

Evaluating the Utility of Grounding Documents with Reference-Free LLM-based Metrics

Yilun Hua^{†*}, Giuseppe Castellucci^{‡*}, Peter Schulam^{‡◊}, Heba Elfardy^{‡◊}, Kevin Small^{‡◊}

[†]Department of Computer Science and Cornell Tech, Cornell University

[‡]Amazon

yilunhua@cs.cornell.edu

Abstract

Retrieval Augmented Generation (RAG)’s success depends on the *utility* the LLM derives from the content used for grounding. Quantifying content utility does not have a definitive specification and existing metrics ignore model-specific capabilities and/or rely on costly annotations. In this paper, we propose *Grounding Generation Utility* (GROGU), a model-specific and reference-free metric that defines utility as a function of the downstream LLM’s generation confidence based on entropy. Despite having no annotation requirements, GROGU is largely consistent with ground-truth relevance annotations in distinguishing ground-truth documents while capturing nuances ignored by LLM-agnostic metrics. We apply GROGU to train a query-rewriter for RAG by identifying high-utility preference data for Direct Preference Optimization. Experiments show improvements by up to 18.2 points in Mean Reciprocal Rank and up to 9.4 points in answer accuracy.

1 Introduction

Retrieval Augmented Generation (RAG) is the dominant approach for providing Large Language Models (LLMs) with knowledge not present in their training data. The basic approach to RAG is straightforward: the user question is sent to a *retriever* that returns documents from an index. The documents are then combined with the original input and sent to a downstream LLM (the *generator*) to generate a response. In practice, however, a RAG system typically has many sub-components that must be carefully tuned to achieve strong performance. For example, a *query-rewriter* (Mo et al., 2023) is often needed to reformulate a question in a conversational interaction, which otherwise omits key information (example in Appendix A). The reformulated queries retrieve more useful grounding

documents for the downstream LLM, thus improving the corresponding response quality.

Tuning a RAG system is nontrivial and typically involves making changes to individual RAG components. In practice, such tuning is usually guided by metrics that rely on annotated labels as references. Examples include the rank of gold passages among retrieved documents or comparing the generated answers with gold ones to compute accuracy. Reference annotations have been used to improve components such as the query-rewriter (e.g., Yoon et al., 2025; Lai et al., 2025) or a re-ranker (e.g., Sun et al., 2025), to provide more *useful* grounding documents to the generator. An important limitation of such previous works is the need for annotated labels, which limits their adoption to tune RAG systems and extension to new domains.

These limitations would be resolved if we can measure the utility of retrieved documents without requiring reference labels. One naive option is to use the *relevance* scores assigned by a retriever, i.e., an LLM-agnostic relation between a query and a document. In practice, such relevance scores are not used for model training due to reasons such as their noisiness (i.e., a document with a high relevance score may have less actual relevance than a lower scored one) (Luan et al., 2021; Wang et al., 2023b; Yoran et al., 2024). We elaborate on other practical issues in Section 2. In this paper, we argue that relevance scores are inherently flawed as an estimation of the utility of documents in a RAG setting due to their LLM-agnostic nature. Even an ideal relevance scorer will ignore important nuances in how much utility an LLM can derive from the grounding documents.

We refer to this *utility* of grounding documents as *Grounding Generation Utility* (GROGU). In this paper, we define GROGU as a *reference-free* metric that captures how useful the documents are to a specific LLM. We define GROGU based on the LLM’s generation confidence and propose a specific for-

*Work done as an intern at Amazon.

*Now at Reddit.

◊Now at Microsoft.

mulation that modifies the standard token entropy calculation for more robust performance, without relying on any annotations. Intuitively, a useful grounding context supplies the LLM with the information it needs to answer the question, which sharpens the LLM’s distribution over the answer tokens relative to generating without a grounding context or with a less useful context. GROGU measures the change in generation confidence on a subset of generated tokens that carry the substantive content of the answer (which we call *key tokens*), so the score is not biased by scaffolding phrases or tokens copied from the question (Section 3.2).

We first evaluate GROGU on its own, with no training involved. These experiments confirm that GROGU recovers the distinctions that gold annotations provide (Section 4.1), and they show two insights that an LLM-agnostic relevance scorer cannot express: **relevance is not the same as utility** (Section 4.2), and **the same document can bring different levels of utility** to different models (Section 4.3). This evaluation also yields a training-free application of GROGU: among several candidate grounding contexts for a query, we send the one with the highest GROGU score to the generator (Appendix D.3). We then use GROGU as a training signal, identifying preference data without any manual annotation¹ to train a query-rewriter for RAG with Direct Preference Optimization (DPO) (Rafailov et al., 2023). We test this method with both sparse and dense retrievers, showing consistent improvements in retrieval performance, by up to 18.2 points in Mean Reciprocal Rank (MRR), as well as in the downstream generator performance, with improvements by up to 9.4 percentage points in accuracy. Our code is available at <https://github.com/yilun-hua/GroGU.git>.

2 Motivation and Background

Existing methods for tuning components of a RAG system primarily use annotated gold passages or gold answers as references: these are often costly and need to be re-executed for every domain. Moreover, this approach does not work well in domains where it is difficult to define a “correct” answer or a gold passage (Anantha et al., 2021). One solution is to use a *reference-free* metric, such as retriever relevance scores. However, these scores

have not served as good training signals to improve RAG components (e.g., query-rewriter, reranker) for several reasons. First, retrievers are typically optimized for recall over precision in document relevance ranking. Luan et al. (2021) and Yu et al. (2024) also point out challenges in using embeddings to accurately match queries to documents. As a result, the retrievers are noisy: given any two documents, the one ranked higher by a retriever may actually have lower relevance than the other (Luan et al., 2021; Wang et al., 2023b; Yoran et al., 2024). A second issue stems from the retriever’s role in a RAG pipeline: (1) retriever’s relevance scores are produced before the document reranking step, thus not reflecting the quality of the reranker or providing reward signals to improve the reranker and (2) for query-rewriter, relevance scores are produced based on rewritten queries but scores of different rewritten queries are not comparable. One that misinterprets the user intent (e.g., asks a different question) may still have documents of higher relevance in the corpus than a correct rewritten query.

Finally, in this paper, we further claim that relevance is fundamentally insufficient for measuring the usefulness of grounding documents because it is LLM-agnostic and thus does not consider how a particular document will be utilized by a particular LLM. We present scenarios where even an ideal relevance scorer would fail, demonstrating the need to distinguish between the generation utility of documents and their retrieval relevance.

Same Document But Different Utility: An LLM-agnostic relevance scorer will assign the same score to a document irrespective of which LLM will utilize the document. However, different models often derive different levels of utility from the same document. Figure 1 shows that two LLMs initially fail to answer a question (using parametric knowledge). When conditioning on the same document, only one of them gives the correct answer. We further show this phenomenon in Section 4.3.

Less Relevance But More Utility: Recently, Cuconasu et al. (2024) reported the counterintuitive finding that adding irrelevant documents to an LLM’s context improves the correctness of the final response (given the context also includes a document with the correct answer). Moreover, the degree to which noise improves response quality depends on the LLM used to generate the response. This result runs counter to the intuition of relevance scores, which always assign low scores to random

¹The only annotation GROGU needs is a small calibration set, used once to set hyper-parameters when applied to a new model (Appendix E). Afterwards, GROGU applies directly to other datasets without further annotation or re-tuning.

Question: Who lives in the blue house in Balamory?
Answer: Edie McCredie
Document: (Title: Balamory) Character Actor Job House Song (s) Miss Hoolie Julie Wilson Nimmo Nursery teacher Green houseNursery ... Edie McCredie Juliet Cadzow Bus driver Blue garage / house When I Honk My Horn (Series 1 - 4) Let Me TakeYou on a Journey (Series 3 — 4) PC Plum Andrew Agnew ...
Qwen-2-1.5b-it (1.5B): Ungrounded: The character named “Bella” ✗ Grounded: The Blue House is owned by Mr. and Mrs. Green ✗
Phi-4 (14B): Ungrounded: The blue house is home to Mr. Bloom. ✗ Grounded: Edie McCredie, played by Juliet Cadzow. ✓

Figure 1: The same document may have different LLM-specific utility. The weaker model (Qwen2-1.5b-it) cannot utilize this document likely due to its challenging format. Both models are sampled 50 times under Qwen’s default sampling configuration, with consistent results.

documents, while these random documents may have high generation utility. We show further evidence of this phenomenon in [Section 4.2](#).

We address these issues by proposing Grounding Generation Utility (GROGU), an LLM-specific metric that directly inspects the downstream LLM’s generation confidence, rather than relying on LLM-agnostic relevance scorers or annotated labels. Because GROGU is computed with the same model that will consume the grounding context, it captures both LLM-specific preferences ([Section 4.3](#)) and cases where less relevant documents actually offer more utility to a model ([Section 4.2](#)). GROGU can significantly automate the fine-tuning of a RAG system by avoiding human annotations while also capturing nuances that relevance scores ignore.

3 Grounding Utility

We design GROGU using the LLM’s generation confidence. In particular, we propose to measure the change in confidence of an LLM when conditioned (and not conditioned) on documents. We focus on the scenario where the ground-truth answer is unknown and evaluation must be done without any reference to ground-truth annotations. Although this approach does not fully replace relevance scores or annotation-based evaluation, we show that it complements existing paradigms.

3.1 Preliminaries

At a high level, GROGU answers the question: given a grounding context, how much more confident is the LLM in its answer than without the context? If the context contains the information the LLM needs, the model’s distribution over the answer tokens sharpens, reducing entropy. We design an algorithm to restrict the entropy calculation to tokens that carry the substantive content of the answer (*key tokens*), making the entropy-based measure better reflect the model’s confidence.

Formally, we define GROGU such that it depends on the specific language model θ , a question q , and a list of retrieved documents $\mathbf{D}_r$ placed in the context window of the model (or the “grounding context” for short). We want to quantify the utility of $\mathbf{D}_r$ for the model θ to generate a response to the question q , that is, $\text{GROGU}_\theta(q, \mathbf{D}_r)$.

