LEARNING RETRIEVAL REFINEMENT FROM GENERATION GRADIENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-Augmented Generation (RAG) systems are fundamentally constrained by retrieval quality, yet user queries often fail to capture the evidence required for accurate downstream generation. We observe that LLM generation gradients inherently encode missing semantic information in retrieved context, providing a useful signal for constructing retrieval refinement pseudo-labels from answer supervision. Based on this insight, we propose Gradient-Driven Query Embedding Refinement (GDER), a novel framework that reformulates query optimization as gradient-driven embedding refinement in dense retrieval space. Instead of costly autoregressive query rewriting, GDER learns lightweight retrieval-space correction offsets directly from generation gradients, substantially reducing both training and inference overhead. GDER requires no modification to either the underlying LLM or retriever and can be seamlessly integrated into existing RAG pipelines. Experiments across multiple open-domain question answering benchmarks demonstrate that GDER consistently improves both retrieval and answer quality while achieving significantly higher efficiency than existing query optimization methods. The model, data, and code are available at `https://anonymous.4open.science/r/gder-60EB`.

1 INTRODUCTION

Retrieval-Augmented Generation (RAG) enhances large language models (LLMs) by incorporating external knowledge, substantially improving performance on knowledge-intensive tasks(Lewis et al., 2020; Izacard et al., 2023; Ma et al., 2023b). However, its effectiveness is fundamentally constrained by retrieval quality, which depends on whether user queries can accurately bridge the semantic gap between user intent and relevant documents in the knowledge base(Xiao et al., 2024; Zhou et al., 2023). In practice, user queries are often incomplete, ambiguous, or semantically misaligned with the evidence required for correct answers, making query optimization a critical technique for bridging this semantic gap and improving downstream generation quality(Maynez et al., 2020).

Existing query optimization methods fall into two paradigms: supervised fine-tuning and reinforcement learning. Supervised approaches train query rewriting models from labeled data(Ma et al., 2023a; Peng et al., 2024), but rely on costly annotations(Tayal & Tyagi, 2024) and heuristic prompt engineering(Wang et al., 2025; 2024; Zhu et al., 2025), limiting scalability. Reinforcement learning methods remove explicit supervision, yet introduce significant overhead. Methods such as LeReT(Hsu et al., 2025) and DeepRetrieval(Jiang et al., 2025) require repeated rewriting–retrieval–feedback loops for policy optimization, incurring high training costs. Their reliance on autoregressive query rewriting at inference time leads to increased latency, making them inefficient for practical deployment.

Beyond these substantial training and inference costs, existing methods largely overlook an intrinsic supervision signal embedded within the LLM generation process. We posit that generation gradients directly encode retrieval deficiencies and therefore provide actionable guidance for retrieval optimization. To validate this hypothesis, we conduct a controlled study under varying retrieval conditions, including oracle evidence, semantically relevant documents, and randomly sampled documents. As shown in Figure 1, our analysis reveals two key findings: generation loss and gradient magnitudes systematically reflect the adequacy of retrieved evidence, and generation gradients exhibit a stable directional alignment with latent semantic deficiencies, defined as the representation gap between imperfectly retrieved documents and oracle evidence in the embedding space. These results show

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.

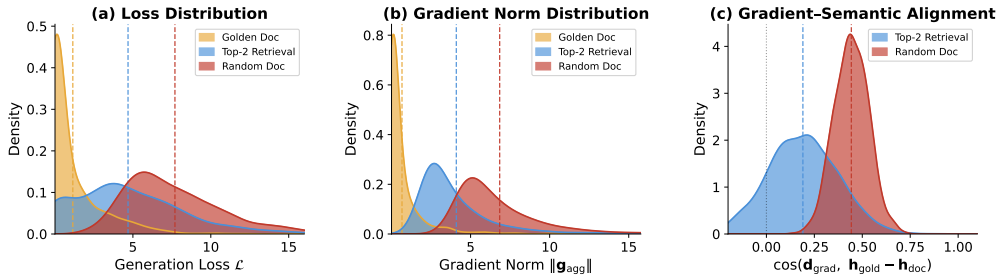


Figure 1: Generation loss distributions under different retrieval conditions (left) and gradient magnitude distributions of the loss with respect to input representations (middle). The right figure shows the directional alignment between generation gradients and latent semantic deficiencies.

that generation gradients consistently point toward the missing semantic information required for correct reasoning, providing a principled and interpretable basis for constructing retrieval refinement pseudo-labels from answer supervision without requiring costly retrieval-specific annotations or large-scale exploration.

Building on these insights, we propose Gradient-Driven Query Embedding Refinement (GDER), a novel framework that reformulates query optimization as a gradient-driven embedding refinement problem. Rather than rewriting queries in text space, GDER directly optimizes query representations in dense retrieval space using intrinsic feedback signals derived from LLM generation gradients. Specifically, GDER learns retrieval-oriented correction directions from generation gradients and trains a lightweight prediction network to estimate retrieval-space refinement offsets. During inference, the learned network directly predicts adaptive embedding refinements without requiring inaccessible gradient computation or costly autoregressive query rewriting. This design enables efficient and generation-aware retrieval optimization.

Our contributions are summarized as follows:

- We identify that LLM generation gradients encode the direction of missing semantic information in retrieved context, and leverage them to construct retrieval refinement pseudo-labels from answer supervision.
- We propose GDER, a framework that reformulates query optimization as gradient-driven embedding refinement, enabling direct optimization in dense retrieval space without autoregressive query rewriting or policy learning.
- We demonstrate that GDER consistently improves retrieval and answer quality across multiple QA benchmarks, while significantly reducing both training and inference overhead compared to existing query optimization methods.

2 Related Work

Heuristic and Prompt-Based Query Optimization. Early query optimization methods primarily improve retrieval quality through heuristic rules or prompt-based strategies(Jeong et al., 2024; Sun et al., 2025; Jiang et al., 2023; Asai et al., 2024; Cheng et al., 2024; Zhu et al., 2025). Heuristic-based approaches refine the original query using signals generated during model inference. For example, DRAGIN extracts salient query elements from attention patterns to construct retrieval-oriented rewritten queries(Su et al., 2024); SeaKR leverages uncertainty-aware internal signals to identify underspecified knowledge components and refine retrieval focus accordingly(Yao et al., 2025); DioR dynamically adjusts query composition by estimating token-level importance and reorganizing query emphasis(Guo et al., 2025). Prompt-based methods instead rely on carefully designed prompts and contextual information to reformulate the original query into retrieval-optimized variants. HyDE generates hypothetical documents to construct retrieval-oriented rewritten queries(Gao et al., 2023). DeepNote RAG and Adaptive-Note RAG further utilize accumulated retrieval notes to iteratively generate refined queries(Wang et al., 2025; 2024). Despite their effectiveness, these approaches remain heavily dependent on handcrafted heuristics, prompt engineering quality and manually

designed rewriting templates. As a result, query optimization remains largely indirect, leading to limited alignment between rewritten queries and downstream generation objectives.

Reinforcement Learning-Based Query Optimization. To address the limitations of heuristic and prompt-based approaches, recent work increasingly formulates query optimization as a reinforcement learning problem, enabling models to learn retrieval strategies directly from task-level feedback(Song et al., 2024; Gao et al., 2025; Guan et al., 2025). LeReT introduces preference-based retrieval optimization that allows LLMs to iteratively explore diverse query formulations(Hsu et al., 2025). SmartRAG further integrates retrieval triggering, query rewriting and answer generation into a unified PPO-based policy framework, directly optimizing retrieval behavior according to downstream task outcomes(Gao et al., 2025). DeepRetrieval extends RL-based query optimization to broader retrieval settings by training LLMs to generate improved search queries through reward-driven optimization(Jiang et al., 2025). However, RL-based methods introduce several practical challenges, including expensive exploration, complex policy optimization and carefully designed reward functions. More importantly, retrieval-oriented rewards are not always fully aligned with final generation quality, since improvements in retrieval metrics do not necessarily translate into evidence that is most beneficial for downstream reasoning and answer generation. Furthermore, existing RL-based methods still rely on autoregressive textual query rewriting, resulting in substantial computational overhead while failing to fully exploit the rich intrinsic gradient and hidden-state signals already embedded within the generation process itself.

