COUNTERFACTUAL EDIT CREDIT WITH SEMANTIC RETRIEVAL FEEDBACK FOR PRODUCT QUERY REWRITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Product queries are often short and underspecified, leaving much of the user's preference implicit. Recent work trains large language models (LLMs) as query rewriters with reinforcement learning (RL) using ranking rewards based on the ranks of labeled relevant products. However, these rewards are sparse when relevant products are not retrieved and coarse when different retrieval outcomes leave their ranks unchanged. We propose Counterfactual Generative Search (CGS), an RL framework that uses dense semantic retrieval feedback to assign counterfactual credit to individual query edits, while retaining ranking reward as the objective for the complete rewrite. CGS uses a relevance teacher to score retrieved products against available evidence about user intent and a semantic list utility to distinguish retrieval outcomes beyond labeled-product ranks. It assigns credit to each edit by comparing retrieval quality when the edit is removed from the complete rewrite and when it is applied to the source query on its own. With extensive experiments on H&M and Amazon ESCI, we show that CGS consistently outperforms baselines, improving NDCG@100 by 10.3% and 4.9%, respectively. We share code at `https://anonymous.4open.science/r/RL_query_rewriter-D8AE`

1 INTRODUCTION

Product search often starts from short and underspecified queries that leave much of the user's actual preference implicit. For instance, a query such as "running shoes" may not reveal what properties of the product the user cares about, making it difficult for the search engine to retrieve the most relevant items. Query rewriting addresses this problem by reformulating or expanding the original query to better express the user's underlying intent. The key challenge is therefore deciding what information should be added to the query to improve retrieval.

Recent work tackles this challenge by learning query rewriting from downstream retrieval feedback (Lin et al., 2025; Nath et al., 2026; Agrawal et al., 2023; Dai et al., 2024). In particular, large language models (LLMs) provide a flexible query rewriter that can introduce rich and specific descriptions. A recent line of work therefore trains LLM-based query generators using reinforcement learning (RL) with a verifiable sequence-level *ranking reward*, such as NDCG, that scores a rewritten query according to the ranks of labeled relevant products (Lin et al., 2025; Nath et al., 2026). Rec-R1 (Lin et al., 2025) uses GRPO (Shao et al., 2024) and assigns the sequence reward uniformly to all generated tokens. OSPO (Nath et al., 2026) estimates generated segments' marginal contributions and redistributes the sequence reward.

These approaches improve retrieval performance, but sequence-level ranking reward creates a credit assignment problem. A rewritten query may contain several changes with different effects on retrieval, yet a single reward does not reveal which changes are responsible. Methods such as OSPO (Nath et al., 2026) address this by estimating segment-level contributions, but these contributions are still derived from ranking reward and inherit its limitations. **First, ranking reward is sparse.** When no target product is retrieved, different rewrites receive the same zero reward even if some retrieve products that better match the user's intent. **Second, ranking reward is coarse.** It only observes ranks of the target-product, so changes to the retrieved list are ignored when those

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.

ranks remain unchanged. Therefore, finer-grained credit alone cannot distinguish query changes whose retrieval effects are invisible to ranking reward.

Our insight. Fine-grained credit is only useful if the feedback can tell whether a particular query change improved retrieval. Re-attributing a sequence-level ranking reward is therefore insufficient when different retrieval outcomes may receive zero or the same reward. In query rewriting, this occurs when target products are missed or their ranks remain unchanged. Effective fine-grained credit assignment therefore requires both feedback that is sensitive to meaningful retrieval changes and a mechanism that attributes those changes to individual edits in the rewritten query.

Present work. Guided by this insight, we propose **C**ounterfactual **G**enerative **S**earch (CGS), an RL framework that combines dense semantic retrieval feedback with counterfactual credit assignment over individual edits in the rewritten query. To distinguish retrieval changes that are invisible to ranking reward, CGS constructs a relevance teacher from the source query, target-product metadata, and user history when available. The teacher scores retrieved products against this intent evidence, and a semantic list utility summarizes the quality of each retrieved list. CGS then decomposes a rewrite into textual edits and evaluates two counterfactual interventions for each edit. Removing an edit from the complete rewrite measures whether it is necessary to maintain the achieved retrieval quality, while applying it to the source query measures whether it is sufficient to improve retrieval on its own. We combine these effects using a minimum inspired by the probability of necessity and sufficiency (PNS) (Pearl, 1999). Our key contributions are:

- **Dense semantic retrieval feedback.** We introduce a semantic retrieval utility that evaluates retrieved products using available evidence about the user intent. Unlike ranking reward, this feedback can distinguish retrieval outcomes when target products are missed or their ranks remain unchanged, providing an informative signal for fine-grained credit assignment. We show that even when ranking reward is zero, the resulting semantic feedback supports positive edit credit for 40.5% of H&M sequences and 55.7% of ESCI sequences.
- **Counterfactual edit credit.** We propose a PNS-inspired credit estimator that attributes changes in semantic retrieval quality to individual query edits. Unlike Rec-R1 (Lin et al., 2025), which assigns sequence reward uniformly, and OSPO (Nath et al., 2026), which redistributes ranking reward across generated segments, our estimator derives edit credit from counterfactual changes in semantic retrieval utility. Extensive experiments show that CGS improves NDCG@100 over the strongest baseline by 10.3% and 4.9% on H&M and ESCI, respectively.

2 Related Work

Query rewriting and retrieval-driven optimization. Query reformulation has long been used to reduce the mismatch between a user's query and relevant documents. Classical approaches modify query terms using relevance feedback (Rocchio, 1971) or estimate useful expansion terms from retrieved documents (Lavrenko & Croft, 2001). Later work learns the reformulation process directly. Nogueira & Cho (2017) train a neural query reformulator with reinforcement learning (RL), using downstream document recall as the reward. In product search, recent methods incorporate product metadata, behavioral signals, or retrieval results into neural and LLM-based query rewriting (Agrawal et al., 2023; Dai et al., 2024; Nguyen et al., 2025; Zhang et al., 2025; Dai et al., 2026). For retrieval-augmented question answering, Ma et al. (2023) train a query rewriter with RL using feedback from a frozen LLM reader. A more recent line of work directly optimizes LLM query rewriters using feedback from a fixed retrieval system. Rec-R1 (Lin et al., 2025) uses ranking reward to train the complete generated rewrite, while OSPO (Nath et al., 2026) decomposes the rewrite into segments and redistributes the ranking reward according to their contributions. Our work focuses on this retrieval-driven RL setting. Rather than changing the rewriting architecture or the context given to the generator, we study the feedback used to train individual query changes when the ranking reward is sparse or cannot distinguish their retrieval effects.

Fine-grained credit assignment. Credit assignment studies how an observed outcome should be attributed to the decisions that produced it. GAE estimates advantages for individual actions from future returns (Schulman et al., 2017), while RUDDER redistributes delayed rewards to earlier decisions through return decomposition (Arjona-Medina et al., 2019). Hindsight Credit Assignment

instead attributes outcomes to past actions based on how likely they were to have produced the observed result (Harutyunyan et al., 2019). Recent LLM post-training methods adapt this idea to generated sequences. VinePPO estimates values for intermediate prefixes using sampled continuations (Kazemnejad et al., 2025), and TEMPO (Tran et al., 2025) propagates credit through sampled response trees. Rewarding Progress measures how each reasoning step changes the likelihood of eventual success under a separate prover policy (Setlur et al., 2025). Other methods exploit task-specific structure. SSVPO uses sequential Shapley values to assign credit to reasoning steps across alternative insertion orders (Li et al., 2026). OSPO assigns credit to generated segments using Owen-Shapley contributions (Nath et al., 2026), while Rank-GRPO, FlexRec, and SAPO define finer credit over recommendation positions, item interventions, or reasoning steps (Zhu et al., 2025; Pan et al., 2026; Zheng et al., 2026). In contrast, we study credit assignment at the level of textual query edits while retaining ranking reward for the complete rewrite.