3.2 Confidence Difference as Utility

We define the utility of the grounding context as the change in the model’s generation confidence, γ , with and without conditioning on this context.

$$\text{GROGU}_\theta(q, \mathbf{D}_r) = \gamma(y_g | q, \mathbf{D}_r) - \gamma(y_u | q) \quad (1)$$

where y_u and y_g are sequences generated when not conditioning/conditioning on the grounding documents, respectively. We explore several formulations for the confidence measure. The most straightforward confidence measure can be derived by taking the negative values of the average perplexity or average entropy of the generated tokens. Both are well-established practices, which we discuss further in [Appendix C.7](#).

Key-token Confidence: When an LLM generates an answer, many tokens can be high-confidence regardless of the grounding context: scaffolding phrases (e.g., “My answer is ...”), parts copied from the question, stop words, or sub-words that always follow other sub-words. These tokens skew the confidence score and hurt the reliability of GROGU (example in [Figure 4](#), [Appendix B](#)).

We propose to compute confidence over only a subset of the generated tokens, which we call *key tokens*. Our *key tokens* selection algorithm applies to different confidence measures and we explain it using entropy as an example. For explanation purposes only, we assume $\mathbf{D}_r$ contains a single document (D_r) to simplify the notation.²

²The definitions based on perplexity and multiple documents are straightforward extensions.

We start by computing the KeyEntropy for the grounded generation, i.e., $\text{KeyEntropy}(y_g | q, \mathbf{D}_r)$, serving as the first term in Equation 1. We obtain the grounded generation y_g via greedy decoding conditioned on $\mathbf{D}_r$ and q , then compute token distributions for the same y_g conditioning only on q . A token y_i in y_g is a key token if

$$|H_\theta(y_i | q, D_r, y_0, \dots, y_{i-1}) - H_\theta(y_i | q, y_0, \dots, y_{i-1})| > \alpha$$

where α is a threshold chosen based on a calibration set and H is the token-level entropy.³ The intuition is that tokens unrelated to the core of the answer (e.g., those copied from the query) typically show little change in entropy with and without the grounding context, so this procedure filters them out. If no token’s entropy change exceeds α , we fall back to the top K% of tokens with the highest entropy. This is based on our observation that when all tokens have small entropy changes, using the token entropy directly gives slightly more robust performance than using the entropy changes. Finally, KeyEntropy is the average entropy of the key tokens. Figure 5 in Appendix B shows an example.

How our design addresses potential confounders

A natural concern is that low entropy may arise from factors unrelated to the usefulness of a document. Our formulation is designed to address exactly this, in two ways: we measure the *change* in entropy induced by the grounding context rather than its absolute level, and we restrict the computation to key tokens. Consider three such confounders. First, *memorized knowledge*: if the model’s parametric knowledge already suffices to answer the query confidently, then adding a grounding context produces a smaller entropy reduction. This is the intended behavior, not an error, because under our model-specific definition a document that tells the model something it already knows does have *less utility to that model*. Second, *prompt regularity*: the phrasing of a prompt can raise or lower generation entropy overall, but the prompt is held identical in the grounded and ungrounded conditions, so it cancels in the difference and any change is attributable to the grounding documents. Third, *specific lexical patterns*: key-token selection is designed precisely to filter tokens whose confi-

dence is driven by lexical regularity rather than by the answer content (Figure 5).

Simplified calculation in practice The KeyEntropy defined above does not directly apply to ungrounded generations. This does not matter in practice: we are only interested in comparing different grounding contexts’ utility given a specific query, so the ungrounded utility in Equation 1 always corresponds to the same query and is a common term that cancels exactly when two contexts are compared (we can just compare the $\text{KeyEntropy}(y_g | q, \mathbf{D}_r)$ for different contexts). Discarding this term is therefore a mathematically exact simplification for ranking rather than a change of metric, and it leaves the interpretation of GROGU as a confidence *gain* from grounding intact; we give the derivation in Appendix C.1. Thus, we don’t need to generate the ungrounded answer y_u , and can save inference cost. Scoring a candidate grounding context is efficient compared with manual annotations, because it just requires the model to answer the query conditioned on the grounding context. We report the cost analysis in Appendix C.6. We also provide a possible definition of ungrounded KeyEntropy in Appendix C.2 for completeness, though it does not affect our experiments.

4 Properties of Grounding Utility

We study GROGU’s properties with three tests, evaluating whether it distinguishes ground-truth documents from documents that do not contain an answer (Section 4.1) and whether the metric captures relationships ignored by relevance scores (Section 4.2 and Section 4.3). We follow Cuconasu et al. (2024) and use the test split of Natural Questions (Kwiatkowski et al., 2019) (unless otherwise specified) and Contriever (Izacard et al., 2022) as our retriever, but only the second test (Section 4.2) is inspired by Cuconasu et al. (2024).

4.1 Identifying Ground-truth Documents

While GROGU is designed to capture utility beyond relevance for a specific LLM, we want it to reflect some of the relationships based on ground-truth annotations. This is to check that GROGU preserves the basic relevance ordering before we test it in Section 4.2 and Section 4.3, where a good metric must not be just a relevance signal. We apply this basic check to the SQuAD dataset (Rajpurkar et al., 2016) and Natural Questions (Kwiatkowski et al., 2019) for quick confirmation, and use more

³We use the standard definition. For the i -th token, $H = -\sum_{j \in V} p_j \log(p_j)$, where p_j is the probability for token j in the vocabulary to be the i -th token, conditioned on the previous tokens in the sequence.

Model	Metric	NQ		SQuAD	
		Gold/ Dist	Gold/ Rand	Gold/ Dist	Gold/ Rand
Phi	PPL	71.3	72.7	87.1	89.3
	KeyPPL	76.9	83.6	89.7	94.5
	Entropy	74.7	76.2	89.2	91.5
	KeyEntropy	77.9**	83.2**	90.7**	95.8**
Qwen	PPL	79.8	85.2	82.5	90.6
	KeyPPL	80.2	87.1	83.2	91.2
	Entropy	81.6	87.2	84.5	91.0
	KeyEntropy	82.4*	88.8**	84.3	91.0

Table 1: Win rates of gold vs. distractor/random with different metrics on Natural Questions (NQ) and SQuAD (Rajpurkar et al., 2016). We report statistical significance between Entropy and KeyEntropy (* and ** indicate $p < 0.1$ and $p < 0.05$ significance, respectively).

complicated datasets in the later experiments (Section 5).

In order to measure this aspect, we create specific tests by leveraging existing datasets having annotations of the gold document that answers a query. In particular, we want to test whether GROGU assigns a higher utility to a gold document (D_{gt}) that contains the answer and a lower utility to other documents. For each query, we curate a set of “distractor” (D_{dst}) and “random” (D_{rnd}) documents. D_{dst} is the document with the highest retriever-assigned relevance but does not contain the answer. For each query q , we compute the utilities of D_{gt} , D_{dst} , and D_{rnd} under different GROGU formulations: i) PPL and KeyPPL, which use the perplexity of all tokens vs. key-tokens; ii) Entropy and KeyEntropy, which use the entropy and key-token entropy. We report the win-rates of the gold document according to each GROGU formulation, e.g., if $GROGU_{\theta}(q, D_{gt}) > GROGU_{\theta}(q, D_{dst})$, the ‘gold’ wins over the ‘distractor.’ An effective metric should lead to high win-rates for the gold document.

Result Table 1 reports the win rates of the gold document against the distractor and the random document when tested with two LLMs: Phi-4-14B (Phi) and Qwen-2.5-7B-Instruct (Qwen). We present additional experiments on two larger models in Appendix F, which show similar trends.

We observe that the entropy-based metrics overall assign higher scores to gold passages than the perplexity-based counterparts. Key-token selection substantially increases the performance of both perplexity and entropy, and KeyEntropy overall leads

to higher win rates. We use the sign test to measure whether the KeyEntropy method meaningfully improves over the simpler Entropy approach (rows corresponding to these methods are highlighted in green). We find that the key-token selection often leads to statistically significant improvements in 6 of 8 comparisons (in the other two comparisons the differences are not statistically significant).

4.2 Less Relevance But More Utility

In order to further understand how relevance and utility differ in RAG, we use the same setting as in Cuconasu et al. (2024)’s experiments with recent LLMs. In particular, we test the setting of using a mixture of one gold document with four random/retrieved documents and measure how GROGU metrics capture the utility. Similarly to Cuconasu et al. (2024), we find that contexts with 1 gold document and 4 random documents (with-rand) achieve a higher accuracy than a mixture of 1 gold document and the top 4 documents returned by a retriever (with-retr). Exact accuracies are reported in Appendix C.5. Closer analysis reveals that though the two types of grounding contexts often lead to the same correctness for a query (‘ties’ in Table 2), there are many queries where conditioning on the with-retr context gives an incorrect answer while conditioning on the with-rand context gives a correct one (the ‘with-rand wins’ row in the table). This suggests that a pure relevance-based metric would fail to reflect the utility of the documents.

	Phi	Qwen
with-retr wins	58	93
with-rand wins	186	321
ties	2645	2475

Table 2: Counts of queries where the with-retr and with-rand contexts lead to different accuracies. The exact accuracies are reported in Table 6 (Appendix C.5).

A good grounding utility metric should predict this counter-intuitive result. To evaluate our GROGU, we extract test instances from Cuconasu et al. (2024) where with-rand generates a correct answer and with-retr generates an incorrect answer (or vice versa). For a query, we define the utility scores as *concordant* if the grounding context (with-rand or with-retr) that generates the correct answer has higher utility. Similarly, *discordant* is defined as the grounding context that generates the incorrect answer has higher utility.

Let C and D be the counts of concordant and discordant cases respectively. We apply the formula of Kendall’s tau correlation $\tau = (C - D)/(C + D)$. This gives a rank correlation between -1 and 1. Intuitively, this reflects how much the observed impact of random documents agrees with the “less than” or “greater than” relationship of the confidence values.

As an additional measure of the relationship between GROGU and correctness, we predict which of two grounding contexts will generate the correct answer by comparing the GROGU scores. We report the binary classification performance of this classifier. Intuitively, the higher the classification performance, the better the grounding utility metric reflects the impact of the documents.

Result As shown in Table 3, all utility metrics have a positive correlation with the generation performance. KeyEntropy outperforms all the other formulations, achieving a correlation (τ) of 0.320 for Phi and 0.377 for Qwen. This suggests that while there exist other factors not reflected in the utility scores that contribute to generation accuracy, KeyEntropy can indeed explain part of the performance differences between with-rand and with-retr. On the other hand, the relevance-based metric (mean relevance score of grounding documents) has a negative correlation by always assigning a lower score to the with-rand context, completely failing to reveal its positive impact. Intuitively, documents that are relevant but not actually useful distract the model and raise the likelihood of alternative, not necessarily correct, answers. This *increases* the model’s generation entropy, so GROGU detects such documents as low-utility even when a relevance scorer ranks them highly. This is precisely the regime in which relevance is not the same as utility.