3 METHOD

We propose **GDER**, a three-stage framework that leverages LLM generation gradients derived from answer supervision to learn query embedding corrections in retrieval space. An overview of the framework is illustrated in Figure 2. Given a question–answer dataset and a dense retrieval index constructed over a document corpus, GDER consists of three stages: (1) gradient-guided data construction, (2) offset network training, and (3) efficient inference.

3.1 PRELIMINARIES AND PROBLEM FORMULATION

Let $\mathcal{E} : \mathcal{Q} \to \mathbb{R}^{d_e}$ denote a dense retrieval encoder (e.g., BGE) (Xiao et al., 2024), where d_e denotes the dimensionality of the retrieval embedding space. Given an input query q, the encoder maps it into a dense representation $\mathbf{e}_q = \mathcal{E}(q)$, which is then used to retrieve the top-k relevant documents docs_0 via maximum inner product search over the document corpus. The retrieved documents are concatenated with the task prompt and provided to a large language model for answer generation (Lewis et al., 2020). The quality of the generated response is evaluated using the autoregressive cross-entropy loss $\mathcal{L}$ over the target answer sequence, which serves as the primary supervision signal for measuring whether the retrieved evidence sufficiently supports downstream generation.

Our key observation is that the gradient of the generation loss with respect to LLM input embeddings encodes fine-grained information about semantic deficiencies in the retrieved context (Chen et al., 2026). When retrieval fails to provide sufficient supporting evidence, the resulting gradients implicitly capture the missing semantic components required for correct answer generation. Based on this observation, we define a correction vector $\boldsymbol{\delta} \in \mathbb{R}^{d_e}$ in retrieval embedding space, where the refined query representation $\mathbf{e}_q^* = \text{normalize}(\mathbf{e}_q + \alpha\boldsymbol{\delta})$ guides retrieval toward more informative evidence and consequently improves downstream generation quality.

However, generation gradients are only available during training and require expensive backpropagation through the LLM, making them impractical for direct use at inference time. To address this limitation, we train a lightweight predictor $F_\phi : \mathbb{R}^{d_h} \to \mathbb{R}^{d_e}$ that maps contextual LLM hidden states $\mathbf{h}_q \in \mathbb{R}^{d_h}$ directly to the desired retrieval correction vector $\boldsymbol{\delta}$(Tolstikhin et al., 2021; Guo et al., 2021). This design reformulates gradient-guided retrieval refinement as an efficient feed-forward prediction problem, enabling scalable and gradient-free query optimization during inference without modifying either the retriever or the backbone language model.

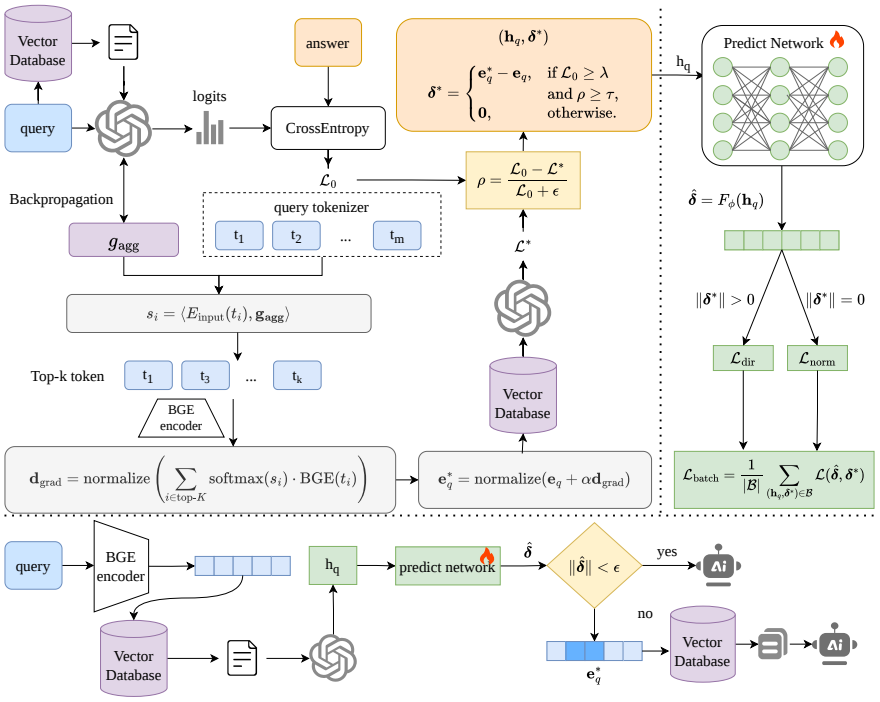


Figure 2: **Overview of the GDER framework.** The framework consists of three stages: (top-left) gradient-guided data collection; (top-right) offset network training; and (bottom) efficient inference.

3.2 Stage I: Gradient-Guided Data Collection

The objective of Stage I is to construct a supervision dataset $\mathcal{S} = \{(\mathbf{h}_q, \boldsymbol{\delta}^*)\}$, which maps contextual query representations to retrieval-space correction vectors that improve downstream answer generation quality. For each training sample (q, a), the supervision target is derived through a gradient-guided retrieval refinement process that tightly couples LLM generation feedback with retrieval optimization.

Initial Retrieval and Generation Feedback. Given an input query q, we first encode it using the dense retriever $\mathcal{E}$ to obtain the initial retrieval embedding $\mathbf{e}_q = \mathcal{E}(q) \in \mathbb{R}^{d_e}$. The resulting embedding is then used to retrieve the top-k supporting documents from the retrieval corpus through maximum inner product search. The retrieved evidence, together with the original query and task prompt, is subsequently provided to the LLM under teacher forcing to compute the answer generation loss:

$$\mathcal{L}_0 = \text{CrossEntropy}(\text{LLM}(q, \text{docs}_0), a). \tag{1}$$

This generation loss directly reflects the extent to which the retrieved evidence sufficiently supports correct downstream reasoning and answer generation. Meanwhile, we extract the final-layer hidden representation $\mathbf{h}_q \in \mathbb{R}^{d_h}$ at the query position, which captures the contextual semantic state of the query conditioned on the retrieved evidence. This representation serves as the supervisory input to the downstream offset prediction network, enabling it to infer retrieval correction directions directly from the LLM hidden states.

Gradient-Induced Retrieval Direction Estimation. To identify semantic deficiencies in the retrieved evidence, we backpropagate the generation loss $\mathcal{L}_0$ with respect to the LLM input embeddings. The resulting gradients encode which semantic components are most critical for reducing generation error and therefore provide intrinsic guidance for retrieval refinement.

Specifically, we aggregate gradients over all answer token positions into a unified correction signal:

$$\mathbf{g}_{\text{agg}} = \frac{1}{|A|} \sum_{t \in A} \frac{\partial \mathcal{L}_0}{\partial \mathbf{E}_{\text{input}}[t]}, \tag{2}$$

where A denotes the set of answer tokens.

