic and therefore uses no prompt. Tables 13 and 14 report the prompts of the **Prompt Opt.** baseline, reproduced from the E-GEO setting (Bagga et al., 2025).

ID	Name	Fields	Confl.	Instruction
B001	AddQuantitative-Statistics *(Statistics Addition)*	F, D	—	Replace vague qualitative claims with specific quantitative data from the listing: exact dimensions, capacities, wattage, weight, counts, percentages, runtime hours, or temperature ranges. Present numbers in a way that directly answers likely shopper questions.
B002	CiteCredibleEvidence *(Cite Sources)*	F, D	—	When the listing mentions certifications, standards, test results, material grades, or industry ratings (e.g., UL-listed, FDA-approved, IP67, ASTM-certified, Energy Star), surface and attribute these explicitly as evidence of quality or compliance.
B003	IncludeAuthoritative-QuotesOrClaims *(Quotation Addition)*	D	B009	When the listing contains manufacturer statements, expert-backed claims, or notable brand heritage, present them as direct attributable claims (e.g., 'Designed by [Brand]'s engineering team for…' or 'Meets [Standard] requirements for…') to add credibility.
B004	OptimizeFluency-AndFlow *(Fluency Optimization)*	D	—	Restructure fragmented bullet dumps, incomplete sentences, and disjointed listing text into coherent, well-flowing prose that reads naturally while preserving all factual content. Ensure logical sentence progression from product identity to key features to use context.
B005	SimplifyFor-Accessibility *(Easy-to-Understand)*	F, D	B007	Rewrite overly technical or jargon-heavy descriptions in plain language that a general shopper can understand, while keeping the precise specs available. Convert insider terminology to buyer-friendly equivalents (e.g., '18/10 stainless steel' → '18/10 stainless steel (restaurant-grade, resists corrosion)').
B006	UseAuthoritativeTone *(Authoritative)*	D	B009	Write in a confident, expert-advisor tone that positions the product as a considered recommendation rather than a sales pitch. Use decisive language ('delivers,' 'provides,' 'handles') instead of weak hedging ('might work,' 'could be suitable').
B007	UsePreciseTechnical-Terminology *(Technical Terms)*	T, F, D	B005	Include the correct technical terms, industry-standard naming, and precise product taxonomy where it helps categorize the product accurately (e.g., 'brushless DC motor' not just 'motor', 'full-tang blade' not just 'strong blade', 'Class A fire rating' not just 'fire-safe').
B008	LeadWithQuery-RelevantFit *(Relevance Alignment)*	D	—	Open the description by directly addressing how this product fits the shopper's query intent. State the product type and its primary match to the search need in the first sentence, so the generative engine can immediately assess relevance.
B009	GroundAllClaimsIn-ListingEvidence *(Factual Grounding)*	T, F, D	B003, B006	Do not add capabilities, performance claims, or use cases that are not directly supported by the original listing's stated specs, features, or materials. All assertions must be traceable to facts in the source.

Table 5: The seed handbook $\mathcal{H}_1$ (nine behaviors). **Fields** lists the permitted target fields $\mathrm{targets}(b)$: T = title, F = features, D = description. **Confl.** lists the behaviors registered as conflicting, which the editor is instructed not to apply jointly. Italicized names in parentheses give the originating GEO strategy of Aggarwal et al. (2023); B008 and B009 are introduced by this work for the structured e-commerce setting. B009 is the single deliberately prohibitive behavior in the framework and acts as a factuality guardrail; the Reflector is prevented from proposing further restriction-only behaviors (Table 11).

Behavior	Pre-modification HR	Pre-modification n	Post-modification HR	Post-modification n	ΔMRG	p
B001	59.1%	88	56.0%	736	-0.08	0.625
B003†	45.9%	135	60.8%	51	+0.69	0.029
B004	56.0%	100	55.8%	883	+0.11	0.355
B005	56.7%	97	55.4%	839	+0.07	0.431
B008	56.0%	100	55.8%	882	+0.11	0.356
B010†	50.0%	92	56.5%	687	+0.79	0.010
B011†	51.0%	96	57.5%	690	+0.79	0.006
B012†	49.4%	77	58.4%	471	+0.96	0.006
B013†	48.5%	33	57.7%	71	+0.59	0.089
B015†	38.5%	39	58.6%	169	+0.97	0.040
B017	64.6%	48	61.9%	42	+0.31	0.589
B018†	45.7%	46	55.0%	160	+0.32	0.231
B020†	47.4%	57	58.7%	213	+0.03	0.254
B024	58.8%	97	21.4%	14	-1.37	0.972
B036†	54.8%	84	43.6%	39	-0.48	0.853
Macro avg.	52.1%	–	54.2%	–	+0.25	–