3 PRELIMINARIES

We formally define generative product search and establish the notation used throughout the paper.

Generative product search. Let $\mathcal{I}$ denote a product catalog, where each product $j \in \mathcal{I}$ has metadata $m(j)$. Each training example contains a source query x, a set of target products $y \subseteq \mathcal{I}$, and optional user context h, which may contain users' history and preferences. A policy π_θ generates a rewritten query $q = (q_1, \ldots, q_T)$ of T tokens, which is passed to a fixed retriever. We denote its top K results by

$$L_K(q) = (j_1^q, \ldots, j_K^q). \tag{1}$$

The ranking reward and semantic list utility can use different retrieval depths, specified in Table 8. We omit the depth from $L(q)$ when clear from context. The goal is to learn a query-generating policy for a fixed retriever and catalog that ranks the labeled relevant products as highly as possible.

Sequence-level ranking reward. We denote the sequence-level ranking reward by $R(q, y)$, which evaluates the complete rewrite q through the ranks of the target products y. Following prior work (Lin et al., 2025; Nath et al., 2026), we use normalized discounted cumulative gain (NDCG) (Järvelin & Kekäläinen, 2002). Specifically, $R(q, y) = \mathrm{DCG@}K(q, y)/\mathrm{IDCG@}K(y)$, where $\mathrm{DCG@}K(q, y) = \sum_{r=1}^{K} \mathbb{I}[j_r^q \in y]/\log_2(r+1)$ and $\mathrm{IDCG@}K(y) = \sum_{r=1}^{\min(K,|y|)} 1/\log_2(r+1)$.

4 METHODOLOGY

We present CGS in four stages. Section 4.1 constructs dense semantic retrieval feedback from a relevance teacher and a semantic list utility. Section 4.2 uses this feedback to assign counterfactual credit to individual edits. Section 4.3 integrates sequence reward and edit credit with GRPO. Section 4.4 then gives a theoretical interpretation of the resulting credit estimator.

4.1 DENSE SEMANTIC RETRIEVAL FEEDBACK

Fine-grained credit assignment requires comparing the retrieval outcomes induced by individual query changes. These comparisons are only informative when the retrieval feedback can distinguish their effects. Ranking reward $R(q, y)$ often remains unchanged when no target product is retrieved or when target-product ranks stay fixed, even if the retrieved products change substantially. We therefore construct a dense semantic retrieval utility that evaluates retrieved products against available evidence about user intent. A relevance teacher scores individual products (Section 4.1.1), and a rank-discounted list utility (Section 4.1.2) aggregates these scores. The resulting utility provides informative feedback for assigning credit to individual query changes.

4.1.1 SEMANTIC RELEVANCE TEACHER

For each training example, we form intent evidence $z = (x, h, m(y))$ from the source query x, optional user history h, and target product metadata $m(y)$. A relevance teacher maps the intent evidence z and a retrieved product j to a score $Y_z(j) \in [0, 1]$, measuring how well the product matches the available evidence about user intent. Our framework is agnostic to the choice of relevance model.

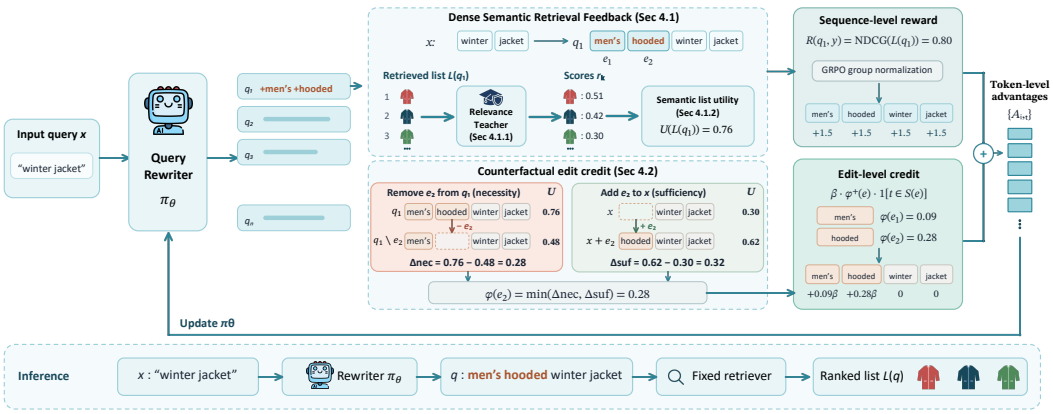


Figure 1: Overview of CGS. During inference, we fix the query rewriter and generate a rewrite q from the source query x and pass it to a fixed retriever.

In our experiments, we use a lightweight metadata matching scorer since credit computation requires scoring several retrieved lists within each RL update. We show in Appendix B.4 the experimental results for different teacher implementations.

We normalize text and canonicalize structured values for each evidence in the intent evidence z. Let $T_d(z)$ denote the terms from field d and $T(j)$ the terms from retrieved product j. We define the compatibility between a product and an evidence field as:

$$c_d(z, j) = \frac{\sum_{t \in T_d(z)} \mathbb{I}[t \in T(j)]}{|T_d(z)|} \in [0, 1]. \tag{2}$$

Let $\mathcal{D}(z)$ denote the evidence fields available for example z. We compute the relevance score of product j as a weighted average of its compatibility with these fields:

$$Y_z(j) = \sum_{d \in \mathcal{D}(z)} w_d(z) c_d(z, j), \qquad w_d(z) \geq 0, \qquad \sum_{d \in \mathcal{D}(z)} w_d(z) = 1. \tag{3}$$

4.1.2 Semantic List Utility

For a retrieved list $L = (j_1, \ldots, j_K)$, we define the semantic list utility as

$$U_z(L) = 1 - \prod_{r=1}^{K} \left(1 - \gamma^{r-1} Y_z(j_r)\right). \tag{4}$$

where $\gamma \in (0, 1]$ controls rank decay. The utility gives more weight to relevant products at higher ranks and diminishing gain as additional relevant products are retrieved. Appendix A provides its probabilistic interpretation and theoretical properties.

4.2 Counterfactual Edit Credit

The semantic utility measures the quality of a retrieved list, but does not identify which changes in a rewrite are responsible for that quality. We therefore assign credit by measuring how the semantic list utility changes when individual query edits are intervened on. A useful edit should contribute to the completed rewrite while also providing evidence that it can improve retrieval on its own. We capture these two views through complementary interventions.

To construct these interventions, we first align the source query x and rewrite q to obtain the set of edits $\mathcal{E}(x, q)$, consisting of inserted and deleted spans. For an edit $e \in \mathcal{E}(x, q)$, let q^{-e} denote the complete rewrite with e removed while all other edits remain fixed, and let x^{+e} denote the source query with only e applied. Together with the original queries q and x, these interventions produce

four retrieval lists, $L(q)$, $L(q^{-e})$, $L(x^{+e})$, and $L(x)$. The teacher evidence z and the retriever remain fixed across all four queries.

