LDCO: EVIDENCE-GROUNDED PREFERENCE TRANSFER FOR USER-SIDE COLD-START RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recommender systems connect users with products at scale, yet they remain largely platform-centered and dependent on target-domain histories. When a user enters an unfamiliar domain with an ambiguous query, three difficulties arise together: target-domain cold start leaves little evidence for personalization, the query omits relevant preferences, and the platform's ranking may not match the user's preferred ordering. We address these coupled problems with Layered Dimension-Conditioned Query Optimization (LDCO), a user-side large language model agent framework that operates through an existing retrieval interface. LDCO constructs domain-level and global profiles from authorized source-domain histories, selects relevant evidence, and maps transferable preferences to target-product attributes. It then uses Monte Carlo tree search on a fuzzy-query dataset to optimize the query-generation prompt through retrieval feedback, and reranks the returned candidates by query–profile relevance. On a user-disjoint Amazon benchmark, LDCO raises Hybrid Hit@100 from 0.3169 to 0.4155, MRR@100 from 0.0365 to 0.0758, and NDCG@100 from 0.0872 to 0.1388 over retrieval with the original fuzzy queries. Under live Amazon retrieval, the full framework raises Precision@15 from 0.2628 to 0.4162. These results show that evidence-grounded preference transfer, retrieval-aware query optimization, and user-side reranking jointly improve cold-start recommendation without modifying the platform's retrieval model.

1 INTRODUCTION

Recommender systems help users navigate large collections of products and content, creating huge value by connecting users with target products and content (Roy & Dutta, 2022; Lin et al., 2025). Shopping platforms use interaction histories and product information to connect consumers with relevant items at scale. As users encounter unfamiliar product categories, however, effective recommendation requires more than identifying generally relevant products. It must connect an incompletely expressed need with the preferences of a particular individual. This motivates a complementary direction for recommendation: user-side assistants that carry personal preferences across domains, communicate those preferences to existing platforms, and organize the returned results around the user's current needs (Xu et al., 2025).

Consider a user who asks, "I want to buy wireless earbuds." Three difficulties arise together. First, the query identifies a product category but leaves preferences such as fit, ease of operation, and durability unspecified: this is *query ambiguity*. Second, the user may have extensive histories in books, clothing, and household products but no electronics interactions: this is *cross-domain cold start*. Third, the platform's ordering may reflect objectives such as engagement, conversion, and advertising revenue alongside relevance, whereas the user seeks items suited to their own requirements: this creates a potential *user–platform objective mismatch* (Jannach & Abdollahpouri, 2023; Zhao et al., 2021). Jointly addressing these difficulties calls for a new approach that integrates cross-domain preference modeling, query optimization, and user-oriented ranking within a unified framework. A useful assistant works by determining what can be inferred about this user, expressing the relevant preferences in a searchable query, and prioritizing suitable items among the returned candidates.

Substantial progress provides the foundations for such an assistant. Cross-domain recommendation (CDR) transfers knowledge between domains: EMCDR learns mappings between domain-specific

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.

embeddings, and CoNet transfers representations through connected networks (Man et al., 2017; Hu et al., 2018). More recent approaches use large language model (LLM)-derived profiles and semantic knowledge to support cross-domain recommendation (Liu et al., 2025a;b). LLM-based profile management and memory mechanisms further demonstrate how textual histories can support personalization (Bang & Song, 2025; Wang et al., 2025a; Huang et al., 2025). In parallel, query-rewriting methods enrich incomplete requests using conversational context or retrieval feedback (Ye et al., 2023; Wang et al., 2025c; Zhang et al., 2025b), while iAgent establishes a user–agent–platform paradigm for instruction-aware recommendation (Xu et al., 2025). Together, these advances make it feasible to place a personalized assistant between a user and an existing shopping platform.

However, an unresolved issue is how to coordinate these capabilities to address the above three difficulties without modifying the platform's models. A cross-domain profile is useful only if its preferences can be expressed as attributes relevant to the current product. Query rewriting can clarify a request yet introduce generic or unsupported attributes when personal evidence is missing. Reranking can prioritize user-relevant candidates, but it cannot recover suitable products that were never retrieved. Conversely, better retrieval does not determine how the returned products should be ordered for the individual. Therefore, the research question is: *How can a user-side assistant jointly address query ambiguity, cross-domain cold start, and objective mismatch using authorized personal histories and an existing platform retrieval interface?*

Addressing this question requires more than concatenating a profile, a rewriting prompt, and a reranker. Source-domain preferences differ in their relevance and level of abstraction. For the earbuds request, repeated preferences for comfortable wear and straightforward operation in clothing and household-product reviews may support target attributes concerning fit and simple pairing; a preference for a particular book genre may contribute little. An overly general profile loses these distinctions, whereas unrestricted transfer risks adding irrelevant constraints. Moreover, a fluent rewrite may not retrieve better products, and platform relevance alone may not reflect the user's priorities. Consequently, the assistant must connect evidence selection, target-specific expression, and final ordering while preserving the explicit request.

In light of this, we propose *Layered Dimension-Conditioned Query Optimization (LDCO)*, an LLM-agent framework that coordinates cross-domain profile construction, retrieval-aware prompt optimization, and user-side reranking. To address cold start, LDCO summarizes source histories into domain-level profiles and combines recurring cross-domain preferences into a global profile. Given a request, it identifies the target domain, selects relevant source profiles, and maps supported preferences to target-domain attribute dimensions. To address query ambiguity, we adapt the Monte Carlo tree search (MCTS) procedure to optimize the query-generation prompt using retrieval feedback (Zhang et al., 2025a). Candidate prompts are evaluated on a fuzzy-query dataset constructed from held-out Amazon interactions, and the selected prompt is frozen before testing. Thus, MCTS optimizes the rewriting strategy offline, while online rewriting combines the current query with profile-supported attributes. For the earbuds example, this can yield a query emphasizing comfortable fit and easy pairing when the history supports those preferences. To address objective mismatch at the ranking stage, an LLM reranks the retrieved candidates according to the personalized query and global profile, introducing an explicit user-side relevance criterion.

This paper makes the following contributions: 1) We formulate a user-centric recommendation setting in which query ambiguity, target-domain cold start, and potential objective mismatch must be addressed together through query generation and candidate reranking. 2) We develop LDCO to connect hierarchical user profiles with target-domain attribute mapping, MCTS-based query-prompt optimization, and preference-conditioned reranking, without updating the platform's retrieval model or index. 3) We evaluate the framework using a user-disjoint Amazon benchmark and live-platform retrieval, examining target-item recovery and ranking quality as complementary outcomes.

On the offline benchmark, comprising 568 prompt-optimization users and 142 held-out test users, the complete framework increases Hybrid Hit@100 from 0.3169 to 0.4155, MRR@100 from 0.0365 to 0.0758, and NDCG@100 from 0.0872 to 0.1388 relative to retrieval with the original fuzzy queries. In the Amazon live-retrieval evaluation with LLM relevance judgments, query optimization increases Precision@15 from 0.2628 to 0.3324, and the reported result after reranking is 0.4162. These results provide evidence of improved retrieval and assessed relevance in the evaluated settings. They support further investigation of user-side assistants that combine persistent personal preferences with current requests while retaining access to existing platform catalogs.

2 SYSTEM MODEL

Platform-Centric Recommendation. We first describe the traditional recommendation platform interface and then define the information available to the user-side agent. A shopping platform maintains a product catalog and uses available user interactions, item information, and current context to generate recommendations. A request may be triggered by browsing the platform. The platform interprets the request, retrieves a candidate subset from its catalog, and ranks the candidates using relevance estimates and its operational objectives. The first K items are displayed, and subsequent clicks, purchases, and ratings can provide feedback for future model updates (Wang et al., 2025b; Jannach & Abdollahpouri, 2023). Let $\mathcal{I}$ denote the catalog and $\mathcal{R}(q, N)$ the platform interface that returns up to N candidates in platform order for query q. This interface encapsulates candidate retrieval and the platform's ranking. However, the platform-centric recommendation relies on target-domain histories and clear user queries. A cross-domain cold-start issue occurs when users enter a target domain with an empty history or express their needs ambiguously.

