**RAG-based Architectures for Drug Side Effect Retrieval using compact LLMs**

Shad Nygren1,\*, Omer Erdogan1,2,3,\*,Pinar Avci1, Andre Daniels1, , Reza Rassool1, Afshin Beheshti1,4,5, Diego Galeano1,6,7+

1Kwaai, CA, USA.

2Department of Computer Engineering, Koc University, Istanbul, Turkey

3KUIS AI Center, Istanbul, Turkey

4Center for Space Biomedicine, McGowan Institute for Regenerative Medicine, Department of Surgery, University of Pittsburgh, Pittsburgh, Pennsylvania, USA.

5Broad Institute of MIT and Harvard, Cambridge, Massachusetts, USA.

6Department of Electronics and Mechatronics Engineering, Facultad de Ingeniería, Universidad Nacional de Asunción - FIUNA, Luque, Paraguay.

Tesabio.ai Inc., DE, USA

\* These authors contributed equally.

+Corresponding author: [dgaleano@ing.una.py](mailto:dgaleano@ing.una.py)

**Abstract**

Drug side effects are a major public health concern, yet off-the-shelf large language models (LLMs) are unreliable to accurately inform about reported drug side effects due to limited training data and domain gaps. We evaluate two open-book architectures that inject curated knowledge from the Side Effect Resource (SIDER 4.1) into LLM workflows: a text-based retrieval-augmented generation (RAG) pipeline and a graph-based variant (GraphRAG) over a Neo4j knowledge graph. On a balanced forward benchmark of 19,520 drug–side‑effect pairs, GraphRAG achieved 100% (Qwen‑2.5‑7B-Instruct) and 99.96% (Llama‑3.1‑8B-Instruct) accuracy; on reverse queries it returned exact drug sets with precision = recall = F1 = 100% at 0.09 s average latency (vs. text‑RAG Format B: F1 = 99.38%, 82.44 s). We also show that a lightweight LLM-based normalization step restores performance under common misspellings of drug names without modifying downstream logic. Taken together, these results indicate that integrating structured knowledge, especially graph representations, markedly improves LLM performance for drug side-effect retrieval, offering a practical path to interactive, evidence-grounded querying of catalogued drug side effect associations in larger language models.

**Introduction**

Drug side effects represent a critical global public health challenge, significantly contributing to morbidity and mortality worldwide1–4. The rapid pace of drug development often outstrips the capacity of healthcare professionals to stay abreast of new medication side effects, particularly outside their primary specialties5,6 .This issue is further complicated when patients report potential adverse reactions, requiring physicians to assess causality during time-constrained appointments. Current tools, such as drug handbooks7, electronic medical records (EMRs)8, and Spontaneous Reporting Systems (FAERS)9,10, while valuable, require time-consuming search capabilities, underscoring the urgent need for more efficient and accessible resources for assessing drug side effects in clinical practice5,11.

Large language models (LLMs)12–17 offer a promising avenue with their intuitive, conversational interfaces, illustrating the potential to streamline clinical workflows and enhance decision-making. These models support semantic search enabling the identification of drugs or diseases associated with specific symptoms18. Despite these advancements in natural language processing, the application of off-the-shelf LLMs in domain-specific tasks such as drug side effect identification has yielded mixed results19–21, frequently struggling with accuracy and reliability in specialized fields like pharmacovigilance14,21. Their limitations stem from knowledge constrained by black-box training data, a propensity for hallucinations, and a general lack of domain-specific expertise, which hinders the effectiveness of LLMs in handling nuanced medical data and generating contextually appropriate insights.