The two interventions test the same edit in complementary contexts. Removing e from the complete rewrite asks whether the achieved utility depends on that edit, while applying e alone to the source query asks whether the edit can improve retrieval by itself. We interpret these as necessity and sufficiency effects. We formally define the corresponding necessity and sufficiency effects as:

$$\Delta_{\mathrm{nec}}(e) = U_z(L(q)) - U_z(L(q^{-e})), \tag{5}$$

$$\Delta_{\mathrm{suf}}(e) = U_z(L(x^{+e})) - U_z(L(x)). \tag{6}$$

A positive necessity effect means that removing e reduces the utility of the complete rewrite, while a positive sufficiency effect means that applying e alone improves utility over the source query. Inspired by the probability of necessity and sufficiency (PNS) (Pearl, 1999), we define the edit-credit score as:

$$\phi_z(e) = \min\left\{\Delta_{\mathrm{nec}}(e), \Delta_{\mathrm{suf}}(e)\right\}. \tag{7}$$

Taking the minimum limits the credit assigned to an edit by the weaker of the two effects. Thus, an edit receives positive credit only when it improves retrieval both in the complete rewrite and on its own. Section 4.4 provides a formal interpretation of this minimum as a worst-context contribution.

Let $\Phi_{\mathcal{B}}$ denote the raw edit-credit scores in the current reward batch. We normalize the scores using the batch median absolute deviation (MAD), retain only positive credit, and clip extreme values:

$$\phi_z^+(e) = \mathrm{clip}\left(\frac{\phi_z(e)}{\max\{\mathrm{MAD}(\Phi_{\mathcal{B}}), \epsilon_\phi\}}, 0, c\right). \tag{8}$$

MAD provides a robust estimate of the credit scale within the batch. The floor $\epsilon_\phi > 0$ prevents unstable normalization when the batch spread is small, while clipping to $[0, c]$ removes nonpositive effects and limits unusually large credit values.

4.3 Integration with GRPO

Following Rec-R1 (Lin et al., 2025) and OSPO (Nath et al., 2026), we optimize the query rewriter using GRPO. We use counterfactual edit credit to provide fine-grained feedback, while retaining the ranking reward $R(q, y)$ as the sequence-level objective for the rewritten query. Per source query, GRPO samples a group $\mathcal{G}$ of candidate rewrites and evaluates their ranking rewards. The sequence advantage measures how well each rewrite performs relative to the other rewrites sampled for the same query.

Let $\mu_{\mathcal{G}}$ and $\sigma_{\mathcal{G}}$ denote the mean and standard deviation of the ranking rewards among rewrites in group $\mathcal{G}$, and let $\epsilon_A > 0$ be a small constant for numerical stability. For rollout i, the sequence advantage is:

$$A_i^{\mathrm{seq}} = \frac{R(q_i, y) - \mu_{\mathcal{G}}}{\sigma_{\mathcal{G}} + \epsilon_A}. \tag{9}$$

A positive A_i^{seq} indicates that the rewrite performs better than the group average, while a negative value indicates that other sampled rewrites perform better. In standard GRPO, this sequence advantage is shared by all response tokens in the rewrite.

We augment this sequence signal with the edit credit from Section 4.2. Let t denote a response-token position and let $S_i(e)$ contain the token positions corresponding to edit e in rollout i. We define the token-level advantage as

$$A_{i,t} = A_i^{\mathrm{seq}} + \beta \sum_{e \in \mathcal{E}(x, q_i)} \mathbb{I}[t \in S_i(e)] \phi_z^+(e), \tag{10}$$

where β controls the strength of the edit-level signal. The edit bonus is added only to tokens associated with positively credited edits, while all other tokens retain the sequence advantage. Because the bonus is additive, it can also make a negative sequence advantage less negative without necessarily making the resulting token advantage positive. $A_{i,t}$ replaces the sequence advantage in the standard clipped GRPO objective. The policy ratio, clipping rule, and KL regularization remain unchanged.

4.4 THEORETICAL INTERPRETATION

The contribution of an edit is the change in semantic list utility caused by adding that edit to a query. This contribution can depend on the other changes in the rewrite. An edit may improve retrieval on its own but become partly redundant when similar information is already present. Conversely, an edit may become more useful when combined with other changes. The sufficiency and necessity tests evaluate the edit with no other edits present and with all other edits present, respectively. We show that, under diminishing or increasing returns, the smaller of these two effects equals the edit's smallest contribution across all combinations of the other edits.

Let $\mathcal{E}(x, q)$ denote the edits that transform the source query x into the completed rewrite q. For any subset $A \subseteq \mathcal{E}(x, q)$, let q_A be the query obtained by applying only the edits in A under the same alignment, so $q_\emptyset = x$ and $q_{\mathcal{E}(x,q)} = q$. We define $F_z(A) = U_z(L(q_A))$ as the semantic list utility after applying the edits in A. For an edit $e \notin A$, its contribution when the edits in A are already present is $m_e(A) = F_z(A \cup \{e\}) - F_z(A)$. The sufficiency effect is therefore $m_e(\emptyset)$, while the necessity effect is $m_e(\mathcal{E}(x, q) \setminus \{e\})$.

Proposition 1 (Minimum edit contribution)**.** *Suppose the contribution of an edit consistently decreases as other edits are added (F_z is submodular), or consistently increases (F_z is supermodular). Then, for every $e \in \mathcal{E}(x, q)$,*

$$\phi_z(e) = \min_{A \subseteq \mathcal{E}(x,q) \setminus \{e\}} m_e(A). \tag{11}$$

Under either condition, $\phi_z(e) > 0$ implies that the edit improves semantic list utility regardless of which other edits are present.

Under diminishing returns, an edit contributes the least when all other edits are present because some of its effect may already be captured by the rest of the rewrite. The necessity effect therefore gives its smallest contribution. Under increasing returns, the edit contributes the least when no other edits are present, so the sufficiency effect gives its smallest contribution. Taking the minimum therefore gives the smallest contribution across all edit contexts under either condition. If edit effects are additive, the contribution does not depend on the other edits, and the two tests recover the same value.

The result also provides a connection to the Shapley value, which averages an edit's contribution over different combinations of the other edits (Shapley, 1953). Under the conditions above, our score gives the smallest contribution across these combinations, so it cannot exceed the Shapley value. When edit interactions are unrestricted, this guarantee no longer holds. Even then, positive credit still has a direct meaning. The edit improves semantic list utility both when applied on its own and when retained in the complete rewrite. Appendix A provides the derivation.

5 EXPERIMENTS

We guide the experiments through five research questions: **RQ1** Does CGS improve retrieval performance over existing query-rewriting and credit-assignment methods? **RQ2**: How does the choice of retrieval utility for counterfactual credit affect performance? **RQ3** How do different constructions of the edit-credit score ϕ affect performance? **RQ4** How does the intent evidence used by the semantic teacher affect performance? **RQ5** How does CGS perform across model scales, and what is its computational cost?

5.1 DATASETS AND EXPERIMENTAL SETUP

Datasets. We evaluate on H&M and Amazon ESCI. H&M contains product queries paired with a future purchase, together with item metadata and user profiles derived from prior purchases (H&M Group, 2022). We treat the future purchase as the target product y and retain examples with a valid catalog target and purchase history. ESCI contains shopping queries with products labeled as Exact, Substitute, Complement, or Irrelevant (Reddy et al., 2022). For ESCI, y consists of the products labeled Exact for each query. In both datasets, during inference, the rewriter receives only the source query, while target metadata and, when available, user history are used only to compute training feedback. Dataset statistics are reported in Table 7 in Appendix B.1.