User Information and Cold-Start Setting. Within this platform setting, personal evidence comes from the user's authorized source histories. For a user u, let $\mathcal{S}_u = \{S_1, \ldots, S_m\}$ denote the source domains with user-authorized interaction histories. Each $\mathcal{H}_u^{S_i}$ contains the available product information and user feedback in domain S_i, such as titles, descriptions, ratings, and reviews. The user submits a natural-language request q_u concerning a target domain T_u. We consider target-domain cold start, which can be formulated as

$$\mathcal{H}_u = \{\mathcal{H}_u^{S_i}\}_{i=1}^{m}, \qquad T_u \notin \mathcal{S}_u \text{ and } \mathcal{H}_u^{T_u} = \varnothing. \tag{1}$$

Here, T_u denotes the domain of the requested product; its prediction from the query is denoted by $\widehat{T}_u$ in Section 3.3. We call q_u an *original fuzzy query* when it expresses a basic product need but leaves relevant attributes or preferences unspecified. For example, "I want to buy wireless earbuds" identifies a category without specifying fit or ease of operation. Notice that query ambiguity concerns incomplete intent expression, whereas cold start concerns missing target-domain history.

User-Centric Recommendation. To use these histories despite the empty target domain, a user-side agent serves the individual user by managing $\mathcal{H}_u$, generating a personalized query q'_u, and reranking the returned candidates. This process is modeled as

$$(\mathcal{H}_u, q_u) \longrightarrow q'_u, \qquad \mathcal{C}_u = \mathcal{R}(q'_u, N), \qquad (\mathcal{H}_u, q'_u, \mathcal{C}_u) \longrightarrow \pi_u. \tag{2}$$

The returned candidate list is $\mathcal{C}_u = (c_{u,1}, \ldots, c_{u,n_u})$, where $c_{u,j} \in \mathcal{I}$ and $n_u \leq N$. The agent produces a reordered list π_u of these candidates; its first $K \leq n_u$ items form the final recommendation. We argue that query optimization can change candidate membership, while reranking changes the ordering within the retrieved set. The agent observes the returned item information but cannot modify the platform's catalog, index, or internal models. The user profile management and the two agent operations are specified in Section 3.

Objective. We aim to improve retrieval and final ranking for an individual user under the user-centric recommendations by jointly addressing three limitations. Cross-domain profile management supplies preference evidence despite the empty target history; MCTS-based prompt optimization improves how an underspecified request and that evidence are expressed for retrieval; preference-conditioned reranking prioritizes candidates according to the individual's needs. The following section implements these complementary operations through the LDCO framework.

3 THE LDCO FRAMEWORK

3.1 OVERVIEW

Figure 1 presents the overview of the user-centric recommendation system from a fuzzy query to personalized recommendations. Given a query, LDCO identifies the target domain, selects relevant source profiles, and maps their preferences to target attributes for query generation; MCTS optimizes the query-generation prompt offline using retrieval feedback; the selected prompt is fixed during online recommendation. The generated query retrieves platform candidates, which user-side reranking then orders by query–profile relevance to produce the final Top-K list.

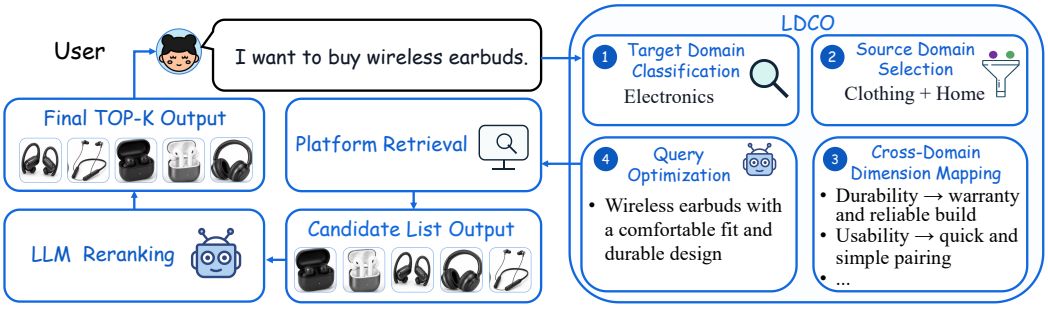


Figure 1: An overview of the user-centric recommendation system. The core module is LDCO, which contains target-domain classification, source-profile selection, and dimension mapping guide query generation. The user-side reranking prioritizes the retrieved candidates.

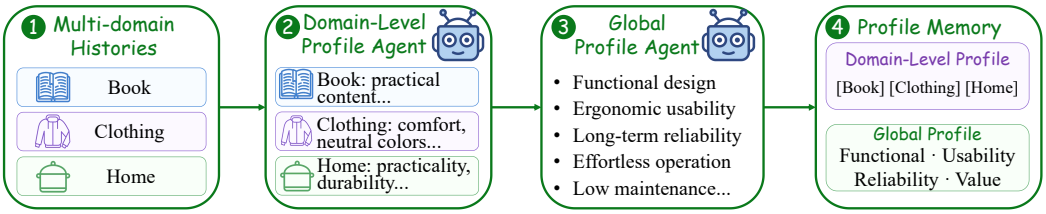


Figure 2: The cross-domain user profile construction pipeline.

3.2 Cross-Domain User Profile Management

The pipeline begins with hierarchical profiles that supply preference evidence when target-domain history is unavailable. As shown in Figure 2, a domain-level profile agent summarizes each source domain history, and a global-profile agent aggregates the resulting profiles:

$$\mathcal{P}_u^{S_i} = g_{\text{domain}}(\mathcal{H}_u^{S_i}; \phi_d), \qquad \mathcal{P}_u = g_{\text{global}}(\{\mathcal{P}_u^{S_i}\}_{i=1}^{m}; \phi_g), \tag{3}$$

where ϕ_d and ϕ_g are the respective generation prompts. Domain-level profiles retain positive and negative attribute preferences from their interaction histories; the global profile summarizes transferable patterns. This separation preserves context while distinguishing local from transferable preferences.

To preserve this distinction during aggregation, evidence constraints limit which preferences enter the global profile. The aggregation prompt normally requires support from at least two source domains for a global preference and retains strong single-domain signals as domain-conditioned information. The current request takes priority whenever it conflicts with historical preferences. Profile inputs, output formats, and generation prompts are provided in Appendix A.

3.3 Layered Dimension-Conditioned Query Optimization

The stored profiles are then translated into request-related attributes. Three layers establish the target context, select source evidence, and map preferences to product attributes before query generation.

Layer 1: Target Domain Classification. Target-domain classification determines the attribute space used for preference transfer. Given an original fuzzy query, the classifier predicts

$$\widehat{T}_u = f_{\text{cls}}(q_u). \tag{4}$$

For "*I want to buy wireless earbuds*," the predicted domain is Electronics, which provides the context for selecting and interpreting source preferences.

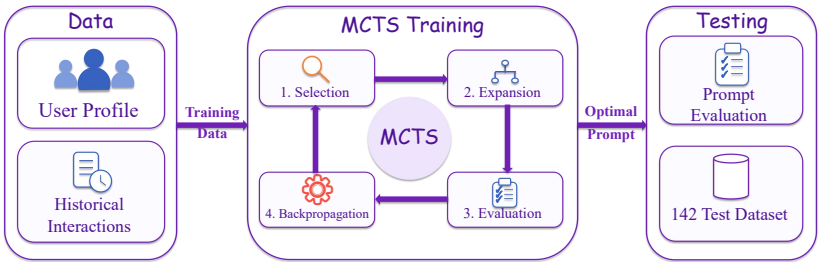


Figure 3: An illustration of the query prompt optimization process.

Layer 2: Source Domain Selection. Conditioned on the predicted target domain, source-domain selection focuses query generation on relevant personal evidence. The selector chooses at most two available domain profiles for the predicted target:

$$\mathcal{P}_u^{\text{sel}} = f_{\text{sel}}\big(\widehat{T}_u, \{\mathcal{P}_u^{S_i}\}_{i=1}^{m}\big). \tag{5}$$

These profiles complement the cross-domain patterns in the global profile.

Layer 3: Cross-Domain Dimension Mapping. The selected evidence is then translated into target-domain attributes through dimension mapping, which filters unsupported transfer. For each applicable dimension, the prompt identifies source evidence, abstracts the underlying preference, and expresses it as a target attribute:

$$\mathcal{M}_u^T = f_{\text{map}}\big(\widehat{T}_u, \mathcal{P}_u^{\text{sel}}, \mathcal{P}_u\big). \tag{6}$$

Here, $\mathcal{M}_u^T$ contains attributes for the predicted target domain and their supporting profile evidence. For example, a documented preference for straightforward operation may support easy pairing for earbuds, whereas an unrelated book-genre preference is discarded. The dimension context and mapping procedure are described in Appendix B.