To overcome these significant challenges, we propose two open-book architectures designed to integrate domain knowledge about drug side effects into a Large Language Model (LLM): Retrieval Augmented Generation (RAG) and GraphRAG. Our first architecture utilizes RAG, which enhances LLMs by retrieving relevant information from an external Pinecone vector database—a HIPAA-compliant database —where drug side effect information is stored as feature vectors. The second architecture utilizes GraphRAG, which leverages a Neo4j graph database to store and efficiently bipartite drug side effect associations. Both frameworks incorporate custom split functions and filtering modules to optimize user prompts for accurate retrieval. Through evaluations on 19,520 drug–side-effect pairs, covering 976 marketed drugs and 3,851 MedDRA terms from the Side Effect Resource (SIDER) 4.1 database, we find that GraphRAG delivers very high accuracy for retrieving known associations under a binary yes/no formulation and under a single side effect query (reverse-query), substantially outperforming a lightweight closed-book LLM baseline and our text-based RAG variants. Within this evaluated setting, integrating structured knowledge—particularly a graph representation—markedly improves retrieval performance, providing a practical route to high-accuracy retrieval of catalogued side effects in LLM workflows.

Retrieval-augmented generation (RAG) architectures have been applied in biomedical and clinical contexts (e.g., BiomedRAG leveraging chunk-based retrieval30) as well as KG-augmented LLMs (e.g., KG-RAG31 combining the SPOKE KG with LLM prompts). Throughout this work, we distinguish closed-book LLMs—models that answer without consulting external evidence (no retrieval; responses rely solely on parametric pre-trained knowledge)—from open-book LLMs, in which the model is provided retrieved evidence at inference time. Our open-book variants include text RAG (document chunks) and GraphRAG (Neo4j knowledge graph). While prior studies (e.g., KG-Rank, RAG232, MKRAG33) explore graph- or knowledge-infused retrieval, our contribution is not to propose RAG/Graph-RAG de novo. Rather, we adapt these concepts specifically to pharmacovigilance, use SIDER 4.1 as a benchmark, and provide a unified, head-to-head comparison of data representations (free text, pairwise, graph) under the same tasks.

**Results**

**A Retrieval-Augmented Generation (RAG) framework for drug side effect retrieval**

Our RAG system was designed for seamless retrieval of drug side effects, utilizing the Side Effect Resource (SIDER) 4.1 database25 that contains drug side effect associations extracted from FDA public documents and drug package inserts. We filtered the database to include drugs with known Anatomical, Therapeutic, and Chemical (ATC) classification and side effects categorized as MedDRA Preferred Terms (PT). Filtering yielded a dataset of 141,209 associations, linking 1,106 marketed drugs to 4,073 unique side effect terms (**Fig. 1a**). Due to the large number of these associations, utilizing the complete dataset for comprehensive evaluation would have been computationally prohibitive. For this assessment, we created a balanced subset of 19,520 drug-side effect pairs, as detailed in the **Methods** section.

To facilitate text-based retrieval, the raw SIDER dataset was processed into two distinct text formats (**Fig. 1b**). "Text Format A" provides a structured, comma-separated list of all known side effects for a given drug (e.g., "The drug metformin may be associated with the following side effects or adverse reactions: shock, peptic ulcer, contusion, …"). In contrast, "Text Format B" presents each drug-side effect pair on a new line, enhancing granularity (e.g., "The drug metformin may cause urticaria as an adverse effect, adverse reaction, or side effect.").

For the RAG pipeline (**Fig. 1c, d**), Text Format A was segmented into chunks using a custom algorithm that splits text at new lines. These chunks were then embedded into a 1,536-dimensional vector space using the OpenAI ada002 embedding model, chosen for its capacity to support up to 8,192 tokens, which is sufficient for even the longest text chunks in Format A (exceeding 10,000 characters). The resulting embeddings were indexed in a Pinecone vector database, enabling rapid similarity-based retrieval.

The RAG query workflow operates as follows: an end-user query (e.g., "Is urticaria an adverse effect of aspirin?") is first embedded using the OpenAI ada002 model and then compare it against the top five most similar entries in the Pinecone database. Concurrently, an LLM-based entity recognition module extracts drug and side effect terms (e.g., “metformin” and “urticaria”) from the query prompt. A subsequent filtering module checks if the identified drug-side effect pair from the query exists within the top five retrieved results. Based on this check, a modified prompt is generated: if a match is found, the prompt states that the side effect has previously been reported for the query drug; otherwise, it specifies that the drug is not known to be associated with the side effect. Our modified prompt has the following structure:

“You are asked to answer the following question with a single word: YES or NO. Base your answer strictly on the RAG Results provided below. After your YES or NO answer, briefly explain your reasoning using the information from the RAG Results. Do not infer or speculate beyond the information provided. Question:\n\n" + query + rag\_results

The variable rag\_results contains the result from RAG. For instance, it can be:

"No, the side effect " + side\_effect\_query + " is not listed as an adverse effect, adverse reaction or side effect of the drug " + drug\_query

Where the drug and side effect query are the terms extracted using the entity recognition module.