Evaluation protocol. We report NDCG@100, Recall@100, AP@100, and MRR@100 for full-catalog retrieval. During training, the ranking reward uses NDCG@1000 on H&M and NDCG@100

DATASET	H&M			ESCI		
METHOD	NDCG@100	Recall@100	AP/MRR@100	NDCG@100	Recall@100	AP/MRR@100
ZERO SHOT LLM INFERENCE						
Qwen2.5-1.5B (zero shot) (Qwen Team, 2024)	0.0459	0.1304	0.0271	0.1876	0.3573	0.1028 / 0.1742
Qwen2.5-3B (zero shot) (Qwen Team, 2024)	0.0366	0.1222	0.0176	0.2345	0.4255	0.1338 / 0.2297
GENERATIVE QUERY REWRITERS						
Rec-R1 (native) (Lin et al., 2025)	0.0563	0.1408	0.0365	0.1868	0.3585	0.1000 / 0.1810
OSPO (native) (Nath et al., 2026)	0.0536	0.1364	0.0346	0.2162	0.4024	0.1191 / 0.2094
RANKING REWARD CREDIT ASSIGNMENT						
GRPO (Shao et al., 2024)	0.1004	0.2099	0.0742	0.2360	0.4272	0.1353 / 0.2252
PPO/GAE (Schulman et al., 2017)	0.0864	0.1852	0.0647	0.2388	0.4261	0.1388 / 0.2378
TEMPO (Tran et al., 2025)	0.0952	0.1980	0.0706	0.2406	0.4368	0.1383 / 0.2293
VinePPO (Kazemnejad et al., 2025)	0.1034	0.2086	0.0779	0.2429	0.4413	0.1423 / 0.2254
OSPO-Prop (Nath et al., 2026)	0.0940	0.2036	0.0679	0.2253	0.4140	0.1281 / 0.2160
COUNTERFACTUAL EDIT CREDIT VARIANTS						
Target-NDCG	0.1018	0.2107	0.0757	0.2339	0.4184	0.1354 / 0.2277
Y_z-NDCG	0.0969	0.2020	0.0715	0.2413	0.4402	0.1387 / 0.2299
CGS	**0.1140**	**0.2205**	**0.0884**	**0.2548**	**0.4539**	**0.1500 / 0.2482**
% improvement over best baseline	+10.3%	+5.1%	+13.5%	+4.9%	+2.9%	+5.4%/ + 4.4%

Table 1: Dense retrieval results. Best results are bolded, and best baseline results are underlined. Results are averaged over three seeds.

on ESCI. H&M has one target product per query, so AP@100 and MRR@100 are identical, while ESCI may contain multiple Exact-labeled products and therefore reports them separately. All main results are averaged over three seeds on a fixed test split.

Baselines. We compare CGS against three groups of baselines. First, we include zero-shot Qwen2.5 (Qwen Team, 2024). Second, we compare with recent RL-based query-rewriting methods, Rec-R1 (Lin et al., 2025) and the original OSPO (Nath et al., 2026). Third, we compare credit-assignment strategies under the same query-rewriting setup, including GRPO (Shao et al., 2024), PPO with GAE (Schulman et al., 2017), TEMPO (Tran et al., 2025), VinePPO (Kazemnejad et al., 2025), and OSPO-Prop (Nath et al., 2026), which uses proportional Owen-Shapley credit.

Implementation details. Our main experiments use Qwen2.5-1.5B as the query rewriter and a fixed MPNet dense retriever (Song et al., 2020). We select checkpoints on the validation set. Appendix B.3 provides the dataset-specific teacher evidence, normalization, and weighting coefficients.

5.2 MAIN RESULTS (**RQ1** & **RQ2**)

Table 1 reports the main retrieval results. To isolate the effect of the utility used for counterfactual credit, we compare three variants that use the same PNS minimum estimator with different edit utilities. Target-NDCG uses target product NDCG and therefore only captures changes in target product rank. Y_z-NDCG instead uses the teacher score $Y_z(j)$ as graded relevance within NDCG, allowing the utility to reflect the semantic relevance and ranking position of retrieved products. CGS uses the semantic list utility U_z.

On H&M, CGS improves over the best baseline for each metric by 10.3% in NDCG@100, 5.1% in Recall@100, and 13.5% in AP/MRR@100. On ESCI, the corresponding improvements are 4.9% in NDCG@100, 2.9% in Recall@100, 5.4% in AP@100, and 4.4% in MRR@100. CGS also outperforms both Target-NDCG and Y_z-NDCG on the two datasets, showing that the semantic list utility provides a more effective signal for counterfactual edit credit. Appendix B.5 provides detailed definitions of the utility variants and the normalization used for Y_z-NDCG.

	H&M			ESCI		
Credit construction	NDCG@100	Recall@100	AP/MRR@100	NDCG@100	Recall@100	AP/MRR@100
R-GRPO (Shao et al., 2024)	0.1004	0.2099	0.0742	0.2360	0.4272	0.1353 / 0.2252
$R + \alpha U_z$	0.0833	0.1964	0.0566	0.2429	0.4350	0.1401 / 0.2393
$R + \Delta_{\text{nec}}$	0.0962	0.1986	0.0714	0.2388	0.4291	0.1363 / 0.2386
$R + \Delta_{\text{suf}}$	0.0977	0.2008	0.0728	0.2311	0.4201	0.1309 / 0.2256
$R + \frac{1}{2}(\Delta_{\text{nec}} + \Delta_{\text{suf}})$	0.0975	0.2044	0.0719	0.1911	0.3657	0.1043 / 0.1799
CGS (R +ϕ_z)	**0.1140**	**0.2205**	**0.0884**	**0.2548**	**0.4539**	**0.1500 / 0.2482**

Table 2: Component ablation. Bold marks the best result and underlining marks the best baseline.

5.3 Ablation Study (**RQ3**)

Table 2 isolates how semantic feedback is used for credit assignment while keeping everything else fixed. We compare adding the semantic list utility U_z directly to the sequence-level reward against edit credit based on necessity, sufficiency, their average, and the proposed counterfactual minimum. For the sequence-level variant, we use $R + \alpha U_z$, with $\alpha = 0.05$ on H&M to account for the larger scale of U_z and $\alpha = 1$ on ESCI. Compared with R-GRPO, counterfactual minimum credit improves NDCG@100 by 13.5% on H&M and 8.0% on ESCI. Adding U_z directly to the sequence-level reward improves performance on ESCI but degrades all metrics on H&M, suggesting that semantic feedback is more effective when assigned to individual edits rather than to the complete rewrite. Necessity and sufficiency alone also underperform the minimum, while averaging the two performs particularly poorly on ESCI. The minimum performs best across all metrics, supporting the use of credit that is positive only when an edit improves both counterfactual comparisons.

	H&M			ESCI		
Teacher evidence	NDCG@100	Recall@100	AP/MRR@100	NDCG@100	Recall@100	AP/MRR@100
Query only	0.0956	0.1960	0.0710	0.2284	0.4147	0.1275 / 0.2262
Target only	0.1113	**0.2215**	0.0848	0.2507	0.4476	0.1464 / 0.2437
Query + target	0.1099	0.2137	0.0849	**0.2548**	**0.4539**	**0.1500 / 0.2482**
Query + target + user profile	**0.1140**	0.2205	**0.0884**	n/a	n/a	n/a

Table 3: Effect of different evidence sources used by the semantic teacher.

5.4 Semantic Teacher Evidence (**RQ4**)

Table 3 studies how the evidence provided to the semantic teacher affects edit credit. Query evidence captures the information explicitly stated in the source query, while target evidence provides additional product attributes, *e.g.* product type, group, and color. On H&M, we also include user-profile evidence derived from prior purchases. On ESCI, combining query and target evidence improves over query-only evidence by 11.6% in NDCG@100, 9.5% in Recall@100, 17.6% in AP@100, and 9.7% in MRR@100. On H&M, target-only evidence achieves the highest recall, while adding the user profile gives the best NDCG and AP/MRR. These results show that target metadata provides substantial additional supervision beyond the source query, while user history can further improve the precision of the resulting edit credit on H&M.

