KRAQ: Optimizing Retrieval-Augmented Generation with Knowledge Graph-Based Questions

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) systems face significant challenges in retrieval diversity and inference latency, limiting their effectiveness in practical scenarios. We introduce *KRAQ*, an innovative approach that employs corpus-derived knowledge graphs to generate high-quality representative questions. These precomputed questions enhance retrieval diversity by serving as diverse retrieval alternatives and reduce inference latency by enabling offline pre-computation of embeddings. Implemented within two practical RAG variants—*Combined Retrieve RAG* and *Efficient Speculative RAG*—*KRAQ* substantially outperforms competitive baselines by up to 48.7 points, achieves accuracy gains of up to 3%, and reduces inference latency by as much as 11.8%. Our results demonstrate *KRAQ*'s potential as a scalable, robust optimization for improving the performance of RAG systems.

1 Introduction

Retrieval-Augmented Generation (RAG) has emerged as the standard paradigm for addressing the static, non-controllable, and limited nature of LLM knowledge (Lewis et al., 2020; Kaddour et al., 2023). By retrieving relevant external information into the model's context, RAG grounds LLM outputs in verifiable evidence and improves factual accuracy. The approach has seen rapid and widespread industrial adoption, becoming arguably one of the most prominent applications of LLM technology in practice (Tully et al., 2024).

Despite its promise, RAG systems continue to face two core challenges. First, traditional retrieval mechanisms often yield results lacking semantic diversity, frequently retrieving passages that are lexically similar both to the query and to one another. This redundancy reduces the overall coverage of retrieved evidence, potentially limiting the quality of generated responses (Barnett et al., 2024). Second, augmenting LLM inputs with more documents not only increases computational overhead and latency but also runs into the 'lost-in-the-middle' problem, where positional biases impair the model's ability to effectively use the provided context (Liu et al., 2024). Recent approaches, such as Speculative RAG (Wang et al., 2024), have aimed to mitigate these issues by leveraging smaller draft models. However, these methods introduce an additional bottleneck by requiring online computation of instruction-conditioned embeddings, limiting scalability and efficiency.

To address these two challenges we propose KRAQ (Knowledge-graph Representative Automatic Questions), a novel methodology designed to simultaneously enhance retrieval diversity and reduce inference latency in RAG systems. The central premise of *KRAQ* is that a carefully curated set of representative questions, pre-generated from corpus-derived knowledge graphs, can serve as reusable proxies for efficient, diverse retrieval. Specifically, *KRAQ* leverages a knowledge graph built from the corpus to identify thematic communities, which are then summarized into concise natural language descriptions. These summaries are subsequently transformed into representative questions using a fine-tuned question-generation LLM. By precomputing and indexing this question set, *KRAQ* creates a reusable asset that enables RAG systems to diversify retrieval and optimize query-time computations.

We validate *KRAQ* through an extensive evaluation across two dimensions. Initially, we directly assess the quality of *KRAQ*-generated questions which outperforms baselines by up to 48.7 points. Subsequently, we evaluate the practical impact of *KRAQ* in two realistic scenarios:

- **Combined Retrieve RAG**: Employs *KRAQ*-generated questions as additional queries, enriching retrieval diversity and leading to more accurate and comprehensive responses.

- **Efficient Speculative RAG**: Uses precomputed *KRAQ*-generated questions to shift expensive embedding computations offline, thereby achieving substantial latency reductions without compromising response quality.

Empirical results on four standard benchmarks—TriviaQA, HotPotQA, BioASQ, and PubHealth—highlight the practical benefits of *KRAQ*, including accuracy improvements up to 3% and inference latency reductions up to 11.8%. These findings underscore *KRAQ*'s value as a practical, scalable enhancement to current RAG frameworks.

2 *KRAQ*

KRAQ transforms a document corpus into a valuable optimization asset: a reusable set of representative questions that succinctly capture its deep semantic structure and anticipate potential user queries. As illustrated in Figure 1, *KRAQ* leverages GraphRAG (Edge et al., 2024) to generate corpus-grounded knowledge graphs, revealing intricate relationships within the corpus. These graphs are subsequently segmented into coherent communities, each summarized into concise textual descriptions via an LLM. A fine-tuned question-generation LLM then transforms these summaries into high-quality representative questions. This precomputed question set then serves as a high-leverage proxy, enabling both diverse retrieval and significant latency reduction in downstream RAG tasks. [1]

2.1 From Corpus to Questions: The *KRAQ* Pipeline

Building upon GraphRAG (Edge et al., 2024), *KRAQ* extracts thematic clusters from the corpus and transforms them into representative questions. Specifically, *KRAQ* utilizes the GraphRAG framework to perform the first four sequential stages of its pipeline:

1. **Knowledge Extraction:** The corpus is segmented into text chunks, from which an LLM extracts structured subject-relation-object triples.
2. **Graph Construction:** Extracted entities are disambiguated and consolidated to form a unified knowledge graph $G = (V, E)$.
3. **Hierarchical Community Detection:** The Leiden algorithm (Traag et al., 2019) is applied recursively, revealing a multi-level hierarchical community structure within the graph. Each hierarchical level represents a distinct, non-overlapping partition of nodes at varying granularities.
4. **Community Summary Synthesis:** An LLM generates concise, natural-language summaries R_i for each community C_i.

KRAQ extends this pipeline by introducing a fifth, novel stage aimed at converting each community summary into a high-quality representative question. This is performed by a LLM fine-tuned for this task. Unlike GraphRAG, which directly uses summaries to answer queries, *KRAQ* converts summaries into questions, creating a reusable corpus-grounded asset. Formally, this question-generation step is defined as:

$$Q_i^K = f_\theta(R_i) \tag{1}$$

where f_θ represents the fine-tuned question-generation model mapping each community summary R_i to its representative question Q_i^K.

Fine-tuning the Question Generator. We fine-tune a pre-trained LLM using synthetic (summary, question) pairs. As existing QA datasets lack suitable community-level summaries, we generate synthetic training data from an external QA corpora not utilized in downstream evaluations. Specifically, given a dataset consisting of a question-answer-evidence tuples (Q, A, E), we employ a teacher model (GPT-4o) to synthesize a representative community summary $R = g(E, Q)$.