The modified prompt is then passed to a compact LLM model (Llama‑3.1‑8B‑Instruct or Qwen‑2.5‑7B‑Instruct) which generates a binary YES/NO response. This binary output was specifically chosen because our evaluation framed drug side effect retrieval as a binary classification problem.

****

**Figure 1. Evaluation of drug side effect retrieval using RAG with compact LLM models. (a)** 19,520 drug-side effect associations (covering 976 drugs and 3,851 side effect terms) were extracted from the SIDER 4.1 database; (b) Drug side effect associations were stored in two text formats: format A (drug to side effect list) and format B (drug-side effect pair); (c) Retrieval Augmented Generation (RAG) to extract drug side effect information from text format A and B; (d) Evaluation procedure. The user writes prompt queries, and we run three different language models (RAG on data A, RAG on data B) to retrieve drug side effect associations from the Pinocone Vector database. Each model is assessed based on its binary response. LLM icon source: https://lobehub.com/icons, MIT license.

**Graph-Based Retrieval Augmented Generation (GraphRAG) for Drug Side Effect Data**

In our GraphRAG framework, drug-side effect associations are precisely modeled as a bipartite graph-based representation, leveraging the extensive SIDER 4.1 database previously described. Within this structure (Fig. 2a), drugs and side effects constitute distinct nodes, and their known relationships are encoded as directed edges, specifically labeled as "may\_cause\_side\_effect". This graph is implemented within a Neo4j database, a robust graph management system that facilitates efficient querying via Neo4j’s query language, Cypher, enabling rapid traversal and retrieval of complex drug-side effect relationships.

The GraphRAG system is designed to process user queries, such as “Is headache an adverse effect of metformin?” (**Fig. 2b**). The workflow begins with an entity recognition module that extracts drug and side effect terms (e.g., “metformin” and “headache”) from the submitted query. These extracted entities are then used to construct a precise Cypher query, showcased with an example below:

cypher = f"""

MATCH (s)-[r:may\_cause\_side\_effect]->(t)

WHERE s.name = 'metformin' AND t.name = 'headache'