This aggregated gradient captures the overall semantic discrepancy between the retrieved evidence and the information required for correct answer generation. To project this signal into retrieval space, we compute token-level importance scores over the original query tokens based on their alignment with the aggregated gradient:

$$s_i = \langle \mathbf{E}_{\text{input}}(t_i), \mathbf{g}_{\text{agg}} \rangle , \tag{3}$$

where $\mathbf{E}_{\text{input}}(t_i)$ denotes the embedding vector of query token t_i after being mapped into the LLM input embedding space. A larger score indicates that the corresponding token is more strongly associated with the semantic deficiency reflected by the generation gradient. The highest-scoring tokens are therefore selected as the top-K tokens for retrieval refinement. Importantly, GDER does not directly transfer the gradient or embedding representations across spaces. Instead, the selected tokens are re-encoded by the retriever to construct the refinement direction.

We then aggregate the corresponding retriever embeddings using softmax-normalized importance weights to construct a retrieval correction direction:

$$\mathbf{d}_{\text{grad}} = \text{normalize}\left(\sum_{i \in \text{top-}K} \omega_i \cdot \mathcal{E}(t_i) \right), \tag{4}$$

where the normalized weight for token t_i is defined as

$$\omega_i = \frac{\exp(s_i)}{\sum_{j \in \text{top-}K} \exp(s_j)} = \text{softmax}(s_i). \tag{5}$$

This procedure transforms generation-gradient-derived token importance into a retrieval-space optimization signal, without requiring explicit alignment between the LLM and retriever representation spaces.

Supervision Target Construction. Using the estimated correction direction $\mathbf{d}_{\text{grad}}$, we construct a refined query embedding:

$$\mathbf{e}_q^* = \text{normalize}(\mathbf{e}_q + \alpha \mathbf{d}_{\text{grad}}), \tag{6}$$

where α controls the magnitude of the refinement step in retrieval space.

We then perform a second retrieval using the optimized query embedding $\mathbf{e}_q^*$, producing a refined set of supporting documents and a corresponding updated generation loss $\mathcal{L}^*$. To assess whether the refined retrieval provides more informative evidence for downstream reasoning and answer generation, we compute the relative reduction in generation loss:

$$\rho = \frac{\mathcal{L}_0 - \mathcal{L}^*}{\mathcal{L}_0 + \epsilon}, \tag{7}$$

where ϵ is a numerical stability constant.

If the initial generation loss is already sufficiently low (i.e., $\mathcal{L}_0 < \lambda$), the retrieved evidence is considered adequately informative and no retrieval correction is required. Otherwise, the refinement is accepted only when the relative improvement satisfies $\rho \geq \tau$, ensuring that only retrieval corrections producing meaningful generation improvements are retained as supervision signals.

The final supervision target is therefore defined as:

$$\boldsymbol{\delta}^* = \begin{cases} \mathbf{e}_q^* - \mathbf{e}_q, & \text{if } \mathcal{L}_0 \geq \lambda \text{ and } \rho \geq \tau, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{8}$$

This formulation naturally produces both positive correction examples and zero-offset abstention examples, enabling the downstream offset network to jointly learn when retrieval embeddings should be refined and when the original retrieval is already sufficient. By leveraging intrinsic generation gradients as self-supervised optimization signals, Stage I establishes an efficient and scalable foundation for adaptive retrieval refinement without requiring external annotations, supervised query rewriting, or iterative reinforcement learning procedures.

3.3 STAGE II: OFFSET NETWORK TRAINING

We train the offset network F_ϕ using a conditional optimization objective that adapts to different types of supervision signals. Given a predicted offset $\hat{\boldsymbol{\delta}} = F_\phi(\mathbf{h}_q)$ and a supervision target $\boldsymbol{\delta}^*$, the training objective is defined as:

$$\mathcal{L}(\hat{\boldsymbol{\delta}}, \boldsymbol{\delta}^*) = \begin{cases} \mathcal{L}_{\text{dir}}, & \text{if } \|\boldsymbol{\delta}^*\| > 0, \\ \beta \cdot \mathcal{L}_{\text{norm}}, & \text{if } \|\boldsymbol{\delta}^*\| = 0, \end{cases} \tag{9}$$

where $\beta < 1$ controls the contribution of zero-supervision samples.

For positive samples, the objective encourages the predicted offset to align with the ground-truth correction direction, enabling the model to learn accurate semantic refinement directions:

$$\mathcal{L}_{\text{dir}} = 1 - \cos(\hat{\boldsymbol{\delta}}, \boldsymbol{\delta}^*). \tag{10}$$

For zero-supervision samples, the objective is to suppress unnecessary perturbations to the original query representation by penalizing the magnitude of the predicted offset, encouraging it to remain close to the zero vector:

$$\mathcal{L}_{\text{norm}} = \|\hat{\boldsymbol{\delta}}\|^2. \tag{11}$$

Given a training batch $\mathcal{B}$, the final optimization objective is defined as the average loss across all training samples:

$$\mathcal{L}_{\text{batch}} = \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{h}_q, \boldsymbol{\delta}^*) \in \mathcal{B}} \mathcal{L}(\hat{\boldsymbol{\delta}}, \boldsymbol{\delta}^*). \tag{12}$$

3.4 STAGE III: EFFICIENT INFERENCE

At inference time, GDER performs adaptive retrieval refinement via a lightweight gating mechanism, without requiring any gradient computation or parameter updates. Given a test query q, we first encode it into a dense representation $\mathbf{e}_q$ using the BGE encoder, and retrieve an initial document set docs_0 from the index. We then obtain a contextual hidden representation $\mathbf{h}_q$ from the hidden-state extraction model through a single forward pass, which is used by the offset network F_ϕ to predict a correction vector $\hat{\boldsymbol{\delta}} = F_\phi(\mathbf{h}_q)$.

The inference procedure is formulated as a conditional retrieval policy:

$$\hat{a}(q) = \begin{cases} \text{Gen}(q, \text{docs}_0), & \|\hat{\boldsymbol{\delta}}\| < \varepsilon \\ \text{Gen}(q, \text{docs}^*), & \text{otherwise} \end{cases} \tag{13}$$

where docs^* is obtained by re-retrieving with the refined query embedding $\mathbf{e}_q^* = \text{normalize}(\mathbf{e}_q + \alpha \cdot \hat{\boldsymbol{\delta}})$.

This formulation defines an adaptive retrieval mechanism that dynamically determines whether refinement of the initial retrieval is necessary based on the magnitude of the predicted correction. The overall inference requires at most two forward passes of the language model and one forward pass of a lightweight MLP(Melas-Kyriazi, 2021; Ding et al., 2022). Importantly, GDER operates without modifying either the retrieval encoder or the language model, enabling seamless integration as a plug-and-play module in existing retrieval-augmented generation systems.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and Evaluation Metrics. We evaluate GDER on several challenging open-domain QA benchmarks, including the two-hop reasoning dataset **HotpotQA** (Yang et al., 2018), the multi-hop reasoning benchmarks **2WikiMultiHopQA** (Ho et al., 2020) and **MuSiQue** (Trivedi et al., 2022), as well as the binary reasoning benchmark **StrategyQA** (Geva et al., 2021). We report **Accuracy (Acc)** and **Token-level F1 (F1)** on HotpotQA, 2WikiMultiHopQA, and MuSiQue, while using **Accuracy (Acc)** as the primary metric for StrategyQA.

Table 1: Main results on four QA benchmarks. We report Accuracy (Acc, %) and Token-level F1 (F1, %) for multi-hop QA datasets, while only Acc is reported for StrategyQA. **Bold** indicates the best result; underline indicates the second best. † denotes methods that require autoregressive generation of new textual queries.