Details of the synthesis prompt and the full fine-tuning procedure are provided in Appendix A. The generated synthetic dataset of (R, Q) pairs is then used to fine-tune our base model via a causal language modeling objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{(R,Q)}\left[\log P_\theta(Q \mid R)\right] \tag{2}$$

Further details on the exact prompt used for fine-tuning are provided in Appendix E.2. This tailored fine-tuning process transforms a general-purpose LLM into a specialized, efficient generator capable of converting thematic summaries into natural, corpus-grounded representative questions. The resulting question set $Q^{\mathcal{K}}$ forms a key asset underpinning *KRAQ*'s downstream retrieval and latency optimizations.

[1] The source code for *KRAQ* and the fine-tuned generator model will be made available upon publication.

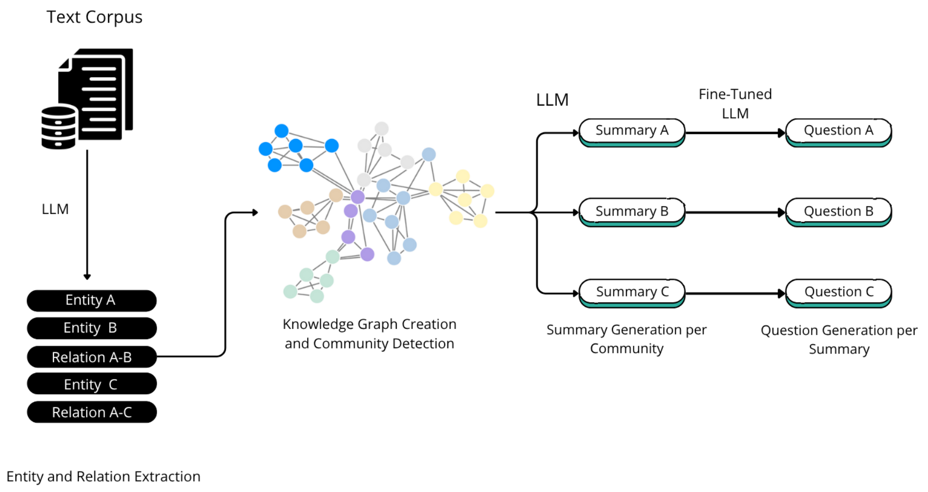
Text Corpus
LLM
Entity A
Entity B
Relation A-B
Entity C
Relation A-C
Entity and Relation Extraction
Knowledge Graph Creation and Community Detection
LLM
Summary A
Summary B
Summary C
Summary Generation per Community
Fine-Tuned LLM
Question A
Question B
Question C
Question Generation per Summary

Figure 1: The *KRAQ* pipeline. The initial stages—from knowledge extraction to community summary synthesis—are implemented using the GraphRAG framework (Edge et al., 2024). *KRAQ*'s novel contribution is the final stage: transforming community summaries into representative questions using a fine-tuned LLM.

3 *KRAQ* for RAG Optimization

The *representative question set* $\mathcal{Q}^K$ generated by *KRAQ* is a versatile asset that can be plugged into existing Retrieval-Augmented Generation (RAG) pipelines with almost no engineering overhead. Below we showcase two orthogonal ways to leverage $\mathcal{Q}^K$: (i) *Combined Retrieve RAG*, which boosts answer accuracy by diversifying the evidence retrieved, and (ii) *Efficient Speculative RAG*, which reduces latency by pre-computing expensive embeddings offline.

3.1 Combined Retrieve RAG: Diversifying Evidence

Motivation. RAG pipelines tend to retrieve documents that are highly similar yet lack semantic diversity, leaving blind spots in the evidence and hurting answer quality. *Combined Retrieve RAG* mitigates this issue by injecting carefully chosen *KRAQ* questions as additional retrieval queries to broad thematic coverage.

Method. The strategy begins by identifying the n questions in the pre-computed set $\mathcal{Q}^K$ that are most semantically similar to the user's query Q. These auxiliary questions are then used to augment the retrieval process. Let Q be the user query, M the total retrieval budget, $\alpha \in [0, 1]$ the share allocated to Q, and n the number of auxiliary questions drawn from $\mathcal{Q}^K$. Algorithm 1 summarizes the procedure.

Algorithm 1 Combined Retrieve RAG

```
Require: User query $Q$; KRAQ set $\mathcal{Q}^K$; budget $M$; primary ratio $\alpha$; # auxiliary questions $n$.
1: $\mathcal{Q}^K_{\text{sim}} \leftarrow \text{TopNSimilar}(Q, \mathcal{Q}^K, n)$
2: $m_{\text{main}} \leftarrow \lfloor \alpha M \rfloor$
3: $m_{\text{aux}} \leftarrow \left\lfloor \frac{M - m_{\text{main}}}{n} \right\rfloor$
4: $D \leftarrow \text{Retrieve}(Q, m_{\text{main}}, \varnothing)$
5: for $q \in \mathcal{Q}^K_{\text{sim}}$ do
6:     $D \leftarrow D \cup \text{Retrieve}(q, m_{\text{aux}}, D)$
7: $A \leftarrow \text{GenerateAnswer}(Q, D)$
8: return $A$
```

Note: RETRIEVE performs a similarity search for a query and returns the k highest-scoring documents *not* present in the exclusion set passed as its third argument.

By fusing evidence retrieved for both the original query and thematically aligned *KRAQ* questions, the model receives a richer, less redundant context, which empirically translates into higher answer accuracy without increasing the retrieval budget.

3.2 Efficient Speculative RAG: Reducing Latency

Motivation. RAG pipelines frequently incur two intertwined problems: (i) *high latency*, as the LLM must attend over an extensive context of retrieved passages, and (ii) *lost-in-the-middle* degradation, where information appearing far from the context's start or end is poorly utilised (Liu et al., 2024). Speculative RAG (Wang et al., 2024) addresses latency by running several lightweight draft generations in parallel, and tackles the 'lost-in-the-middle' problem by conditioning each draft on a much smaller, focused subset of the evidence. However, constructing these subsets introduces a new bottleneck: it relies on query-specific *instruction-conditioned* embeddings that must be computed *online* for every retrieved document. Our goal is to eliminate this step—thereby cutting latency further—while preserving answer quality.