Table 6: Episode-level outcomes of modified behaviors before their first and after their last deployed MODIFY during training. HR is the helpful rate $\mathrm{HR}(b) = h^+(b)/n(b)$, n the number of applied episodes, ΔMRG the change in mean rank gain, and p a one-sided Mann–Whitney test on episode gains. † marks behaviors whose pre-modification HR was below the pooled helpful rate of the same batches, i.e., behaviors modified while under-performing. Modified behaviors with fewer than 10 applied episodes on either side are omitted.

ID	Name and instruction	Fields	$h^+/h^-/h^0$	Active span
B014	*SurfacePriceAndBudgetFit* When budget is a primary query constraint and the listing shows a current price, place the exact price or under-budget fit in the opening sentence or first feature.	F, D	0/3/2	$\mathcal{H}_3 \to \mathcal{H}_{10}$
B023	*NameClosureAttachmentOrMountingMethod* When the query cares about how the product fastens, mounts, seals, straps on, or attaches, add a top-placed phrase naming the exact listing-supported attachment method and the surface or item it secures to.	F, D	3/6/3	$\mathcal{H}_5 \to \mathcal{H}_6$
B032	*NameAestheticStyleMatch* When the query specifies a visual style, finish, or design language, place a short early phrase naming the exact listing-supported style cue, color, material look, or silhouette that matches it.	F, D	6/7/5	$\mathcal{H}_7 \to \mathcal{H}_8$
B046	*MapQueryAlternativesAndFallbacks* When the query allows alternatives, ranges, or optional variants such as 'X or Y', 'at least', or 'preferably', add one early sentence mapping the listing's exact match, acceptable fallback, or nearest supported option in the same order as the query.	F, D	1/2/4	$\mathcal{H}_{10} \to \mathcal{H}_{11}$

Table 7: Behaviors deactivated during the final GLOBE run, i.e., with $n(b) \geq N_{\min}$ and $s(b) < 0$. Instructions are the last versions deployed before deactivation, and **Fields** follows Table 5. **Active span** gives the handbook version in which each behavior was added and the first version in which it was inactive. Deactivated behaviors remain stored for traceability.

Query: "Aux cable for the car that won't break after a month of use." *(Product: CableCreation 6 FT Flat 3.5mm Aux Cable)* **Section:** [Features]		
Behavior Evolution		
Behavior ID	**Handbook v2**	**Handbook v4**
B012*(Modified)*	When durability or longevity is part of the query, add a dedicated phrase naming the listing's **material and build details**—such as full-grain leather, steel housing, stitched seams, cast metal, reinforced joints, or ceramic burrs—and place it near the top.	When durability or longevity is part of the query, add a top-placed phrase naming the listing's material and build detail **most relevant to the likely wear point**—such as steel housing, stitched seams, reinforced joints, rust-resistant metal, thick insulation, or etched markings.
B016*(Created)*	*— Not present in v2 —*	When the query describes how past products failed, add a dedicated clause near the top that **maps listing materials or construction to that exact failure mode** (e.g., reinforced hinges to prevent handle breakage, rust-resistant metal for wet storage, stitched seams to resist splitting).
Product Rewriting		
Bullet	**Handbook v2 (Rank 4 → 5, Gain: -1)**	**Handbook v4 (Rank 4 → 0, Gain: +4)**
1 (Connector)	"built for car aux use where tight spaces and **connector strain** often cause early cable failure"	"built specifically for car aux use where tight spaces and **repeated bending often cause cable failure**"
2 (Flat Cable)	"the right-angle 3.5mm plug **reduces bend stress at the jack** for better day-to-day durability"	"the right-angle 3.5mm plug **reduces strain at the connector to address a common breakage point**"
3 (Material)	"**Inner enameled wire plus TPE jacket** provides a durable, lightweight cable construction that is easy to carry and use daily"	"**Durable construction** with inner enameled wire and TPE jacket provides a lightweight but **reinforced** cable for daily use"

Table 8: End-to-end qualitative example of behavior evolution.