Dataset	H&M			ESCI		
Method	NDCG@100	Recall@100	AP/MRR@100	NDCG@100	Recall@100	AP/MRR@100
Zero shot LLM inference						
Qwen2.5-3B (zero shot) (Qwen Team, 2024)	0.0366	0.1222	0.0176	0.2345	0.4255	0.1338 / 0.2297
Query generators						
Rec-R1 (native) (Lin et al., 2025)	0.0638	0.1610	0.0409	0.1998	0.3761	0.1092 / 0.1908
OSPO (native) (Nath et al., 2026)	0.0370	0.1240	0.0178	0.2343	0.4268	0.1337 / 0.2275
Ranking reward credit assignment						
GRPO (Shao et al., 2024)	0.1012	0.2006	0.0771	0.2368	0.4308	0.1339 / 0.2257
PPO/GAE (Schulman et al., 2017)	0.0990	0.2118	0.0713	0.2328	0.4260	0.1310 / 0.2265
TEMPO (Tran et al., 2025)	<u>0.1048</u>	<u>0.2174</u>	0.0777	<u>0.2449</u>	<u>0.4491</u>	0.1392 / 0.2380
VinePPO (Kazemnejad et al., 2025)	0.1015	0.1972	<u>0.0782</u>	0.2443	0.4398	<u>0.1398</u> / <u>0.2402</u>
OSPO-Prop (Nath et al., 2026)	0.0864	0.1978	0.0596	0.2248	0.4128	0.1259 / 0.2210
Counterfactual edit credit variants						
Target-NDCG	0.1108	0.2146	0.0857	0.2404	0.4434	0.1355 / 0.2356
Y_z-NDCG	0.1058	**0.2228**	0.0780	0.2523	0.4582	0.1466 / 0.2420
CGS	**0.1162**	0.2215	**0.0905**	**0.2677**	**0.4730**	**0.1570 / 0.2700**
% improvement over best baseline	$+10.9\%$	$+1.9\%$	$+15.7\%$	$+9.3\%$	$+5.3\%$	$+12.3\%/+12.4\%$

Table 4: Qwen2.5-3B dense retrieval results.

5.5 Model Scaling (**RQ5**)

Table 4 examines whether the gains of CGS persist with a larger query rewriter. On H&M, the zero-shot 3B model performs worse than its 1.5B counterpart, while several RL-based methods improve only modestly. In contrast, CGS maintains its gains and remains the strongest method across the reported metrics. On ESCI, the larger backbone benefits several methods, but CGS shows a larger improvement over the strongest baseline than in the 1.5B setting. These results suggest that the benefit of CGS is complementary to improvements from a stronger query rewriter.

5.6 Credit-Signal Analysis

We next examine the edit-credit signal itself. In particular, we ask whether semantic counterfactual credit provides supervision when the ranking reward is uninformative, and whether the necessity and sufficiency tests capture different aspects of an edit's retrieval effect. Ranking reward is zero for 77.8% of H&M sequences and 49.7% of ESCI sequences. These sequences provide no ranking-based distinction between the generated rewrites. However, among these zero-reward sequences, our method assigns positive edit credit to 40.5% on H&M and 55.7% on ESCI. Thus, the semantic feedback recovers training signal for a substantial fraction of examples that receive no signal from the ranking reward. At the same time, only 28.7% of H&M edits and 41.2% of ESCI edits receive positive credit, showing that the method is selective.

Measure	H&M	ESCI
$R = 0$ sequences (%)	77.8	49.7
$R = 0$ with positive edit credit (%)	40.5	55.7
Positive-credit edits (%)	28.7	41.2
$\text{sign}(\Delta_{\text{nec}}) = \text{sign}(\Delta_{\text{suf}})$ (%)	61.4	71.1

Table 5: Credit-signal analysis.

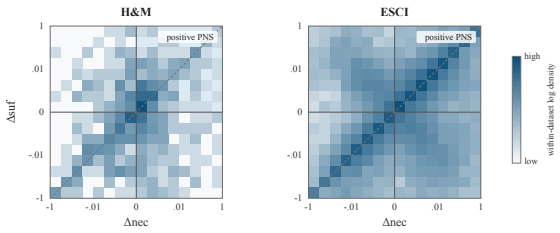


Figure 2: Distribution of Δ_{nec} and Δ_{suf} used for edit credit.

Figure 2 shows the necessity and sufficiency effects for individual edits. The x-axis measures how much utility is lost when an edit is removed from the complete rewrite, and the y-axis measures how much utility is gained when the edit is added to the source query. An edit receives positive credit only when both effects are positive, corresponding to the upper-right quadrant. Many edits are positive under only one test, showing that an edit can help on its own without remaining useful in the full rewrite, or vice versa. This supports evaluating both effects when assigning credit.

5.7 Efficiency Analysis

Method	Extra policy rollouts	Retrieval evaluations	Credit unit
GRPO	0	1	Sequence
PPO/GAE	0	1	Token
TEMPO	0	1	Token prefix
VinePPO	nS	$\leq 1 + nS$	Step
OSPO-Prop	0	$\leq 1 + C$	Segment
CGS	0	$\leq 2 + 2E$	Edited segment

Table 6: Policy rollouts and retrieval evaluations per sampled response, with shared endpoints counted once.

CGS assigns edit-level credit without additional policy rollouts. For a rewrite with E edits, we evaluate the source query and complete rewrite once, then evaluate two counterfactual queries per edit: one with the edit removed and one with it applied alone. This gives at most $2 + 2E$ retrieval evaluations, so the cost grows linearly with the number of edits.

Table 6 compares these evaluation counts across credit estimators. VinePPO uses nS additional rollouts, where n is the number of rollouts per prompt and S is the number of nonterminal step boundaries. OSPO-Prop instead requires up to C additional retrieval evaluations for its nonempty coalitions.

6 Conclusion

We introduced CGS, an RL framework for product query rewriting that assigns counterfactual credit to individual edits using semantic retrieval feedback, while retaining ranking reward for complete rewrites. CGS evaluates each edit through counterfactual removal and standalone application. Across H&M and Amazon ESCI, CGS consistently outperforms query-rewriting and credit-assignment baselines at both model scales. Ablations show that semantic list utility provides effective edit credit, including for many sequences with zero ranking reward.

AI USE STATEMENT

We used generative AI tools, primarily OpenAI GPT-5.6 and GPT-6, during several stages of this work. Early in the project, we used these tools to help search for and identify relevant literature. We also used them to assist with parts of the implementation and to provide feedback on code. All AI-assisted implementation was reviewed and verified by the authors, who take full responsibility for the final results. During paper preparation, we used generative AI to revise author-written drafts for clarity and readability, and to help refine the presentation of methodological, theoretical, and experimental discussions. The research decisions, experimental design, analysis, and final claims were made and verified by the authors. We reviewed all AI-assisted content and take full responsibility for the final manuscript.

REFERENCES

Sanjay Agrawal, Srujana Merugu, and Vivek Sembium. Enhancing e-commerce product search through reinforcement learning-powered query reformulation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Aijun Dai, Zhenyu Zhu, Haiqing Hu, Guoyu Tang, Lin Liu, and Sulong Xu. Enhancing e-commerce query rewriting: A large language model approach with domain-specific pre-training and reinforcement learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 4439–4445, 2024. doi: 10.1145/3627673.3680109.