Draft–then–Verify Paradigm. Speculative RAG (i) retrieve a pool $\mathcal{D}$, (ii) embeds each passage with a *query-conditioned* encoder $\mathcal{E}(d_i \mid Q)$ and clusters the embeddings into k topical buckets, (iii) builds k evidence subsets by sampling one document per cluster, (iv) lets k lightweight draft models answer Q in parallel, and (v) asks a verifier LLM to choose the best draft.

The Latency Bottleneck. Speed gains vanish at step (ii): the query-conditioned embeddings $\mathcal{E}(d_i \mid Q)$ must be computed *online* for every retrieved passage, adding a forward pass per document and becoming the dominant source of latency.

Pre-computation with *KRAQ*. We solve this bottleneck by replacing the query-specific embedding with the closest pre-computed *KRAQ* proxy. Intuitively, if two queries are semantically similar, conditioning on one should yield embeddings *good enough* for clustering documents for the other.

Offline Phase (one-time) For every document d_i and every *KRAQ* question $Q^K_j \in \mathcal{Q}^K$ we store

$$E_{\text{pre}}(d_i, Q^K_j) \;=\; \mathcal{E}\big(d_i \mid Q^K_j\big).$$

Online Phase (per query)

1. Retrieve a document pool $\mathcal{D}$ for query Q.
2. Select its nearest *KRAQ* proxy

$$Q^K_{\text{sim}} \;=\; \arg\max_{Q^K_j \in \mathcal{Q}^K} \cos(\text{emb}(Q), \text{emb}(Q^K_j)).$$

3. Fetch the cached embeddings $E_{\text{pre}}(d_i, Q^K_{\text{sim}})$ for all $d_i \in \mathcal{D}$—a pure lookup.
4. Cluster these embeddings and proceed with the standard draft–then–verify pipeline.

This simple substitution removes *all* query-time embedding calls, dropping latency by $\approx \mathtt{O}(\#\text{docs})$ forward passes while keeping the rest of Speculative RAG intact.

Algorithm 2 Efficient Speculative RAG (online)

```
Require: Query $Q$; cache $E_{\text{pre}}$; KRAQ set $\mathcal{Q}^K$
1: $\mathcal{D} \leftarrow \textsc{RetrieveDocs}(Q)$
2: $Q^K_{\text{sim}} \leftarrow \textsc{NearestKraq}(Q, \mathcal{Q}^K)$
3: $E \leftarrow \{\, E_{\text{pre}}(d, Q^K_{\text{sim}}) \mid d \in \mathcal{D} \,\}$
4: $\{c_1, \dots, c_k\} \leftarrow \textsc{KMeans}(E, k)$
5: // sample one doc per cluster to form $k$ evidence subsets
6: // run $k$ draft models in parallel
7: // verifier selects the best draft
8: return best answer $\hat{A}$
```

As the proxy Q^K_{sim} is drawn from a *representative*, domain-wide question set; empirically (Sec. 5) we find that its embeddings preserve the relative document topology required for effective clustering delivering a substantial latency reduction while maintaining a highly competitive response quality.

4 Experimental Setup

We conducted a series of experiments to validate the performance of *KRAQ*. We detailed our datasets, implementation choices, evaluation metrics, and baselines below.

4.1 Datasets

We evaluated on four standard QA benchmarks: **TriviaQA** (Joshi et al., 2017): open-domain questions authored independently of the evidence. **HotPotQA** (Yang et al., 2018): multi-hop questions that require reasoning across documents. **BioASQ** (Tsatsaronis et al., 2015): biomed questions; we score against the *exact_answer* entity list. **PubHealth** (Kotonya and Toni, 2020): public-health claims to be verified as true, false, or mixed.

For each benchmark we iteratively sampled questions and their evidence until the resulting corpus reached $\sim$5 M unique tokens, keeping experiments tractable on our hardware.

4.2 Evaluation Metrics

Question quality. We assess the semantic alignment of *KRAQ*-generated questions against benchmark reference questions. For each reference question, we find its nearest *KRAQ* neighbor via cosine similarity and compute:

- **Relevance:** Average BERTScore F_1 (Zhang et al., 2020);
- **Relevance@τ** % of pairs whose BERTScore F_1 meets or exceeds $\tau \in \{0.70, 0.75, 0.80\}$

RAG performance.

- **Exact Match** – fraction of answers containing the gold string.
- **LLM judge** – We used GPT-4.1 to rate semantic and factual equivalence (Zheng et al., 2023) (prompt in App. E.6).
- **Latency** – median wall-clock time per query; for parallel stages we adopt the slowest branch time (see App. D).

4.3 Baselines

To rigorously evaluate our contributions, we compared our methods against carefully designed baselines that allowed us to isolate the impact of each component of our framework.

***KRAQ* Question Quality.**

- **Random Baseline:** To test the core hypothesis that structured, thematic context is superior to unstructured context, this baseline generates questions from a sampling of the corpus. The method randomly samples a variable number of text chunks (2 to 7), concatenates them, and then prompts an LLM to produce a question (see Appendix E.3). This allows us to directly measure the value added by *KRAQ*'s systematic, graph-based content structuring.
- **Non-Finetuned *KRAQ*:** This baseline followed the same pipeline as *KRAQ* but used a non-fine-tuned model for the final question generation step. This allowed us to isolate the specific contribution of our fine-tuned model.

Combined Retrieve RAG.

- **Traditional RAG:** We compared our method against a standard RAG system that used the same LLM and retriever. For a given user query, it retrieved the top-M documents based solely on their semantic similarity to the original query. This direct comparison, using the prompt detailed in Appendix E.4, allowed us to quantify the gains of our retrieval diversification strategy.

Efficient Speculative RAG Baseline. The baseline for our efficiency-focused application was our re-implementation of the Speculative RAG framework from Wang et al. (2024). We followed the original design, but introduced necessary adaptations for numerical stability and model handling, which are fully detailed in Appendix C. This version computed the instruction-conditioned embeddings online for every user query. By comparing against this baseline, we could directly measure the reduction in latency and the corresponding trade-off in accuracy achieved by our optimization.