Method	HotpotQA		2WikiMultiHopQA		MuSiQue		StrategyQA
	Acc	F1	Acc	F1	Acc	F1	Acc
Qwen2.5-7B-Instruct							
Naive RAG	41.98	48.93	33.81	36.96	12.54	19.12	69.71
Prompt Rewriting†	38.28	32.40	31.98	28.70	13.66	17.78	57.86
LeReT†	42.11	50.50	22.32	18.54	10.18	13.64	66.98
DeepRetrieval†	35.45	40.59	34.67	38.56	12.65	21.39	69.95
Dense PRF	41.05	40.11	31.37	5.62	11.75	3.28	65.20
HyDE	25.14	28.26	30.20	32.24	8.35	2.46	62.79
GDER (Ours)	**44.12**	**52.14**	**36.90**	**40.04**	**15.54**	**22.43**	**70.71**
Qwen2.5-14B-Instruct							
Naive RAG	47.78	44.52	40.69	33.44	17.80	21.29	67.64
Prompt Rewriting†	35.78	34.56	41.11	30.45	14.56	16.98	56.64
LeReT†	48.02	44.98	40.88	34.09	**19.24**	**24.03**	68.13
DeepRetrieval†	47.11	42.35	41.76	34.38	15.46	19.22	65.57
Dense PRF	42.11	44.34	35.4	9.84	12.15	4.22	63.1
HyDE	30.58	33.19	36.84	37.19	10.56	3.98	59.12
GDER (Ours)	**49.80**	**46.68**	**42.87**	**35.58**	18.76	22.12	**68.82**
LLaMA-3-8B-Instruct							
Naive RAG	41.97	48.93	38.98	22.12	12.41	11.83	64.66
Prompt Rewriting†	30.99	13.54	21.84	9.41	14.56	8.73	54.46
LeReT†	40.15	50.11	31.42	18.33	13.46	12.11	64.43
DeepRetrieval†	42.30	50.04	39.11	23.14	11.21	10.52	60.68
Dense PRF	40.32	38.54	33.08	8.64	10.59	3.09	63.8
HyDE	24.17	27.49	32.37	19.59	7.67	2.89	60.29
GDER (Ours)	**43.12**	**51.14**	**41.19**	**25.21**	**15.33**	**13.15**	**65.71**

Retrieval Setup. Since GDER performs query optimization directly in the dense retrieval embedding space, all experiments are conducted under a purely dense retrieval setting. We use the BGE-base-en-v1.5 encoder (Xiao et al., 2024) to encode the document corpus offline and construct retrieval indexes using FAISS (Douze et al., 2025) for efficient maximum inner product search. For each query, we retrieve the top-k documents with $k = 5$.

Generation Models. To evaluate the generalizability of GDER across different backbone LLMs, we conduct experiments using three representative instruction-tuned generation models: **Qwen2.5-7B-Instruct** (Team, 2024; Yang et al., 2024), **Qwen2.5-14B-Instruct** (Team, 2024; Yang et al., 2024), and **LLaMA-3-8B-Instruct** (Grattafiori et al., 2024). In addition, all experiments consistently use Qwen2.5-7B-Instruct as the embedding extraction model for hidden state representation. All models are loaded in bfloat16 precision and remain fully frozen throughout both training and inference.

Baselines. We compare GDER with several representative retrieval and query optimization baselines, including **Naive RAG**, which performs standard retrieval using the original query embedding without query optimization. For prompt-based methods, **Prompt-based Rewriting** relies on carefully designed prompts to rewrite the original query using both the input question and the initially retrieved documents, followed by a second retrieval step for answer generation. We further include reinforcement learning-based methods such as **LeReT** (Hsu et al., 2025) and **DeepRetrieval** (Jiang et al., 2025), which formulate query optimization as a policy learning problem and improve retrieval through reward-driven or preference-based optimization. To provide closer comparisons with embedding-space retrieval refinement, we additionally include **HyDE** and **Dense PRF**. **HyDE** improves retrieval by generating a hypothetical document and using its embedding as the refined query representation. **Dense PRF** refines the query representation by incorporating information from the initially retrieved documents. For a fair comparison, we set its interpolation coefficient α to 0.5, consistent with the correction coefficient used in GDER.

4.2 Main Results

Main Results Analysis. As shown in Table 1, GDER consistently achieves strong performance across all datasets and backbone models, demonstrating the effectiveness of gradient-driven query embedding refinement for retrieval-augmented generation. Compared with **Naive RAG**, GDER achieves consistent improvements across different backbones, showing that directly optimizing the query representation can effectively improve the quality of retrieved evidence. Compared with text-based query rewriting methods such as **Prompt Rewriting**, **LeReT**, and **DeepRetrieval**, GDER operates directly in the retrieval embedding space and avoids autoregressive query generation. More importantly, unlike methods that rely on manually designed prompts or reward-driven optimization, GDER derives its supervision signal from generation gradients under answer supervision, providing a more direct connection between query refinement and the downstream answer generation objective. Compared with embedding-space refinement methods such as **Dense PRF** and **HyDE**, GDER further exploits generation-gradient information to construct query-specific refinement directions, leading to consistently stronger performance across the evaluated benchmarks.

The results also show that GDER maintains consistent improvements across both Qwen2.5 and LLaMA-3 backbones, demonstrating that the learned refinement mechanism is not tightly coupled to a specific generation model or training dataset. Although the refinement supervision is constructed from the HotpotQA training set, the offset predictor learns a general mapping from the contextual representations of queries and retrieved documents to retrieval refinement directions, rather than dataset-specific retrieval heuristics. Since the supervision is derived from generation loss gradients, it captures the semantic mismatch between the query and retrieved context in terms of the downstream generation objective, enabling the learned refinement capability to transfer across different QA benchmarks. The retrieval-oriented improvements of GDER are further evaluated using retrieval metrics, with detailed results provided in the appendix D.1.

5 Analysis

5.1 Ablation Study: Gradient-Guided vs. Alternative Refinement Strategies

To investigate the contributions of gradient-derived supervision and the cross-space refinement mechanism, we evaluate three variants: **Random Perturbation**, which replaces the learned offset with a random direction of the same magnitude; **GDER w/o LLM Gradient**, which estimates token importance using retrieval-space relevance without generation gradients; and **GDER w/ Learned Space Alignment**, which uses a lightweight linear mapping between the LLM and BGE representation spaces. These variants allow us to examine the roles of the gradient-derived supervision and the proposed cross-space refinement mechanism.

Table 2: Ablation study comparing random perturbation, retriever-space weighting without LLM gradients, learned space alignment, and the full GDER model using Qwen2.5-7B-Instruct.

Method	**HotpotQA**		**2WikiMultiHopQA**		**MuSiQue**	
	Acc	F1	Acc	F1	Acc	F1
No Refinement	41.98	48.93	33.81	36.96	12.54	19.12
Random Perturbation	30.12	35.01	10.05	13.10	13.11	19.48
GDER w/o LLM Gradient	39.55	44.15	28.34	31.76	10.51	15.46
GDER w/ Learned Space Alignment	41.20	17.98	34.11	36.66	11.38	16.49
GDER (Full)	**44.12**	**52.14**	**36.90**	**40.04**	**15.54**	**22.43**

As shown in Table 2, GDER consistently outperforms all ablation variants across the evaluated benchmarks. Random perturbation performs substantially worse than both GDER and No Refinement, showing that arbitrary embedding perturbations are insufficient for effective refinement. Removing the LLM gradient also causes clear performance degradation, while the learned space alignment variant fails to match the full model. These results indicate that the gains of GDER come from the query-specific refinement directions derived from answer-supervised generation gradients, rather than from arbitrary perturbation or explicit global alignment between the two representation spaces.

5.2 Inference Efficiency Compared to RL-based Methods

We compare the average inference time of GDER and existing query optimization (QO) baselines across four datasets in Table 3. Prompt-based and RL-based methods incur substantially higher inference costs due to autoregressive query rewriting and multi-stage inference. In contrast, GDER performs query optimization directly in the embedding space without text generation, reducing the optimization process to lightweight forward computation. These results demonstrate that GDER achieves significantly lower inference cost while maintaining competitive performance, highlighting the efficiency advantage of embedding-level query refinement over autoregressive query optimization.