Model	Metric	τ	Acc (%)	F1 (%)
Phi	PPL	0.172	58.6	52.3
	KeyPPL	0.238	61.9	57.7
	Entropy	0.295	64.8	56.6
	KeyEntropy	0.320	66.0	59.6
	Relevance	-0.525	23.8	–
Qwen	PPL	0.266	63.3	55.6
	KeyPPL	0.275	63.8	56.8
	Entropy	0.348	67.4	58.1
	KeyEntropy	0.377	68.8	60.6
	Relevance	-0.551	22.5	–

Table 3: Simple classifier results and correlation (τ) between utility and accuracy. Relevance-based classifiers always predict the same label so its F1 is undefined.

4.3 Same Document But Different Utility

Different LLMs may derive different levels of utility from the same context, i.e., a document that helps one model may not be useful for another one. This is a long-tail problem that occurs less frequently in datasets but can nonetheless still lead to statistically significant performance differences. We instantiate the phenomenon with a controlled *positional* variation, which connects it to the order effects and “lost-in-the-middle” behavior documented in long-context LLMs (Liu et al., 2024). We construct synthetic contexts that contain the same information but vary in the physical layout of that information. We accomplish this by first selecting ten documents for each question (including the gold document and nine retrieved documents) and then creating three alternative contexts by varying where the gold document sits among the documents (first, fifth, or last). Our objective is to demonstrate that our GROGU scores can select contexts that improve answer correctness over randomly choosing a gold-containing context and that the preferences indicated by GROGU scores for one model do not apply to other models. Figure 2 presents the test procedure. Notably, the random baseline in this test would have high performance by construction: all three candidate contexts contain the gold passage and differ only in its position within the prompt. The test measures whether GROGU can identify the most useful of several already-relevant contexts for a specific model, targeting the LLM-specific utility. Given the long-tail nature of this phenomenon, the improvement margin of GROGU is expected to be small but statistically significant.

We develop this controlled *positional* variation evaluation because it holds the information content of the context exactly constant, so any difference in utility is attributable to the model rather than to the content. This evaluation is narrow by design: it keeps the comparison controlled while still showing that GROGU tracks model-specific utility. Characterizing this phenomenon of “Same Document But Different Utility” more broadly requires more evaluation scenarios, which we hope our work will motivate.

Result: Table 4 presents the result. We focus on grounding utility based on Entropy and KeyEntropy.⁴ $\Phi(\mathcal{D}_{\Phi})$ denotes the responses generated using Phi when given contexts selected by Phi’s

⁴Other metrics give similar conclusions (Appendix D.1).

Input: Two LLMs with parameters θ_i and θ_j ; a set of questions Q ; k grounding contexts for each $q \in Q$
Procedure:

1. For model LM_{θ_i} :
 - (a) For each question $q \in Q$:
 - i. Compute the utility of each grounding context with θ_i .
 - ii. Select context with highest utility: $D_{\theta_i}^q$.
 - (b) Construct set of best contexts: $\mathcal{D}_{\theta_i} = \{D_{\theta_i}^q \mid q \in Q\}$.
 - (c) Compute average accuracy: $\text{LM}_{\theta_i}(\mathcal{D}_{\theta_i})$.
2. Repeat for LM_{θ_j} to obtain $\mathcal{D}_{\theta_j}$ and $\text{LM}_{\theta_j}(\mathcal{D}_{\theta_j})$.
3. Additionally construct $\mathcal{D}_{\text{rand}}$ which randomly selects a grounding context for each question.
4. Compute generation accuracy for these scenarios.
 - (a) Same LLM for utility scoring and answer generation: $\text{LM}_{\theta_i}(\mathcal{D}_{\theta_i})$, $\text{LM}_{\theta_j}(\mathcal{D}_{\theta_j})$
 - (b) Different LLM for utility scoring and answer generation: $\text{LM}_{\theta_i}(\mathcal{D}_{\theta_j})$, $\text{LM}_{\theta_j}(\mathcal{D}_{\theta_i})$
 - (c) Randomly select grounding context for each question: $\text{LM}_{\theta_i}(\mathcal{D}_{\text{rand}})$, $\text{LM}_{\theta_j}(\mathcal{D}_{\text{rand}})$

Figure 2: Procedure to test the utility of the same document for different LLMs.

	Metric		
	Entropy	KeyEntropy	Random
$\text{Phi}(\mathcal{D}_{\text{Phi}})$	73.3	74.7	72.9
$\text{Phi}(\mathcal{D}_{\text{Qwen}})$	72.1	72.5	
$\text{Qwen}(\mathcal{D}_{\text{Qwen}})$	68.8	69.3	66.0
$\text{Qwen}(\mathcal{D}_{\text{Phi}})$	66.1	66.3	

Table 4: Generation Accuracy for different context scenarios. We present the random baselines in the last column, which correspond to $\text{Phi}(\mathcal{D}_{\text{rand}})$ and $\text{Qwen}(\mathcal{D}_{\text{rand}})$.

GROGU scores. Similarly, $\text{Phi}(\mathcal{D}_{\text{Qwen}})$ denotes responses generated by Phi given contexts selected by Qwen’s GROGU scores.⁵ We highlight two conclusions: i) using GROGU to select which context to use for response generation improves over a random choice (this aligns with findings in the previous two experiments); ii) for a given model, using the best context of another model does not perform as well as using the best context selected by this model’s own GROGU scores (all the accuracy gaps are statistically significant).

This experiment establishes that LLMs can prefer different physical layouts of the same information (in this case different gold passage positions) and that GROGU can help to select the layout that leads to higher accuracy. We expect that other variations in how information is expressed (e.g., format, phrasing, and so on) will also lead to different

⁵We use similar notation for Qwen rows.

impacts on different models.

5 Query-Rewriter Training

In Section 4, we observed that KeyEntropy achieves the best performance across the tests, so we use it as our GROGU metric to train a query-rewriter. Under this scenario, a user and a RAG system have a multi-turn conversation. The user’s questions will be based on past turns and the system needs to rewrite the query before sending it to the retriever. Query-rewriting is an important step for RAG since human interactions with modern LLMs are highly conversational. Figure 3 in Appendix A provides an illustration. In this section, we present our method of training a query-rewriter by using GROGU without any annotated labels.

5.1 Training Data Creation

Our data preparation and training resemble Yoon et al. (2025)’s supervised pipeline (RetPO) but we replace their steps that require gold annotations with GROGU. In the original method, RetPO prompts GPT-4 with three different strategies to achieve a diverse sample of 25 rewrites for each question. It then sends those rewrites to a retriever and measures the quality of the rewrite based on the rank of the gold passage in the top 100 passages the rewrite retrieves. Informed by this retrieval performance, it selects the best rewrites to construct a dataset for SFT warmup and a preference dataset for DPO training (by pairing better and worse rewrites based on their gold passage ranks).

Unlike RetPO, we do not assume access to gold passage labels: we use GROGU to evaluate retrieval success. For each rewrite, we take the top 10 retrieved documents and send them to an LLM generator to compute GROGU. We take the rewrite corresponding to the highest GROGU for SFT warmup and we pair the rewrites with the highest and lowest GROGU for DPO. Because we want the preferred and dispreferred rewrites in our DPO dataset to be different enough, we filter out 50% of the preference pairs based on the GROGU gap between the preferred and dispreferred (removing those with smaller gaps). The rest of the procedure and the raw data follow RetPO. Note that GROGU is used only to construct the preference data, so its cost is incurred once at training time and the trained rewriter adds no overhead at inference. We further discuss these implementation details in Appendix E.

5.2 Training Experiment

We use the same models as before for the main experiments and additionally show our method generalizes to larger models in [Appendix F](#). We discuss our model choice in [Appendix C.3](#).

The most comparable baseline RetPO uses llama2-7b-chat. Thus we attempted to use Qwen and Phi to reproduce their experiments. We reproduced half of their experiments (those involving BM25) because they only released the curated preference data for these experiments. We observed similar performance with Qwen and Phi, not outperforming the results reported in their paper. Because of this observation and not having the curated preference data to reproduce their ANCE experiments, we use the performance reported in the original RetPO paper as our main baseline.

We exactly follow [Yoon et al. \(2025\)](#) for our evaluation. We test our method on the sparse retriever BM25 and the dense retriever ANCE ([Xiong et al., 2021](#)). We then benchmark the RAG system on top of: TopiOCQA ([Adlakha et al., 2022](#)) and QReCC ([Anantha et al., 2021](#)). More details about the setup are provided in [Appendix E](#).

Baselines We compare with major baselines that use annotations. The most comparable one is RetPO as we closely follow their pipeline. We also report additional baselines: ConvGQR ([Mo et al., 2023](#)) uses gold answers and combines query rewriting with expansion; IterCQR ([Jang et al., 2024](#)) uses gold passages and iterative training; ADACQR ([Lai et al., 2025](#)) uses gold passages and aligns the rewriter with both the sparse and dense retrievers. We provide a discussion of other related but not comparable methods in [Appendix G](#).

Evaluation We use retrieval metrics based on gold passages (MRR, Recall@K) for retrieval results. We evaluate the LLM’s answers with a GPT-4o-mini judge that compares the generated answers with gold answers based on the judge prompt from [Ni et al. \(2024\)](#). We also report word overlap metrics for LLM’s answers as in [Adlakha et al. \(2022\)](#).

5.3 Training Results

We report the retrieval and generation performance. In the provided tables, ORIGINAL refers to the untrained original models, BASE SFT refers to directly distilling from randomly sampled GPT-4 rewrites from the same raw dataset as our method, instead

of doing data selection with GroGU.⁶ Finally, GroGU refers to our method described above, i.e., an SFT warmup followed by a DPO stage. [Appendix E](#) provides details about our training setup.