4.4 Implementation Details

All experiments were conducted on a single NVIDIA RTX 3090 GPU (24GB VRAM).

Model: Our primary model for all generative tasks was LLaMA 3.1-8B-Instruct(Grattafiori et al., 2024). We used a 4-bit AWQ quantized version to manage resources, with inference served by vLLM (Kwon et al., 2023). This model served as both the base model for all fine-tuning experiments and as the off-the-shelf instruct model for baseline comparisons and other non-fine-tuned generative roles (e.g., Drafter, Verifier).

Graph Construction: *KRAQ* pipeline utilized the open-source **GraphRAG** framework (Edge et al., 2024). We used its default configuration without prompt tuning (see Appendix B for details), with chunks of 300 tokens and 50 tokens overlap.

Embedding Models: We used two embedding models. For general-purpose semantic retrieval, `nomic-embed-text` (Nussbaum et al., 2024) with relevance calculated via cosine similarity. For instruction-conditioned embeddings, we used `InBedder-RoBERTa`(Peng et al., 2024).

Vector Store: We used the Qdrant (Vasnetsov et al., 2021) vector database for efficient indexing and similarity search.

Hyperparameters Configuration. The hyperparameters for our RAG applications were selected based on the characteristics of each dataset. The specific prompts used for each generative task are detailed in Appendix E.

Combined Retrieve RAG. For Table 2, we used $n = 2$ similar *KRAQ* questions and a retrieval proportion $\alpha = 0.5$. The total number of retrieved documents (M) was 15 for TriviaQA, HotPotQA, and BioASQ, and 10 for PubHealth.

Speculative RAG. Configurations for Table 5 and 6 were: (i) **BioASQ:** $N_{\text{retrieved}} = 18$, $k = 5$ clusters, $m = 10$ drafts.[2], (ii) **HotPotQA:** $N_{\text{retrieved}} = 10$, $k = 4$ clusters, $m = 8$ drafts, and (iii) **TriviaQA & PubHealth:** $N_{\text{retrieved}} = 10$, $k = 2$ clusters, $m = 5$ drafts.

5 Results

We first validate the semantic quality of questions generated by our core *KRAQ* methodology, then assess the downstream impact on accuracy and latency in our two RAG applications.

[2]The higher number of retrieved documents for BioASQ was selected to better cover the multiple distinct evidence sources often needed for list-based biomedical answers.

5.1 Question Quality

We first evaluated *KRAQ*'s ability to generate semantically relevant questions that cover the breadth of a corpus. As shown in Table 1, our fine-tuned *KRAQ* model delivered a consistently better performance than baselines.

The results isolate the dual benefits of our approach. First, the significant gap between *KRAQ* and the Random baseline (e.g., a 48.7-point difference in Relevance@0.75 on TriviaQA) underscores the value of the graph-based structuring, which ensures thematic coherence. Second, the equally large gap between *KRAQ* and the **Instruct** baseline (e.g., a 20.7-point difference on the same metric) demonstrates that our specialized fine-tuning is critical for transforming community summaries into high-quality, natural questions. The comparatively lower performance on PubHealth across all methods is likely attributable to its claims-based format, which diverges from the standard question structure *KRAQ* was trained on.

Dataset	**Metric**	***KRAQ***	**Non-FT *KRAQ***	**Random**
TriviaQA	Rel.	**78.1**	75.5	72.2
	R@.70	**93.0**	90.6	72.0
	R@.75	**71.0**	50.3	22.3
	R@.80	**33.0**	15.0	3.6
HotPotQA	Rel.	**74.2**	72.8	69.5
	R@.70	**84.0**	75.0	42.7
	R@.75	**40.4**	29.9	5.6
	R@.80	**10.0**	4.7	0.3
PubHealth	Rel.	**68.5**	68.0	66.7
	R@.70	**33.6**	30.3	15.6
	R@.75	**4.8**	3.4	1.1
	R@.80	**0.4**	0.26	0.03
BioASQ	Rel.	**79.0**	77.9	74.1
	R@.70	93.1	**93.6**	84.3
	R@.75	**73.8**	70.8	42.7
	R@.80	**42.6**	34.9	8.6

Table 1: *KRAQ* question generation performance (scores are percentages, R@ is Relevance@). Our fine-tuned *KRAQ* model generated more relevant questions.

5.2 Combined Retrieve RAG Results

The results presented in Table 2 validate our central hypothesis: diversifying the retrieval context with *KRAQ*-generated questions leads to more accurate RAG systems. The improvements are consistent across both Exact Match (EM) and an LLM-as-a-Judge evaluation. For standard QA datasets like TriviaQA, HotPotQA, and PubHealth, Combined Retrieve RAG consistently improves both literal precision and semantic correctness by furnishing the LLM with a more comprehensive and less redundant evidence set.

The case of BioASQ, however, reveals a key trade-off inherent to retrieval diversification. While enriching the context is beneficial for generating comprehensive narrative answers, it can be detrimental for tasks demanding exhaustive recall of a list of specific entities. BioASQ exemplifies such a task, as its answers are lists of entities and our EM evaluation requires matching at least 50% of them for a correct score.[3] We hypothesize that by broadening the thematic scope, our method may occasionally replace a highly specific document containing key entities with a more general one, leading to lower scores on these strict, list-based metrics. This highlights a key insight: the optimal retrieval strategy may be task-dependent. While our diversification proves highly effective for standard QA formats, it may need to be adapted for entity-centric retrieval scenarios, an improvement we leave for future work.

Dataset	**EM (%)**		**LLM-Judge**	
	Trad.	**Combined**	**Trad.**	**Combined**
HotPotQA	57.0	**58.6**	70.3	**71.3**
TriviaQA	88.6	**89.0**	91.0	**92.3**
PubHealth	65.5	**66.2**	65.5	**66.2**
BioASQ	**69.6**	67.5	**76.0**	74.6

Table 2: Accuracy of *Combined Retrieve RAG* vs. Traditional RAG, using Exact Match (EM) and an LLM-as-a-Judge (GPT-4.1) evaluation.

Ablation Studies. To better understand the interplay between the original query and *KRAQ*'s supplementary questions, we conducted an ablation study varying the retrieval proportion (α) and the number of questions (n). This analysis, performed on HotPotQA and presented in Table 3, reveals an optimal configuration. Performance peaks at an α of 0.75, confirming that *KRAQ* questions are most effective as a strategic supplement to the primary user query, yielding the best results in both Exact Match and the LLM-Judge evaluation. The study on n suggests that while performance remains relatively stable, a smaller number of highly-focused auxiliary questions may be sufficient, as expanding to four questions does not yield further gains. This indicates that with tuned parameters, performance can be optimized effectively.

[3]The modified EM criterion for BioASQ's list-based answers considers a response correct if it contains at least 50%

Varying n ($\alpha = 0.5$)			Varying α ($n = 2$)		
n	**EM**	**LLM-Judge**	α	**EM**	**LLM-Judge**
1	58.7	**73.0**	0.25	56.0	69.0
2	58.7	71.3	0.50	58.6	71.3
3	58.7	72.0	0.75	**59.6**	**74.0**
4	57.0	72.3	1.00	57.0	70.3

Table 3: Ablation study for Combined Retrieve RAG on HotPotQA, varying the number of auxiliary questions (n) and the retrieval proportion (α).

To validate our core hypothesis that question quality directly translates to downstream performance, a second ablation study (Table 4) examines the impact of the generator model on final RAG accuracy. The results are unequivocal: our Fine-tuned *KRAQ* model leads to the best performance across both Exact Match and the LLM-Judge evaluation. This confirms a direct causal link: higher quality, semantically relevant questions translate into more effective retrieval and more precise final answers.

***KRAQ* Question Source**	**EM (%)**	**LLM-Judge (%)**
Fine-tuned *KRAQ*	**67.5**	**74.6**
Random Baseline	65.3	71.6
Non-Finetuned *KRAQ*	65.1	72.5

Table 4: Impact of *KRAQ* generator quality on Combined Retrieve RAG performance on BioASQ.

5.3 Efficient Speculative RAG Results

Our second application targeted the efficiency of the Speculative RAG framework. We compared our implementation of the original algorithm against our optimized version, *Efficient Speculative RAG*, evaluating the trade-off between latency and accuracy.

Latency. Table 5 confirms a significant reduction in time computation (Appendix D). *Efficient Speculative RAG* achieved speedups across all datasets, with latency reductions ranging from 2.7% on HotPotQA to a substantial 11.8% on PubHealth. This result demonstrates that using a *KRAQ* question as a proxy to pre-compute instructed embeddings is a valid strategy for removing the main computational bottleneck from the online inference path.

Accuracy. Having established the significant efficiency gains, we next evaluated whether this speedup came at the cost of accuracy. The results in Table 6 show that our optimization maintains a

of the reference items, after normalization.

Dataset	**SR (s)**	**ESR (s)**	**Reduction (%)**
HotPotQA	3.01	**2.93**	2.7%
TriviaQA	3.91	**3.51**	10.2%
PubHealth	3.81	**3.36**	11.8%
BioASQ	4.32	**3.97**	8.1%

Table 5: Latency comparison between Speculative RAG (SR) and Efficient Speculative RAG (ESR). Inference time reductions.

highly competitive performance. While the original Speculative RAG method holds a consistent, albeit marginal, edge across both Exact Match and the LLM-Judge evaluation, the accuracy of *Efficient Speculative RAG* remains remarkably close. This establishes a clear trade-off: a substantial reduction in latency for a negligible impact on response quality, confirming that using a *KRAQ* question as a proxy is a robust strategy for document clustering.

Dataset	**EM (%)**		**LLM-Judge (%)**	
	SR.	**ESR.**	**SR.**	**ESR.**
HotPotQA	**44.3**	44.0	**54.3**	54.0
TriviaQA	**77.6**	75.3	**82.6**	80.6
PubHealth	**58.3**	58.0	**58.3**	58.0
BioASQ	**51.4**	50.6	**63.8**	62.0

Table 6: Accuracy comparison between our implementation of Speculative RAG (SR) and Efficient Speculative RAG (ESR).

Ablation Studies. To further validate our efficiency-focused approach, two additional studies were conducted. First, as shown in Table 7, we confirmed that proxy question quality is crucial for maintaining accuracy. The results show that using questions from our Fine-tuned *KRAQ* model is essential for achieving the highest accuracy in Efficient Speculative RAG, outperforming the other methods in both Exact Match and the LLM-Judge evaluation. This establishes a direct link between the quality of the proxy question and the final response, as better proxies lead to more relevant document clustering. This result highlights the need for our specialized, graph-aware fine-tuning process to generate the most effective proxies.

***KRAQ* Question Source**	**EM (%)**	**LLM-Judge (%)**
Fine-tuned *KRAQ*	**75.3**	**80.6**
Random Baseline	73.3	79.3
Non-Finetuned *KRAQ*	73.0	76.3

Table 7: Impact of *KRAQ* generator quality on Efficient Speculative RAG performance on TriviaQA.

Second, we analyzed how latency scaled with the number of initially retrieved documents ($N_{\text{retrieved}}$). As shown in Table 8, the latency advantage of our efficient method grew as more documents were retrieved. This is because the cost of online embedding generation for the original method increases linearly with the number of documents, while our method's lookup cost remains nearly constant. This result highlights the scalability of our optimization, demonstrating its particular value for complex queries that require a larger evidence set.

$N_{\text{retrieved}}$	**Original Latency (s)**	**Efficient Latency (s)**
10	3.01	**2.93**
15	3.09	**2.98**
20	3.23	**2.99**

Table 8: Latency comparison on HotPotQA as the number of retrieved documents ($N_{\text{retrieved}}$) increases.

6 Related Work

Our work lies at the intersection of Retrieval-Augmented Generation (RAG) optimization and knowledge-guided question generation.

RAG and Retrieval Optimization. The foundational RAG architecture (Lewis et al., 2020) established the "retrieve-then-read" paradigm. Recent work has focused on improving the retrieval step, often through query expansion. For instance, Rackauckas (2024) proposed RAG-Fusion, which generates multiple query variants online and re-ranks the combined results using Reciprocal Rank Fusion (RRF). Similarly, Li et al. (2024) introduced a dual-mode mechanism that applies semantic perturbations to the input query. While these methods broaden the search, their query variations are generated in isolation from the underlying corpus structure. Our *Combined Retrieve RAG* differs fundamentally by leveraging a pre-computed set of questions that are holistically corpus-aware. These questions, derived from the knowledge graph's global thematic structure, enable a grounded and comprehensive diversification that other methods lack. Crucially, because this asset is pre-computed offline, our design avoids introducing online inference latency, allowing for richer retrieval without a performance penalty.

Knowledge-Guided Question Generation. Using structured knowledge to guide Question Generation (QG) is an active area of research. Methodologies often leverage underlying semantics to inform what to ask. For example, some approaches construct local, single-document graphs to identify salient sentences for question generation (Do et al., 2023), while others identify key concepts across a corpus and retrieve evidence for each to ground the generated questions (Noorbakhsh et al., 2025). Recent frameworks like GraphRAG (Edge et al., 2024) have further advanced this by using global knowledge graphs to generate community-level summaries for direct QA, and can also *reactively* generate suggested follow-up questions based on a user's query history.

While *KRAQ* builds upon a similar foundation of semantic structuring, its approach and purpose are fundamentally different. Instead of generating content for direct user consumption (like summaries or suggested queries), *KRAQ*'s sole purpose is to proactively pre-compute a comprehensive question set to serve as a reusable optimization tool. We repurpose this multi-granularity, thematically-grounded set to directly enhance the performance and efficiency of downstream RAG systems, a contribution distinct from prior work.

7 Conclusion and Future Work

In this work, we introduced *KRAQ*, a novel framework that addresses RAG's dual challenges of low retrieval diversity and high latency. We demonstrated that by creating a pre-computed asset of high-quality, representative questions from a corpus knowledge graph, we can significantly enhance RAG systems. Our empirical results validate this strategy: *KRAQ* surpasses question generation baselines by up to 48.7 points, while its applications in *Combined Retrieve RAG* and *Efficient Speculative RAG* achieve accuracy gains of up to 3% and latency reductions of up to 11.8%, respectively.

While these applications serve as powerful proof-of-concept, our core contribution is broader: we propose that proactively modeling a corpus's deep semantic structure opens a new research direction for creating reusable, structure-aware optimization assets. Promising avenues for future work could follow this path, starting with enhancing the core *KRAQ* methodology with finer-grained graph features. Further research could also explore advanced applications, such as leveraging question hierarchies for multi-level retrieval, optimizing the pre-computation cost for *Efficient Speculative RAG* (Appendix D.3), and developing new tools for automated QA dataset generation and interactive corpus exploration.

Limitations

While our framework demonstrates strong performance, we identify four key limitations that also point toward fruitful avenues for future research. **1) Pre-processing Cost:** *KRAQ* requires a significant, one-time computational investment to build the graph, which may be prohibitive for very large or dynamic corpora. **2) Proxy Mismatch Risk:** The efficiency gains in our framework rely on a *KRAQ* question being a good proxy for the user query. This assumption's validity diminishes for queries that fall outside the core thematic distribution of the corpus, as no suitable proxy question may exist. **3) Error Propagation:** The quality of generated questions is dependent on the initial graph construction; biases or errors in the graph will propagate downstream. **4) Task Format Sensitivity:** Our results suggest that the benefits of retrieval diversification are less pronounced for tasks requiring highly structured, list-based answers (e.g., BioASQ). The enriched context, while effective for generating descriptive answers, may encourage the model to produce more narrative responses, which can be penalized by strict list-matching evaluation metrics.

Ethical Considerations

The deployment of *KRAQ* warrants ethical diligence. **1) Bias Amplification:** By design, *KRAQ* reflects a corpus's dominant themes, which can amplify societal biases present in the source data and affect downstream tasks. **2) Potential for Misuse:** The framework could be applied to disinformation corpora to automatically generate misleading questions. **3) Data Privacy:** When used on sensitive datasets, there is a risk that generated summaries or questions could inadvertently reveal private information. Responsible data governance and auditing of the generated questions are essential on sensible corpora.

References

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven failure points when engineering a retrieval augmented generation system. *arXiv preprint arXiv:2401.05856*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free Dolly: Introducing the world's first truly open instruction-tuned LLM. Databricks Blog, April 11 2023.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36.

Xuan Long Do, Bowei Zou, Shafiq Joty, Tran Tai, Liangming Pan, Nancy Chen, and Ai Ti Aw. 2023. Modeling what-to-ask and how-to-ask for answer-unaware conversational question generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10785–10803, Toronto, Canada. Association for Computational Linguistics.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Aaron Grattafiori and 1 others. 2024. The Llama-3 herd of models. *arXiv preprint arXiv:2407.21783*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*.

Neema Kotonya and Francesca Toni. 2020. Explainable automated fact-checking for public health claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7740–7754, Online. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th ACM SIGOPS Symposium on Operating Systems Principles (SOSP '23)*, Koblenz, Germany.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Zhicong Li, Jiahao Wang, Zhishu Jiang, Hangyu Mao, Zhongxia Chen, Jiazhen Du, Yuanxing Zhang, Fuzheng Zhang, Di Zhang, and Yong Liu. 2024. DMQR-RAG: Diverse multi-query rewriting for RAG. *arXiv preprint arXiv:2411.13154*.

NelsonF. 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.

Kimia Noorbakhsh, Joseph Chandler, Pantea Karimi, Mohammad Alizadeh, and Hari Balakrishnan. 2025. Savaal: Scalable concept-driven question generation to enhance human learning. *arXiv preprint arXiv:2502.12477*.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*.

Letian Peng, Yuwei Zhang, Zilong Wang, Jayanth Srinivasa, Gaowen Liu, Zihan Wang, and Jingbo Shang. 2024. Answer is all you need: Instruction-following text embedding via answering the question. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 459–477, Bangkok, Thailand. Association for Computational Linguistics.

Zackary Rackauckas. 2024. RAG-Fusion: a new take on retrieval-augmented generation.

Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):Article 5233.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, and 1 others. 2015. An overview of the BioASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):Article 138.

Tim Tully, Joff Redfern, and Derek Xiao. 2024. 2024: The state of generative AI in the enterprise. Menlo Ventures–Perspective Blog. Encuesta a 600 organizaciones; reporta un 51% de adopción de RAG en producción.

Andrey Vasnetsov and 1 others. 2021. Qdrant: Vector similarity search engine and vector database. `https://github.com/qdrant/qdrant`. GitHub repository.

Zilong Wang, Zifeng Wang, Long Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Speculative RAG: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv:2407.08223*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *Proceedings of the International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Datasets and Benchmarks Track*.

A *KRAQ* Generator Fine-tuning Details

This section provides additional details on the fine-tuning process for our core *KRAQ* question generator model.

A.1 Fine-tuning Dataset and Training Performance

To train our question generator, we created a synthetic dataset of (Summary, Question) pairs. We synthesized this data from two external, publicly available corpora to avoid any overlap with our evaluation benchmarks:

- **Dolly-v2** (Conover et al., 2023): A dataset of 15k instruction-following records generated by humans, valued for its diversity and natural language quality.
- **MusiQue** (Trivedi et al., 2022): A multi-hop question-answering dataset that provides complex questions requiring multi-step reasoning.

From these datasets, we extracted (Question, Evidence) pairs to synthesize the (Summary, Question) pairs used for fine-tuning. The specific prompts used for summary synthesis and question generation are detailed in Appendix E.1 and E.2, respectively.

The fine-tuning process demonstrated stable convergence, as illustrated by the validation loss curve in Figure 2.

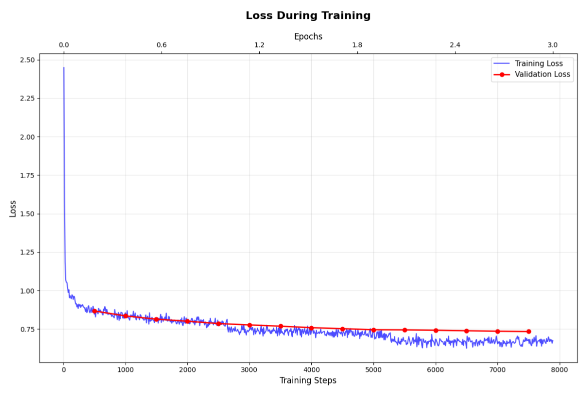

Figure 2: Validation loss curve during the fine-tuning of the *KRAQ* question generator model. The model shows steady convergence.

A.2 Hyperparameter Configuration

The *KRAQ* question generator model was fine-tuned from LLaMA 3.1-8B-Instruct using QLoRA (Dettmers et al., 2023) with the following key hyperparameters:

- **Learning Rate:** 2e-4
- **Batch Size:** 2
- **Epochs:** 3
- **Optimizer:** Paged AdamW (8-bit)
- **LoRA rank (r):** 64
- **LoRA alpha:** 16
- **LoRA dropout:** 0.05

B GraphRAG Configuration: Prompt Tuning

The GraphRAG framework includes a *PromptTuning* feature that automatically adapts its internal prompts to a specific domain. To determine the optimal configuration, we compared its performance against the default, non-tuned prompts on the BioASQ dataset.

Metric	No Tuning	Prompt Tuning
Relevance	**79.0**	78.8
R@.70	**93.1**	92.9
R@.75	73.8	**74.8**
R@.80	**42.6**	40.6

Table 9: Comparison of *KRAQ* performance with and without GraphRAG's prompt tuning on BioASQ.

As shown in Table 9, the performance was comparable, with the default configuration slightly outperforming the tuned version in most cases. Given the added complexity of the tuning step, we opted to use the default GraphRAG prompts for all experiments in this paper.

C Speculative RAG Implementation Notes

This section details the key practical adaptations made during our re-implementation of the Speculative RAG framework (Wang et al., 2024). These notes cover our solutions for ensuring numerical stability in score calculation and our empirical findings on drafter model fine-tuning, which informed our final implementation choices.

C.1 Score Calculation

The Speculative RAG framework ranks candidate drafts using a score ρ_j derived from three components:

$\rho_j^{\textbf{draft}}$**:** The probability of the Drafter model generating the rationale and response: $P(\beta_j|Q,\delta_j) + P(\alpha_j|Q,\delta_j,\beta_j)$.

$\rho_j^{\textbf{self-contain}}$**:** The joint probability of the response and rationale, measuring internal coherence: $P(\alpha_j, \beta_j|Q, \delta_j)$.

$\rho_j^{\textbf{self-reflect}}$**:** The Verifier model's confidence in the rationale, estimated as the probability of it generating a positive affirmation (e.g., "Yes").

The final score is the product of these components: $\rho_j = \rho_j^{\text{draft}} \cdot \rho_j^{\text{self-contain}} \cdot \rho_j^{\text{self-reflect}}$.

The scoring method proposed by Wang et al. (2024) relies on calculating the full probability of a sequence S, which is theoretically the product of its token probabilities:

$$P(S) = \prod_{t \in S} P(t)$$

However, in practice, this calculation leads to **numerical underflow**[4] for long sequences, where the result collapses to zero and makes it impossible to differentiate between drafts. To address this, we implemented a numerically stable confidence metric, $P_{\text{conf}}(S)$, based on the average log-probability of the tokens in a sequence S of length L:

$$P_{\text{conf}}(S) = \exp\left(\frac{1}{L}\sum_{t \in S} \text{logprob}(t)\right)$$

This metric normalizes for sequence length and is robust against underflow. We used this confidence score to compute the three score components, preserving the multiplicative structure of ρ_j. This pragmatic adaptation was crucial for a functional implementation.

C.2 Drafter Model Fine-tuning

We conducted an experiment to fine-tune the M_{Drafter} model (LLaMA 3.1-8B-Instruct) to generate both a response and a rationale, following the methodology in Wang et al. (2024) using the same datasets for training that we used in the KRAQ Generator Finetuning (Appendix A.1). However, as shown in Table 10, the fine-tuned model performed worse than the base instruction-tuned model across both Exact Match and the LLM-Judge evaluation.

[4]We use LLaMA-3-8B with AWQ INT4. Logits are computed in FP16/32 after dequantization, but reduced precision increases underflow risk, motivating P_{conf}.

Table 10: Performance on TriviaQA using a fine-tuned Drafter vs. the base instruct model, evaluated with EM and an LLM-Judge (GPT-4.1). The fine-tuned model led to a drop in accuracy.

Drafter Model Variant	EM (%)	LLM-Judge (%)
Base Instruct Drafter	**77.6**	**82.6**
Fine-tuned Drafter	71.6	76.3

Given this negative result, we opted to use the non-fine-tuned LLaMA 3.1-8B-Instruct model for all Speculative RAG experiments. To obtain structured output, we used the prompt in Appendix E.5, instructing the model to return its response and rationale in a JSON format.

D Latency Estimation for Efficient Speculative RAG

This section details the methodologies used to estimate the latency of *Efficient Speculative RAG* and its comparison to the original framework, given the practical constraints of our single-GPU experimental setup.

D.1 Modeling Parallelism Latency

Our experiments were conducted on a single GPU, which necessitated serial execution of the m draft generation and verification steps common to both Speculative RAG frameworks. To provide a realistic latency estimate for a multi-GPU environment where these steps could run in parallel, we adopted a standard simulation approach.

For each user query, we first recorded the individual execution times for generating each of the m drafts ($t_1^{\text{draft}}, ..., t_m^{\text{draft}}$) and verifying each of them ($t_1^{\text{verify}}, ..., t_m^{\text{verify}}$). The total latency for the parallelized draft-and-verify phase was then estimated by summing the maximum time of each stage:

$$\text{Latency}_{\text{draft-verify}} = \max_{j=1..m}(t_j^{\text{draft}}) + \max_{j=1..m}(t_j^{\text{verify}})$$

This estimated time was then added to the other serial components of the pipeline (e.g., retrieval, clustering) to calculate the total end-to-end latency reported in our results. This conservative estimation, based on the longest-running task, is a common practice for modeling the performance of parallelized systems in a serialized environment.

D.2 Simulating Pre-Computed Embedding Retrieval

A full offline pre-computation of all instructed embeddings for *Efficient Speculative*

RAG—calculating $\mathcal{E}(d_i|Q_j^K)$ for every document d_i and every *KRAQ* question Q_j^K—was computationally prohibitive for this work. To measure the latency of our method nonetheless, we simulated the retrieval of pre-computed embeddings during the online phase.

For a given query Q, we identified the most similar *KRAQ* question Q_{sim}^K and computed the instructed embeddings online for the retrieved documents using Q_{sim}^K as the instruction. To estimate the final latency of our efficient method, we **subtracted** the time taken for this on-the-fly computation. To account for the lookup cost, we then **added** the time taken to retrieve the initial documents as a conservative proxy, assuming that fetching embeddings from a vector store would have a similar or lower time complexity. This simulation allowed us to evaluate the accuracy of our method using the correct proxy-instructed embeddings while providing a fair estimate of the latency gains.

D.3 Hypothesis on Optimized Pre-computation

While a full pre-computation is costly, we hypothesize that it is not necessary. As a direction for future work, we propose a **selective pre-computation** strategy. For each *KRAQ* question Q_j^K, one would first retrieve the top-N most semantically similar documents (e.g., $N = 1000$) using a standard embedding model. The expensive instructed embedding calculation, $\mathcal{E}(d_i|Q_j^K)$, would then be performed only for this much smaller subset of documents. This would dramatically reduce the offline cost, making the approach practical for very large corpora while likely preserving most of the performance benefits. Validating this hypothesis remains a key avenue for future research.

E Prompts

E.1 Community Summary Synthesis Prompt

This prompt was used with a teacher model (GPT-4o) to generate a summary R from evidence E and a target question Q for the fine-tuning dataset creation.

```
Given this evidence and knowing that
we want to generate a question
about {target_question}, create a
community-style summary that:

1. Begins with "This community centers
   around..."
2. Describes the main topic that
   connects the entities.
3. Lists key members or elements.
4. IMPORTANT: Do not reference the
   specific question.
5. Make the summary concise (max 5
   sentences).

Evidence:
{evidence}
```

E.2 *KRAQ* Question Generator Prompt

This prompt was used for fine-tuning the *KRAQ* generator model and for inference to generate questions from community summaries.

```
Given this summary of a document
collection, generate a natural question
that a person might ask when looking for
this information.  The question should
be:

• Simple and straightforward
• Written in conversational language
• Focused on the main topic or event
• Something a real person would ask

Now generate a question for this
summary:
{summary}
```

E.3 Random Baseline Question Generator Prompt

This prompt was used to generate questions for the random baseline, using concatenated random text chunks as context.

```
Given these random fragments, generate a
natural, concise question that someone
might ask about the themes or topics
present in these passages. The question
should:

• Be short and to the point
• Focus on a common theme or connection
• Be something a real person would ask

Fragments:
{combined_content} Generate only ONE
concise question:
```

E.4 Traditional RAG Answer Generation Prompt

This base prompt was used for the final answer generation step in Traditional RAG and Combined Retrieve RAG.

```
Below is an instruction that describes
a task.  Write a response using the
evidence provided for it. Evidence:
{context} Instruction:
{query}
```

E.5 Speculative RAG Drafter Prompt

This prompt instructed the M_{Drafter} model to produce a response and rationale in a structured JSON format.

```
Response to the instruction. Also
provide a concise rationale that
justifies the response. Instruction:
{instruction} Evidence:
{evidence} Your response must be a valid
JSON object with the following format:
{{'response': 'your response here',
'rationale': 'your rationale here'}}
```

E.6 LLM-as-a-Judge Prompt

This prompt was used to evaluate the semantic correctness of generated answers from our RAG applications.

```
You are an expert evaluator. Your task
is to determine if the generated answer
correctly responds to the question
according to the reference answer.

Question: {question}
Generated Answer: {generated_answer}
Reference Answer: {reference_answer}

Respond with ONLY a single digit:
1 - CORRECT
0 - INCORRECT

Your verdict (just the digit 1 or 0):
```