Aijun Dai, Jixiang Zhang, Haiqing Hu, Guoyu Tang, Lin Liu, and Ziguang Cheng. Relevance matters: A multi-task and multi-stage large language model approach for e-commerce query rewriting. *arXiv preprint arXiv:2603.02555*, 2026.

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, and Remi Munos. Hindsight credit assignment. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

H&M Group. H&M Personalized Fashion Recommendations. Kaggle competition, 2022. URL https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002. doi: 10.1145/582415.582418. URL https://doi.org/10.1145/582415.582418.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29557–29590, 2025.

Victor Lavrenko and W. Bruce Croft. Relevance-based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 120–127. ACM, 2001. doi: 10.1145/383952.383972.

Yugu Li, Zehong Cao, Jianglin Qiao, and Siyi Hu. SSVPO: Effective step-level credit assignment for RL training of language models. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4da4dd613cc9e63932d643b9c6e8f86f-Abstract-Conference.html.

Jiacheng Lin, Tian Wang, and Kun Qian. Rec-R1: Bridging generative large language models and user-centric recommendation systems via reinforcement learning. *arXiv preprint arXiv:2503.24289*, 2025.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5303–5315. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.322. URL https://aclanthology.org/2023.emnlp-main.322/.

Abhijnan Nath, Alireza Bagheri Garakani, Tianchen Zhou, Fan Yang, Yan Gao, and Nikhil Krishnaswamy. Owen-shapley policy optimization: A principled rl algorithm for generative search llms. *arXiv preprint arXiv:2601.08403*, 2026.

Duy A. Nguyen, Rishi Kesav Mohan, Shimeng Yang, Pritom Saha Akash, and Kevin Chen-Chuan Chang. MiniELM: A lightweight and adaptive query rewriting framework for e-commerce search optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6952–6964, 2025. doi: 10.18653/v1/2025.findings-acl.363.

Rodrigo Nogueira and Kyunghyun Cho. Task-oriented query reformulation with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 574–583. Association for Computational Linguistics, 2017. doi: 10.18653/v1/D17-1061.

Yijun Pan, Weikang Qiu, Qiyao Ma, Mingxuan Ju, Tong Zhao, Neil Shah, and Rex Ying. FlexRec: Adapting llm-based recommenders for flexible needs via reinforcement learning. *arXiv preprint arXiv:2603.11901*, 2026.

Judea Pearl. Probabilities of causation: Three counterfactual interpretations and their identification. *Synthese*, 121(1–2):93–149, 1999.

Qwen Team. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024.

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

Joseph J. Rocchio. Relevance feedback in information retrieval. In Gerard Salton (ed.), *The SMART Retrieval System: Experiments in Automatic Document Processing*, pp. 313–323. Prentice-Hall, 1971.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/98711dea460bdefe0e651ca23ec98ba2-Abstract-Conference.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Lloyd S. Shapley. A value for n-person games. In *Contributions to the Theory of Games II*, pp. 307–317. Princeton University Press, 1953.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pp. 16857–16867, 2020.

Hieu Tran, Zonghai Yao, and Hong Yu. Exploiting tree structure for credit assignment in rl training of llms. *arXiv preprint arXiv:2509.18314*, 2025.

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*, 2025.

Zaiyi Zheng, Guanghui Min, Yaochen Zhu, Liang Wu, Liangjie Hong, Chen Chen, and Jundong Li. SAPO: Step-aligned policy optimization for reasoning-based generative recommendation. *arXiv preprint arXiv:2605.17648*, 2026.

Yaochen Zhu, Harald Steck, Dawen Liang, Yinhan He, Vito Ostuni, Jundong Li, and Nathan Kallus. Rank-GRPO: Training llm-based conversational recommender systems with reinforcement learning. *arXiv preprint arXiv:2510.20150*, 2025.

A Theoretical Interpretation

The semantic list utility combines product-level relevance scores, while edit credit measures how this utility changes under query edits. We first establish properties of the semantic list utility, then prove Proposition 1 and examine how edit interactions and normalization affect the resulting credit.

A.1 Properties of the Semantic List Utility

To study how the utility combines product scores, we hold each retrieved product and its rank fixed. Let $v = (j, r)$ denote product j at rank r, with rank-discounted relevance score $p_v = \gamma^{r-1} Y_z(j) \in [0, 1]$. For a set A of product–rank pairs, define

$$C_z(A) = 1 - \prod_{v \in A} (1 - p_v). \tag{12}$$

For a retrieved list $L = (j_1, \ldots, j_K)$, taking $A = \{(j_r, r) : r = 1, \ldots, K\}$ gives $C_z(A) = U_z(L)$.

Proposition 2 (Semantic list utility)**.** *C_z is normalized, bounded in $[0, 1]$, monotone, and submodular over fixed product–rank pairs.*

Proof. The empty product gives $C_z(\emptyset) = 0$, and $p_v \in [0, 1]$ implies $C_z(A) \in [0, 1]$. Adding a pair $v \notin A$ changes the utility by

$$C_z(A \cup \{v\}) - C_z(A) = p_v \prod_{u \in A} (1 - p_u) \geq 0, \tag{13}$$

so C_z is monotone. If $A \subseteq B$ and $v \notin B$, then

$$p_v \prod_{u \in A} (1 - p_u) \geq p_v \prod_{u \in B} (1 - p_u), \tag{14}$$

because each additional factor lies in $[0, 1]$. Thus, the gain from adding the same product–rank pair decreases as more pairs are already present, establishing submodularity. □

Equation 13 explains the diminishing gain in the semantic list utility. Once the current list already contains highly relevant products, another relevant product contributes less additional utility. This result concerns fixed product–rank pairs. Query edits can change both the retrieved products and their ranks, so it does not imply that the edit-level utility defined below is submodular.

The utility also admits a probabilistic interpretation. Let $B_v \sim \text{Bernoulli}(p_v)$ indicate whether product–rank pair v satisfies the underlying need, independently given z. Then

$$\Pr\left(\bigcup_{v \in A} \{B_v = 1\} \,\middle|\, z \right) = 1 - \prod_{v \in A} (1 - p_v) = C_z(A). \tag{15}$$

Under this interpretation, the semantic list utility is the probability that at least one retrieved product satisfies the need.

A.2 Minimum Edit Contribution

Fix a source query x, a complete rewrite q, and the edit set $\mathcal{E}(x, q)$. For any $A \subseteq \mathcal{E}(x, q)$, let q_A be the query obtained by applying only the edits in A under the original alignment. As in Section 4.4, define $F_z(A) = U_z(L(q_A))$. For an edit $e \notin A$, its contribution when the edits in A are already present is $m_e(A) = F_z(A \cup \{e\}) - F_z(A)$. The sufficiency effect is $m_e(\emptyset)$, while the necessity effect is $m_e(\mathcal{E}(x, q) \setminus \{e\})$.

Proof of Proposition 1. If edit effects are additive, then F_z is modular and can be written as $F_z(A) = c + \sum_{f \in A} a_f$. In this case, $m_e(A) = a_e$ for every $A \subseteq \mathcal{E}(x, q) \setminus \{e\}$, so necessity and sufficiency recover the same contribution.

Suppose instead that edit contributions decrease as more edits are present, so F_z is submodular. Then $m_e(A) \geq m_e(B)$ whenever $A \subseteq B \subseteq \mathcal{E}(x,q) \setminus \{e\}$. Therefore,

$$m_e(\emptyset) \geq m_e(A) \geq m_e(\mathcal{E}(x,q) \setminus \{e\}). \tag{16}$$

The necessity effect is therefore the smallest contribution across all combinations of the other edits.