Retrieval Results [Table 5](#) reports the retrieval performance of different methods. Overall, we see consistent improvements on all the retrieval metrics with our method compared with not using utility-guided data selection (Base SFT). This trend is consistent across retrievers, LLMs, and benchmarks. Under the BM25+QReCC setting, Qwen (GroGU) outperforms the Base SFT baseline by 13.5 in MRR and Phi (GroGU) outperforms it by 13.7 in MRR. Compared with Original, GroGU also improves by up to 18.2 (Qwen, BM25+QReCC). Importantly, our method achieves performance close to or sometimes better than some reference-based baselines. For BM25+TopiOCQA, Qwen (GroGU) and Phi (GroGU) are close to the most comparable baseline (RetPO) in all metrics. GroGU also outperforms RetPO under the ANCE+TopiOCQA setting. Overall, the results demonstrate the need to finetune a query rewriter (given the big improvement over ORIGINAL) and our method offers an easy, annotation-free framework.

Generation Performance Generation metrics are less informative for conversational query-rewriting, due to the questions’ free-form, open-ended nature. Related works ([Yoon et al., 2025](#); [Lai et al., 2025](#); [Zhu et al., 2025](#)) only use retrieval performance. We provide our trained models’ generation results in [Appendix D.2](#), while noting that the retrieval performance is a more reliable measure. Our method consistently provides statistically significant improvements in the downstream LLM’s generation accuracy, by up to 9.4 points. The improvements are smaller than those with retrieval metrics, likely because the accuracy metric underestimates the actual performance in free-form QA. Yet, they still show our method’s advantage.

6 Related Work

Prior methods to improve RAG systems have primarily used metrics based on annotated labels. Some use the rank of the gold passages to quantify retrieval performance (e.g., [Jang et al., 2024](#);

⁶We also conduct an ablation run with randomly selected preference pairs for DPO ([Appendix D.4](#)). The effects are similar to BASE SFT, since both exclude GROGU-based context selection. We just focus on BASE SFT for result analysis here.

Type	Method	TopiOCQA				QReCC			
		MRR	R@10	R@20	R@100	MRR	R@10	R@20	R@100
Sparse (BM25)	QWEN - Original	14.6	27.8	35.5	52.7	27.7	43.6	50.7	65.4
	Base SFT	17.4	32.8	41.6	60.5	32.4	51.4	60.1	75.5
	GroGU	25.7	46.1	54.7	72.0	45.9	67.2	74.9	85.6
	PHI - Original	16.6	31.3	39.1	58.0	36.8	53.7	61.9	75.3
	Base SFT	15.7	29.0	37.8	55.1	32.1	50.4	58.6	73.5
	GroGU	27.2	48.8	58.6	74.8	45.8	66.2	73.7	84.3
	RetPO	28.3	48.3	-	73.1	50.0	69.5	-	89.5
	ConvGQR	12.4	23.8	-	45.6	44.1	64.4	-	88.0
	IterCQR	16.5	29.3	-	54.1	46.7	64.4	-	85.5
	ADACQR	28.3	48.9	-	71.2	55.1	76.5	-	93.7
Dense (ANCE)	QWEN - Original	28.1	47.7	54.1	65.1	29.6	45.7	51.0	59.9
	Base SFT	33.9	56.8	64.2	76.1	35.2	54.1	61.1	71.2
	GroGU	37.8	61.7	68.7	79.8	40.2	60.8	67.5	76.8
	PHI - Original	31.6	53.6	60.6	73.1	36.0	54.9	61.3	70.6
	Base SFT	29.7	51.3	59.6	72.6	32.9	51.8	58.5	69.4
	GroGU	38.1	61.1	68.3	79.6	39.4	59.8	67.1	76.3
	RetPO	30.0	49.6	-	68.7	44.0	66.7	-	84.6
	ConvGQR	25.6	41.8	-	58.8	42.0	63.5	-	81.8
	IterCQR	26.3	42.6	-	62.0	42.9	65.5	-	84.1
	ADACQR	38.5	58.4	-	75.0	45.8	67.3	-	83.8

Table 5: Sparse and dense retriever performance on TopiOCQA and QReCC. Results are presented in two groups: reference-free methods and baselines that require ground truths. The best result in each group is shown in bold. Baseline models’ performance scores are those reported in their original papers, which do not report Recall@20.

Lai et al., 2025). These metrics have served as reward functions to build DPO training data (Yoon et al., 2025) or for online RL (Zhu et al., 2025). Other works send documents to the downstream LLM and compare the generated answer with the gold one(s), using metrics such as accuracy, Exact Match, and the likelihood of gold answer (Mo et al., 2023; Zhang et al., 2024; Sun et al., 2025). These works also improve an individual component of RAG, e.g., the retriever (Salemi and Zamani, 2024) or the reranker (Sun et al., 2025). We focus on an annotation-free metric to improve the query-rewriter, though it likely applies to these other components too. Additionally, our metric is based on the idea of key token extraction, which is also used in Fang et al. (2025) but they focus on evaluating language modeling in long-context settings using the perplexity assigned to reference sequences. We apply key token selection to measure generation confidence and design it for the RAG setting. Our work is also related to scoring frameworks for RAG outputs (e.g., RAGAS (VibrantLabs, 2024; Es et al., 2024), ARES (Saad-Falcon et al., 2024), and DIRAS (Ni et al., 2025)), which address problems orthogonal to ours. We discuss them in Appendix C.4.

7 Conclusion and Future Directions

We define GROGU, a reference-free metric for the utility an LLM derives from its grounding documents for Retrieval-Augmented Generation. GROGU selects the key tokens of a grounded gener-

ation and measures the change in entropy they contribute, using no gold passages, gold answers, or per-domain annotation. We propose new evaluation scenarios that give more insight into retrieval relevance and grounding utility, and show that GROGU improves LLM performance through better context selection or by training a query rewriter.

Our experiments demonstrate that retrieval relevance and grounding utility are distinct quantities: relevance is a property of a document in the context of a query, whereas utility is a property of a document, query, *and* the model that uses it. The same grounding context provides different levels of utility to different LLMs (Section 4.3), and, counterintuitively, less relevant documents can carry more utility than higher-ranked ones (Section 4.2). No LLM-agnostic relevance scorer can express either effect, while GROGU captures both. When applied to model training, GROGU-selected preference data trains a query-rewriter that substantially improves retrieval and answer generation.

We hope future research will apply GROGU to improve more RAG components that decide what reaches the LLM generator; for example, training rerankers with RL rewards based on GROGU instead of ground-truth annotations and expanding the training data to broader domains where annotations are unavailable. We also hope our work will motivate more research on the phenomena of “relevance $\neq$ utility” and “same document, different utility levels”, studying them in more scenarios and improving RAG pipelines accordingly.

Limitations

In this paper, we try to re-think the retrieval quality evaluation from the perspective of utility to an LLM in a RAG setting, assuming no access to any references/labels. To show this, we dedicate a substantial part of the paper to the motivation and the analysis of the properties of our utility metric GROGU with ad-hoc experiments. From a practical perspective, we then test the metric on a single task, i.e., query-rewriting, showing robust improvements in both retrieval and generation performance. We didn't specifically explore training other RAG components in this paper, which is already dense in content, but we expect GROGU to be also applicable to other components of RAG systems, e.g., the reranking stage. We do show that GROGU can potentially be used for a context selection step in a RAG pipeline (Appendix D.3).

GROGU also requires access to the downstream LLM's token-level output distributions, since it computes entropy over the vocabulary. Many commercial LLM APIs do not expose these distributions. Some cloud-based models return top- k token probabilities. Computing GROGU with top- k token probabilities (rather than over the entire vocabulary) would be a promising alternative, though it is not the exact metric used in the paper.

As we pointed out throughout the paper, evaluating open-ended, free-form generation by comparing the generated answer with the gold one(s) can oftentimes underestimate the actual model performance. We still report this evaluation as part of our results to provide a complete picture but we dedicate more space to the retrieval performance as it is more suitable to show the impact of our method.

Ethics Statement

Any Retrieval Augmented Generation system, regardless of retrieval quality, may still accidentally include misleading, non-factual, or other inappropriate documents that may be used by the downstream LLM to produce harmful content. Therefore, these systems should be deployed with caution and users should be informed of these limitations. Additionally, for some application scenarios, external modules may be necessary to detect inappropriate contents and prevent them from being propagated to the LLM's generation.

References

- Vaibhav Adlakha, Shehzaad Dhuliawala, Kaheer Suleman, Harm de Vries, and Siva Reddy. 2022. [Topi-OCQA: Open-domain Conversational Question Answering with Topic Switching](#). *Transactions of the Association for Computational Linguistics*, 10:468–483.
- Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2021. [Open-Domain Question Answering Goes Conversational via Question Rewriting](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. [The Power of Noise: Redefining Retrieval for RAG Systems](#). In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. [RAGAS: Automated Evaluation of Retrieval Augmented Generation](#). In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*.
- Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. 2025. [What is Wrong with Perplexity for Long-context Language Modeling?](#) In *Proceedings of the Thirteenth International Conference on Learning Representations*.
- Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. [Unsupervised dense information retrieval with contrastive learning](#). *Transactions on Machine Learning Research*.
- Yunah Jang, Kang-il Lee, Hyunkyung Bae, Hwanhee Lee, and Kyomin Jung. 2024. [IterCQR: Iterative Conversational Query Reformulation with Retrieval Guidance](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.
- Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. [How Can We Know When Language Models Know? On the Calibration of Language Models for Question Answering](#). *Transactions of the Association for Computational Linguistics*, 9:962–977.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural Questions: A Benchmark for Question Answering](#)