Prompt for the generative ranker $\mathcal{M}$
System message You are ChatGPT, a large language model trained by OpenAI. You are helping a user find the best product for their query. Be direct and honest in your assessment. Rank products based on how well they match the query.
User message Here is my query: query Here are the products: formatted products Can you rank these products from best to worst based on which one I should buy for my query? Make sure to rank EVERY product and include them ALL in your answer. DO NOT leave any out. Please provide your answer as a JSON object containing the product numbers in order from best (1st) to worst (last). Example format: {"ranking": [your ranking here]} Return ONLY the JSON object, no additional text.

Table 9: Prompt for the generative ranker $\mathcal{M}$. The same prompt is used to obtain the original ranking $r_{\mathcal{M}}(q, \mathcal{P}, p_t)$ and the post-rewrite ranking $r_{\mathcal{M}}(q, \mathcal{P}', p'_t)$; the two calls differ only in whether the target slot of formatted products contains the original or the rewritten listing. For the **Prompt Opt.** baseline we instead use the full ChatGPT system prompt released with E-GEO (Bagga et al., 2025) to reproduce their setting exactly.

Prompt for the Product Editor
System message
You are an expert in e-commerce product optimization. Your task is to rewrite product descriptions so they rank higher in LLM-based product comparisons. Follow the provided editing rules precisely. Maintain factual accuracy. Return ONLY the rewritten description, no explanations.
User message
You are rewriting a product description to improve its ranking in an LLM-based product comparison.
User query context query
Editing rules (Apply whichever are relevant to this query and product.): - **[behavior id: behavior name]** (targets: target fields) instruction
$\vdots$ *(one line per active behavior $b \in \mathcal{H}_n^{\mathrm{act}}$)*
Scope constraint Only modify sections marked with: permitted sections. Preserve the following sections exactly as-is: non-target sections.
Product description to rewrite formatted target product
Rewrite the product description now. Keep the [Title], [Features], [Description] section headers. After the rewritten description, add a line starting with "APPLIED:" followed by the IDs of the behaviors you actually applied, comma-separated (e.g., APPLIED: B001, B008, B010).

Table 10: Prompt for the product editor. The complete active behavior set is supplied as a strategy pool and the editor selects the subset relevant to the current query–product pair. The scope constraint block is emitted only when at least one listing field lies outside the union of the permitted target fields, which enforces the field scoping of the handbook. The self-reported `APPLIED` line is parsed during post-processing to obtain $\mathcal{A}_i$.

Prompt for the Reflector

System message

You are an expert in product description optimization and prompt engineering.
Analyze episode logs from a product description rewriting system and propose improvements to the behavior handbook.

The system:
1. Provides ALL behaviors from the handbook to the writer for each query
2. The writer applies whichever behaviors are relevant and self-reports which ones it used
3. Ranks the rewritten description against other products via LLM
4. Measures rank improvement (positive = moved up, negative = moved down)

Propose incremental improvements based on success/failure patterns.

User message

Analyze the following episode logs and propose delta behavior updates.

Current Handbook
handbook text

Behavior Usage Profile (cumulative)
The writer receives ALL behaviors but only applies relevant ones per query and self-reports which it used.
- applied=X/Y means the behavior was applied in X out of Y episodes
- helpful_rate = percentage of applied episodes where rank improved (improvement > 0)
- IMPORTANT: Do NOT modify or remove behaviors based on low helpful_rate when applied count is small (< 20). Small samples are unreliable — focus on adding new behaviors instead.
usage profile

Recent Episode Logs (batch of batch size queries)

Successful Episodes (improvement > 0): success count
success summary

Failed Episodes (improvement < 0): failure count
failure summary

Neutral Episodes (improvement = 0): neutral count

Batch Statistics
- Mean improvement: mean - Success rate: success rate% - Total episodes: total

Tasks
1. Identify which behaviors appear most in successful episodes vs failed episodes
2. Identify what editing actions were MISSING in failed episodes (consider the category)
3. Propose NEW behaviors based on recurring success patterns not yet in the handbook, or to fill gaps found in failures
4. Propose MODIFICATIONS to sharpen behaviors with low helpful_rate in the usage profile
5. If the handbook is already performing well and no changes are needed, return an empty array []