If edit contributions instead increase as more edits are present, so F_z is supermodular, the inequalities reverse:

$$m_e(\emptyset) \leq m_e(A) \leq m_e(\mathcal{E}(x,q) \setminus \{e\}). \tag{17}$$

The sufficiency effect is then the smallest contribution. Under either condition,

$$\phi_z(e) = \min_{A \subseteq \mathcal{E}(x,q) \setminus \{e\}} m_e(A). \tag{18}$$

Thus, $\phi_z(e) > 0$ implies that the edit improves semantic list utility for every combination of the other edits.

The Shapley value averages the contribution of edit e over different combinations of the other edits:

$$\mathrm{Sh}_e(F_z) = \sum_{A \subseteq \mathcal{E}(x,q) \setminus \{e\}} \frac{|A|!\,(|\mathcal{E}(x,q)| - |A| - 1)!}{|\mathcal{E}(x,q)|!}\, m_e(A). \tag{19}$$

The weights are nonnegative and sum to one. Under either diminishing or increasing returns, $\phi_z(e)$ is the smallest contribution in this average and therefore lower-bounds $\mathrm{Sh}_e(F_z)$.

With unrestricted edit interactions, this guarantee need not hold. The minimum still has a direct interpretation because

$$\min\{a, b\} = \max\{c \in \mathbb{R} : c \leq a \text{ and } c \leq b\}. \tag{20}$$

Thus, $\phi_z(e)$ is the largest value that does not exceed either measured effect. Positive credit therefore requires the edit to improve utility both on its own and within the completed rewrite. □

A.3 Effect of Edit Interactions

To illustrate how interactions between edits affect credit, consider a model with standalone edit effects and pairwise interactions:

$$F_z(A) = c + \sum_{f \in A} a_f + \sum_{\{f,g\} \subseteq A} b_{fg}. \tag{21}$$

Here, a_e is the contribution of edit e on its own, while b_{ef} measures how edit f changes that contribution. Substituting this model into the two counterfactual comparisons gives

$$\Delta_{\mathrm{suf}}(e) = a_e, \tag{22}$$

$$\Delta_{\mathrm{nec}}(e) = a_e + \sum_{f \in \mathcal{E}(x,q) \setminus \{e\}} b_{ef}, \tag{23}$$

$$\phi_z(e) = a_e + \min\left\{0, \sum_{f \in \mathcal{E}(x,q) \setminus \{e\}} b_{ef}\right\}. \tag{24}$$

If the other edits increase the contribution of e, the minimum retains its standalone contribution a_e. If they reduce its contribution, the minimum uses the smaller contribution measured within the complete rewrite. Thus, the score reflects the weaker of the edit's standalone and combined effects.

A.4 Effect of Utility Scaling on Normalized Credit

MAD normalization makes edit credit invariant to positive affine transformations of the semantic list utility when the denominator floor is inactive. Let $\widetilde{U} = aU + b$ with $a > 0$. The offset b cancels in each counterfactual difference, giving $\widetilde{\Delta}_{\mathrm{nec}} = a\Delta_{\mathrm{nec}}$, $\widetilde{\Delta}_{\mathrm{suf}} = a\Delta_{\mathrm{suf}}$, and $\widetilde{\phi} = a\phi$. Likewise,

$$\mathrm{MAD}(a\Phi_{\mathcal{B}}) = a\ \mathrm{MAD}(\Phi_{\mathcal{B}}). \tag{25}$$

When the denominator floor is inactive, the scale factor therefore cancels:

$$\mathrm{clip}\left(\frac{\widetilde{\phi}_z(e)}{\mathrm{MAD}(\widetilde{\Phi}_{\mathcal{B}})}, 0, c\right) = \mathrm{clip}\left(\frac{\phi_z(e)}{\mathrm{MAD}(\Phi_{\mathcal{B}})}, 0, c\right). \tag{26}$$

When the floor is active, rescaling can change the magnitude of normalized credit but does not change which edits receive positive credit. For $\epsilon_\phi > 0$ and $c > 0$, Equation 8 gives $\phi_z^+(e) > 0$ exactly when $\phi_z(e) > 0$. Thus, normalization and clipping preserve the requirement that both counterfactual effects are positive.

B Additional Experimental Details

B.1 Dataset Construction

Dataset	Catalog items	Train	Validation	Test	User context
H&M	105,542	24,960	5,000	5,000	✓
ESCI	1,367,729	4,510	100	798	×

Table 7: Dataset statistics.

H&M. *Queries and targets.* Each H&M example consists of a search query paired with one target product. We join the public search queries with their relevance judgments using query IDs and retain queries of 1–30 words with at least one relevant product in the catalog. We also require the target product to have complete metadata, including its name, product type, product group, and color. After removing duplicate query–target pairs, we use the first relevant article as the target when multiple relevant products are available.

User profiles. The H&M search data do not identify the customer who issued each query, so the original query author's purchase history is unavailable. To provide user-context information, we associate each query–target pair with a customer who purchased the target product. This customer serves only as a source of preference history and may not be the original query author. We require at least five purchases before the target purchase and summarize up to 20 earlier purchases into a profile containing age, frequently purchased product types, colors, sections, garment groups, and recent items.

Splits. We sample 24,960 training, 5,000 validation, and 5,000 test examples using seed 20260629 while preserving the distribution of target product groups across splits. Splitting is performed at the example level rather than the customer level, so the same profile customer may appear in more than one split.

ESCI. *Queries and targets.* Each ESCI example consists of a shopping query and its Exact-labeled products. We use U.S. query–product pairs from the ESCI small-version release and retain Exact products with valid metadata and a known Amazon category. We then group products by normalized query, so a single query can have multiple target products.

Splits. The ESCI training partition contains 4,610 queries. We reserve 100 queries for validation and use the remaining 4,510 for training after shuffling. The official 798-query evaluation set is used for testing and covers Video Games, Baby Products, Office Products, and Sports and Outdoors.

B.2 Implementation Details

Table 8 lists the hyperparameters used in the main 1.5B experiments. On both datasets, edit credit is defined as the minimum of the necessity and sufficiency effects, then normalized and clipped before being added to the token-level advantage after GRPO group normalization.

Setting	H&M	ESCI
Policy initialization	Qwen2.5-1.5B-Instruct	
Retriever	fixed all-mpnet-base-v2 FAISS	
Rank decay γ	0.85	0.85
Edit-credit coefficient β	0.02	0.05
MAD floor ϵ_ϕ	0.02	0.02
GRPO stability constant ϵ_A	10^{-6}	10^{-6}
Normalized-credit clip c	3.0	3.0
PPO ratio clip	0.2	0.2
Actor learning rate	10^{-6}	10^{-6}
KL coefficient	0.001	0.001
Rollouts per query	8	8
Train / validation batch size	32 / 32	32 / 32
PPO mini / micro batch size	128 / 2	128 / 2
Sampling (temperature, top-p, top-k)	$(0.6, 0.95, -1)$	$(0.6, 0.95, -1)$
Maximum prompt / response length (tokens)	512 / 356	256 / 356
Training ranking reward	NDCG@1000	NDCG@100
Semantic list utility depth	20	100

Table 8: Hyperparameters for the primary experiments.

B.3 Metadata Relevance Teacher

We instantiate the relevance teacher in Equation 3 with a lightweight metadata-matching model. The teacher scores how well each retrieved product matches the available evidence about user intent. On H&M, the evidence includes the source query, target-product metadata, and user profile, with weights $(.30, .60, .10)$, respectively. The target metadata includes product text, product type, product group, and color, while the user profile summarizes purchases made before the target purchase. On ESCI, we use the source query and target-product metadata with weights $(.40, .60)$. The target metadata includes product text, category, brand, and color.