- Research. *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Yilong Lai, Jialong Wu, Congzhi Zhang, Haowen Sun, and Deyu Zhou. 2025. **AdaCQR: Enhancing Query Reformulation for Conversational Search via Sparse and Dense Retrieval Alignment**. In *Proceedings of the 31st International Conference on Computational Linguistics*.
- Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2022. **ELMER: A Non-Autoregressive Pre-trained Language Model for Efficient and Effective Text Generation**. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. **Lost in the Middle: How Language Models Use Long Contexts**. *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. **Sparse, dense, and attentional representations for text retrieval**. *Transactions of the Association for Computational Linguistics*, 9:329–345.
- Fengran Mo, Kelong Mao, Yutao Zhu, Yihong Wu, Kaiyu Huang, and Jian-Yun Nie. 2023. **ConvGQR: Generative Query Reformulation for Conversational Search**. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- Jingwei Ni, Tobias Schimanski, Meihong Lin, Mrinmaya Sachan, Elliott Ash, and Markus Leippold. 2025. **DIRAS: Efficient LLM Annotation of Document Relevance for Retrieval Augmented Generation**. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.
- Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. 2024. **Mixeval: Deriving wisdom of the crowd from llm benchmark mixtures**. In *Proceedings of the 38th International Conference on Neural Information Processing Systems (NeurIPS)*.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. **Direct preference optimization: Your language model is secretly a reward model**. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS)*, volume 36.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. **SQuAD: 100,000+ questions for machine comprehension of text**. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.
- Julian Risch, Timo Möller, Julian Gutsch, and Malte Pietsch. 2021. **Semantic answer similarity for evaluating question answering models**. In *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*.
- Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. **ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems**. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.
- Alireza Salemi and Hamed Zamani. 2024. **Towards a Search Engine for Machines: Unified Ranking for Multiple Retrieval-Augmented Large Language Models**. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- Jiashuo Sun, Xianrui Zhong, Sizhe Zhou, and Jiawei Han. 2025. **DynamicRAG: Leveraging Outputs of Large Language Model as Feedback for Dynamic Reranking in Retrieval-Augmented Generation**. In *Proceedings of the 39th International Conference on Neural Information Processing Systems (NeurIPS)*.
- VibrantLabs. 2024. Ragas: Supercharge your llm application evaluations. <https://github.com/vibrantlabsai/ragas>.
- Liang Wang, Nan Yang, and Furu Wei. 2023a. **Query2doc: Query expansion with large language models**. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.
- Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. 2023b. **Learning to filter context for retrieval-augmented generation**. Preprint, arXiv:2311.08377.
- Orion Weller, Benjamin Van Durme, Dawn Lawrie, Ashwin Paranjape, Yuhao Zhang, and Jack Hessel. 2025. **Promptriever: Instruction-trained retrievers can be prompted like language models**. In *Proceedings of the Thirteenth International Conference on Learning Representations*.
- Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. **Approximate nearest neighbor negative contrastive learning for dense text retrieval**. In *Proceedings of the Ninth International Conference on Learning Representations*.
- Jiacheng Xu, Shrey Desai, and Greg Durrett. 2020. **Understanding Neural Abstractive Summarization Models via Uncertainty**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.
- Chanwoong Yoon, Gangwoo Kim, Byeongguk Jeon, Sungdong Kim, Yohan Jo, and Jaewoo Kang. 2025. **Ask Optimal Questions: Aligning Large Language Models with Retriever’s Preference in Conversation**.

In *Findings of the Association for Computational Linguistics: NAACL 2025*.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. [Making retrieval-augmented language models robust to irrelevant context](#). In *Proceedings of the Twelfth International Conference on Learning Representations*.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. [RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs](#). In *Proceedings of the 38th International Conference on Neural Information Processing Systems (NeurIPS)*, volume 37.

Tianhua Zhang, Kun Li, Hongyin Luo, Xixin Wu, James Glass, and Helen Meng. 2024. [Adaptive Query Rewriting: Aligning Rewriters through Marginal Probability of Conversational Answers](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Changtai Zhu, Siyin Wang, Ruijun Feng, Kai Song, and Xipeng Qiu. 2025. [ConvSearch-R1: Enhancing Query Reformulation for Conversational Search with Reasoning via Reinforcement Learning](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.

A Additional Illustration

We present additional illustration in Figure 3.

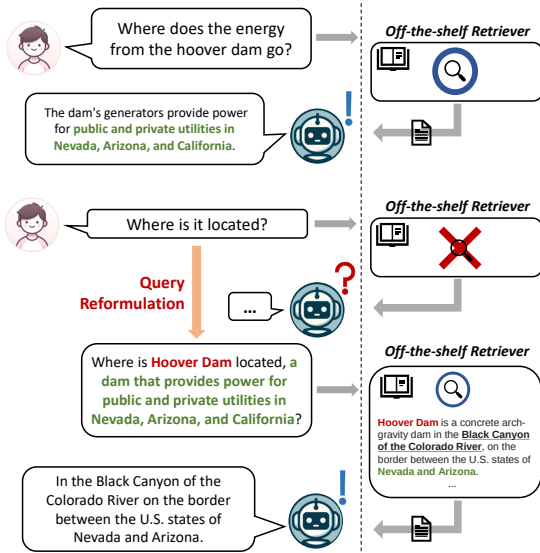


Figure 3: Query-rewriting illustration from Jang et al. (2024). The user’s queries are dependent on the conversation history and need to be reformulated into standalone queries for the retriever.

B Examples

We present additional examples in Figure 4 and Figure 5.

C Additional Discussion

C.1 Simplified GROGU

In Section 3.2 we noted that the ungrounded term of Equation 1 can be dropped when comparing grounding contexts. We show here that this is an exact identity for ranking rather than a change of metric.

Fix a query q and consider two candidate grounding contexts $\mathbf{D}_1$ and $\mathbf{D}_2$. Both correspond to the same ungrounded generation y_u , so

$$\begin{aligned} \text{GROGU}_\theta(q, \mathbf{D}_1) - \text{GROGU}_\theta(q, \mathbf{D}_2) &= [\gamma(y_g | q, \mathbf{D}_1) - \gamma(y_u | q)] \\ &\quad - [\gamma(y_g | q, \mathbf{D}_2) - \gamma(y_u | q)] \\ &= \gamma(y_g | q, \mathbf{D}_1) - \gamma(y_g | q, \mathbf{D}_2). \end{aligned}$$

The ungrounded term $\gamma(y_u | q)$ is constant across all candidates for the same query and cancels exactly. Consequently,

$$\arg \max_{\mathbf{D}} \text{GROGU}_\theta(q, \mathbf{D}) = \arg \max_{\mathbf{D}} \gamma(y_g | q, \mathbf{D}),$$

so ranking candidates by the grounded confidence alone induces exactly the same ordering as ranking them by the full GROGU definition. The interpretation of GROGU as the confidence *gain* caused by grounding is therefore unaffected by this simplification: we are still measuring that gain, and we simply avoid recomputing a term that is identical for every candidate.

We emphasize the scope of this equivalence. It holds whenever the compared contexts share the same query, which is the setting of every application in this paper (selecting among grounding contexts for a query, and constructing preference pairs among rewrites of the same query). It does not hold when comparing utilities *across* different queries, where the ungrounded terms differ and the full definition is required.

C.2 Definition of ungrounded generation confidence

We define the key tokens of the ungrounded generation to satisfy

$$H_\theta(y_i | q, y_0, \dots, y_{i-1}) > \alpha,$$

where y is the ungrounded generation. If no tokens satisfy this criterion, the top $K\%$ tokens in entropy will be used. Under this definition, tokens with small entropy will not contribute to our uncertainty

Question: Who hosted They Think It's All Over?

Gold Answers: ['Lee Mack', 'Nick Hancock']

Grounded on a retrieved passage that **does not** contain the answer

- Generated Answer: Jonathan Ross hosted "They Think It's All Over." ✗
- Entropy: 0.466

Grounded on a gold passage

- Generated Answer: Nick Hancock, followed by Lee Mack ✓
- Entropy: 0.569

Entropy by token:

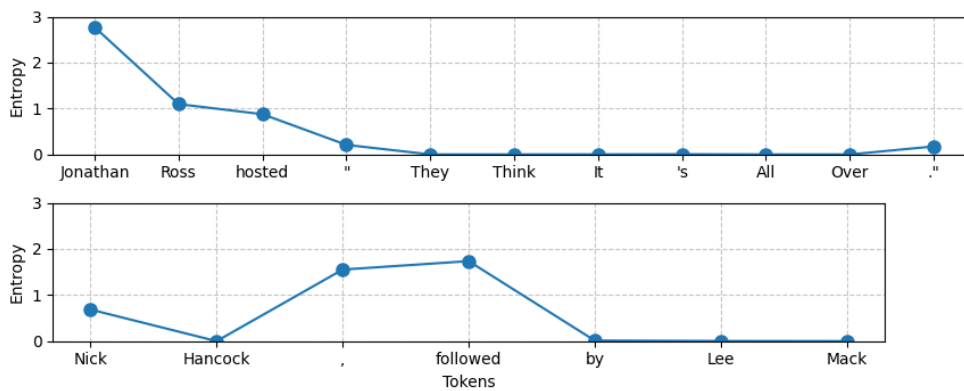


Figure 4: Example showing that non-answer tokens can skew the uncertainty/confidence measure. The model answers the question correctly when grounded on the gold passage but the average entropy is higher, indicating a lower confidence. This is because the incorrect answer repeats part of the question which skews down the average entropy of the generation.

estimation. This helps mitigate the spurious correlations of non-key tokens because we empirically find that when the LLM is not given a grounding context, the low-entropy tokens in the generated answer are mostly non-answer tokens. We note that this simple definition should only be used for ungrounded generations because for a grounded generation, an important answer token may actually have low entropy due to the presence of the same/related tokens in the grounding document.

C.3 Model Choices

The models covered in this paper are the main-stream models that have completely open licenses (Apache and MIT licenses) and have model sizes that we can efficiently train with our compute resources. Some other "open models" were not chosen due to the legal constraints of their specific

terms and conditions (e.g., Llama models are only fully open-sourced to companies that have fewer than 700 million monthly active users).

C.4 Discussion on RAG Scoring Frameworks

A separate line of work develops scoring frameworks for RAG outputs. They address problems distinct from ours and rely on assumptions that conflict with the reference-free, training-free, cross-domain setting that motivates GROGU.

RAGAS (VibrantLabs, 2024; Es et al., 2024) is a representative evaluation library in this area, but it does not provide a usefulness metric comparable to ours. It either requires annotations (e.g., grading notes, gold references) or reports generic judge-assigned relevance, both of which we want to avoid. Our paper already compares against several other recent methods that rely on annotations

Question: Who hosted They Think It's All Over?

Gold Answers: ['Lee Mack', 'Nick Hancock']

Tokens selected by our KeyEntropy algorithm for calculation are in blue.