Table 3: Inference efficiency comparison of GDER and query optimization (QO) baselines using Qwen2.5-7B-Instruct as the backbone model. We compare the computational cost of the QO stage and the overall end-to-end inference time across four datasets.

Method	QO Type	QO Time (ms)	Generation Stages	Infer Time (ms)
Prompt Rewriting	Autoregressive	1862.2	Rewrite + Answer	3033.0
LeReT	Autoregressive	590.4	Rewrite + Answer	1761.2
DeepRetrieval	Autoregressive	680.7	Rewrite + Answer	1851.5
GDER (Ours)	Embedding Offset	**233.1**	Forward + Answer	**1403.9**

5.3 Efficiency Comparison with Naive RAG

Compared with **Naive RAG**, GDER introduces additional inference costs from three components: (1) contextual representation extraction, (2) lightweight query offset prediction, and (3) adaptive retrieval refinement. The first two components are applied to all queries, while adaptive retrieval refinement is activated only when further optimization is required. GDER supports both unified and decoupled deployment settings. In the unified setting, the contextual representation h_q is directly obtained from the deployed generation model without an additional model copy, resulting in only 0.29 GB additional memory consumption. The decoupled setting deploys a separate hidden state extractor, providing greater flexibility across different generation models at the cost of additional memory. Under the unified setting, the detailed overhead is shown in Table 4. The overall average latency overhead is 233.1 ms. These results show that GDER introduces moderate additional computation and limited memory overhead compared with Naive RAG, while providing the benefits of query-specific retrieval refinement. Further experimental results and analyses are provided in the Appendix D.

Table 4: Additional inference overhead of GDER compared with Naive RAG under the unified model setting.

Module	Activation Rate	Latency (ms)	Additional Memory (GB)
Contextual Representation Extraction	100%	115.5	$\sim$0
Query Offset Predictor	100%	45.1	0.29
Adaptive Retrieval Refinement	25.5%	284.4	0
Overall Additional Cost	–	**233.1**	**0.29**

6 Conclusion

We propose GDER, a framework that leverages LLM generation gradients derived from answer supervision to optimize queries in dense retrieval space. We show that these gradients naturally capture the direction of missing semantic information in retrieved context, providing a principled and task-relevant signal for retrieval optimization. Based on this insight, GDER reformulates query optimization as embedding-space optimization, avoiding costly autoregressive query rewriting and policy optimization. The framework is fully compatible with standard RAG pipelines and enables efficient gradient-free inference via a lightweight predictor. Overall, this work demonstrates that intrinsic generation gradients provide a simple and effective mechanism for aligning retrieval with downstream reasoning in retrieval-augmented generation systems.

AI USE STATEMENT

We used generative AI tools only for language editing and polishing of the manuscript. All research ideas, experimental results, technical claims, and final text were reviewed and verified by the authors, who take responsibility for the final content of this work.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Siyuan Chen, Hang Ding, Kangxiaoyu, and Jiechao Gao. GRO-RAG: Gradient-aware re-rank optimization for multi-source retrieval-augmented generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=5zdubHFutd.

Qinyuan Cheng, Xiaonan Li, Shimin Li, Qin Zhu, Zhangyue Yin, Yunfan Shao, Linyang Li, Tianxiang Sun, Hang Yan, and Xipeng Qiu. Unified active retrieval for retrieval augmented generation. *arXiv preprint arXiv:2406.12534*, 2024.

Xiaohan Ding, Chunlong Xia, Xiangyu Zhang, Xiaojie Chu, Jungong Han, and Guiguang Ding. Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition, 2022. URL https://arxiv.org/abs/2105.01883.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2025. URL https://arxiv.org/abs/2401.08281.

Jingsheng Gao, Linxu Li, Ke Ji, Weiyuan Li, Yixin Lian, yuzhuo fu, and Bin Dai. SmartRAG: Jointly learn RAG-related tasks from the environment feedback. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OCd3cffulp.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1762–1777, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.99. URL https://aclanthology.org/2023.acl-long.99/.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021. doi: 10.1162/tacl_a_00370. URL https://aclanthology.org/2021.tacl-1.21/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. Deeprag: Thinking to retrieve step by step for large language models. *arXiv preprint arXiv:2502.01142*, 2025. URL https://arxiv.org/abs/2411.11504.

Hanghui Guo, Jia Zhu, Shimin Di, Weijie Shi, Zhangze Chen, and Jiajie Xu. DioR: Adaptive cognitive detection and contextual retrieval optimization for dynamic retrieval-augmented generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2953–2975, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.148. URL https://aclanthology.org/2025.acl-long.148/.

Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu, and Shi-Min Hu. Beyond self-attention: External attention using two linear layers for visual tasks, 2021. URL https://arxiv.org/abs/2105.02358.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Sheryl Hsu, Omar Khattab, Chelsea Finn, and Archit Sharma. Grounding by trying: LLMs with reinforcement learning-enhanced retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=BPAZ6yW3K7.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24(1), January 2023. ISSN 1532-4435.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.389. URL https://aclanthology.org/2024.naacl-long.389/.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=u9JXu4L17I.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting for retrieval-augmented large language models, 2023a. URL https://arxiv.org/abs/2305.14283.

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

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1906–1919, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.173. URL https://aclanthology.org/2020.acl-main.173/.

Luke Melas-Kyriazi. Do you even need attention? a stack of feed-forward layers does surprisingly well on imagenet, 2021. URL https://arxiv.org/abs/2105.02723.

Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. Large language model based long-tail query rewriting in taobao search, 2024. URL https://arxiv.org/abs/2311.03758.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.409. URL https://aclanthology.org/2024.acl-long.409/.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12991–13013, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.702. URL https://aclanthology.org/2024.acl-long.702/.

Jiashuo Sun, Xianrui Zhong, Sizhe Zhou, and Jiawei Han. DynamicRAG: Leveraging outputs of large language model as feedback for dynamic reranking in retrieval-augmented generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=NuCtKoflsV.

Anuja Tayal and Aman Tyagi. Dynamic contexts for generating suggestion questions in rag based conversational systems. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, pp. 1338–1341. ACM, May 2024. doi: 10.1145/3589335.3651905. URL http://dx.doi.org/10.1145/3589335.3651905.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, 2021. URL https://arxiv.org/abs/2105.01601.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Ruobing Wang, Daren Zha, Shi Yu, Qingfei Zhao, Yuxuan Chen, Yixuan Wang, Shuo Wang, Yukun Yan, Zhenghao Liu, Xu Han, Zhiyuan Liu, and Maosong Sun. Retriever-and-memory: Towards adaptive note-enhanced retrieval-augmented generation, 2024. URL https://arxiv.org/abs/2410.08821.

Ruobing Wang, Qingfei Zhao, Yukun Yan, Daren Zha, Yuxuan Chen, Shi Yu, Zhenghao Liu, Yixuan Wang, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. DeepNote: Note-centric deep retrieval-augmented generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 19688–19715, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1073. URL https://aclanthology.org/2025.findings-emnlp.1073/.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings, 2024. URL https://arxiv.org/abs/2309.07597.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng

Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Liu Weichuan, Lei Hou, and Juanzi Li. SeaKR: Self-aware knowledge retrieval for adaptive retrieval augmented generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27022–27043, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1312. URL https://aclanthology.org/2025.acl-long.1312/.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

Jia Zhu, Hanghui Guo, Weijie Shi, Zhangze Chen, and Pasquale De Meo. Radio: Real-time hallucination detection with contextual index optimized query formulation for dynamic retrieval augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26129–26137, 2025.