Query Generation. Using these mapped attributes, the query generator combines the explicit request with supported preferences to produce a personalized retrieval query:

$$q'_u(p) = f_{\text{opt}}\big(q_u, \mathcal{P}_u, \mathcal{P}_u^{\text{sel}}, \mathcal{M}_u^T; p\big), \tag{7}$$

where p is the query-generation prompt. The prompt preserves the requested product and explicit constraints while adding relevant preference terms. For example, LDCO can transform "I want to buy wireless earbuds" into the structured demand {product: wireless earbuds; fit: comfortable; setup: easy pairing; build: durable}, and then into the catalog-facing query "comfortable wireless earbuds with easy pairing and durable build," provided that every added attribute is supported by the user's profiles. Appendix B.3 gives further end-to-end examples. The effectiveness of this formulation depends on the prompt p, motivating the retrieval-based optimization below.

3.4 Query Prompt Optimization with MCTS

To align query generation with retrieval performance, MCTS selects prompts using execution feedback on a fuzzy-query dataset. We adapt AFlow's search procedure so that each node represents a candidate prompt (Zhang et al., 2025a). As shown in Figure 3, starting from a manual seed, the optimizer selects and revises prompts, executes their rewritten queries, and records retrieval rewards to guide subsequent search. Profile construction, classification, source selection, and dimension mapping remain fixed during this search.

The search uses Hybrid Hit@100 to select the candidate prompts that recover held-out target items. For the optimization set $\mathcal{V}$, it selects

$$p^* = \arg\max_{p \in \mathcal{P}_{\text{MCTS}}} J(p), \qquad J(p) = \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} \mathbb{I}[a_u \in \mathcal{R}_{\text{Hybrid}}(q'_u(p), 100)], \tag{8}$$

where a_u is the target item, $\mathcal{P}_{\text{MCTS}}$ contains the visited prompts, and $\mathbb{I}$ is the indicator function. Target identifiers are used only to compute rewards and are withheld from query generation and prompt revision. The selected prompt is frozen before evaluation on disjoint test users. Appendix C details the MCTS algorithm; Appendix D describes the fuzzy-query dataset.

3.5 LLM-BASED USER-SIDE RERANKING

Once the optimized query has retrieved candidates, user-side reranking determines their final order using an individual relevance criterion. After retrieving $\mathcal{C}_u = \mathcal{R}(q'_u(p^*), N)$, the LLM scores candidates against the personalized query and global profile and produces

$$\pi_u = f_{\text{rank}}\big(q'_u(p^*), \mathcal{P}_u, \mathcal{C}_u\big). \tag{9}$$

The first K items form the final recommendation. This step addresses potential objective mismatch by prioritizing query–profile relevance within the retrieved set, complementing query optimization without changing the platform's internal objectives. Appendix B details the reranking process.

4 EXPERIMENTS

We organize the evaluation around four research questions: **RQ1:** Does the full framework improve offline target recovery and ranking over the original fuzzy query across target domains and cutoffs? **RQ2:** How do cross-domain profiles, layered query optimization, MCTS prompt search, and LLM reranking contribute to the gains? **RQ3:** Do query optimization and reranking remain effective under live Amazon retrieval? **RQ4:** How sensitive is the framework to the retrieval backbone, and what online latency and API cost does it incur?

4.1 EXPERIMENTAL SETUP

Implementation. GPT-4o-mini with temperature 0.1 is used as the execution model for profile construction, query generation, reranking, and relevance judging. During offline MCTS prompt search, GPT-5.5 is used exclusively as the optimizer model to propose candidate-prompt revisions; GPT-4o-mini executes each candidate prompt to generate rewritten queries for retrieval evaluation. GPT-4.1 serves only as the online LLM judge under the same protocol for all conditions.

Datasets. The benchmark isolates cross-domain cold start through a user-disjoint split of the five-core Amazon Reviews 2023 dataset (Hou et al., 2024). Books, Clothing_Shoes_and_Jewelry, and Home_and_Kitchen provide 10–40 source interactions per domain and 30–120 interactions per user. For each user, one five-star interaction from the unseen Beauty_and_Personal_Care or Electronics target domain is held out. The resulting 710 users are split into 568 users for MCTS prompt optimization and 142 users for testing, including 69 Beauty and 73 Electronics users.

Baselines. The *Full model* combines hierarchical profiles, all three LDCO layers, the MCTS-optimized prompt, and LLM reranking. The *Fuzzy query* sends the original incomplete request directly to retrieval. *w/o Profile* uses the optimized generation prompt without personal evidence; *w/o Layers 1/2/3* uses the hierarchical profile without target classification, source selection, or dimension mapping; and *w/o LLM rerank* retains the hierarchical profiles and all three LDCO layers but omits LLM reranking. Offline retrieval uses BM25, Vector retrieval with `BAAI/bge-small-en-v1.5` (Xiao et al., 2023); Hybrid is the main backbone.

Metrics. The metrics separately measure candidate recovery and ranking quality. In the offline evaluation, the held-out product is the sole target: Hit@K indicates whether it appears in the first K results, MRR@K measures its reciprocal rank, and NDCG@K discounts its position logarithmically. In live Amazon retrieval, where the historical target may be unavailable, an LLM judge evaluates each returned item against the original request and a fixed reference profile. It assigns binary relevance for Precision@K and MRR@K, and four-level relevance in $\{0, 1, 2, 3\}$ for NDCG@K. We also report the cosine similarity between each generated query and its held-out target product as a retrieval-independent measure of query informativeness.

4.2 OFFLINE PERFORMANCE COMPARISON

Main Results (RQ1). The Full model improves both target recovery and ranking under cross-domain cold start. Table 1 compares the Fuzzy query and Full model across target domains. Hybrid retrieves 500 candidates; the Full model reranks only the top 100, while positions 101–500 retain the original Hybrid order.

Table 1: Performance of the Full model and the Fuzzy query with Hybrid retrieval.

		$K = 100$			$K = 300$			$K = 500$		
Target domain	**Baselines**	**Hit**	**MRR**	**NDCG**	**Hit**	**MRR**	**NDCG**	**Hit**	**MRR**	**NDCG**
Overall	Fuzzy query	0.3169	0.0365	0.0872	0.4789	0.0375	0.1090	0.5352	0.0376	0.1156
Overall	Full model	0.4155	0.0758	0.1388	0.5634	0.0767	0.1585	0.6338	0.0768	0.1667
Beauty	Fuzzy query	0.4058	0.0532	0.1159	0.5507	0.0540	0.1352	0.6087	0.0542	0.1419
Beauty	Full model	0.4203	0.0709	0.1379	0.6232	0.0719	0.1642	0.6812	0.0720	0.1710
Electronics	Fuzzy query	0.2329	0.0207	0.0601	0.4110	0.0218	0.0843	0.4658	0.0220	0.0908
Electronics	Full model	0.4110	0.0805	0.1397	0.5068	0.0812	0.1531	0.5890	0.0814	0.1626

Table 2: Ablation results under Hybrid retrieval at $K = 100$.

Variant	**Profile**	**LDCO**	**LLM rerank**	**Hit@100**	**MRR@100**	**NDCG@100**
Fuzzy query	none	×	×	0.3169	0.0365	0.0872
w/o Profile	none	✓	×	0.3453	0.0458	0.0993
w/o Layers 1/2/3	hierarchical	×	✓	0.3803	0.0745	0.1313
w/o LLM rerank	hierarchical	✓	×	0.4155	0.0577	0.1236
Full model	hierarchical	✓	✓	0.4155	0.0758	0.1388

The gains are consistent across domains and cutoffs, with the largest recovery benefit in the Electronics domain. Overall Hit@100 rises from 0.3169 to 0.4155 (31.1%), MRR@100 from 0.0365 to 0.0758 (107.7%), and NDCG@100 from 0.0872 to 0.1388 (59.2%). Electronics Hit@100 increases from 0.2329 to 0.4110, whereas Beauty shows a smaller Hit@100 gain but stronger improvements at larger cutoffs and in ranking metrics. These results answer RQ1 by showing that profile-grounded query generation recovers more target items and that the complete pipeline places them earlier.

Profile and Query Optimization Ablation (RQ2). The ablation study attributes the overall gain to complementary improvements in request specification, preference transfer, and candidate ordering. We compare the five variants on all 142 test queries using Hybrid retrieval at $K = 100$.

Table 2 shows that the optimized prompt improves Hit@100 from 0.3169 to 0.3453 without personal evidence. The Full model further improves Hit@100 from 0.3803 to 0.4155 and NDCG@100 from 0.1313 to 0.1388 over removing Layers 1/2/3, supporting selective transfer. Reranking preserves Hit@100 at 0.4155 while increasing MRR@100 from 0.0577 to 0.0758 and NDCG@100 from 0.1236 to 0.1388. Thus, query optimization improves candidate recovery, whereas reranking improves candidate ordering.

Prompt Optimization Comparison. MCTS execution feedback produces a query-generation strategy that generalizes beyond the optimization users. Table 3 compares the manual seed prompt with the selected prompt on the disjoint test set at $K = 100$. Both variants use the same hierarchical profiles, LDCO components, Hybrid retriever, and LLM reranker; only the query-generation prompt differs.