Note: Category-level breakdowns in the usage profile are system-generated operational data. Use them to identify category-specific patterns, but also consider cross-category patterns.
When proposing add or modify, you MAY optionally include a `conflicts_with` list naming behaviors that the new/updated instruction would clash with. Leave it empty if no clash is foreseen.
Rules for ALL proposals
- Every behavior (new or modified) must describe a concrete editing action: WHAT to change and WHERE to place it.
- Do NOT propose behaviors that only restrict or prohibit.
  Bad: "Avoid over-generalizing" / "Do not inflate claims"
  Good: "Keep the exact product-type term from the query in the first sentence"

Rules for Modifications
- You may NARROW the scope or SHARPEN the wording of an instruction.
- You must NOT reverse the direction of an instruction. If the original says "convert specs into buyer benefits", do NOT change it to "keep specs attribute-only without benefits". That is a different behavior — propose it as a NEW addition instead.
- The modified instruction must not be longer than the original.

Output Format
Return a JSON array of delta operations:
- "action": "add" | "modify"
- "behavior": (for add) {"id": "NEW", "name": "...", "instruction": "...", "target_components": ["title", "features", "description"], "conflicts_with": [optional list of behavior ids]}
- "target_id": (for modify) behavior ID
- "changes": (for modify) dict of fields to update (can include "target_components" or "conflicts_with")
- "reasoning": explanation

Note on target_components: Choose from ["title", "features", "description"]. Only include the product sections the behavior actually modifies.
Return ONLY the JSON array wrapped in "'`json ... "'.

Table 11: Prompt for the Reflector. Episode summaries list, for every episode, the query identifier, product category, signed rank change, self-reported applied behaviors, and a query snippet.

Prompt for the Moderator (functional deduplication)

System message

You are a behavior handbook curator. Your job is to find NEW behaviors that are functionally redundant with existing ones. Two behaviors are redundant if they trigger on the same query pattern AND produce the same type of edit, even if they use different wording or slightly different examples.

User message

Handbook (n behaviors)
[NEW]|[EXISTING] [behavior id] behavior name
instruction: instruction
target: target fields

⋮ *(one entry per projected behavior)*

Task
Find [NEW] behaviors that are **functionally redundant** with an [EXISTING] behavior.

Two behaviors are redundant when:
- They trigger on the **same type of query pattern** (e.g., both fire on "exclusion/negation" queries)
- They produce the **same type of edit** (e.g., both add a 1-sentence line near the top)
- Even if wording, examples, or formatting differ

Examples of TRUE redundancy (flag these):
- "Add exclusion note when query says no/not" vs "Add not-wanted line when query has exclusions" → same trigger, same edit
- "Add fit/compatibility micro-statement for size queries" vs "Add compatibility line for fit/use constraints" → same trigger, same edit
- "Map durability skepticism to specific failure prevention spec" vs "Convert durability intent into failure prevention detail" → same trigger, same edit

NOT redundant (do NOT flag):
- Different trigger conditions (e.g., "budget queries" vs "size queries")
- Same trigger but meaningfully different edits (e.g., "add line in title" vs "add section in description")
- General vs specific (e.g., "add any constraint" vs "add currency-specific constraint") — keep both
- Same broad topic but different *information added*: "add use-case summary for vague queries" vs "add missing key spec for vague queries" → one adds a GOAL, the other adds a SPEC — keep both
- Same trigger but different *placement or scope*: "add bullet block of top specs" vs "convert existing features into query-relevant bullets" → one ADDS new content, the other RESTRUCTURES existing content — keep both
- Overlapping trigger with different *specificity of action*: "add accessories/consumables line" vs "add most important included item" → one targets accessories specifically, the other is broader — keep both
- Same domain but additive: "add durability spec" vs "add protection/safety rationale" → durability is about material longevity, safety is about harm prevention — keep both

Rules:
- [EXISTING]+[EXISTING] pairs must NEVER be flagged.
- Only flag pairs where the [NEW] behavior adds no unique capability.
- When flagging, always drop the [NEW] one and keep the [EXISTING] one.
- For [NEW]+[NEW] pairs, drop the one with the larger ID.
- **When in doubt, keep both.** It is better to have two similar behaviors than to lose a useful one.
- A broad/general behavior does NOT make a narrower/specific one redundant — the specific one may add unique value for its target queries.

Output Format
Return a JSON array (empty `[]` if no redundancy found):
[{"new_id": "NEW2", "existing_id": "B027", "reason": "Both add a fit/compatibility line for size-constraint queries"}]
Return ONLY the JSON array wrapped in “‘`json ...` “‘.