A Limitations

While GDER demonstrates consistent improvements across multiple benchmarks, it has several limitations. First, since query optimization is performed directly in the dense embedding space, GDER is inherently restricted to dense retrieval systems and cannot be straightforwardly extended to sparse retrieval methods such as BM25, which may limit its applicability in retrieval pipelines that rely on lexical matching. Second, GDER requires a large language model to perform a forward pass at inference time to extract the query hidden representation $\mathbf{h}_q$ as input to the offset network F_ϕ, which introduces non-trivial GPU memory overhead and may pose deployment challenges in resource-constrained environments. Third, GDER is primarily evaluated on open-domain question answering tasks, and its effectiveness on other retrieval-augmented generation settings, such as multi-turn dialogue or long-form generation, remains underexplored.

B Theoretical Analysis

B.1 Generation Gradients as a Directional Signal for Missing Semantics

Let $\mathcal{L}(\mathbf{H})$ denote the generation loss with respect to the LLM input representations $\mathbf{H}$. The aggregated generation gradient is defined as

$$\mathbf{g}_{\text{agg}} = \nabla_{\mathbf{H}} \mathcal{L}. \tag{14}$$

For a small perturbation $\Delta\mathbf{H}$, the first-order Taylor expansion gives

$$\mathcal{L}(\mathbf{H} + \Delta\mathbf{H}) \approx \mathcal{L}(\mathbf{H}) + \langle \mathbf{g}_{\text{agg}}, \Delta\mathbf{H} \rangle . \tag{15}$$

Therefore, $\mathbf{g}_{\text{agg}}$ characterizes the direction in the input representation space that is most sensitive to the generation objective, while $-\mathbf{g}_{\text{agg}}$ corresponds to the local loss-reducing direction. In GDER, however, the gradient is not used to directly update the LLM input embeddings. Instead, we use it as a directional signal to identify the token representations most relevant to the generation objective.

Specifically, for each query token t_i, we compute its importance score as

$$s_i = \langle \mathbf{E}_{\text{input}}(t_i), \mathbf{g}_{\text{agg}} \rangle , \tag{16}$$

and select the top-K tokens with the largest scores. This score measures the alignment between a token representation and the generation-gradient direction, identifying tokens that are most strongly associated with the semantic deficiency reflected by the generation loss. The generation gradient therefore serves as a directional proxy for the missing semantic information needed by the downstream generation objective.

Importantly, the selected tokens remain in the LLM representation space only for token identification. GDER does not directly transfer $\mathbf{g}_{\text{agg}}$ or $\mathbf{E}_{\text{input}}(t_i)$ to the retrieval space. Instead, the selected tokens are re-encoded using the retriever encoder E_{Ret} to obtain retrieval-space representations, which are then aggregated according to their gradient-derived importance scores to construct the query refinement direction:

$$\boldsymbol{\delta}^* = \sum_{t_i \in \mathcal{T}_K} \alpha_i E_{\text{Ret}}(t_i), \qquad \alpha_i = \frac{\exp(s_i)}{\sum_{t_j \in \mathcal{T}_K} \exp(s_j)}. \tag{17}$$

Thus, GDER uses generation gradients to identify the most informative semantic components and subsequently expresses these components in the retriever space for query refinement, without requiring explicit alignment between the LLM and retriever representation spaces.

C Implementation Details

C.1 Stage I: Gradient-Guided Data Collection Details

Hyperparameter Search via Pilot Study. Prior to large-scale data collection, we conduct a pilot study on 500 randomly sampled training examples from HotpotQA (seed = 132) to determine the optimal hyperparameter configuration for Stage I. The search space covers four hyperparameters: the number of top-K query tokens ($K \in \{2, 3, 4, 6\}$), the correction magnitude $\alpha \in \{0.1, 0.3, 0.5, 1.0\}$, the loss threshold $\lambda \in \{0.5, 1.0, 2.0, 3.0, 5.0\}$, and the minimum gain ratio $\tau \in \{0.01, 0.02, 0.05, 0.1\}$, yielding $4 \times 4 \times 5 \times 4 = 320$ candidate configurations. To avoid redundant computation, gradient directions are pre-cached per K and reused across (λ, τ) settings. Each configuration is evaluated by a composite score:

$$\text{Score} = 0.6\,\overline{\rho}_{\text{pos}} + 0.4\,s_{\text{ratio}}, \quad s_{\text{ratio}} = \begin{cases} 1.0 & \text{if } 0.25 \le r_{\text{pos}} \le 0.40, \\ 0.7 & \text{if } 0.20 \le r_{\text{pos}} \le 0.45, \\ 0.3 & \text{if } 0.15 \le r_{\text{pos}} \le 0.50, \\ 0.0 & \text{otherwise}, \end{cases} \tag{18}$$

where $\overline{\rho}_{\text{pos}}$ is the average gain ratio among positive samples and r_{pos} is the positive sample ratio. Table 5 reports the top-5 configurations.

Table 5: Top-5 hyperparameter configurations from the pilot study on 500 HotpotQA samples. r_{pos} denotes the positive sample ratio; $\overline{\rho}_{\text{pos}}$ denotes the average gain ratio of positive samples.

Rank	K	α	λ	τ	r_{pos} (%)	n_{pos}	$\overline{\rho}_{\text{pos}}$	Score
1	3	1.00	0.50	0.01	25.0%	125	0.3776	0.6266
2	2	1.00	0.50	0.02	25.0%	125	0.3585	0.6151
3	2	0.50	0.50	0.02	25.4%	127	0.3563	0.6138
4	2	1.00	0.50	0.01	25.4%	127	0.3531	0.6119
5	2	0.50	0.50	0.01	26.0%	130	0.3484	0.6091

Data Collection Algorithm. Using the optimal hyperparameters identified above, we run large-scale data collection on the full training split of HotpotQA. The complete procedure is described in Algorithm 1.

Algorithm 1: Gradient-Guided Data Collection (Stage I)

Input: Datasets $\{(q_i, a_i)\}$, Retriever, LLM, LLM Encoder E, BGE encoder $\mathcal{E}$
Input: Hyperparameters: α, K, λ, τ, retrieve top-k
Output: Training dataset $\mathcal{S} = \{(\mathbf{h}_q, \boldsymbol{\delta}^*)\}$

foreach *sample* (q, a) **do**
  $\mathbf{e}_q \leftarrow \mathcal{E}(q)$; $\quad \text{docs}_0 \leftarrow \text{Retrieve}(\mathbf{e}_q,\, k)$;
  $\mathcal{L}_0,\, \mathbf{h}_q,\, \mathbf{g}_{\text{agg}} \leftarrow \text{ForwardBackward}(q,\, \text{docs}_0,\, a,\, \text{LLM})$;
  if $\mathcal{L}_0 < \lambda$ **then**
    Append $(\mathbf{h}_q,\ \boldsymbol{\delta}^* = \mathbf{0})$ to $\mathcal{S}$;
    `// Initial retrieval sufficient`
    continue;
  $s_i \leftarrow \langle \mathbf{E}(t_i),\, \mathbf{g}_{\text{agg}} \rangle$ for each query token t_i;
  Select top-K tokens; compute softmax weights $\{\omega_i\}$;
  $$\mathbf{d}_{\text{grad}} \leftarrow \text{normalize}\left(\sum_{i \in \text{top-}K} \omega_i \cdot \mathcal{E}(t_i) \right);$$
  $\mathbf{e}_q^* \leftarrow \text{normalize}(\mathbf{e}_q + \alpha \cdot \mathbf{d}_{\text{grad}})$;
  $\text{docs}^* \leftarrow \text{Retrieve}(\mathbf{e}_q^*,\, k)$;
  $\mathcal{L}^* \leftarrow \text{Forward}(q,\, \text{docs}^*,\, a,\, \text{LLM})$;
  $\rho \leftarrow (\mathcal{L}_0 - \mathcal{L}^*) \,/\, (\mathcal{L}_0 + \epsilon)$;
  if $\rho \geq \tau$ **then**
    Append $\left(\mathbf{h}_q,\ \boldsymbol{\delta}^* = \mathbf{e}_q^* - \mathbf{e}_q\right)$ to $\mathcal{S}$;
    `// Positive sample`
  else
    Append $(\mathbf{h}_q,\ \boldsymbol{\delta}^* = \mathbf{0})$ to $\mathcal{S}$;
    `// Zero sample`
return $\mathcal{S}$;