For each evidence field, we normalize its terms and compare them with the normalized metadata terms of the retrieved product. The field-level compatibility score is the fraction of evidence terms matched by the product, with terms mapped to the same normalized form treated as matches. The weighted combination of these field-level scores gives the final teacher score $Y_z(j)$.

Target metadata and user profiles are used only during training to compute semantic feedback. The query rewriter receives only the source query, and the fixed retriever receives only the generated rewrite.

	H&M			ESCI		
Method / teacher	NDCG@100	Recall@100	AP/MRR@100	NDCG@100	Recall@100	AP/MRR@100
			Qwen2.5-1.5B			
GRPO	0.1004	0.2099	0.0742	0.2360	0.4272	0.1353 / 0.2252
CGS (MPNet)	0.1101	0.2115	0.0801	0.2366	0.4241	0.1380 / 0.2296
CGS	0.1140	0.2205	0.0884	0.2548	0.4539	0.1500 / 0.2482
			Qwen2.5-3B			
GRPO	0.1012	0.2006	0.0771	0.2368	0.4308	0.1339 / 0.2257
CGS (MPNet)	0.1134	0.2143	0.0821	0.2415	0.4396	0.1402 / 0.2417
CGS	0.1162	0.2215	0.0905	0.2677	0.4730	0.1570 / 0.2700

Table 9: Teacher implementations experiment results with 1.5B and 3B rewriters.

B.4 Teacher Implementation Comparison

Table 9 compares our lightweight metadata-overlap teacher with an MPNet-based semantic teacher at both 1.5B and 3B model scales. The overlap teacher consistently gives the strongest overall retrieval performance, while the MPNet teacher provides smaller or less consistent gains over GRPO.

The difference is especially clear on ESCI, where the overlap teacher improves all reported metrics at both model sizes, whereas the MPNet variant remains close to the GRPO baseline. On H&M, both teacher implementations improve over GRPO, but the overlap teacher again performs best. These results suggest that the proposed framework does not depend on a particular teacher architecture, while the simple metadata-overlap teacher provides sufficient supervision in our setting.

B.5 Retrieval Utility Ablations (**RQ2**)

Table 1 compares three choices of retrieval utility for computing the necessity and sufficiency effects. All three variants use the same sequence-level ranking reward R, minimum edit-credit rule, positive-credit filtering, and MAD normalization. They differ only in how the four counterfactual retrieval lists are scored.

Target-NDCG. This variant computes the counterfactual effects directly from labeled-target NDCG:

$$\Delta_{\mathrm{nec}}^{\mathrm{target}}(e) = R(q, y) - R(q^{-e}, y), \tag{27}$$

$$\Delta_{\mathrm{suf}}^{\mathrm{target}}(e) = R(x^{+e}, y) - R(x, y). \tag{28}$$

The edit credit is the minimum of these two effects. Because R depends only on target-product ranks, this variant cannot distinguish counterfactual retrievals when those ranks remain unchanged.

Y_z-NDCG. This variant replaces binary target relevance with the teacher score $Y_z(j)$ as graded relevance within NDCG. It can therefore distinguish retrieved products by both semantic relevance and ranking position.

Because the four counterfactual queries may retrieve different products, independently normalized NDCG values would not be directly comparable. For each edit e, we therefore form a common candidate pool $\mathcal{C}_e$ from the products retrieved by q, q^{-e}, x^{+e}, and x, removing duplicates. We rank this pool by Y_z to construct a single ideal DCG and use the same denominator for all four retrieval lists:

$$V_{z,e}^{\mathrm{NDCG}}(L) = \frac{\displaystyle\sum_{r=1}^{K} \frac{2^{Y_z(j_r)} - 1}{\log_2(r+1)}}{\displaystyle\sum_{r=1}^{\min(K, |\mathcal{C}_e|)} \frac{2^{Y_z(j^\star_{e,r})} - 1}{\log_2(r+1)}}, \qquad \mathcal{C}_e = \mathrm{uniq}\big(L(q) \cup L(q^{-e}) \cup L(x^{+e}) \cup L(x)\big), \tag{29}$$

where $(j^\star_{e,1}, j^\star_{e,2}, \ldots)$ orders the products in $\mathcal{C}_e$ by decreasing teacher score. Using a common denominator ensures that differences between the four utilities reflect changes in the retrieved products and their ranks rather than changes in NDCG normalization. We use the same teacher evidence and weights as in the full model, described in Appendix B.3.

CGS. Our full model uses the semantic list utility U_z from Equation 4. Unlike the NDCG variants, U_z directly combines the teacher scores of the retrieved products, giving greater weight to relevant products at higher ranks while providing diminishing gains as additional relevant products are retrieved. The teacher scores and edit-credit construction are otherwise identical to those used in Y_z-NDCG.

B.6 Credit-Signal Analysis

Table 10 examines whether PNS provides edit-level feedback when ranking reward is zero and how the necessity and sufficiency effects relate to each other. Figure 2 shows their joint distribution. The H&M sample contains 553 sequences and 1,186 edits, with both effects available for 830 edits. The ESCI sample contains 6,204 sequences and 10,584 edits. The reported correlation compares $\phi_z(e)$ with the mean of the two effects, $\frac{1}{2}(\Delta_{\mathrm{nec}}(e) + \Delta_{\mathrm{suf}}(e))$.

Ranking reward is zero for 77.8% of H&M sequences and 49.7% of ESCI sequences. Among these zero-reward sequences, 40.5% and 55.7% still receive positive PNS edit credit. Thus, edits can

Measure	H&M	ESCI
$R = 0$ sequences (%)	77.8	49.7
$R = 0$ with positive edit credit (%)	40.5	55.7
Positive-credit edits (%)	28.7	41.2
$\mathrm{sign}(\Delta_{\mathrm{nec}}) = \mathrm{sign}(\Delta_{\mathrm{suf}})$ (%)	61.4	71.1
$\rho\left(\phi_z, \frac{\Delta_{\mathrm{nec}}+\Delta_{\mathrm{suf}}}{2}\right)$	0.853	0.828
$\Delta_{\mathrm{nec}} \leq \Delta_{\mathrm{suf}}$ (%)	49.6	56.5

Table 10: Credit-signal analysis. ρ denotes Spearman correlation.

improve semantic list utility even when no target product is retrieved. Across all edits, 28.7% on H&M and 41.2% on ESCI receive positive credit.

Necessity and sufficiency agree in sign for 61.4% of H&M edits and 71.1% of ESCI edits. PNS credit has Spearman correlations of 0.853 and 0.828 with their mean. The ordering $\Delta_{\mathrm{nec}} \leq \Delta_{\mathrm{suf}}$ holds for 49.6% and 56.5% of edits, respectively. These are edits whose contribution on the source query is at least as large as their contribution within the complete rewrite.

B.7 Discussion

Our results suggest that fine-grained credit assignment requires both informative retrieval feedback and a mechanism for attributing that feedback to individual query edits. The semantic list utility distinguishes retrieval changes beyond target-product ranks, while the counterfactual tests determine which edits produced those changes. The ablations show that using semantic feedback for edit-level credit is more effective than adding it directly to the sequence-level reward.

The method also has several limitations. Credit quality depends on the available intent evidence and relevance teacher, and H&M user profiles come from customers who purchased the target product rather than the original query author. In addition, counterfactual credit requires extra retrieval evaluations for each edit, although no additional policy rollouts are needed. Future work could explore stronger learned teachers, observed user context, and joint adaptation of the rewriter and retriever.