Grounded on a retrieved passage that **does not** contain the answer

- Generated Answer: Jonathan Ross hosted "They Think It's All Over." ✗
- Entropy: 0.466
- KeyEntropy: 1.024

Grounded on a gold passage

- Generated Answer: Nick Hancock, followed by Lee Mack ✓
- Entropy: 0.569
- KeyEntropy: 0.569

Entropy by token:

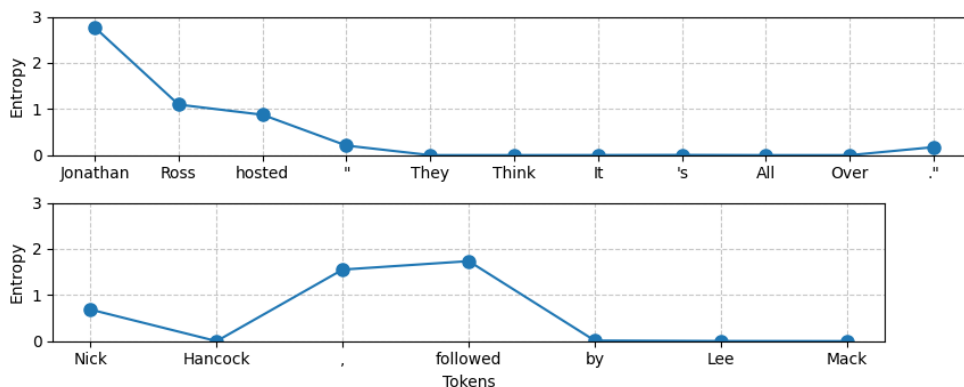


Figure 5: Example showing that KeyEntropy mitigates the issue of non-answer tokens. The model answers the question correctly when grounded on the gold passage but the average entropy is higher, indicating a lower confidence. KeyEntropy identifies and omits the tokens in the incorrect answer that repeat part of the question, which skews the entropy calculation.

(Section 5.2, Section 6).

Other approaches in this area, such as ARES (Saad-Falcon et al., 2024) and DIRAS (Ni et al., 2025), focus on domain-specific, fine-tuned judges, where a separate evaluator LLM is fine-tuned to produce scores. ARES requires domain-specific human annotations. DIRAS replaces human annotations with annotations distilled from a strong frontier model and then trains an open-source annotator. In both cases the resulting evaluator is domain-specific: a new round of annotation collection and a new evaluator must be produced for each new dataset or domain. This is incompatible with our goal of a single metric that can be applied across datasets without per-domain training.

Additionally, DIRAS only produces a relevance score that is independent of the downstream LLM that will consume the document, so it inherits the LLM-agnostic limitations identified in Section 2 and demonstrated empirically in Section 4.2 and Section 4.3.

C.5 Accuracies in the Less Relevance But More Utility experiment

Table 6 reports the exact accuracies behind the analysis in Section 4.2, where we compare an LLM’s accuracy when conditioned on a context of 1 gold document and 4 random documents (with-rand) against its accuracy when conditioned on a context of 1 gold document and the top 4 retrieved docu-

ments (with-retr). For both Phi and Qwen, the with-rand context yields higher accuracy than the with-retr context, replicating the finding of [Cuconasu et al. \(2024\)](#) with more recent LLMs and motivating the per-query concordant/discordant analysis in the main text.⁷

	Phi	Qwen
with-retr Acc (%)	87.0	80.3
with-rand Acc (%)	91.4	88.2
with-retr wins	58	93
with-rand wins	186	321
ties	2645	2475

Table 6: Full version of [Table 2](#) including the exact LLM accuracies (%) when conditioned on with-retr and with-rand contexts.

C.6 Computational Cost of GROGU Scoring

GROGU is designed to replace manual annotation, which requires an annotator to identify at least one ground-truth document for every query, out of a large corpus, and which must be repeated for every new domain. GROGU’s fully automated process is substantially cheaper. Its main computational overhead is the same as asking the model to answer the query. LLM-judge-based approach could be an alternative but is not more efficient: any judge also requires at least one model call per candidate context, typically with a larger judge model and/or chain-of-thought decoding in order to be reliable, which amounts to more computation than a single forward pass through the downstream LLM. Judges additionally tend to require annotated references or per-domain evaluator training ([Appendix C.4](#)), which conflicts with our reference-free, cross-domain setting.

Cost measurement [Table 7](#) reports the average wall-clock time to compute the GROGU score for a single candidate grounding context, i.e., the top-10 documents retrieved by one rewrite. The candidates for a query are independent of one another, so scoring is fully parallelizable and can be optimized with many inference-acceleration techniques. [Table 7](#) just uses the vLLM 0.11.0 library’s default continuous batching and other default settings and the A100 GPU. More modern hardware, inference implementation, and optional dependencies will likely further reduce the time cost.

⁷We follow [Cuconasu et al. \(2024\)](#)’s steps to compute accuracy, which treats the correctness of an answer as binary (either correct or incorrect).

Downstream LLM	TopiOCQA (s)	QReCC (s)
Qwen2.5-7B	0.11	0.26
Phi-4 (14B)	0.22	0.49
Qwen2.5-32B	0.26	0.59
Olmo-3.1-32B	0.27	0.60

Table 7: Average wall-clock time in seconds to compute the GROGU score for a single candidate grounding context (the top-10 documents retrieved by one rewrite). Computed from a randomly sampled 2500 (query, rewrite) pairs from each dataset. Measured on 1×A100 80GB GPU, except for the 32B models, which use 2×A100 80GB.

One-time cost at training time In our main application ([Section 5](#)), GROGU is used only to construct preference data and adds zero overhead at inference time. Because this is a one-time cost, we extract a single preference pair from a sizable pool of 25 candidate rewrites, and therefore 25 grounding contexts. This 25-candidate setting is specific to data collection, where we deliberately score a large and diverse rewrite pool. A training-free deployment use of GROGU, such as selecting among the grounding contexts produced by a few different retrieval pipelines, involves far fewer candidates per query.

C.7 Further Discussion on Entropy

Entropy as an established confidence measure Using the token-level entropy of an off-the-shelf LLM to reflect its generation confidence or uncertainty is established practice ([Xu et al., 2020](#); [Jiang et al., 2021](#); [Li et al., 2022](#)). We therefore treat the motivation for entropy itself as well established, and focus our contribution on improving how entropy is applied to the RAG setting through key-token selection.

Entropy versus Perplexity The two measures differ in how much of the model’s output distribution they use. A language model’s perplexity over a generated sequence y of length t is

$$\text{PPL}(y) = \exp \left\{ -\frac{1}{t} \sum_{i=1}^t \log p_{\theta}(y_i | y_{<i}) \right\},$$

which depends only on the probability assigned to the token that was actually generated, i.e., the most likely token at each position under greedy decoding. The token-level entropy we use instead sums over the entire vocabulary at each position ([Section 3.2](#)), so it also reflects how the remaining probability mass is distributed. Two grounding contexts can

therefore yield nearly identical perplexity while leaving the model with different amounts of uncertainty, which perplexity cannot distinguish but entropy can. This difference likely contributes to the results that the entropy-based formulations consistently outperform their perplexity-based counterparts in [Section 4](#).

Relationship to uncertainty calibration

GROGU is designed to work with off-the-shelf models, so we do not include a calibration stage that updates model weights, and our approach already produces strong empirical results. Models can nonetheless be calibrated through finetuning so that their generation entropy better tracks the correctness of their output ([Jiang et al., 2021](#)), and we would expect a better-calibrated model to yield a sharper GROGU signal. We leave training-time interventions for such calibration to future work.

D Additional Results

D.1 Same Document But Different Utility

[Table 8](#) presents additional results on the Same Document But Different Utility experiment introduced in [Section 4.3](#). The same analysis with entropy-based metrics from [Section 4.3](#) still apply to the perplexity-based metrics. We see that the perplexity-based utility metrics can also help identifying the contexts that give higher performance than the random baseline and an LLM generator achieves the best performance if the utility scorer is the same model (e.g., $\text{Phi}(\mathcal{D}_{\text{Phi}})$ instead of $\text{Phi}(\mathcal{D}_{\text{Qwen}})$). This further supports our analysis in [Section 4.3](#). In general, using KeyPPL to select grounding contexts gives the higher performance than just standard perplexity, further indicating the benefits of key token selection. We also note that when Phi is used as both the generator and utility scorer, KeyPPL leads to slightly better performance than the corresponding KeyEntropy from [Table 4](#) (75.3 vs. 74.7) but this difference is not statistically significant and KeyEntropy still outperforms KeyPPL most of the time across the tasks.

D.2 Downstream Generator Performance

We measure the performance of the downstream LLM that uses the documents retrieved by the rewrites. Because word-overlap-based metrics (Precision, Recall, F1 and Exact Match) are problematic for evaluating free-form generation ([Risch](#)

	Utility Metric		
	PPL	KeyPPL	Random
$\text{Phi}(\mathcal{D}_{\text{Phi}})$	74.0	75.3	72.9
$\text{Phi}(\mathcal{D}_{\text{Qwen}})$	71.9	72.4	
$\text{Qwen}(\mathcal{D}_{\text{Qwen}})$	68.4	69.2	66.0
$\text{Qwen}(\mathcal{D}_{\text{Phi}})$	65.8	66.5	

Table 8: Generation accuracy for different scenarios of the Same Document But Different Utility test for perplexity-based utility metrics

[et al., 2021](#)),⁸ we focus on the accuracy judged by the LLM judge. We adapt the LLM-judge for QA tasks from [Ni et al. \(2024\)](#) to evaluate the generator’s answers. We use GPT-4o-mini as the judge and prompt it to produce a binary label for each of the generator’s answer. An answer is considered correct as long as the LLM judge matches any of the gold answers. The LLM judge’s output is reported as accuracies (“acc”) in [Table 9](#). The LLM judge is more suitable for the free-form, open-ended setting than the word-overlap metrics, though it would still underestimate the generation performance when there are multiple semantically different correct answers but the provided gold answers are limited.

As presented in [Table 9](#), we observe that our method is able to consistently provide statistically significant improvements in the downstream LLM accuracy.⁹ For example, GROGU trained models are able to improve the accuracy of Qwen from 48.8% (Original) to 56.2% for TopiOCQA and from 51.4% to 60.5% for QReCC when using a BM25 retriever. We observe similar outcomes for the Phi model. When using the ANCE retriever the improvements are smaller in magnitude, as they start from a higher baseline, e.g., Phi improves from 66.7% to 70.1% in TopiOCQA. Notably, we notice that our method is improving also with respect to the BaseSFT approach, with gains around 4-8 points in the BM25 setting and .5-4 points with ANCE. The magnitude of changes is smaller than those with the retrieval results, likely a result of the accuracy metric underestimating the actual performance in free-form QA. Yet, they still show that

⁸The evaluation of free-form, open-ended questions is difficult because they often have many correct answers ([Anantha et al., 2021](#)) while the benchmarks come with only one or few gold answers. Thus it can under-estimate the LLM’s accuracy. LLM judge mitigates this by better recognizing semantically equivalent answers but does not solve the issue fully.