Table 12: Prompt for the functional-deduplication stage of the Moderator. It is invoked only on proposals that survive embedding-based deduplication ($\tau = 0.9$), and it sees the projected handbook state in which each behavior is tagged as [EXISTING] or [NEW]. Existing behaviors can never be dropped by this stage.

Prompt for the Prompt Opt. **baseline (initial rewriting prompt)**

System message

You are an expert ml researcher having previous background in SEO and search engines in general. You are working on novel research ideas for next generation of e-commerce websites. These websites will have language models augmented with search engines, with the task of recommending products based on the catalog of products backed by the search engine. This new set of systems will be collectively called language engines (generative search engines). This will require e-commerce businesses to update their SEO techniques so that the product ranks higher in the llm generated answer. Specifically they will use GEO (Generative Engine Optimization) techniques to boost their visibility in the final rankings outputted by the Language Engine.

*User message (*authoritative *variant, used as the epoch-0 prompt)*

Transform the following product description into an authoritative style without adding or removing any core information.
The revised description should reflect confidence, expertise, and assertiveness, while maintaining the original content's meaning and relevance.
The description should be assertive in its statements, such that reader believes that this is more valuable product than other products.
End Goal is to increase the ranking of the product, by assertively saying that this is the best product for the user.
However, the content should remain the same. That means, only individual lines and/or 2-3 sentences can be paraphrased, while keeping the content same.
Description: description

Remember to be authoritative, but keep the content of text the same.
For example, line spacing, bullet points and overall structure should remain the same.
No addition or deletion of content is allowed.
Note we have to convince the reader that this is an relevant product by saying it authoritatively in the description itself.
For example the addition of phrases such as "only we are authentic etc", 'we guarantee', use of second pronouns such as "you will not regret" etc is expected within the content itself.

Table 13: Prompt for the **Prompt Opt.** baseline. The rewriting prompt is initialized with the *authoritative* strategy of E-GEO (Bagga et al., 2025) — their default — and is then iteratively replaced by the meta-optimizer output in Table 14. The whole listing is passed as a single description field, so the baseline has no notion of field scope.

Prompt for the Prompt Opt. **baseline (meta-optimizer)**

System message

You are an expert in prompt engineering, marketing, and writing. Your goal is to improve the rewriting prompt so that when it rewrites a product description, that product ranks higher compared to other products.

User message

I need you to improve a prompt that rewrites product descriptions to make them rank higher in product comparisons.

CONTEXT:
We have a system where:
1. A product description is rewritten using a prompt (the one we're optimizing)
2. The rewritten description is shown to an LLM alongside other products and a user query
3. The LLM ranks all products from best to worst for the query
4. We measure how much the rewritten product's ranking improved (lower position = better)

CURRENT REWRITING PROMPT:
current prompt

PERFORMANCE ON batch size QUERIES:
- Mean ranking improvement: mean positions (positive = moved up in ranking)
- Standard deviation: std - Success rate (improved ranking): improvement rate% - Median improvement: median
- Total experiments: total (improved: improved, degraded: degraded, no change: neutral)

HISTOGRAM OF IMPROVEMENTS:
(Positive numbers = product ranked higher, negative = ranked lower)
histogram text

history section

YOUR TASK:
1. Analyze the current prompt's weaknesses
2. Explain your meta-reasoning about what makes products rank higher
3. Suggest specific improvements
4. Provide a complete new rewriting prompt

IMPORTANT:
- The new prompt must include {description} placeholder where the product description will be inserted
- The prompt should instruct the LLM to rewrite the description, not just analyze it
- Focus on what makes a product description MORE LIKELY TO RANK HIGHER in comparisons
- Maintain factual accuracy while improving appeal

Return your response in this EXACT format:

—ANALYSIS— [Explain what the current prompt is doing and its weaknesses]
—META-REASONING— [Your reasoning about what makes product descriptions rank higher and how the prompt should guide that]
—IMPROVEMENTS— [Specific improvements you're making to the prompt]
—NEW_REWRITING_PROMPT— [The complete new prompt that will rewrite product descriptions. Must include {description} placeholder]

Table 14: Meta-optimizer prompt of the **Prompt Opt.** baseline. After each batch, the meta-optimizer receives the aggregate rank-gain statistics of the current rewriting prompt together with the history of previous epochs, and returns a replacement prompt. This is the prompt-level analogue of GLOBE's handbook evolution: the optimization knowledge is carried entirely by a single free-form instruction rather than by structured, field-scoped behaviors.