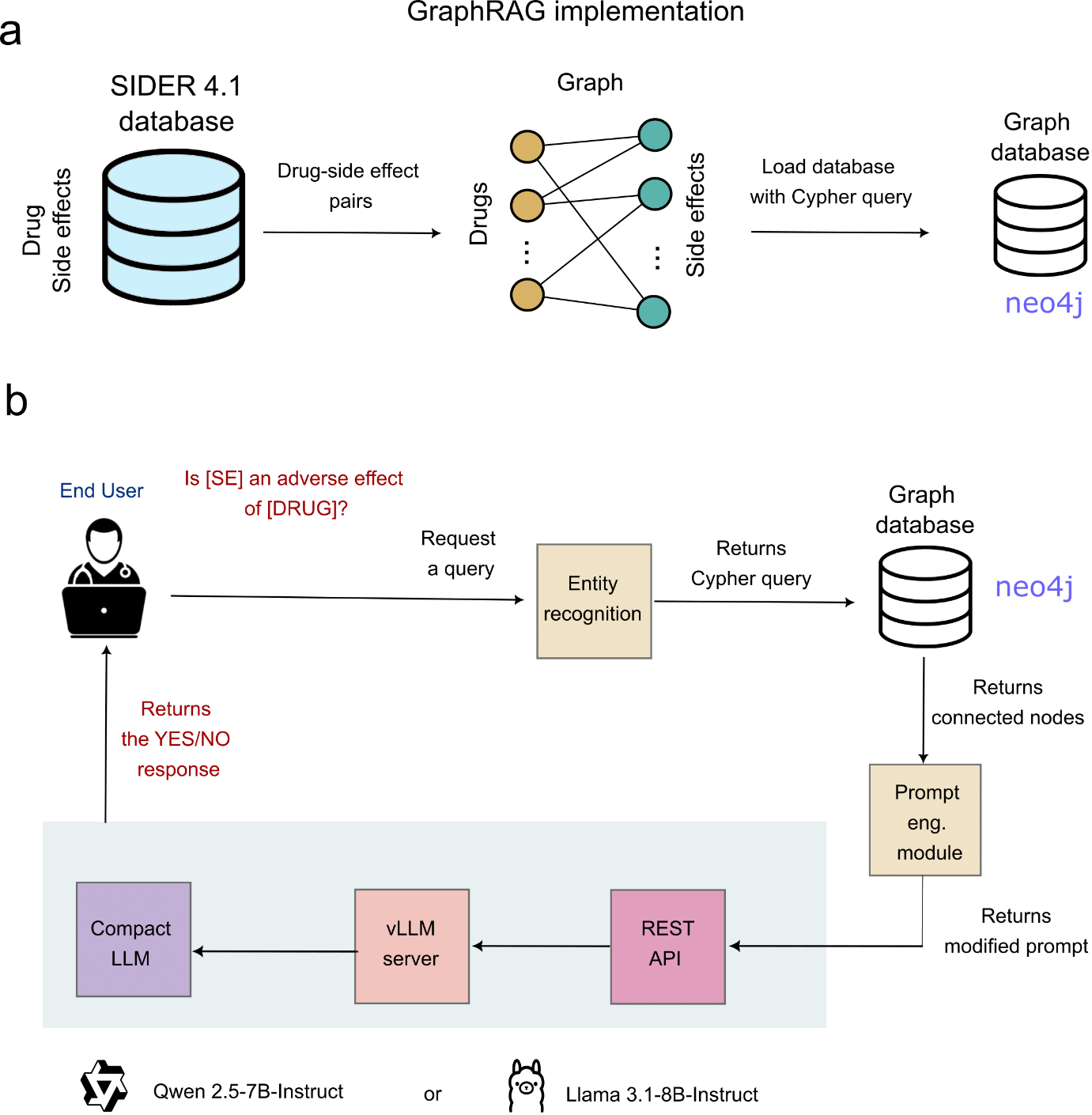
RETURN s, r, t

"""

Which is executed against the Neo4j database. This query efficiently identifies the presence or absence of a direct edge between the specified drug and side effect nodes, returning matching associations or an empty result accordingly.

A prompt engineering module then processes the retrieved results to generate a context-specific input for the language model. If a match is found, the prompt is modified to state, “Metformin is known to be associated with headache as a side effect”. Conversely, if no association is found, the prompt states, “Metformin is not known to be associated with headache as a side effect”. This prompt modification strategy is identical to that employed in our RAG architecture, ensuring a consistent approach to informing the language model. This refined prompt is fed into a lightweight LLM (Llama‑3.1‑8B‑Instruct or Qwen‑2.5‑7B‑Instruct) model, which generates a binary YES/NO response. As with our RAG framework, this binary output is a deliberate choice, aligning with our evaluation's formulation of drug side effect retrieval as a binary classification problem to predict the presence or absence of a drug side effect. The system is served via vLLM with a lightweight REST API layer that orchestrates Neo4j graph lookups, supporting low-latency, scalable real-time queries.

This GraphRAG approach offers distinct advantages over traditional text-based retrieval methods for pharmacovigilance. By representing drug-side effect associations as a bipartite graph, it enables exact, relationship-driven queries that significantly reduce ambiguity and enhance retrieval accuracy. Its integration with Neo4j facilitates complex traversals, such as identifying group of drugs associated with a query side effect, while the use of the LLM ensures future escalation to larger reasoning models that will provide user-friendly responses with more medical context.