Data Collection Statistics. For large-scale training data construction on HotpotQA, we use Qwen2.5-7B-Instruct as the teacher LLM and BGE-base-en-v1.5 as the dense retriever, with retrieve top-k fixed to 5 throughout all experiments. Based on the pilot hyperparameter search, the final data collection configuration is set to $K = 3$, $\alpha = 1.0$, $\lambda = 0.5$, and $\tau = 0.01$. Under this configuration, we collect a total of 10,000 training samples, among which 27.0% are positive samples requiring non-zero query correction vectors ($\boldsymbol{\delta}^* \neq \mathbf{0}$), ensuring sufficient supervision diversity for subsequent predictor training.

Training Data Construction Overhead The only LLM backpropagation in GDER is performed once during the offline target construction stage of Stage I. For each training sample, the process involves initial retrieval, LLM forward and backward computation for generation-gradient extraction, retrieval refinement, and generation loss computation with the refined results. The complete target construction process takes 1.51 s per sample, with LLM backward computation being the main computational cost. For the full training set, this offline process requires 4.19 hours and reaches a peak GPU memory consumption of 26.55 GB. Once the refinement targets are constructed, the query offset predictor is trained using standard supervised learning without further LLM backpropagation. Thus, compared with retrieval refinement methods that do not require LLM gradients, GDER mainly introduces a one-time offline preprocessing cost, while no gradient computation is required during predictor training or inference.

Model Architecture and Optimization Setup. The offset prediction network F_ϕ is implemented as a lightweight multi-layer perceptron (MLP) that maps the query hidden representation $\mathbf{h}_q \in \mathbb{R}^{3584}$ to the retrieval correction vector $\hat{\boldsymbol{\delta}} \in \mathbb{R}^{768}$. To improve generalization and reduce overfitting, we adopt a compact architecture with hidden dimension 512, 2 fully connected layers, and dropout rate 0.3, which provides an effective balance between expressive capacity and optimization stability.

Training is performed using AdamW with learning rate 1×10^{-3}, weight decay 1×10^{-3}, batch size 128, and cosine annealing scheduling over 30 epochs. We reserve 10% of the collected training data as a validation set and apply gradient clipping with maximum norm 1.0 to ensure stable optimization throughout training.

C.2 Stage III: Efficient Inference Details

Inference Architecture and Deployment Configuration. During deployment, GDER operates as a plug-and-play retrieval refinement module composed of four components: (1) the BGE-base-en-v1.5 dense retriever, (2) the hidden-state extraction LLM, (3) the trained offset predictor F_ϕ, and (4) the final answer generation LLM. By default, Qwen2.5-7B-Instruct is used for hidden-state extraction, while the answer generation model can be flexibly selected. The framework also supports decoupled extractor-generator configurations for cross-model generalization experiments.

Adaptive Gating and Query Refinement. To minimize unnecessary retrieval overhead, we apply a norm-based gating mechanism using threshold $\varepsilon = 0.2$. If $\|\hat{\boldsymbol{\delta}}\| < \varepsilon$, the system directly uses the original retrieval results for generation. Otherwise, a refined query embedding is constructed, which is then used to re-retrieve a new set of documents for downstream answer generation.

All experiments are conducted on a server with 2 NVIDIA A100 (40GB) GPUs for both training and inference.

Test Dataset and Retrieval Index Statistics. We summarize dataset splits and retrieval index configurations in Table 6. HotpotQA, 2WikiMultiHopQA, and MuSiQue use their validation splits, while StrategyQA uses the training split following prior work. All datasets are indexed independently using a dense semantic retrieval index built with BGE-base-en-v1.5 and FAISS inner-product search.

Table 6: Dataset splits, dataset sizes, semantic retrieval index type, and index sizes across all evaluation benchmarks.

	HotpotQA	2WikiMultiHopQA	MuSiQue	StrategyQA
Split	Valid	dev	Valid	Train
Dataset Size	7,405	2417	11,067	2,290
Semantic Index	hotpotqa.index	2wikimultihopqa.index	musique.index	wiki.index
Index Size	139416	5233329	430139	21015324

D Additional Experiments

We conduct additional experiments to further analyze the behavior of GDER.

D.1 Retrieval Performance

To further evaluate the impact of GDER on retrieval quality, we report retrieval-level results on three multi-hop QA benchmarks. We use Recall@5, nDCG@5, and MRR@5 to compare GDER with Naive RAG and representative retrieval refinement baselines under the same retrieval settings. The results are shown in Table 7.

GDER achieves the best retrieval performance across all three benchmarks and consistently outperforms Naive RAG in Recall@5, nDCG@5, and MRR@5. It also improves over the representative retrieval refinement baselines, demonstrating that GDER provides a positive effect on retrieval quality. These results are consistent with the improvements observed in downstream QA performance, supporting the effectiveness of generation-gradient-driven query refinement for improving retrieved evidence.

D.2 Analysis of Inference-Time Threshold ε.

Table 8 shows that increasing the threshold ε consistently reduces both the query refinement rate (Q-Rate) and the document change rate (D-Rate), indicating fewer queries are being optimized and

Table 7: Retrieval performance of GDER and baseline methods on multi-hop QA benchmarks.

Dataset	Method	Recall@5	nDCG@5	MRR@5
HotpotQA	Naive RAG	70.00	70.82	87.26
	Prompt Rewriting†	64.08	64.62	80.41
	Dense PRF	67.43	64.35	78.18
	HyDE	56.40	55.48	69.16
	GDER	**79.77**	**80.46**	**91.77**
2Wiki	Naive RAG	73.72	76.47	94.54
	Prompt Rewriting†	64.76	65.19	80.60
	Dense PRF	69.76	66.51	79.80
	HyDE	50.19	48.85	62.25
	GDER	**77.66**	**79.36**	**98.45**
MuSiQue	Naive RAG	49.97	52.90	76.88
	Prompt Rewriting†	47.62	50.29	71.85
	Dense PRF	44.26	44.05	63.55
	HyDE	44.06	45.60	65.96
	GDER	**54.77**	**56.66**	**80.53**

fewer retrieval results are modified. In terms of performance, accuracy first improves as ε increases from 0.10 to 0.20, reaching the best results at $\varepsilon = 0.20$, and then degrades when ε further increases to 0.25. This suggests that a moderate refinement frequency achieves the best balance between retrieval efficiency and optimization effectiveness, while too aggressive or too conservative triggering can both harm performance.

Table 8: Sensitivity analysis of inference-time threshold ε. Acc measures answer accuracy. Q-Rate denotes the fraction of queries that trigger refinement, and D-Rate measures the fraction of retrieved documents that change after query optimization. Results show stable improvements with moderate thresholds, with optimal performance at $\varepsilon = 0.2$.

