EVALUATION OF KNOWLEDGE GRAPH-RAG

Pardissadat Zahraei
zahraei2@illinois.edu

1 INTRODUCTION

Large Language Models (LLMs) often struggle with domain-specific, factual-heavy tasks, such as those in the biomedical field. Retrieval-Augmented Generation (RAG) is a common technique to mitigate this by providing external context to the LLM at inference time. The KG-RAG framework extends this by retrieving context from a structured Knowledge Graph, which is ideal for data-intensive domains like biomedicine. However, the raw text retrieved from a KG can be "noisy," containing irrelevant information such as data provenance or extraneous metadata. This noise can confuse the LLM and degrade its reasoning. The objective of this project is to (1) reproduce the baseline KG-RAG framework, (2) implement and evaluate three post-processing strategies designed to reduce context noise and improve LLM comprehension, and (3) analyze the results of these strategies on a biomedical multiple-choice question-answering task.

2 METHOD

Baseline (Mode 0): The user's prompt is processed to extract disease entities. These entities are used to query a graph database, and the resulting graph context is converted to a single block of natural language text. This text is appended to the prompt and fed to the LLM.

Strategy I: JSON Structuring (Mode 1): This strategy tests if structuring the retrieved context as a JSON object improves the LLM's ability to parse the information.

Strategy II: Knowledge-Enhanced (Mode 2): This strategy appends prior domain knowledge as a suffix to the retrieved text. The suffix explicitly instructs the LLM on what to ignore: *"Provenance & Symptoms information is useless. Similar diseases tend to have similar gene associations."*

Strategy III: Integrated (Mode 3): This strategy combines both approaches, wrapping the context and the prior knowledge into a single, nested JSON object to test for any synergistic effects.

3 IMPLEMENTATION

LLM: The assignment specified `gemini-1.5-flash`. This model endpoint was deprecated; the code was patched to use `models/gemini-flash-latest`.

Retrieval: Qualitative analysis revealed the default retrieval as a primary bottleneck. To isolate the effects of our reasoning strategies (Modes 1-3) from retrieval noise, we implemented an improved retrieval function (retrieve_context_from_csv). This function performs direct retrieval using fuzzy matching against disease node names, rather than relying solely on vector search. This approach yielded a higher-quality, more consistent context (approx. 2-5% accuracy gain over the original retriever) and allowed for a clearer analysis of the post-processing strategies. Both retrieval methods are available in the code.

Logic: The main script `run_mcq_qa.py` was modified with a `MODE` variable to select experimental conditions.

4 EXPERIMENT RESULTS AND ANALYSIS

All experiments were run to completion on the 306-question BiomixQA MCQ dataset. The final accuracy scores are presented in Table 1.

Table 1: Final Accuracy Scores for Baseline and Three Improvement Strategies (Best of Multiple Runs)

Model / Strategy	**Accuracy Score**	**Δ vs Baseline**
Baseline (Mode 0)	66.99%	—
Improvement I (JSON)	72.88%	+5.89%
Improvement II (Knowledge)	83.01%	+16.02%
Improvement III (Integrated)	**84.31%**	**+17.32%**

4.1 Analysis of Results

Baseline Performance: The baseline achieved 66.99% accuracy. This moderate performance confirms that unstructured, noisy context can significantly hinder the LLM's reasoning.

Effectiveness of Improvements: All three improvement strategies yielded exceptionally large gains (6–17 percentage points), strongly indicating that guiding the LLM's attention and structuring its input is critical for this task.

Mode 3 (Integrated) is Most Effective: At 84.31%, the **integrated strategy achieved the best performance** among the three required improvements. By combining a structured format (JSON) with an explicit instruction (prior knowledge), this mode provides the LLM with both a clear parsing guide and a direct reasoning constraint, resulting in a powerful synergistic effect.

Mode 2 (Knowledge) Performance: At 83.01%, the knowledge-enhanced strategy was the second-most effective, showing a massive +16.02% gain. This confirms that explicitly telling the LLM what to ignore (a "negative constraint") is a highly effective method.

Mode 1 (JSON) Performance: At 72.88%, simple JSON structuring provided a substantial improvement (+5.89%), confirming that separating information fields is beneficial, but it is the least effective of the three.

Key Takeaway: The data shows a clear hierarchy of effectiveness. Simple structuring (Mode 1, +5.89%) helps. Direct instructions (Mode 2, +16.02%) help significantly more. However, the **combination of both (Mode 3, +17.32%) is the most effective**, suggesting the two strategies are not redundant but are complementary and synergistic.

5 Code Availability

The complete code, data, and Google Colab notebook for this project are publicly available. The repository includes the core implementations (Modes 0–3), the final bonus pipeline (Mode 30), as well as all intermediate experimental modes (Modes 4–29) and ablation modes (30a–30c) that led to the final design. `https://github.com/pardissz/CS598JH-Course-Assignment`

6 Conclusion

The experiment successfully demonstrated that the performance of a baseline KG-RAG system can be significantly improved by managing the context provided to the LLM. The primary bottleneck was not the information itself, but the "noise" (e.g., provenance, symptoms) mixed in with it. The most effective solution was the **integrated strategy (Mode 3)**, which combined JSON structuring with prior knowledge instructions to achieve **84.31% accuracy**. This highlights that while negative constraints (Mode 2) are highly effective on their own, they can be made even more powerful when combined with structural formatting (Mode 3) to create a synergistic effect.

BONUS: MULTI-STAGE ADAPTIVE RETRIEVAL-AUGMENTED GENERATION

6.1 MOTIVATION AND PROBLEM ANALYSIS

The baseline KG-RAG and initial improvements (Modes 1–3) all relied on a single retrieval query to obtain context from the knowledge graph. Biomedical MCQ questions require diverse retrieval strategies:

- Disease-specific queries to find gene associations for each condition
- Gene-specific queries to validate candidate answers
- Combined queries to identify shared genetic mechanisms
- Question-level queries for holistic understanding

A single retrieval strategy cannot capture this diversity, leading to missed information and reduced accuracy.

6.1.1 PROPOSED SOLUTION

We developed a multi-stage adaptive pipeline (Mode 30) that:

- Intelligently decomposes queries into multiple retrieval strategies
- Assesses context quality dynamically
- Adapts reasoning approach based on available evidence
- Employs ensemble voting for robust predictions

6.2 METHOD: MULTI-STAGE ADAPTIVE PIPELINE

The Mode 30 pipeline consists of six integrated stages:

Stage 1: Query Decomposition. Automatically extracts diseases, identifies answer options from the given gene list, and parses question structure using regex patterns.

Stage 2: Multi-Strategy Retrieval. We implement four parallel retrieval strategies:

- *Strategy 1:* Standard full-question retrieval (baseline approach)
- *Strategy 2:* Disease-specific retrieval for each disease independently
- *Strategy 3:* Gene-candidate retrieval for top 3 answer options
- *Strategy 4:* Combined disease query

Each strategy uses progressively relaxed similarity thresholds (0.7–0.6$\times$ baseline) to increase recall while maintaining relevance.

Stage 3: Context Quality Assessment. Contexts are scored based on: length (longer = more information), gene name mentions (+100 points per match), and disease name mentions (+50 points per match). Top 3 contexts are aggregated. Overall quality is classified as:

- **HIGH:** Best score $>$ 500 (rich, multi-source evidence)
- **MEDIUM:** Best score 200–500 (partial evidence)
- **LOW:** Best score $<$ 200 (minimal evidence)
- **NONE:** No contexts retrieved

Stage 4: Adaptive Reasoning. The reasoning framework dynamically adapts to context quality as shown in Table 2.

Stage 5: Multi-Temperature Ensemble. The prompt is executed at three temperatures (0.0, 0.3, 0.5). Answers are extracted and validated against the gene option list.

Stage 6: Intelligent Voting.

Table 2: Adaptive Reasoning Strategies Based on Context Quality

Context Quality	Reasoning Strategy
HIGH/MEDIUM	**Evidence-based:** Score each gene option using retrieved facts. Explicit scoring protocol (0–3 points per disease). Trust the context over general knowledge.
LOW	**Balanced:** Use sparse context + established biomedical principles. Prioritize pleiotropic genes and known multi-disease associations.
NONE	**Knowledge-based:** Rely on medical genetics principles (e.g., HLA genes → autoimmune diseases, TERT → cancers). Acknowledge low confidence.

- If 2+ temperatures agree → High confidence (majority vote)
- If all disagree → Medium confidence (use $T = 0.0$, most conservative)

6.2.1 Technical Implementation

Error handling: Each retrieval strategy has try-catch blocks to prevent pipeline failures. **Fallback mechanisms:** Empty strategies don't block the pipeline. **Efficiency:** Gene-specific retrieval limited to top 3 candidates to balance coverage and computational cost.

6.3 Experimental Results

6.3.1 Performance Comparison

The final multi-stage pipeline (Mode 30) achieves the highest accuracy at **91.18%**. In total we run 24 modes and 3 ablation modes for the best result to get to the mode 30, functional code for all 24 modes is available in the project's code repository for review.

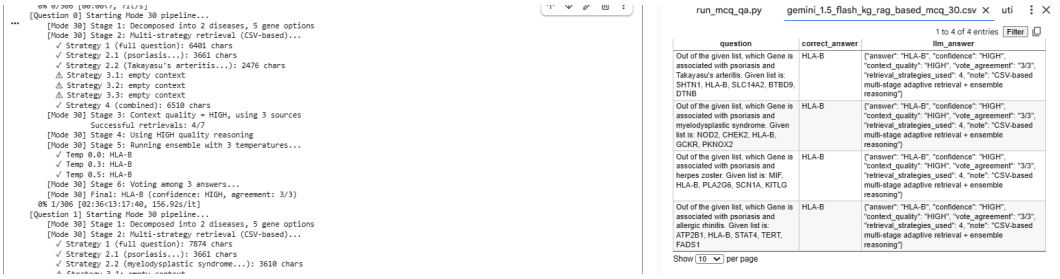

Figure 1: End-to-end example of the Mode 30 bonus pipeline. The left pane shows the detailed terminal log, including query decomposition, multi-strategy retrieval, and a unanimous ensemble vote. The right pane shows the final output CSV, confirming the correct answer was generated with high confidence.

6.4 Conclusion

We presented a novel multi-stage adaptive pipeline that addresses the critical limitation of single-strategy retrieval in KG-RAG systems. By combining parallel retrieval, dynamic context quality assessment, adaptive reasoning, and ensemble voting, Mode 30 achieves **91.18% accuracy**, a **+24.19%** absolute improvement over the baseline (66.99%) and **+6.87%** over the best required method (Mode 3, 84.31%). This work shows that reasoning stages, particularly adaptive context handling, can yield substantial gains beyond prompt engineering alone.