Table 3: Prompt-optimization comparison with Hybrid retrieval at $K = 100$.

Prompt	**Hit@100**	**MRR@100**	**NDCG@100**
Original seed prompt	0.3239	0.0347	0.0864
MCTS-optimized prompt	0.4155	0.0758	0.1388

The optimized prompt improves all three test metrics despite being selected only with Hit@100 feedback. Hit@100 rises from 0.3239 to 0.4155, MRR@100 from 0.0347 to 0.0758, and NDCG@100 from 0.0864 to 0.1388. The Hit gain confirms alignment with the search objective, while the MRR and NDCG gains show that the learned rewriting strategy also moves target items earlier.

Table 4: Cosine similarity between queries and held-out target products.

Target domain	Fuzzy query	LDCO query
Beauty	0.5303	0.6063
Electronics	0.5385	0.6428
Overall	0.5345	0.6251

4.3 Online Retrieval Validation (RQ3)

This section tests whether the offline findings survive in Live Amazon retrieval with respect to the query–target similarity and the LLM-as-a-Judge evaluation.

Retrieval Similarity Analysis. We compute cosine similarity between each held-out target product and its original or LDCO-generated query.

Table 4 shows that the similarity gain holds in both target domains and is larger for Electronics. Overall similarity increases from 0.5345 to 0.6251; Beauty rises from 0.5303 to 0.6063, and Electronics from 0.5385 to 0.6428. This domain pattern agrees with the offline recovery results.

LLM-as-a-Judge Evaluation. The LLM evaluates each product against the original fuzzy query and fixed reference profile using the labels defined in Section 4.1.

Table 5: Online retrieval performance on the Amazon platform.

Online condition	Precision@15	MRR@15	NDCG@15
Fuzzy query	0.2628	0.4754	0.8423
LDCO query	0.3324	0.5148	0.8537
LDCO query + LLM reranking	0.4162	0.7571	0.9786

Table 5 compares platform-ranked results for the Fuzzy and LDCO queries with the reranked LDCO results. The live results reproduce the two-stage behavior observed offline. Query optimization raises Precision@15 from 0.2628 to 0.3324 (26.5%), MRR@15 from 0.4754 to 0.5148, and NDCG@15 from 0.8423 to 0.8537. Reranking further raises these metrics to 0.4162, 0.7571, and 0.9786, respectively. Query optimization therefore improves which products are retrieved, while reranking places user-relevant products earlier. This answers RQ3.

4.4 Retrieval Backbone and Efficiency Analysis (RQ4)

This section evaluates the efficiency of the proposed framework. Using the companion Amazon retrieval repository, we evaluate the Full model on the 142 test queries with BM25, Vector retrieval, and their Hybrid combination.

Table 6: Performance of the full model across retrieval backbones at $K = 100$.

Backbone	Hit@100	MRR@100	NDCG@100
BM25	0.3239	0.0805	0.1275
Vector	0.3380	0.0547	0.1074
Hybrid	0.4155	0.0758	0.1388

Table 6 shows that Hybrid provides the strongest candidate pool, achieving the highest Hit@100 (0.4155) and NDCG@100 (0.1388), while BM25 gives the highest MRR@100 (0.0805). Lexical matching can place an exact match early, whereas Hybrid broadens target coverage by combining lexical and semantic evidence; this result justifies its use in the main experiments. Excluding

reusable offline profile construction, the online path from target classification through reranking averages 30 seconds and USD 0.00233 per query. These results answer RQ4 by showing compatibility with multiple backbones and quantifying the current execution cost.

5 Related Work

5.1 Cross-Domain Recommendation and User Profiling

CDR offers a foundation for personalizing recommendations when target-domain interactions are scarce or unavailable (Zang et al., 2023). EMCDR learns mappings between domain-specific user embeddings, while CoNet transfers representations through connected recommendation networks (Man et al., 2017; Hu et al., 2018). These approaches exploit cross-domain interaction data during model training. Recently, LLM-based approaches provide a promising basis for this setting. LLM4CDSR incorporates hierarchical profiles into cross-domain sequential recommendation (Liu et al., 2025a;b). Prompt-based explainable recommendation further demonstrates the use of textual evidence across domains (Petruzzelli et al., 2024). Complementary profiling research includes review-based profile management, recommendation-oriented profile generation, and domain-level aggregation (Bang & Song, 2025; Wang et al., 2025a; Su et al., 2025); MR.Rec connects memory retrieval with reasoning for personalization (Huang et al., 2025). These advances establish the usefulness of textual user representations. Using them for retrieval further requires selecting source-domain evidence and expressing it as relevant target-product attributes. This requirement motivates LDCO's combination of domain-level and global profiles.

5.2 Query Optimization and User-Side Recommendation

Personalized rewriting uses user histories to formulate candidate queries, while conversational rewriting and feedback-based methods enrich incomplete requests with contextual information (Cho et al., 2021; Ye et al., 2023; Wang et al., 2025c). In e-commerce, LESER optimizes query expansion using search-engine feedback, and OptAgent combines simulated shoppers with evolutionary query search (Zhang et al., 2025b; Handa et al., 2025). AFlow further demonstrates how MCTS and execution feedback can optimize agentic workflows (Zhang et al., 2025a). These studies motivate evaluating rewrites through their retrieval outcomes; our adaptation searches query-generation prompts using Hybrid Hit@100, conditions rewriting on cross-domain profile evidence, and freezes the selected prompt before testing. Retrieval improvement, however, addresses only which products become available for recommendation. Multi-objective recommendation research highlights that platform and user priorities may differ (Jannach & Abdollahpouri, 2023), and iAgent introduces an instruction-aware intermediary between the user and platform (Xu et al., 2025). Building on this boundary, LDCO adds preference-conditioned reranking to prioritize retrieved items according to the user's personalized query and profile.

6 Conclusion

This paper introduced a user-centric LLM-agent architecture that personalized recommendations across domains without modifying platform retrieval. LDCO jointly addressed target-domain cold start, ambiguous queries, and potential user–platform objective mismatch by coordinating cross-domain profiles, MCTS-optimized query generation, and preference-conditioned reranking. The experiments showed a clear division of labor: profiles and dimension mapping supplied transferable evidence, optimized queries exposed it to retrieval, and reranking moved user-relevant candidates earlier. On held-out users, the framework raised Hybrid Hit@100 from 0.3169 to 0.4155 and NDCG@100 from 0.0872 to 0.1388; under live Amazon retrieval, it raised Precision@15 from 0.2628 to 0.4162. These gains showed the value of addressing the three problems jointly. More broadly, the results suggested that platforms could provide catalogs and retrieval while a user-side agent carried preferences across domains, offering a practical foundation for portable, user-centric recommendation.

AI USE STATEMENT

In this work, we used generative AI tools for manuscript grammatical polishing, academic sentence refinement, and minor experiment code syntax assistance. We have not used generative AI tools for research idea design, core algorithm design, experimental design, or result reasoning. Generative AI models that serve as part of the proposed algorithm and experimental baselines are research artifacts described in the main paper and do not fall under this disclosure statement.

All AI-assisted text and auxiliary code has been fully reviewed, revised, and verified by the authors to ensure technical correctness. The authors take full and final responsibility for all content, claims, and artifacts presented in this paper.

REPRODUCIBILITY STATEMENT

The main paper specifies the LDCO architecture, optimization objective, datasets, baselines, evaluation metrics, model configurations, and user-disjoint evaluation protocol. The appendices provide the profile inputs and prompts, layered query-generation and reranking interfaces, MCTS search procedure and stopping rule, fuzzy-query construction process, dataset statistics, and Oracle protocol. Together, these materials document the evidence boundaries and implementation steps needed to reproduce the reported offline evaluation and the protocol used for live-platform evaluation; live results may additionally vary with the platform's changing catalog and retrieval system.

ETHICS STATEMENT

This study uses the public Amazon Reviews 2023 dataset for offline evaluation and product listings returned by Amazon for live retrieval; it does not recruit human participants or report directly identifying user information. The benchmark keeps optimization and test users disjoint, and held-out target items are not exposed to profile construction, query generation, prompt revision, or reranking. The proposed user-side setting assumes that individuals authorize which histories the agent may process. This assumption is a system requirement rather than a consent mechanism established by the present experiments. A real deployment should therefore provide informed and revocable authorization, data minimization, secure storage, access control, and deletion of profiles and histories. Because a personalized query can reveal inferred preferences to a retrieval platform, deployments should also disclose this transfer and allow users to inspect or suppress profile-derived attributes before submission. The framework may inherit biases from interaction data, language models, and platform rankings; consequently, its outputs require monitoring and should not be used as the sole basis for consequential decisions. The simulation codes are given in the supplementary material.

REFERENCES

Seunghwan Bang and Hwanjun Song. LLM-based user profile management for recommender system. In *Proceedings of the GENNEXT Workshop at ACM SIGIR*, 2025.

Eunah Cho, Ziyan Jiang, Jie Hao, Zheng Chen, Saurabh Gupta, Xing Fan, and Chenlei Guo. Personalized search-based query rewrite system for conversational AI. In *Proceedings of the Workshop on Natural Language Processing for Conversational AI*, pp. 179–188, 2021. doi: 10.18653/v1/2021.nlp4convai-1.17.

Divij Handa, David Blincoe, Orson Adams, and Yinlin Fu. OptAgent: Optimizing query rewriting for e-commerce via multi-agent simulation. *arXiv preprint arXiv:2510.03771*, 2025.

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.

Guangneng Hu, Yu Zhang, and Qiang Yang. CoNet: Collaborative cross networks for cross-domain recommendation. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pp. 667–676, 2018. doi: 10.1145/3269206.3271684.

Jiani Huang, Xingchen Zou, Lianghao Xia, and Qing Li. MR.Rec: Synergizing memory and reasoning for personalized recommendation assistant with LLMs. *arXiv preprint arXiv:2510.14629*, 2025.

Dietmar Jannach and Himan Abdollahpouri. A survey on multi-objective recommender systems. *Frontiers in Big Data*, 6:1157899, 2023. doi: 10.3389/fdata.2023.1157899.

Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, and Weinan Zhang. How can recommender systems benefit from large language models: A survey. *ACM Transactions on Information Systems*, 43(2):1–47, 2025. doi: 10.1145/3678004.

Qidong Liu, Xiangyu Zhao, Yejing Wang, Zijian Zhang, Howard Zhong, Chong Chen, Xiang Li, Wei Huang, and Feng Tian. Bridge the domains: Large language models enhanced cross-domain sequential recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1582–1592, 2025a. doi: 10.1145/3726302.3729911.

Xinyi Liu, Ruijie Wang, Dachun Sun, Dilek Hakkani-Tür, and Tarek Abdelzaher. Uncovering cross-domain recommendation ability of large language models. In *Companion Proceedings of the ACM Web Conference*, pp. 2736–2743, 2025b. doi: 10.1145/3701716.3717850.

Tong Man, Huawei Shen, Xiaolong Jin, and Xueqi Cheng. Cross-domain recommendation: An embedding and mapping approach. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 2464–2470, 2017. doi: 10.24963/ijcai.2017/343.

Alessandro Petruzzelli, Cataldo Musto, Lucrezia Laraspata, Ivan Rinaldi, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. Instructing and prompting large language models for explainable cross-domain recommendations. In *Proceedings of the ACM Conference on Recommender Systems*, pp. 298–308, 2024. doi: 10.1145/3640457.3688137.

Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1):59, 2022. doi: 10.1186/s40537-022-00592-5.

Hang Su, Yun Yang, Tianyang Liu, Xin Liu, Peng Pu, and Xuesong Lu. Personalized question answering with user profile generation and compression. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 4744–4763, 2025. doi: 10.18653/v1/2025.findings-emnlp.255.

Lu Wang, Di Zhang, Fangkai Yang, Pu Zhao, Jianfeng Liu, Yuefeng Zhan, Hao Sun, Qingwei Lin, Weiwei Deng, Dongmei Zhang, Feng Sun, and Qi Zhang. LettinGo: Explore user profile generation for recommendation system. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2985–2995, 2025a. doi: 10.1145/3711896.3737024.

Peiyao Wang, Zhan Shi, Amina Shabbeer, and Ben London. Off-policy evaluation of candidate generators in two-stage recommender systems. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, RecSys '25, pp. 350–359, New York, NY, USA, 2025b. Association for Computing Machinery. doi: 10.1145/3705328.3748057.

Yujing Wang, Hainan Zhang, Liang Pang, Binghui Guo, Hongwei Zheng, and Zhiming Zheng. MaFeRw: Query rewriting with multi-aspect feedbacks for retrieval-augmented large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):25434–25442, 2025c. doi: 10.1609/aaai.v39i24.34732.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general Chinese embeddings. *arXiv preprint arXiv:2309.07597*, 2023.

Wujiang Xu, Yunxiao Shi, Zujie Liang, Xuying Ning, Kai Mei, Kun Wang, Xi Zhu, Min Xu, and Yongfeng Zhang. iAgent: LLM agent as a shield between user and recommender systems. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18056–18084, 2025. doi: 10.18653/v1/2025.findings-acl.928.

Fanghua Ye, Meng Fang, Shenghui Li, and Emine Yilmaz. Enhancing conversational search: Large language model-aided informative query rewriting. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5985–6006, 2023. doi: 10.18653/v1/2023.findings-emnlp.398.

Tianzi Zang, Yanmin Zhu, Haobing Liu, Ruohan Zhang, and Jiadi Yu. A survey on cross-domain recommendation: Taxonomies, methods, and future directions. *ACM Transactions on Information Systems*, 41(2):1–39, 2023. doi: 10.1145/3548455.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, 2025a.

Yipeng Zhang, Bowen Liu, Xiaoshuang Zhang, Aritra Mandal, Canran Xu, and Zhe Wu. LESER: Learning to expand via search engine-feedback reinforcement in e-commerce. *arXiv preprint arXiv:2509.05570*, 2025b.

Xiangyu Zhao, Changsheng Gu, Haoshenglun Zhang, Xiwang Yang, Xiaobing Liu, Jiliang Tang, and Hui Liu. DEAR: Deep reinforcement learning for online advertising impression in recommender systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1):750–758, 2021. doi: 10.1609/aaai.v35i1.16156.

Appendix Overview

The appendices provide the implementation and evaluation details omitted from the main text. Appendix A specifies the profile inputs, prompts, and outputs underlying Equation 3. Appendix B details the three LDCO layers and the reranking boundary. Appendix C gives the MCTS procedure, reward, stopping rule, and selected prompts. Appendix D documents fuzzy-query construction and the user-disjoint benchmark. Appendix E defines the retrieval-only Oracle protocol used in the query-quality analysis.

A Cross-Domain User Profile Management

A.1 Profile Inputs

Profile construction uses only authorized source-domain evidence. For each source domain S_i, the history $\mathcal{H}_u^{S_i}$ contains the product title, description, rating, and review text for each interaction. It excludes the constructed target-domain fuzzy query, target-domain history, held-out target item, and relevance label. This boundary ensures that the profiles available to LDCO exist before the user enters the target domain.

A.2 Domain-Level Profiles Generation

Domain-level profiles preserve the context in which each preference is observed. For every source domain, the system generates

$$\mathcal{P}_u^{S_i} = g_{\text{domain}}\big(\mathcal{H}_u^{S_i}; \phi_d\big).$$

Each profile separates positive and negative attribute-level evidence. Positive evidence describes attributes that the user praises or repeatedly selects; negative evidence describes criticism, avoidance, or low-rated attributes. Each field is limited to 50 words. If the history does not support a negative preference, the corresponding field remains empty rather than being inferred.

System Prompt for Domain-Level User Profile Generation

Role: Grounded domain profiling agent for cross-domain cold-start recommendation.
Task: Generate a structured domain-level profile by extracting transferable positive and negative attribute-level preferences from a user's interaction history.
Input Configuration:

- Product Information: title and description
- User Feedback: rating and review text
- Source Domain: the product domain of the interaction history

Profile Constraints:
- Ground every preference in the provided interaction history.
- Use concrete attribute-level terms rather than generic descriptions.
- Retain both positive and negative signals; return an empty negative summary when no clear negative signal exists.
- Limit the positive and negative summaries to 50 words each.

Output Format: {"positive_summary": "...", "negative_summary": "..."}.

A.3 Global Profile Generation

The global profile retains only preferences that can plausibly transfer across domains. The system aggregates the domain profiles as

$$\mathcal{P}_u = g_{\text{global}}\big(\mathcal{P}_u^{S_1}, \ldots, \mathcal{P}_u^{S_m}; \phi_g\big).$$

The global profile does not copy source-product names. It summarizes transferable choice dimensions, reusable query-to-product reasoning patterns, negative-transfer guards, and domain-conditioned reserve signals. A global preference normally requires support from at least two source domains; a strong single-domain signal remains domain-conditioned and can be used only when the current request makes that domain relevant.