⁹The previous works we used in retrieval evaluation do not report the downstream LLM performance, so we are only comparing the untrained models vs. the models we trained.

Type	Method.	TopiOCQA					QReCC				
		EM	f1*	prec*	recall*	Acc	EM	f1*	prec*	recall*	Acc
BM25	QWEN - Original	17.7	39.8	47.5	46.3	48.8	1.0	28.8	23.8	49.3	51.4
	Base SFT	19.0	42.5	50.3	48.7	52.4	1.1	30.8	25.7	51.6	56.0
	GroGU	20.3	44.7	52.6	51.3	56.2	1.2	32.3	26.8	53.9	60.5
	PHI - Original	5.7	32.3	27.0	63.6	56.4	0.4	32.9	25.7	54.2	59.2
	Base SFT	5.5	32.2	26.8	63.7	57.2	0.2	31.9	24.8	53.0	58.1
	GroGU	6.6	35.9	30.0	69.8	65.8	0.4	34.1	26.5	56.5	62.6
ANCE	QWEN - Original	19.2	43.1	50.5	50.5	54.3	0.9	27.9	22.9	48.7	50.4
	Base SFT	20.9	46.7	54.0	55.1	60.2	1.1	30.1	24.7	51.6	55.7
	GroGU	20.8	46.9	54.6	54.7	60.7	1.1	31.1	25.6	52.9	58.3
	PHI - Original	6.2	36.0	29.8	70.2	66.7	0.4	32.0	24.8	53.3	57.6
	Base SFT	6.2	36.1	29.9	70.5	66.2	0.3	31.5	24.4	52.6	57.5
	GroGU	7.0	37.7	31.2	73.1	70.1	0.4	33.0	25.6	55.0	60.1

Table 9: Downstream LLM performance on TopiOCQA and QReCC. We calculate the word overlap metrics f1*, prec*, recall* based on the method from Adlakha et al. (2022). A test instance (a query) may have more than one gold answers. For each model answer, we compute f1, precision, and recall with respect to every gold answer. We then take the mean of the maximum f1, precision, and recall values for each answer.

our annotation-free method can provide sizable improvements in generation accuracy.

D.3 Additional Context Selection Results

Besides training a query-rewriter, GROGU can also be applied directly as a context selection step to improve the quality of the grounding contexts sent to the downstream LLM. Section 4.3 provides preliminary evidence by showing that GROGU identifies the more useful context among marginally different candidates. More direct evidence comes from our preference data construction process itself: GROGU identifies the more useful grounding contexts, and the rewritten query that produced them is labeled as the preferred data.

This context selection application can also be understood via a related training-free scenario. Suppose we cannot train a query-rewriter but can use GPT-4 to generate a diverse set of rewritten queries per instance as we did for preference data construction. Each rewritten query yields a potentially different group of grounding documents, and GROGU can select the group most useful for the downstream LLM. Because this scenario is effectively a byproduct of our preference data construction, we observed that the contexts selected by GROGU consistently achieved higher MRR and Recall@K than the average across all contexts. Since the result is consistent and largely a consequence of the same mechanism that drives our main training results (Section 5), we report a single representative example in Table 10 rather than for every possible combination.

	MRR	Recall@10	Recall@20
GROGU selection	48.6	77.9	84.0
No selection	37.5	59.7	68.5

Table 10: GROGU context selection application. ‘GROGU selection’ represents choosing the group of contexts that received the highest GROGU score for each instance. ‘No selection’ represents the average of all grounding contexts (retrieved by all 25 GPT4-generated queries per instance).

D.4 DPO with Randomly Paired Preferences

As a quick additional ablation, we construct the DPO preference pairs by sampling two rewrites randomly from the same 25-candidate pool for each query, without GROGU scoring, and train with the same pipeline otherwise. Table 11 reports the result. Performance is close to the Base SFT baseline on both benchmarks and well below GROGU. Random pairs carry no utility signal, so the DPO stage adds little over the SFT. The gap between random pairing and GROGU therefore reflects the contribution of the utility signal rather than of the DPO stage itself.

E Additional Implementation Details

E.1 Experiment Implementation

Grounding Utility Hyperparameters We tune the α and K hyperparameters for KeyEntropy with the test from Section 4.1. We tune them on a small calibration set (100 questions) from the train split of the SQuAD dataset (Rajpurkar et al., 2016), which is disjoint with any of the data we use in this paper. For α , We search over a range between

Method	TopiOCQA				QReCC			
	MRR	R@10	R@20	R@100	MRR	R@10	R@20	R@100
Base SFT	17.4	32.8	41.6	60.5	32.4	51.4	60.1	75.5
Random DPO	16.9	32.5	41.7	60.2	32.0	50.7	59.0	74.6
GroGU	25.7	46.1	54.7	72.0	45.9	67.2	74.9	85.6

Table 11: Retrieval performance when the DPO preference pairs are formed at random (Random DPO). The model is Qwen and the retriever is BM25. Base SFT and GroGU rows are repeated from Table 5 for reference. As expected, the Random DPO performance is similar to Base SFT.

0 and 0.5 with a step size of 0.05. For K , we search over a range between 0 and 1 with a step size of 0.1. We use $\alpha = 0.05$ and $K = 0.1$ for both Qwen-2.5-7b-instruct and Phi-4 in our paper. For larger LLMs, we noticed that increasing α beyond 0.05 consistently hurt KeyEntropy’s performance and made KeyEntropy underperform the standard entropy. This observation led us to additionally search over the α range (0.01, 0.05) for the two 32B LLMs at a step size of 0.01. As a result, we use $\alpha = 0.02$ for Qwen-2.5-32b-instruct and $\alpha = 0.01$ for Olmo-3.1-32B-Instruct. While this means that different models potentially require different hyperparameters, it is not an issue in practice. We just need to apply hyperparameter tuning on a small calibration set for each model. All our experiments in the paper use the same hyperparameter values for a given model, showing that the hyperparameters generalize across datasets and task settings once tuned. We note that this tuning step does not conflict with our reference-free claim: it is a one-time step per model, the 100 calibration labels it uses already exist in a public dataset, and so no new annotation is collected. Once tuned, applying GroGU to a new domain, dataset, or task requires no reference labels, which is the sense in which GroGU is reference-free throughout this paper.

Training Data We randomly sample 15555 queries from RetPO’s raw TopiOCQA dataset and 13238 queries for its raw QReCC dataset. These are about half the size of the data RetPO used for their data preparation and training pipeline. We choose these dataset sizes to save training compute. We take the 25 GPT-4 generated rewrites for each query from RetPO. There are some queries that RetPO only generated 15 queries, we followed their procedure and generated the remaining 10 queries using GPT-4. For our main experiments, we use the same LLM for producing GroGU and query-rewriting and use the same retriever for training and evaluation. For experiments with larger models, we use a smaller model as the query-rewriter and use

the larger models to assign GroGU scores (further explained in Appendix F.2).

Utility Scoring The downstream LLM assigns the GroGU score by taking the top 10 documents retrieved by the rewritten query. This corresponds to how the downstream LLM would use the documents in a simple RAG pipeline, where no reranker is involved and the downstream LLM just takes the top 10 documents.

Though not a focus of this paper, this approach can be easily extended to optimizing a rewriter while assuming access to a common reranker that re-ranks the top 100 retrieved documents and sends K documents to the downstream LLM (e.g., 5 or 10 documents). In this case, we would utilize the GroGU of the top- K documents after the reranker as the training signal. In either case, the way we use GroGU perfectly matches how the downstream LLM would use the documents. The LLM would see the same K documents we measure the GroGU for, which is a feature of our method instead of a compromise. Because this is an initial paper to propose the new paradigm of using reference-free metrics for RAG, we focus on the simpler setup without a reranker, which already strongly supports our paradigm with the significant improvements in retrieval performance (e.g., Recall@10, 20, and 100). We expect the second setup with reranker to produce similar results given that our method also substantially improves Recall@100, which would directly affect the quality of the documents after reranking.

Hyperparameters for Training We use the same values from RetPO (Yoon et al., 2025) for all hyperparameters, except for the learning rates. We use $2.0e-5$ for SFT warmup following (Yoon et al., 2025). We search over the learning rates $\{5.0e-7, 5.0e-6, 5.0e-5\}$ for DPO training. Table 12 present the full table of learning rates for DPO training. Similar to RetPO, we train our models for up to 3 epochs for SFT warmup and 3

Phi		
	TopiOCQA	QReCC
BM25	5e-6	5e-7
ANCE	5e-7	5e-6
Qwen		
	TopiOCQA	QReCC
BM25	5e-6	5e-6
ANCE	5e-7	5e-6

Table 12: Learning rates for different experiment scenarios.

epochs for DPO. For our SFT baseline (Base SFT), we train it for up to 6 epochs. We take the checkpoint with the smallest validation loss. A single experiment run (SFT+DPO) takes about 2.5 hours with 8 A100 80GB GPUs.

Statistical Test We use the sign test for all statistical testing. We use $p < 0.05$ as criterion for statistical significance unless otherwise specified.

E.2 Softwares Used

- transformers==4.48.2 (BSD-3 license)
- pyserini==0.22.1 (Apache-2.0 license)
- torch==2.7.1 (BSD-3 license)
- vLLM==0.11.0 (Apache-2.0 license)
- Open-sourced codebase of Cuconasu et al. (2024) <https://github.com/florin-git/The-Power-of-Noise>
- Open-sourced codebase of Yoon et al. (2025). <https://github.com/dmis-lab/RetPO?tab=readme-ov-file>
- GPT-4 (gpt-4-0125-preview) and GPT-4o-mini (gpt-4o-mini-2024-07-18) <https://platform.openai.com/docs/models>

AI assistance: We used AI assistants for questions about LaTeX commands and to polish the language of the paper. All content and claims are the authors’ own and were verified by the authors. Our use of all softwares is consistent with their intended use.