	HotpotQA		2WikiMultiHopQA		MuSiQue		StrategyQA	
ε	Acc	Q/D-Rate	Acc	Q/D-Rate	Acc	Q/D-Rate	Acc	Q/D-Rate
0.10	43.86	48.56 / 52.19	36.84	59.64 / 58.33	14.89	62.18 / 52.45	69.87	32.54 / 35.84
0.15	43.52	27.50 /26.80	**37.20**	36.80 / 31.10	15.45	46.44 / 41.86	70.02	15.42 / 18.80
0.20	**44.12**	16.04 / 14.35	36.90	25.53 / 19.71	**15.54**	35.13 / 31.78	**70.71**	5.93 / 5.78
0.25	42.12	11.40 /12.15	35.48	20.36 / 12.31	15.06	24.46 / 20.56	70.21	2.17 / 2.09

D.3 Analysis of Optimized Queries.

To better understand the effectiveness of query refinement, we analyze query subsets based on whether GDER activates optimization. Since GDER defaults to Naive RAG when no refinement is triggered, performance differences only arise on queries where refinement is applied (i.e., need_refine=True).

Table 9: Performance comparison on HotpotQA queries grouped by whether GDER activates query refinement. When optimization is skipped, GDER is equivalent to Naive RAG.

Subset	Samples	Naive Acc	GDER Acc	Naive F1	GDER F1
Triggered Optimization	1186	23.47	36.83	48.72	68.76
Skipped Optimization	6219	45.51	45.51	48.97	48.97

Table 9 shows that the queries selected for refinement are significantly more difficult for standard retrieval, with Naive RAG achieving only 23.47 Acc on this subset. After applying GDER, performance improves substantially to 36.83 Acc, while F1 increases from 48.72 to 68.76, demonstrating that gradient-driven query refinement effectively addresses challenging retrieval cases where initial retrieval quality is poor. In contrast, for queries where refinement is not activated, GDER remains

Algorithm 2: Iterative GDER Inference Procedure

Input: Query q, dense retriever $\mathcal{E}$, retrieval index, hidden-state extractor LLM, offset predictor F_ϕ, threshold ε, refinement scale α, maximum iterations $T_{\max}$

Output: Predicted answer $\hat{a}$, total refinement iterations T

$\mathbf{e}_q^{(0)} \leftarrow \mathcal{E}(q)$;
$\text{docs}^{(0)} \leftarrow \text{Retrieve}(\mathbf{e}_q^{(0)}, k)$;
$T \leftarrow 0$;
while $T < T_{\max}$ **do**
  $\mathbf{h}_q^{(T)} \leftarrow \text{ExtractHidden}(q, \text{docs}^{(T)})$;
  $\hat{\boldsymbol{\delta}}^{(T)} \leftarrow F_\phi(\mathbf{h}_q^{(T)})$;
  if $\|\hat{\boldsymbol{\delta}}^{(T)}\| < \varepsilon$ **then**
    break;
  $\mathbf{e}_q^{(T+1)} \leftarrow \text{normalize}(\mathbf{e}_q^{(T)} + \alpha \cdot \hat{\boldsymbol{\delta}}^{(T)})$;
  $\text{docs}^{(T+1)} \leftarrow \text{Retrieve}(\mathbf{e}_q^{(T+1)}, k)$;
  $T \leftarrow T + 1$;
$\hat{a} \leftarrow \text{Generate}(q, \text{docs}^{(T)})$;
return $\hat{a}$, T;

identical to Naive RAG, preserving baseline performance without introducing unnecessary modifications. These findings indicate that the gating mechanism successfully identifies hard queries that benefit most from optimization, enabling GDER to focus refinement on difficult cases while maintaining computational efficiency and stable performance on easier examples.

D.4 Gate Activation Rate and Retrieval Dynamics

To further analyze the behavior of the gating mechanism, Figure 3 presents the gate activation rate and document change rate across four benchmarks. The results in the figure show a clear correlation between task complexity and gate activation behavior. Specifically, 2WikiMultiHopQA and MuSiQue, which require more complex multi-hop reasoning, exhibit higher activation rates, while HotpotQA, a relatively simpler two-hop dataset, shows a moderate activation rate. In contrast, StrategyQA, which is a binary question-answering task with low reasoning complexity, yields the lowest activation rate. These results suggest that the gating mechanism can effectively adapt to the intrinsic reasoning difficulty of different tasks.

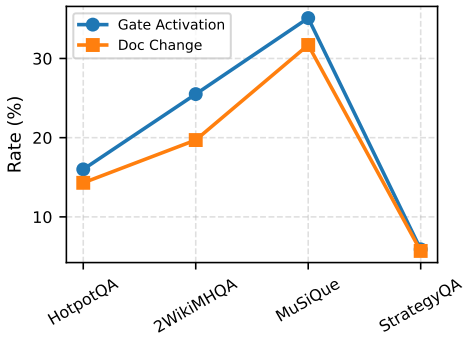


Figure 3: Gate activation rate and document change rate across four benchmarks.

We further observe a consistent gap between gate activation rate and document change rate across all datasets. In particular, not every activated gate leads to substantial changes in the retrieved documents, indicating that the model often performs conservative refinement even when correction is triggered. This behavior implies that the gating mechanism captures a graded notion of uncertainty rather than a strict binary decision, enabling the model to balance retrieval stability with adaptive refinement when needed.

D.5 Iterative Retrieval Refinement.

Although the default GDER inference process performs only a single refinement step, the framework naturally supports iterative retrieval optimization without requiring any additional training. After each refinement round, the newly retrieved documents can be re-evaluated by the hidden-state extraction

model to produce an updated query representation, which is then passed through the offset predictor F_ϕ to estimate a new correction vector. This process can be repeated until the predicted offset norm falls below the gating threshold ε or a predefined maximum iteration budget $T_{\max}$ is reached. The full iterative inference procedure is summarized in Algorithm 2.

Table 10: Comparison of single-step and iterative GDER across four datasets using Qwen2.5-7B-Instruct. Avg. Iter denotes the average number of refinement iterations per query across four datasets, and Avg. Time reports the average end-to-end inference time per query across four datasets (ms).

Method	Avg. Iter	Avg. Time	HotpotQA		2Wiki		MuSiQue		StrategyQA
			Acc	F1	Acc	F1	Acc	F1	Acc
GDER ($T_{\max} = 1$)	0.2066	1403.9	44.12	52.14	36.90	40.04	15.54	22.43	70.71
GDER ($T_{\max} = 3$)	0.8302	1719.9	44.09	51.08	36.93	40.13	15.76	22.72	70.88

The results in Table 10 indicate that GDER exhibits a stable and robust behavior under iterative retrieval refinement. Although the framework allows multiple refinement steps in principle, most queries are effectively resolved within the first iteration due to the strong corrective signal provided by the gradient-driven offset predictor. Subsequent iterations tend to contribute diminishing improvements, as the query representation gradually converges and the estimated offset magnitude decreases below the gating threshold. At the same time, additional refinement steps introduce non-negligible computational overhead without consistently yielding proportional gains, suggesting that the model naturally reaches a near-optimal retrieval state early in the inference process. Overall, these observations highlight that GDER primarily functions as an efficient single-step refinement mechanism, while its iterative extension serves as a lightweight and optional enhancement rather than a necessary component for performance improvement.

E PROMPTS

Table 11: Prompt template for general answer generation and data collection.

General Answer Generation And Data Collection Prompt

Answer the following question based on the provided references. Only give me the answer and do not output any other words.

References:
{References}

Question: {Query}
Answer:

Table 12: Prompt template for StrategyQA (Binary decision).

StrategyQA Answer Generation Prompt

Answer the following question based on the provided references. Only give me 'true' or 'false' as your answer and do not output any other words.

References:
[1] {Reference 1}
[2] {Reference 2}
...

Question: {Query}
Answer:

Table 13: Prompt template for query rewriting baseline.

Query Rewriting Prompt (Baseline Method)

Based on the following reference documents and the original question, generate a better search query that can retrieve more relevant information. Only output the new query and nothing else.

Reference documents:
{Context}

Original question: {Question}
New search query: