

Beyond Average Oracle Gains: Split-based Bottleneck Diagnosis for Modular RAG

Chaewon Sung
Sookmyung Women's University
Seoul, Republic of Korea
scw226@sookmyung.ac.kr

Chong-Kwon Kim
Korea Institute of Energy Technology
Seoul, Republic of Korea
ckim@kentech.ac.kr

Yoonhyuk Choi
Sookmyung Women's University
Seoul, Republic of Korea
chldbsgur123@gmail.com

Abstract

Retrieval-augmented generation (RAG) systems are commonly implemented as modular pipelines with retrieval, reranking, and generation stages. End-to-end QA metrics measure final answer quality, but they provide limited insight into where failures arise. A common diagnostic approach is oracle intervention, where gold evidence is injected at a specific stage while the rest of the pipeline remains unchanged. However, reporting oracle results only as average gains can lead to coarse conclusions and hide subset-specific failure patterns. We revisit stage-wise oracle analysis for modular RAG and introduce a split-based bottleneck diagnosis approach based on gold-evidence intervention. Instead of relying only on global oracle comparisons, we analyze how intervention gains vary across subsets defined by retrieval outcome and question type, and we examine representative failure cases. In a 500-sample study on Natural Questions, we compare a baseline pipeline with Oracle-R, Oracle-Re, and Oracle-G settings. The results show that intervention gains are small when retrieval already succeeds but substantially larger when retrieval fails. They also show that different question types benefit most from different oracle settings, while some numerical cases remain difficult even under oracle correction because of answer-format mismatch.

CCS Concepts

• Information systems → Question answering.

Keywords

Retrieval-Augmented Generation, Stage-wise Oracle Analysis, Gold-Evidence Intervention, Bottleneck Diagnosis, Question Answering

ACM Reference Format:

Chaewon Sung, Chong-Kwon Kim, and Yoonhyuk Choi. 2026. Beyond Average Oracle Gains: Split-based Bottleneck Diagnosis for Modular RAG. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '26)*, Aug. 9-13, 2026, Jeju, South Korea. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/>

1 Introduction

Retrieval-augmented generation (RAG) systems are commonly evaluated using end-to-end answer quality metrics such as Exact Match (EM) and F1 [7, 15]. However, such aggregate evaluation often

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

KDD '26, Aug. 9-13, 2026, Jeju, South Korea

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1329-3/25/03

<https://doi.org/10.1145/>

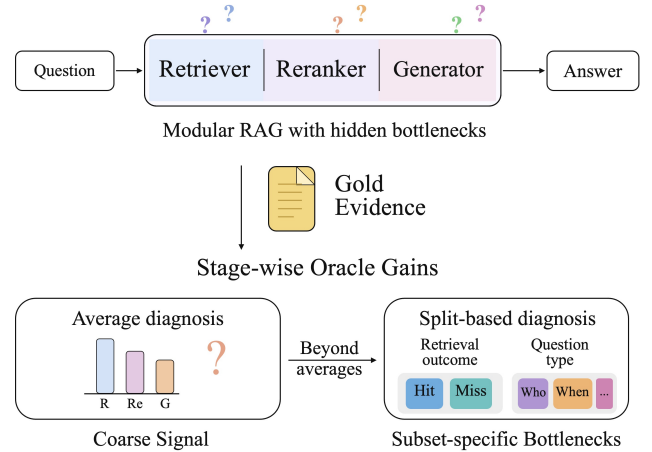


Figure 1: Conceptual motivation for our approach. Average stage-wise oracle gains can obscure heterogeneous bottlenecks, whereas split-based diagnosis reveals subset-specific failure patterns in modular RAG.

obscures which part of the pipeline is responsible for failures. In modular RAG pipelines, errors may arise from retrieval, reranking, or generation, but end-to-end scores alone do not indicate which stage is currently the main source of performance loss. A straightforward way to analyze stage bottlenecks is through oracle intervention, in which one component is corrected using gold evidence while the others remain unchanged. For example, one may force the retriever to include the gold supporting passage or directly provide gold evidence to the generator. While intuitive, this analysis is often reported only through average gains across oracle settings. As a result, it tends to yield coarse conclusions that fail to capture heterogeneous failure patterns across instances. This can raise the concern that standard open-domain QA evaluation may miss deeper reasoning and diagnostic distinctions [12].

This motivates a subset-aware view of oracle analysis. The stage that most constrains a modular RAG pipeline need not be identical for all questions. Instead, it may vary with instance-level conditions such as retrieval success and question form [2, 5, 10, 19, 20]. Therefore, we move beyond global oracle comparison and introduce a *split-based bottleneck analysis approach* based on gold-evidence intervention. Specifically, we examine how oracle gains differ across question subsets defined by retrieval outcome and surface question type. Further, we inspect representative qualitative failure cases on the benchmark dataset. Using a 500-sample study on Natural Questions (NQ) [14], we compare four settings: a standard baseline pipeline, Oracle-R, Oracle-Re, and Oracle-G (see Section 3.1 for

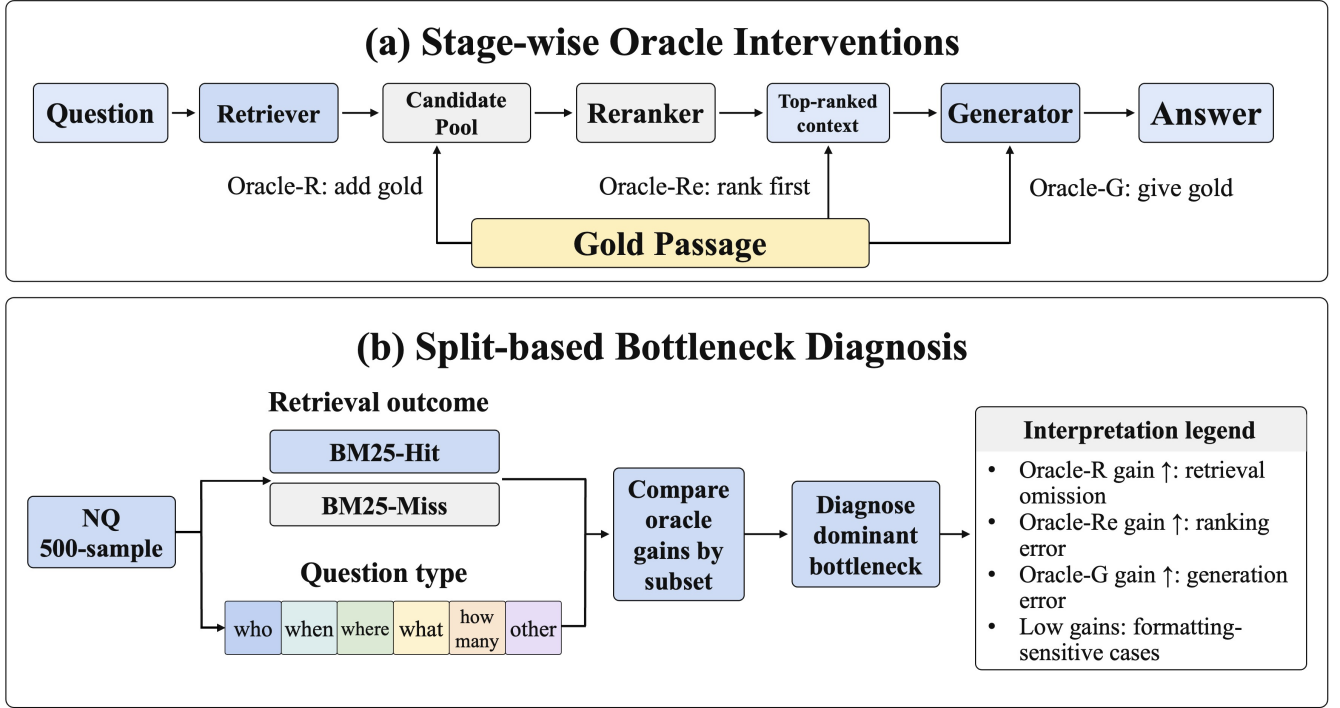


Figure 2: Overview of the proposed stage-wise oracle intervention and split-based bottleneck analysis approach for modular RAG. (a) Oracle interventions inject or prioritize the gold passage at different stages of the pipeline. (b) Split-based analysis compares oracle gains across retrieval-outcome and question-type subsets to identify subset-specific bottleneck patterns.

more details). The baseline pipeline consists of BM25 retrieval, a cross-encoder reranker, and an LLM generator. We stratify questions by whether the gold passage appears in the BM25 top- k results (retrieval hit vs. miss), as well as by surface question type. This allows us to test whether different stages appear relatively more limiting under different conditions.

Our central claim is that average oracle gains can hide substantial subset-level variation. When retrieval already succeeds, downstream processing appears to play a relatively larger role. In contrast, when retrieval fails, downstream improvements alone cannot recover missing evidence, making retrieval correction more consequential. Thus, average oracle comparison alone is insufficient for characterizing bottlenecks in modular RAG pipelines. The contributions of this paper are as follows:

- We propose a split-based bottleneck analysis approach that stratifies questions by retrieval outcome and question type under gold-evidence intervention.
- We provide empirical evidence that oracle gains alone may be insufficient to characterize bottlenecks in modular RAG.
- We show that oracle gains vary by retrieval outcome and question type. We further analyze representative qualitative failure cases that explain retrieval-driven, downstream-driven, and oracle-resistant errors.

2 Related Work

Retrieval-augmented generation. Early open-domain QA systems such as DrQA established the retrieve-then-read paradigm [4]. Subsequent work incorporated retrieval into language model pretraining [8] and used retrieved passages for generation over multiple documents [9]. RAG has since become a standard approach for knowledge-intensive NLP [7, 15]. Recent studies further improve RAG with adaptive retrieval, self-reflection, evidence-aware generation, and interleaved retrieval-reasoning [1, 2, 10, 19]. These studies mainly aim to improve end-to-end performance or generation quality under retrieval-augmented settings [2, 10, 15]. In contrast, our work focuses on diagnosing which part of a modular RAG pipeline limits performance under different conditions.

Oracle-based and component-level analysis. Component-level analysis is commonly used to understand modular NLP systems by replacing or controlling one component and measuring the resulting performance change [3]. In RAG pipelines, gold-evidence intervention provides a natural way to probe retrieval and downstream behavior because retrieval quality strongly affects final QA performance [7, 15]. For instance, one can inject the gold passage, promote it in the ranked list, or provide it directly to the generator to examine how corrected evidence access changes the output [7, 12]. However, such oracle results are often summarized as aggregate gains in end-to-end evaluation settings, making it difficult to determine whether the same bottleneck pattern holds [7, 12].

Instance-dependent behavior in RAG. Recent work suggests that retrieval needs and generation behavior vary across questions [2, 10, 19]. Adaptive and self-reflective RAG methods condition retrieval or generation decisions on the input instance [2, 10]. Corrective and context-aware approaches also show that retrieved evidence can help or hurt depending on its relevance and how it is used [5, 20]. These findings motivate our diagnostic question: if RAG behavior is instance-dependent, then bottlenecks may also vary by subset [2, 10, 20]. Based on this observation, we use split-based oracle analysis to examine how bottleneck patterns change with retrieval outcome and question type.

3 Methodology

3.1 Stage-wise Gold-Evidence Intervention

We consider a modular RAG pipeline with retrieval, reranking, and generation. Given a question q , the pipeline retrieves candidate passages, reranks them, and generates an answer conditioned on the top-ranked passages. To analyze stage-level bottlenecks, we define three oracle-style intervention settings:

- **Baseline:** The pipeline uses BM25, a reranker, and an LLM generator. No oracle is applied.
- **Oracle-R:** The gold supporting passage is forced into the retrieved candidate set, while reranking and generation remain unchanged. This setting corrects retrieval omission while preserving downstream behavior.
- **Oracle-Re:** The gold supporting passage is injected into the candidate pool and forced to rank first, while the generator remains unchanged. This controls for both retrieval omission and ranking error, allowing downstream behavior to be analyzed under corrected evidence access.
- **Oracle-G:** The generator is directly given the gold supporting passage as context, bypassing retrieval and reranking. This setting evaluates generation behavior under direct access to gold evidence, but it is not a generation oracle that forces the gold answer.

Remark. We interpret the above settings as controlled *gold-evidence interventions*, which are not perfectly isolated module-level upper bounds. Our goal is not to claim strict causal decomposition, but to compare how performance changes when evidence access is corrected at different points in the pipeline.

3.2 Split-based Bottleneck Analysis

We hypothesize that bottleneck patterns are instance-dependent. To test this, we stratify questions into subsets and compare intervention gains within each.

- **Retrieval outcome:** We divide questions into BM25-Hit and BM25-Miss subsets based on whether the annotated gold supporting passage appears in the top- k retrieved candidates. This split directly captures whether the retrieval stage succeeds under the baseline retriever.
- **Question type:** We group questions by surface form (*who*, *when*, *where*, *what*, and *how many/how much*) to examine whether each question type exhibits different bottlenecks.

In this paper, we instantiate the approach using two simple and interpretable split criteria. The goal is not to claim exhaustiveness,

Table 1: Overall performance of stage-wise gold-evidence interventions on NQ with $n=500$. ΔEM and $\Delta F1$ indicate absolute improvements over the baseline (RQ1).

Metric	Base	Oracle-R	Oracle-Re	Oracle-G
EM	.416	.494	.496	.518
F1	.579	.685	.697	.697
ΔEM	-	+.078	+.080	+.102
$\Delta F1$	-	+.106	+.118	+.118

but to show that even lightweight subset analysis reveals failure patterns that average oracle gains alone do not expose.

4 Experiments

We aim to address the following Research Questions (RQs):

- **RQ1:** How do stage-wise gold-evidence interventions affect overall pipeline performance?
- **RQ2:** How do bottleneck patterns vary across question-type subsets?
- **RQ3:** How does the observed bottleneck pattern vary with retrieval outcome?
- **RQ4:** What qualitative failure cases explain the bottleneck patterns observed in the quantitative analysis?

4.1 Datasets, Baselines, and Implementation

Dataset. We use the validation split of Natural Questions (NQ) [14] and select 500 examples with short answers. NQ is one of several standard open-domain QA benchmarks [11, 17, 22] that provide annotated answer spans and supporting passages. For each example, we extract the question, the gold short answer, and a gold supporting passage built from a local context window around the annotated answer span. Then, we construct a passage-level mini-corpus from these gold supporting passages. This controlled setup makes stage-wise comparisons easier to interpret because retrieved and oracle evidence are defined at the same granularity. However, it does not represent a full open-domain retrieval setting.

4.2 (RQ1) Overall Effect of Gold-Evidence Intervention

Table 1 reports overall performance. All oracle settings improve over the Baseline, indicating that retrieval, reranking, and generation each leave room for improvement. Here, ΔEM and $\Delta F1$ denote absolute improvements over the Baseline setting. Oracle-G achieves the highest EM (.518), while Oracle-Re and Oracle-G reach the same F1 (.697). The aggregate gains are positive but close: Oracle-R reaches EM .494, Oracle-Re .496, and Oracle-G .518. These results do not indicate a single uniformly dominant bottleneck, which motivates the subset-level analysis.

The gain from Oracle-R suggests that some baseline errors are caused by retrieval omission, since adding the gold supporting passage improves EM by .078 and F1 by .106. The additional gain from Oracle-Re is small in EM but larger in F1, indicating that ranking the gold passage first helps partial answer overlap more than exact-match accuracy. Oracle-G achieves the best EM, but it still does not solve all cases, which means that some errors remain

Table 2: Oracle performance by question type on NQ. Best EM per subset is shown in bold (RQ2).

Setting	who (175)		when (87)		where (64)		what (88)		how many (17)	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Baseline	.429	.579	.517	.574	.359	.576	.466	.643	.118	.360
Oracle-R	.509	.696	.632	.713	.438	.666	.477	.713	.177	.458
Oracle-Re	.531	.718	.644	.732	.391	.662	.546	.741	.177	.458
Oracle-G	.543	.706	.701	.797	.406	.648	.511	.708	.177	.458

even when gold evidence is directly provided. Overall, the aggregate results suggest that retrieval access, evidence ranking, and answer generation all affect performance, but they do not show whether the same bottleneck pattern holds for every subset.

4.3 (RQ2) Bottlenecks across Question-type Subsets

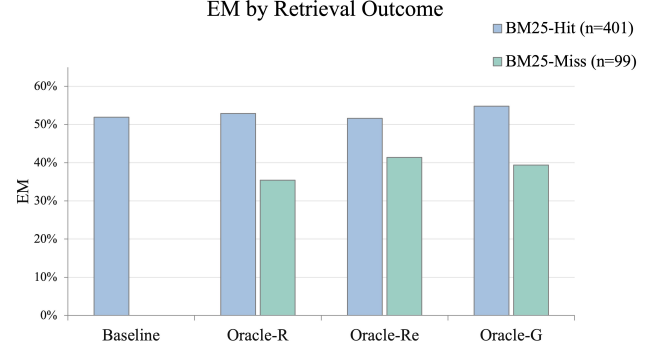
Whether the same bottleneck pattern holds across question types is difficult to see from aggregate oracle gains alone. We therefore compare oracle effects across surface question types. Table 2 shows that the best-performing oracle setting differs by subset, which is consistent with the view that bottlenecks are not globally fixed. Because some subsets are small and several differences are modest, we interpret these patterns as diagnostic signals rather than statistically definitive rankings. The five main surface question types exhibit different intervention patterns.

When questions benefit most from Oracle-G, which points to downstream answer extraction or formulation as a possible bottleneck for temporal queries. One possible reason is that even when relevant evidence is available, the model must still select a precise date or time span to match the annotated short answer. For *what* questions, Oracle-Re achieves the highest EM (.546). This pattern points to the importance of evidence ranking for broad semantic queries, where the gold passage may be present but not placed optimally for generation. In contrast, *how many* questions show only small and uniform improvement, with EM increasing from .118 in the baseline to .177 under all three oracle settings. Because this subset is small ($n = 17$), the result should be treated as preliminary. Still, the pattern suggests that some numerical cases may be limited by answer formatting rather than evidence access. Overall, the question-type split reveals variation that is not visible from aggregate oracle gains alone.

4.4 (RQ3) Bottlenecks by Retrieval Outcome

On the 500-sample subset, BM25 retrieves the annotated gold passage within the top 10 for 80.2% of questions (401/500), with an average rank of 1.58 among hits. This recall is likely higher than in a full open-domain setting because the mini-corpus is constructed from gold supporting passages. The results should therefore be interpreted within this controlled diagnostic setup. Figure 3 reports EM separately for BM25-Hit and BM25-Miss examples. This split reveals a sharper contrast than the aggregate results in Table 1.

On BM25-Hit samples, intervention gains are modest: the largest improvement is Oracle-G over Baseline (+.030 EM). Once relevant evidence is retrieved, many remaining errors appear to arise downstream. This is consistent with prior work showing that language

**Figure 3: EM within each retrieval-outcome subset. Values are computed separately for BM25-Hit and BM25-Miss subsets.**

models may fail to use relevant evidence even when it appears in context [16]. On BM25-Miss samples, the Baseline achieves EM .000 and F1 .000 in our experiment. This zero score is not implied by the split definition, where a model could recover the answer from alternative retrieved passages or from parametric knowledge. Instead, it is an empirical observation in our controlled mini-corpus. When the annotated gold passage is absent from the BM25 top-10, the baseline does not produce answers with EM or token-level F1 overlap against the gold short answers. Oracle interventions then recover substantial performance, suggesting that access to the annotated evidence is a major limiting factor for this subset.

One further observation is that Oracle-Re slightly outperforms Oracle-G on the BM25-Miss subset (EM .414 vs. .394). Oracle-Re injects the gold passage into a multi-passages context, whereas Oracle-G provides it alone; the richer context may affect generation behavior under retrieval failure, for example, by changing answer formatting or refusal behavior. Given the scale of this pilot study, we treat this as a hypothesis rather than a definitive conclusion.

4.5 (RQ4) Qualitative Failure Cases

Table 3 presents representative question-level failure cases. The examples illustrate three error patterns: missing evidence, overly broad answer extraction, and answer-format mismatch. In the numerical case, all settings output 291 episodes instead of the annotated answer 291, which remains incorrect under EM. These cases show that modular RAG failures are not fully captured by average oracle gains. Retrieval-driven failures appear in BM25-Miss examples, where the baseline cannot produce a supported answer because the gold evidence is absent from the retrieved context.

Table 3: Representative question-level failure cases from the NQ 500-sample study (RQ4).

Case type	Example question	Observed outputs	Interpretation
Retrieval failure	who died in the plane crash greys anatomy	Gold: Lexie. Baseline: I don't know. Oracle-R/Re/G: Lexie.	The baseline fails without gold evidence, while the oracle intervention recovers the exact answer.
Extraction failure	where did the first African American air force unit train	Gold: Moton Field, the Tuskegee Army Air Field. Baseline: Moton Field, Tuskegee Army Air Field, Tuskegee University, Alabama. Oracle-G: Moton Field, the Tuskegee Army Air Field.	Evidence is available, but the baseline output is overly broad, causing an exact-match failure.
Oracle-resistant failure	how many episodes are there in dragon ball z	Gold: 291. Baseline/Oracle-R/Oracle-Re/Oracle-G: 291 episodes.	All settings remain incorrect under EM due to answer-format mismatch.

In these cases, Oracle-Re and Oracle-G often recover the answer, indicating that evidence access is the primary limiting factor.

Downstream extraction failures occur even when retrieval succeeds. Here, the relevant passage is available, but the model produces an overly long or incorrectly formatted answer, causing an exact-match failure. This explains why BM25-Hit examples still benefit modestly from Oracle-G. Finally, some examples remain oracle-resistant. Numerical questions may fail an exact match because the generated answer includes an additional unit or descriptor, such as 291 episodes instead of 291. These errors are not fully attributable to retrieval or reranking. Instead, they reflect answer normalization, formatting, or metric sensitivity. Therefore, the qualitative cases support the quantitative finding that modular RAG failures are heterogeneous and are not fully characterized by average oracle gains alone.

5 Conclusion

Stage-wise oracle analysis is often summarized through aggregate gains, but such averages provide only a coarse view of bottlenecks in modular RAG. We introduced a split-based analysis approach that compares intervention gains across retrieval-outcome and question-type subsets. In a 500-sample Natural Questions study, the results indicate that bottleneck patterns are not well described by a single global trend. We also find variation across question types, with temporal queries showing larger downstream gains and numerical queries showing signs of oracle-resistant behavior. Qualitative cases illustrate three distinct error types: retrieval-driven, downstream-driven, and oracle-resistant. Within this controlled setup, the results point to different directions for improvement depending on the retrieval outcome: retrieval robustness for queries where gold evidence is missed, and answer extraction precision when retrieval already succeeds. Whether these priorities generalize to other settings remains an open question.

6 Potential Limitations

This study may have limitations. First, the experiments use a controlled mini-corpus constructed from gold supporting passages, which may yield higher BM25 recall than a full open-domain setting. Second, the experiment is limited to 500 examples from a single dataset, so the observed patterns may not generalize across datasets. Third, EM and token-level F1 are sensitive to answer-format mismatch, as shown by the oracle-resistant numerical cases. Future work should extend the analysis to additional datasets, retrievers, and reference-free evaluation frameworks such as RAGAS [6].

Acknowledgments

This work was supported by Sookmyung Women's University Research Grants (1-2503-2027).

References

- [1] Akari Asai, Matt Gardner, and Hannaneh Hajishirzi. 2022. Evidentiality-guided Generation for Knowledge-Intensive NLP Tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics*. 2226–2243.
- [2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations*.
- [3] Yonatan Belinkov and James Glass. 2019. Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics* 7 (2019), 49–72.
- [4] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 1870–1879.
- [5] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, et al. 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*.
- [6] 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*. 150–158.
- [7] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, et al. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997* (2023).
- [8] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval Augmented Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning*. 3929–3938.
- [9] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*. 874–880.
- [10] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *Proceedings of the 2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 6930–6950.
- [11] Mandar Joshi, Eunsol Choi, Daniel 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*. 1601–1611.
- [12] Ehsan Kamalloo, Charles L. A. Clarke, and Davood Rafiei. 2023. Limitations of Open-Domain Question Answering Benchmarks for Document-level Reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2123–2128.
- [13] Vladimir Karpukhin, Barlas Öğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 6769–6781.
- [14] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics* 7 (2019), 452–466.
- [15] 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*, Vol. 33. 9459–9474.
- [16] 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. In *Transactions of the Association for Computational Linguistics*, Vol. 12. 157–173.
- [17] 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*. 2383–2392.
- [18] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. In *Foundations and Trends in Information Retrieval*, Vol. 3. 333–389.
- [19] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. 10014–10037.
- [20] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective Retrieval Augmented Generation. *arXiv preprint arXiv:2401.15884* (2024).
- [21] Jianlv Yang, Shitao Xiao, Liang Zhang, Kang Leng, Zhicheng Dou, and Ting Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. *arXiv preprint arXiv:2402.03216* (2024).
- [22] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William 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*. 2369–2380.

Table 4: Main scripts and their functions.

File	Purpose
<code>run_experiments.py</code>	Runs baseline and oracle settings
<code>add_labels.py</code>	Adds question-type labels
<code>analyze_hit_miss.py</code>	Computes BM25-Hit/Miss results
<code>analyze_question_type.py</code>	Summarizes results by question type
<code>extract_cases.py</code>	Extracts qualitative cases

Supplementary Material

We use `BM25Okapi` [18] from the `rank_bm25` package as the retriever, `BAAI/bge-reranker-v2-m3` [21] as the cross-encoder reranker, and `gpt-4o-mini` as the generator. BM25 is a simple and widely used sparse retrieval baseline [18], which makes intervention effects easier to interpret without the additional training dynamics of dense retrievers such as DPR [13]. The retrieval module implements BM25 retrieval and cross-encoder reranking using the BGE reranker model, while the generation module uses `gpt-4o-mini` to produce answers. The generator is prompted in a zero-shot setting to return the shortest supported answer span, copy from the context when possible, and avoid answering when the context does not support the answer. Retrieved passages are concatenated into a single context, and the model is prompted to answer only from the provided evidence.

As shown in Table 4, the codebase is organized into four components: evaluation, preprocessing and execution scripts, core RAG modules, and oracle intervention modules. The script directory includes corpus construction, Natural Questions sample preparation, experiment runners, and analysis scripts. The oracle module implements the three intervention settings: Oracle-R, Oracle-Re, and Oracle-G. We evaluate all settings using Exact Match (EM) and

token-level F1, which are computed in the evaluation module. In the baseline pipeline, BM25 retrieves candidate passages, the cross-encoder scores each query-passage pair and reranks the candidates, and the generator produces the final answer. In Oracle-R, the gold passage is added to the retrieved candidate pool. In Oracle-Re, the candidate pool is constructed to include the gold passage, which is then forced to the first rank before generation. In Oracle-G, the gold passage is directly provided to the generator as supporting context. All settings are evaluated using EM and token-level F1 on a 500-sample subset of Natural Questions containing short-answer annotations.

For split-based analysis, we use two lightweight labeling procedures. For retrieval-outcome analysis, `analyze_hit_miss.py` labels each question using the original BM25 top-10 retrieval results before oracle intervention. A question is labeled as BM25-Hit if its stored `golden_passage` exactly appears in the retrieved list, and as BM25-Miss otherwise. The script then computes average EM and F1 for the baseline and oracle settings within each subset.

For question-type analysis, `add_labels.py` assigns surface-form labels using rule-based prefix matching. Questions beginning with *who*, *when*, *where*, *what*, or *how many/how much* are assigned to the corresponding type, while all remaining questions are labeled as Other. These labels are stored in the experiment outputs and summarized by the question-type analysis script, which reports average EM and F1 for each oracle setting by question type.

Representative qualitative cases are extracted from the experiment outputs. We select examples recovered after retrieval misses, BM25-Hit cases where the baseline still fails, and numerical cases that remain incorrect under oracle settings. These examples form the qualitative analysis in Table 3.

Code availability. The implementation is available at github.com/sungchaewon/rag-diagnostics.