F Additional Results With Larger LLMs

We present additional results on 32B LLMs, which largely follow the trends of our main experiments with the 7B and 14B models. The models tested

are Qwen-2.5-32B-Instruct and Olmo-3.1-32B-Instruct. We choose these models for the same consideration explained in Appendix C.3.

F.1 Inference-only Experiments

Table 13 reports the win rates of the gold document against the distractor and the random document, as defined in Section 4.1 “Identifying Ground-truth Documents”. We observe that all metrics give non-trivial performance in identifying the gold documents, much greater than random guess (50%). The entropy-based metrics still overall achieve better performance than their perplexity-based counterparts. KeyEntropy also still achieves the best performance on average, though Entropy’s performance is close to KeyEntropy’s.

Table 14 is also consistent with the findings in Section 4.2, where all the utility metrics show a positive correlation with the generation performance, with KeyEntropy significantly outperforming all the other formulations.

Finally, Table 15 shows the results from the Same Document But Different Utility experiments (defined in Section 4.3). It shows that both the 32B models still display some level of order effects bias and our metric can help identify the order of documents that’s most useful to a model, increasing Olmo’s performance from 63.8 (random baseline) to 66.2 and Qwen’s performance from 83.8 (random baseline) to 84.7. We notice that the documents assigned higher utility by Qwen (i.e., $\mathcal{D}_{\text{Qwen}}$) also lead to a slightly higher accuracy for Olmo (64.9), though they are still not as effective as those assigned high utility by Olmo itself ($\mathcal{D}_{\text{Olmo}}$), which gives an accuracy of 66.2.

F.2 Training Experiments

Setup Due to the constraints of our computational resources, our experiments with larger models do not use every combination of experiment conditions that we used in our main experiments. Nonetheless, the experiments we conducted still demonstrate that our metric generalizes well to larger models. Specifically, we use the Qwen-2.5-7B-instruct model as the backbone of the query-rewriter and use a larger 32B model as the downstream LLM that assigns the utility scores. It’s different from our main experiments’ setting, where we used the same small model as both the query-rewriter and the downstream LLM. This new setting actually represents a realistic scenario where the query-rewriter is a relatively small model and

the downstream LLM is larger. In our previous main experiments, we used the same retriever for training data construction and evaluation. For large model experiments, we only use the BM25 retriever when constructing the training datasets but we also evaluate the final trained rewriter with ANCE.

Results As presented in [Table 16](#), we see consistent, substantial improvement over the off-the-shelf model and Base SFT, a trend similar to our main results in [Table 5](#). Additionally, even though we only used BM25 as the retriever to construct the training data, we still see large improvements when evaluating with the ANCE retriever. This means that the query-rewriter trained with one retriever generalizes well to another retriever, which is consistent with the findings in the original RetPO paper ([Yoon et al., 2025](#)).

Model	Metric	NQ		SQuAD		Avg
		Gold/ Dist	Gold/ Rand	Gold/ Dist	Gold/ Rand	
Qwen	PPL	77.2	79.3	81.8	87.9	81.6
	KeyPPL	79.0	82.8	82.4	89.9	83.5
	Entropy	79.8	82.9	85.3	90.9	84.7
	KeyEntropy	79.9	84.0	84.3	91.7	85.0
Olmo	PPL	72.0	75.7	80.1	85.9	78.4
	KeyPPL	71.6	77.6	80.0	87.0	79.1
	Entropy	76.6	81.3	84.8	90.3	83.3
	KeyEntropy	75.9	83.2	85.5	91.4	84.0

Table 13: 32B models’ win rates of gold vs. distractor/random with different metrics on Natural Questions (NQ) and SQuAD.

Model	Metric	τ	Acc (%)	F1 (%)
Qwen	PPL	0.189	59.4	53.4
	KeyPPL	0.246	62.3	54.4
	Entropy	0.246	62.3	55.3
	KeyEntropy	0.288	64.4	57.5
Olmo	PPL	0.068	53.4	52.8
	KeyPPL	0.107	55.4	54.9
	Entropy	0.160	58.0	57.1
	KeyEntropy	0.186	59.3	58.7

Table 14: Simple classifier performance and the correlation between utility and accuracy (τ) for 32B models.

	Metric				Random
	PPL	KeyPPL	Entropy	KeyEntropy	
Olmo($\mathcal{D}_{\text{Olmo}}$)	64.4	65.2	65.7	66.2	63.8
Olmo($\mathcal{D}_{\text{Qwen}}$)	65.1	64.8	64.8	64.9	
Qwen($\mathcal{D}_{\text{Qwen}}$)	84.0	84.2	84.5	84.7	83.8
Qwen($\mathcal{D}_{\text{Olmo}}$)	83.1	84.3	83.1	83.5	

Table 15: Generation Accuracy for different contexts scenario for 32B models. We present the random baselines in the last column, which correspond to Olmo($\mathcal{D}_{\text{rand}}$) and Qwen($\mathcal{D}_{\text{rand}}$).

Type	Method	TopiOCQA				QReCC				
		MRR	R@10	R@20	R@100	MRR	R@10	R@20	R@100	
Sparse (BM25)	Before	Post-training	14.6	27.8	35.5	52.7	27.7	43.6	50.7	65.4
		Base SFT	17.4	32.8	41.6	60.5	32.4	51.4	60.1	75.5
		GroGU (Qwen-32b)	27.2	49.8	59.3	74.5	47.0	67.4	75.2	85.6
		GroGU (O1mo-32b)	26.8	48.1	57.4	74.3	45.1	65.9	73.6	84.8
	RetPO		28.3	48.3	-	73.1	50.0	69.5	-	89.5
	ConvGQR		12.4	23.8	-	45.6	44.1	64.4	-	88.0
	IterCQR		16.5	29.3	-	54.1	46.7	64.4	-	85.5
	ADACQR		28.3	48.9	-	71.2	55.1	76.5	-	93.7
	Before	Post-training	28.1	47.7	54.1	65.1	29.6	45.7	51.0	59.9
		Base SFT	33.9	56.8	64.2	76.1	35.2	54.1	61.1	71.2
Dense (ANCE)		GroGU (Qwen-32b)	41.1	63.3	70.8	81.7	40.7	60.8	67.4	76.5
		GroGU (O1mo-32b)	39.6	63.0	69.9	80.1	40.6	60.8	67.5	76.7
	RetPO		30.0	49.6	-	68.7	44.0	66.7	-	84.6
	ConvGQR		25.6	41.8	-	58.8	42.0	63.5	-	81.8
	IterCQR		26.3	42.6	-	62.0	42.9	65.5	-	84.1
	ADACQR		38.5	58.4	-	75.0	45.8	67.3	-	83.8

Table 16: Sparse and dense retriever performance on TopiOCQA and QReCC. Results are presented in two groups: reference-free methods and baselines that require ground-truth annotations. The best result in each group is shown in bold. GroGU (Qwen-32b) and GroGU (O1mo-32b) represent experiments where the preference data are produced by the utility scores assigned by the 32B models, showing that our metric generalizes well to larger models. Baseline models’ performance scores are those reported in their original papers, which do not report Recall@20.

G Related But Not Comparable Methods

We report the performance of the methods that are related but not comparable with ours in Table 17. Promptriever (Weller et al., 2025) is based on a versatile LLM and designed to understand any input instruction/text to retrieve relevant documents. One may expect it to understand the conversational context and the instruction to find documents that answer the last question without any training. However, we empirically find that it performs much worse than our method or the comparable baseline RetPO, which are finetuned for the conversational query-rewriting task.

Query2Doc (Wang et al., 2023a) uses few-shot examples to prompt an LLM to write a pseudo-document that answers the given question. Given the LLM’s parametric knowledge of what may be relevant and the longer query (a passage rather than a question), one may expect this method to be applicable to conversational query-rewriting. We find that it still performs worse than our method or other baselines, despite incurring substantially higher generation costs. It only performs on par with or better than our method in a small number of scenarios (such as with Phi-4 on the QReCC dataset).

ConvSearch-R1 (Zhu et al., 2025) requires gold labels and is thus not a fair baseline to compare with ours. It also uses reasoning chains and on-line RL training, which require substantially more compute for both training and inference than our method and other baselines. Because of this, it is not surprising that ConvSearch-R1 overall outperforms the comparable baseline, RetPO. However, the high demand for inference-time compute makes it less suitable for practical applications, where the retrieval process needs to be done efficiently to minimize latency and compute cost. Additionally, ConvSearch-R1’s benefits become notably smaller at Recall@100, so in practical retrieval pipelines where a reranker would be applied to the top-100 documents, the performance gap would have even less impact on the final grounding documents’ quality.

Finally, we expect that our metric can replace ConvSearch-R1’s training reward that is based on gold passage labels, and leave this study to future work. For this paper, we choose RetPO as the framework to test our metric because ConvSearch-R1 requires too much resource to train (beyond our current capacity) and it is not suitable for ef-

ficient practical applications due to the high inference cost.

Type	Method	TopiOCQA				QReCC			
		MRR	R@10	R@20	R@100	MRR	R@10	R@20	R@100
BM25	GroGU - QWEN	25.7	46.1	54.7	72.0	45.9	67.2	74.9	85.6
	GroGU - PHI	27.2	48.8	58.6	74.8	45.8	66.2	73.7	84.3
	RetPO	28.3	48.3	-	73.1	50.0	69.5	-	89.5
	Query2Doc - Qwen	16.7	29.2	35.0	47.7	38.5	54.4	60.7	70.7
	Query2Doc - Phi	24.5	42.2	49.9	66.2	48.6	67.6	73.8	82.5
ANCE	ConvSearch-R1	-	59.6	-	80.1	-	77.2	-	89.0
	GroGU - QWEN	37.8	61.7	68.7	79.8	40.2	60.8	67.5	76.8
	GroGU - PHI	38.1	61.1	68.3	79.6	39.4	59.8	67.1	76.3
	RetPO	30.0	49.6	-	68.7	44.0	66.7	-	84.6
	Query2Doc - Qwen	35.7	56.2	63.2	74.5	40.5	59.5	66.3	74.6
	Query2Doc - Phi	36.5	58.5	65.6	77.6	43.6	64.1	70.9	78.7
	ConvSearch-R1	-	72.0	-	86.3	-	70.6	-	82.8
	Promptriever	7.4	12.8	15.4	22.6	11.4	19.9	24.9	32.1

Table 17: Performance of other not comparable methods. “-” indicates values not reported in the method’s paper.