System Prompt for Global User Profile Generation

Role: Cross-domain transferable reasoning agent for cold-start query optimization.
Task: Infer transferable user characteristics, reusable query-to-product reasoning patterns, and negative transfer guards from multiple domain-level profiles.
Input Configuration:
- Source Evidence: positive and negative attribute-level preferences retained by each domain profile

Reasoning Constraints:
- Use only the provided domain profiles; do not restate source-domain facts or preferred products.
- Treat a preference as globally transferable only when it is normally supported by at least two domains.
- Preserve strong single-domain signals as domain-conditioned reserve rather than global preference.
- Do not generate target-domain facts, brands, specifications, certifications, or full search queries.
- Express retained signals as attributes usable for retrieval, filtering, or ranking.

Output Format: Structured JSON containing transferable characteristics, query-to-product methods, negative transfer guards, and domain-conditioned reserve signals.

A.4 Outputs Used by LDCO

The two profile levels serve different downstream roles. Layer 2 selects at most two domain-level profiles as request-specific evidence, whereas the global profile supplies stable cross-domain patterns to dimension mapping, query generation, and reranking. Explicit requirements in the current query always override historical preferences. Consequently, a profile can support an attribute only when the later mapping step finds a target-relevant interpretation; profile text is never copied directly into the retrieval query.

B Layered Mapping and Reranking Details

This appendix expands the interfaces summarized in Sections 3.3 and 3.5. Layer 1 returns one target-domain label; Layer 2 returns no more than two source-domain profile identifiers; Layer 3 returns target-domain attributes paired with their supporting profile evidence; and query generation returns one catalog-facing query. Keeping these outputs distinct makes unsupported transfer detectable before retrieval.

B.1 Target Classification and Source Selection

The first two layers restrict which evidence can reach dimension mapping. The target-domain classifier consumes only the Fuzzy query q_u, preventing historical preferences from changing the requested product category. The source selector then considers the predicted target domain and the available domain-level profiles, returning at most two profiles with evidence relevant to the request. It can return fewer profiles when the remaining histories are unrelated.

For the wireless-earbuds example, Layer 1 predicts Electronics. Layer 2 can select Clothing, Shoes and Jewelry when that profile contains evidence about comfort or fit, and Home and Kitchen when it contains evidence about ease of use or durability. A Books profile containing only genre preferences is not selected. Selection identifies potentially useful evidence; it does not yet authorize any target-domain attribute, which remains the responsibility of Layer 3.

B.2 Dimension Context and Preference Transfer

As defined in Equation 4, LDCO first predicts the target domain $\widehat{T}_u$ from the original fuzzy query q_u. The predicted target domain determines the attribute space within which cross-domain preference transfer is considered. Conditioned on $\widehat{T}_u$, the source-domain selector then constructs $\mathcal{P}_u^{\mathrm{sel}}$ according to Equation 5, containing at most two domain-level profiles that provide the most relevant available source evidence for the target request.

The dimension context organizes the attributes considered during cross-domain transfer. It contains type-level dimensions, bridge dimensions shared across product types, and source–target pair mappings. Type-level dimensions describe attributes commonly relevant to a target product type. Bridge dimensions represent reusable choice criteria that may appear differently across product domains, such as usability, reliability, compatibility, maintenance effort, or value sensitivity. Source–target pair mappings provide more specific interpretation guidance for transferring a dimension from a particular source domain to the predicted target domain.

Applicable source–target mappings are ordered by their pair weights before being supplied as context for preference translation. These weights prioritize mappings that are more suitable for the current source–target domain pair. However, the dimension context specifies only which correspondences may be considered. It does not itself establish that user u has a preference for a particular attribute. A dimension is used only when the selected profiles $\mathcal{P}_u^{\mathrm{sel}}$ or global profile $\mathcal{P}_u$ contain relevant supporting evidence.

Following Equation 6, the dimension-mapping function produces the target-domain mapping result

$$\mathcal{M}_u^T = f_{\mathrm{map}}\left(\widehat{T}_u, \mathcal{P}_u^{\mathrm{sel}}, \mathcal{P}_u\right).$$

The function f_{map} performs preference translation in three steps. First, it identifies profile evidence associated with an applicable dimension. Second, it abstracts the evidence into a domain-neutral choice criterion. Third, it interprets this criterion as a concrete target-domain attribute that can be used for query generation. The mapping result $\mathcal{M}_u^T$ therefore contains only target-domain attributes that are supported by the selected profile evidence and are relevant to the predicted target domain.

For example, a preference for straightforward setup in a source-domain profile can be interpreted as a usability-related choice criterion. When $\widehat{T}_u$ is Electronics and the request concerns wireless earbuds, the dimension context may support translating this criterion into an attribute such as "easy pairing." In contrast, a preference for mystery-book genres has no relevant target-domain interpretation for earbud attributes and is excluded from $\mathcal{M}_u^T$. Thus, LDCO transfers underlying choice logic rather than directly copying source-domain product words into the target query.

A candidate mapping is discarded when profile evidence is absent, when the abstracted preference is irrelevant to the requested product type, or when the dimension context does not provide a valid target-domain interpretation. Negative preferences are handled conservatively: they affect query generation only when they can be expressed as a concrete and relevant target-domain constraint. Otherwise, they remain profile information and are not converted into query attributes.

Finally, the attributes in $\mathcal{M}_u^T$ are passed to the query generator together with the original fuzzy query q_u, selected domain profiles $\mathcal{P}_u^{\mathrm{sel}}$, and global profile $\mathcal{P}_u$, as specified in Equation 7. Explicit query

requirements, including product type, model, compatibility, size, audience, and usage scenario, always take precedence over inferred preferences. If no evidence-grounded mapping is available, $\mathcal{M}_u^T$ remains empty and query generation preserves the original request rather than introducing unsupported attributes.

B.3 End-to-End Query Transformation Examples

The following illustrative examples make the LDCO interface concrete. Each example separates the original fuzzy query, the authorized profile evidence, the structured demand produced by dimension mapping, and the final query generated with the MCTS-selected prompt. The attributes are illustrative rather than outputs from the held-out benchmark, and each may enter a deployed query only when supported by the stated profile evidence.

Illustrative LDCO Query Transformations

Example 1: Wireless earbuds (Electronics)
Fuzzy query: I want to buy wireless earbuds.
Profile evidence: Clothing reviews repeatedly favor comfortable fit; Home and Kitchen reviews favor straightforward setup and durable construction.
Structured demand: {product: wireless earbuds; fit: comfortable; setup: easy pairing; build: durable}.
Optimized query: Comfortable wireless earbuds with easy pairing and durable build.

Example 2: USB switch (Electronics)
Fuzzy query: Can you help me find a USB switch for two computers?
Profile evidence: The global profile favors reliable products that are simple to configure and perform as described.
Structured demand: {product: USB switch; compatibility: two computers; reliability: high; setup: simple}.
Optimized query: Reliable USB switch for sharing devices between two computers with simple setup.

Example 3: Travel bottles (Beauty and Personal Care)
Fuzzy query: Can you help me find travel bottles for camping?
Profile evidence: Home and Kitchen reviews favor leak-resistant containers and durable reusable materials.
Structured demand: {product: travel bottles; scenario: camping; closure: leak-resistant; build: durable and reusable}.
Optimized query: Leak-resistant durable reusable travel bottles for camping.

These examples also expose the grounding constraint: a preference such as a favored book genre has no valid mapping to earbud fit, USB-switch setup, or bottle closure and is therefore omitted. If no relevant evidence remains, the structured demand contains only the explicit request and the generated query stays close to the fuzzy query.

B.4 Candidate Representation and Reranking Scope

Following Equation 9, the user-side reranker takes the personalized query $q'_u(p^*)$, the global user profile $\mathcal{P}_u$, and the retrieved candidate list $\mathcal{C}_u$ as input, and produces the final ranked list π_u. The reranker operates after query generation and platform retrieval; therefore, it evaluates the relevance of already retrieved products rather than searching the catalog or generating new candidates. Let

$$\mathcal{C}_u^{(100)} = (c_{u,1}, c_{u,2}, \ldots, c_{u,100})$$

denote the first 100 candidates returned by Hybrid retrieval. Each candidate $c_{u,j}$ is represented by its ASIN, title, and description. The ASIN serves only as a stable identifier for matching the LLM output with the corresponding retrieved product. For each candidate in this prefix, the reranker produces a query–profile relevance assessment:

$$r_{u,j} = g_{\text{rank}}\left(q'_u(p^*), \mathcal{P}_u, c_{u,j}\right), \qquad c_{u,j} \in \mathcal{C}_u^{(100)}.$$

Here, g_{rank} denotes the LLM-based relevance assessment used by the reranking function in Equation 9. The personalized query expresses the current request and supported transferred preferences, whereas the global profile provides stable cross-domain user context.

The reranked prefix is obtained by ordering the first 100 candidates according to their relevance assessments:

$$\pi_{u,1:100} = \mathrm{Rank}\left(\mathcal{C}_u^{(100)}; r_{u,1}, r_{u,2}, \ldots, r_{u,100}\right).$$

Reranking changes order but not candidate membership. Candidates below the reranked prefix retain the platform order, giving

$$\pi_u = \left(\pi_{u,1:100}, c_{u,101}, \ldots, c_{u,N}\right).$$

The agent therefore cannot recover an item absent from $\mathcal{C}_u$, and reranking cannot change Hit@100 because the first-100 candidate set remains fixed. This boundary explains why the ablation in Table 2 attributes Hit gains to query generation and ranking gains to reranking.

C MCTS Prompt Optimization

C.1 Search Objective

MCTS optimizes the query-generation instruction against retrieval outcomes rather than linguistic quality. AFlow supplies candidate selection, LLM-based expansion, execution evaluation, and experience backpropagation; we adapt its workflow search so that each node represents an LDCO query-generation prompt. GPT-5.5 serves as the optimizer model and proposes prompt revisions during MCTS expansion. GPT-4o-mini serves as the execution model and applies each candidate prompt to generate rewritten queries for Hybrid retrieval evaluation.

Search starts from a manually designed seed prompt p_0. Profile construction, target-domain classification, source-domain selection, dimension mapping, the Hybrid retriever, and all model settings remain fixed. Reranking is not part of prompt evaluation. Performance differences between nodes therefore reflect only the generated queries.

C.2 MCTS Search Cycle

Each MCTS round follows four stages.

- **Initialization.** The manually designed seed prompt p_0 is used as the root node and evaluated on the optimization set to obtain its initial Hybrid Hit@100 reward.
- **Selection.** A parent prompt is selected according to its historical reward. Following AFlow's soft mixed-probability selection strategy, the search balances exploitation of high-reward prompts with exploration of alternative candidates, reducing the risk of local optima.
- **Expansion.** GPT-5.5, acting as the optimizer model, modifies the selected parent prompt to create a new candidate prompt. Modifications may refine cross-domain preference use, attribute-mapping constraints, query phrasing, or restrictions against unsupported attribute expansion. Each child prompt records its parent and the corresponding modification.
- **Execution Evaluation and Backpropagation.** GPT-4o-mini, acting as the execution model, applies the candidate prompt to the optimization samples and generates rewritten queries. Hybrid retrieval evaluates these queries, and the held-out target identifier is used only after retrieval to compute Hybrid Hit@100. The resulting reward, prompt modification, and improvement or failure relative to the parent are recorded and backpropagated through the search tree to guide subsequent selection and expansion.

C.2.1 Algorithmic Procedure

Algorithm B1: Refined MCTS for LDCO Prompt Optimization

Model assignment: `LLMMutate` uses GPT-5.5 to propose prompt revisions, whereas `LLM` uses GPT-4o-mini to execute each candidate prompt and generate rewritten queries.

Require: Seed prompt p_0, optimization set $\mathcal{V}$, search budget $T = 15$, candidate-prompt shortlist size B, fixed context builder C, and Hybrid retriever R.
Ensure: Selected prompt p^*.

1: Evaluate p_0 on $\mathcal{V}$; initialize candidate set $\mathcal{P} = \{p_0\}$ and experience memory $\mathcal{E}$.
2: for $t = 1, \ldots, T$ **do**
3: $\mathcal{P}_B \leftarrow \mathrm{GetTopRounds}(\mathcal{P}, B)$.
4: $p_{\mathrm{parent}} \leftarrow \mathrm{PenalizedBoltzmannSelect}(\mathcal{P}_B, \mathcal{E})$.
5: $r \leftarrow \mathrm{HeuristicCritic}(p_{\mathrm{parent}}, \mathcal{E})$.
6: $\hat{p} \leftarrow \mathrm{LLMMutate}(p_{\mathrm{parent}}, r, \mathcal{E})$.
7: **if** $\hat{p}$ cannot produce a valid one-line retrieval query **then**
8: Record the failed mutation in $\mathcal{E}$ and continue.
9: **end if**
10: scores $\leftarrow []$.
11: **for each** $(x_u, q_u, a_u) \in \mathcal{V}$ **do**
12: $q'_u \leftarrow \mathrm{LLM}(\hat{p}, C(x_u, q_u))$.
13: scores.append$(\mathbb{I}[a_u \in R_{\mathrm{Hybrid}}(q'_u, 100)])$.
14: **end for**
15: $J(\hat{p}) \leftarrow \mathrm{mean}(\mathrm{scores})$.
16: $\mathrm{UpdateExperience}(p_{\mathrm{parent}}, \hat{p}, J(\hat{p}), \mathcal{E})$.
17: $\mathcal{P} \leftarrow \mathcal{P} \cup \{\hat{p}\}$.
18: end for
19: return $p^* = \arg\max_{p \in \mathcal{P}} J(p)$.

Here, x_u denotes the execution-model-visible profile and dimension context, while a_u is the reward-only target identifier. The execution loop produces one binary reward per optimization user: whether a_u appears in the Hybrid Top-100 results. A valid query is a nonempty, single-line catalog query without JSON, labels, explanations, or Markdown; query validation does not inspect a_u. Search stops after the predefined budget of 15 rounds, and an invalid mutation is recorded as failed experience before the next round.

C.3 Retrieval-Aware Reward

Let $\mathcal{V}$ denote the prompt-optimization set, q_u the original fuzzy query, and a_u the target item associated with that query. The reward for candidate prompt p is

$$J(p) = \frac{1}{|\mathcal{V}|} \sum_{(q_u, a_u) \in \mathcal{V}} \mathbb{I}\left[a_u \in \mathcal{R}_{\mathrm{Hybrid}}(q'_u(p), 100)\right].$$

Here, $\mathcal{R}_{\mathrm{Hybrid}}(q'_u(p), 100)$ denotes the Top-100 results returned by Hybrid retrieval with optimized query $q'_u(p)$, and $\mathbb{I}[\cdot]$ is the indicator function. This reward is equivalent to Hybrid Hit@100 on the optimization set. MCTS therefore optimizes prompts for target-item recovery rather than query fluency alone.

C.4 Data Isolation and Prompt Selection

MCTS search uses only the 568-user optimization set. For each sample, the execution model receives the user context and original fuzzy query. The target-item identifier is used only after retrieval to calculate the reward; it is not supplied to the query-generation model, prompt optimizer, or critic. The 142 test users remain isolated throughout the search.

Starting from p_0, we run 15 MCTS rounds and select the candidate prompt with the highest reward:

$$p^* = \arg \max_{p \in \mathcal{P}_{\mathrm{MCTS}}} J(p),$$

where $\mathcal{P}_{\mathrm{MCTS}}$ is the set of candidate prompts visited during MCTS search. The selected prompt p^* is frozen after search and used unchanged for final evaluation on the held-out test users.

C.5 Seed and Optimized Prompt Comparison

To make the prompt-search outcome transparent and reproducible, we present below the manually designed seed prompt p_0 and the final prompt p^* selected by MCTS. Both prompts are reproduced

without modification from their corresponding search states. Neither prompt contains the identity of the target item or any information from the held-out test set.

Seed Prompt p_0 for Query Optimization

Role: Product-search catalog title normalizer.
Task: Convert the user's shopping request into one concise Amazon-style retrieval query.
Input Configuration:
- Original fuzzy shopping request and target domain.
- Global user profile and source-domain profiles.
- Cross-domain transferable dimensions and domain-pair transfer guidance.

Reasoning Constraints:
- Remove conversational wording and non-product purchase intent, while retaining terms that define the product or appear on it.
- Preserve explicit functions, audiences, sizes, compatibility requirements, materials, models, term lengths, and use cases.
- Use profile evidence only when it provides concrete and relevant attributes for the requested product type; ignore vague or unrelated signals.
- Do not transfer source-domain product names unless they are valid searchable attributes, such as a brand, model, material, size, or compatibility term.
- Avoid unsupported subjective descriptors, beauty claims, ingredients, and certifications.
- Prefer a query of 4–12 words and apply catalog defaults only to recognized sparse requests.

Output Format: Return only one nonempty, single-line product-search phrase, without JSON, metadata keys, labels, Markdown, quotes, bullets, or explanations.

MCTS-Optimized Prompt p^*

Role: Product-search catalog title normalizer.
Task: Convert the user's shopping request into one concise Amazon-style retrieval query.
Input Configuration:
- Original fuzzy shopping request and target domain.
- Global user profile and source-domain profiles.
- Cross-domain transferable dimensions and domain-pair transfer guidance.

Reasoning Constraints:
- Preserve all explicit product functions, audiences, dimensions, compatibility details, materials, models, term lengths, and use cases.
- Transfer profile evidence only when it yields concrete, searchable attributes for the same product type; reject vague or unrelated signals.
- Suppress unsupported subjective qualities, beauty claims, ingredients, certifications, and features such as keyboards, USB ports, adjustable temperature, subwoofers, Bluetooth, 4K, rechargeable batteries, or waterproofing.
- Preserve exact device models, generations, and sizes for cases, covers, chargers, cables, tablets, phones, laptops, and Kindle products.
- Apply expanded sparse-query catalog rules, including targeted normalizations for an iPad Air 2 case, a 1-inch curling iron, and a surge suppressor, together with other product-specific defaults learned in earlier rounds.
- Prefer a query of 4–12 words, but retain essential specifications when a longer query is necessary.
- Before answering, verify that the query contains no labels or structured output, excludes unsupported gift intent, and preserves compatibility and sizing.

Output Format: Return only one nonempty, single-line product-search phrase, without JSON, metadata keys, labels, Markdown, quotes, bullets, or explanations.

The comparison indicates that MCTS preserves the original objective of producing a concise retrieval query while refining the instructions governing cross-domain evidence use, target-domain attribute mapping, and unsupported query expansion. In particular, p^* places greater emphasis on retaining only user-supported preference signals and preventing the generation of speculative brands, categories, specifications, or product attributes. It also strengthens the output-format constraint so that the execution model returns only a valid one-line retrieval query. These refinements are selected exclusively according to Hybrid Hit@100 on the optimization set.

C.6 Search Trace and Reproducibility Contract

The complete contents of the initial and selected prompts are reported in Appendix C.5. In addition, for every attempted mutation, the search trace stores the round number, parent prompt identifier, critic or mutation message, execution status, and Hybrid Hit@100 reward. Failed mutations remain in the trace as negative experience but are excluded from candidate ranking.

The mutation model can access only the seed prompt, selected parent prompt, and processed experience from the optimization split. It cannot access held-out target ASINs, test-user records, test rewards, or final test metrics. After round 15, the prompt with the highest optimization reward is selected once and frozen. No test result is fed back into MCTS.

D Fuzzy-Query Benchmark Construction

D.1 Fuzzy Query Construction

Fuzzy-query construction creates incomplete target-domain requests without exposing the held-out product to LDCO. For each user, the pipeline takes the title, description, and review of one five-star target-domain interaction and applies the two stages in Figure 4. The resulting query retains the product need and necessary context but removes the title, ASIN, soft preferences, and uniquely identifying attribute combinations. The held-out item subsequently serves only as the retrieval label.

Demand-Extraction Agent. The first agent converts the interaction into three fields: `hard_constraint_phrase` for the product type and indispensable compatibility conditions, `soft_preferences` for nonessential attributes, and `optional_scene` for a possible use context. This intermediate representation separates what the fuzzy query can retain from the details reserved for evaluation.

Fuzzy-Query Agent. The second agent expresses the hard constraint and, when necessary, the usage context as a short natural-language request. It omits `soft_preferences`, target identifiers, and uniquely identifying details. LDCO therefore has to recover useful specificity from source-domain profiles rather than from leaked target attributes.

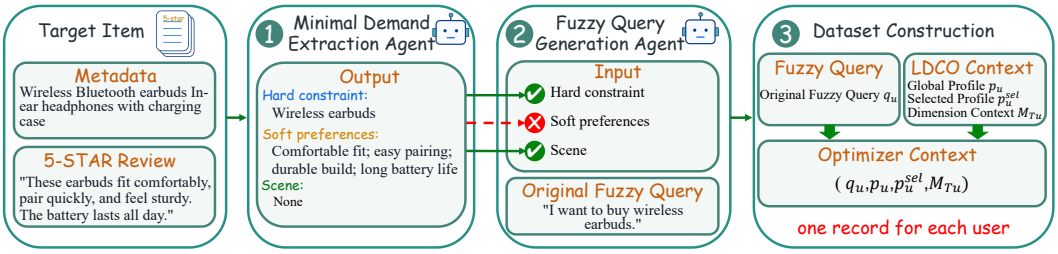


Figure 4: Fuzzy Query Construction.

Fuzzy Query Construction Example. The following example illustrates the construction of one sample from the Beauty and Personal Care target domain. Target-item information and the review are shown solely to explain the offline construction procedure.

Example of Fuzzy Query Construction

Target Domain: Beauty and Personal Care.
Held-out Target Item: Travel bottles with leak-proof silicone containers and a lanyard.
User Review: The set is convenient for camping and shower use.
Demand Extraction Output:

- `hard_constraint_phrase`: Travel bottles
- `soft_preferences`: Leak-proof, silicone, with a lanyard

- `optional_scene`: Camping

Generated Original Fuzzy Query: Can you help me find travel bottles for camping?

In this example, the final query retains only the product type and use context. Fine-grained preferences, including leak-proof construction, silicone material, and the lanyard, are excluded from the query to avoid revealing target-item details and to leave room for later personalization through LDCO.

D.2 User-Level Split and Leakage Control

The benchmark separates prompt selection from final evaluation at the user level. Of the 710 users, 568 form the optimization split used to evaluate MCTS candidates, and 142 form the test split used only after p^* is frozen. The test split contains 69 Beauty and Personal Care users and 73 Electronics users. No interaction, profile, query, target identifier, reward, or metric from a test user is available during prompt search.

Each sample maintains two information channels. The execution-model-visible channel contains the fuzzy query, predicted target domain, global profile, selected source profiles, and dimension context. The reward-only channel contains the held-out target identifier and is accessed only after retrieval to compute Hit@100. This separation implements the information boundary assumed by Equation 8.

D.3 Prompt-Optimization Dataset

Each optimization instance connects an incomplete request to an executable retrieval reward. Its visible input combines the original fuzzy query with the predicted target domain, global profile, selected source-domain profiles, and cross-domain dimension context.

The instance stores the held-out target identifier separately as reward metadata. This identifier is not included in the query-generation input, optimizer prompt, or critic experience; it is used only to test membership in the retrieved Top-100 list. Thus, the target item remains an evaluation label rather than an input feature.

Prompt-Optimization Dataset Example. The following example shows one retrieval-aware instance used for prompt optimization. The user context is abbreviated for presentation; in implementation, it contains the complete global and source-domain profiles together with the relevant cross-domain dimension mapping context.

Example of a Prompt-Optimization Instance

Execution-Model-Visible Context:
- Target Domain: Electronics
- Global Profile: The user values practical, functional, and durable products.
- Relevant Source-Domain Evidence: The user prefers items that are reliable, easy to use, and perform as described.
- Target-Domain Dimension Context: Compatibility, reliability, and setup convenience.

Original Fuzzy Query: Can you help me find a USB switch for two computers?
Illustrative Rewritten Query: USB switch for sharing devices between two computers.
Reward-Only Metadata: Held-out target identifier `B006Z0Q2SI`; this identifier is withheld from the prompt optimizer and used only after Hybrid retrieval to compute Hybrid Hit@100.

For this instance, the optimizer receives only the user context and Fuzzy query and produces a rewritten query. Hybrid retrieval is then executed with that query, after which the held-out target identifier determines the binary Top-100 reward. This separation prevents target leakage while retaining retrieval-based feedback for MCTS.

E Oracle Query Protocol

We define an *Oracle query* as a diagnostic query generated with the same user-side context as LDCO plus the held-out target product metadata. The Oracle condition measures the value of unavailable

target-specific information rather than a deployable method. The Fuzzy query exposes only the incomplete request; the LDCO query additionally uses authorized source-domain profiles and mapped attributes; and the Oracle query additionally receives the held-out target product metadata. Target metadata are available only to this diagnostic condition and are never exposed to profile construction, LDCO query generation, MCTS prompt search, or reranking.

All three conditions follow the same retrieval-only protocol. Hybrid retrieval returns candidates in its native order, and Hit@100, MRR@100, and NDCG@100 are computed directly from that order. The reranker f_{rank} is disabled, so the comparison isolates query information from candidate reordering.

Table 7: Query-quality comparison with Hybrid retrieval at $K = 100$, without LLM reranking.

Query type	**Hit@100**	**MRR@100**	**NDCG@100**
Fuzzy query	0.3169	0.0365	0.0872
LDCO query	0.4155	0.0577	0.1236
Oracle query	0.7042	0.2312	0.3231

The LDCO–Fuzzy gap in Table 7 measures information recovered from cross-domain evidence, whereas the Oracle–LDCO gap measures target-specific information unavailable under cold start. LDCO improves Hit@100 from 0.3169 to 0.4155 without accessing the target item but remains below the Oracle value of 0.7042. This gap bounds what profile-grounded query generation can recover from the available evidence.