**Figure 2 - GraphRAG framework for drug side effect retrieval.** (a) The SIDER 4.1 database, which contains drug-side effect pairs, is transformed into a graph structure where drugs (orange nodes) and side effects (green nodes) are connected by "may\_cause\_side\_effect" edges. It is then loaded into a Neo4j graph database using Cypher queries. (b) Workflow of the GraphRAG system: an end-user submits a query (e.g., "Is [SE] an adverse effect of [DRUG]?"), which undergoes entity recognition to extract drug and side effect terms. A Cypher query retrieves matching associations from the Neo4j database, and a prompt engineering module refines the input for a lightweight LLM model to generate a binary YES/NO response. LLM icon source: https://lobehub.com/icons, MIT license.

**Performance evaluation for forward queries: from drug-side effect pair to binary answer**

To quantify performance on drug–side-effect retrieval, we evaluate whether the model’s final output to multiple single drug queries is correct against SIDER-derived ground truth. We constructed a balanced evaluation set by sampling 10 positives (documented MedDRA Preferred Terms) and 10 negatives (MedDRA terms not linked to that drug) for each drug with at least 10 known associations, yielding 19,520 pairs across 976 drugs and 3,851 side-effect terms. We compared four architectures: (i) Closed-book LLM (LLM-only, no retrieval from any database), (ii) Open-book LLM — RAG (Format A: drug→list of side effects), (iii) Open-book LLM — RAG (Format B: drug-side effect pairs per sentence), and (iv) Open-book LLM — GraphRAG (drug-side effect bipartite graph), using two compact models (Qwen-2.5-7B-Instruct and Llama-3.1-8B-Instruct). Execution flow for each method is illustrated with an example in **Supplementary Figures 1, 2, 3** and **4**.

For our evaluations, we used two small/compact LLMs because we test at large scale (many thousands of queries) and the task is binary retrieval with ground-truth evidence, so we do not require long-form reasoning from the model; using compact models (Qwen-2.5-7B-Instruct and Llama-3.1-8B-Instruct) reduces cost and latency per query while preserving accuracy once retrieval is in place. We report the accuracy, F1, precision, sensitivity, and specificity of the LLM’s emitted YES/NO under each architecture. The results are reported in **Tables 1, 2,** and **3**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Architecture type** | **Accuracy** | **F1 Score** | **Precision** | **Sensitivity** | **Specificity** |
| **Closed-book LLM (no retrieval)** | 62.90% | 0.494 | 0.776 | 0.363 | 0.895 |
| **Open-book LLM — RAG (Format A: drug→list)** | 86.67% | 0.858 | 0.919 | 0.805 | 0.928 |
| **Open-book LLM — RAG (Format B: pairs)** | 96.50% | 0.967 | 0.936 | 0.999 | 0.931 |
| **Open-book LLM — GraphRAG** | 100.00% | 1 | 1 | 1 | 1 |

**Table 1**: Comparative results (balanced SIDER subset; single-drug, binary queries) using Qwen2.5-7B.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Architecture type** | **Accuracy** | **F1 Score** | **Precision** | **Sensitivity** | **Specificity** |
| **Closed-book LLM (no retrieval)** | 63.21% | 0.534 | 0.728 | 0.422 | 0.842 |
| **Open-book LLM — RAG (Format A: drug→list)** | 84.54% | 0.819 | 0.987 | 0.7 | 0.991 |
| **Open-book LLM — RAG (Format B: pairs)** | 95.86% | 0.96 | 0.924 | 0.999 | 0.918 |
| **Open-book LLM — GraphRAG** | 99.96% | 1 | 1 | 1 | 1 |

**Table 2:** Comparative results (balanced SIDER subset; single-drug, binary queries) using Llama 3.1-8B-Instruct.

|  |  |  |  |
| --- | --- | --- | --- |
| **Architecture type** | **Qwen2.5-7B-Instruct** | **Llama-3.1-8B-Instruct** | **Difference/winner** |
| **Closed-book LLM (no retrieval)** | 62.90% | 63.21% | Llama +0.31% |
| **Open-book LLM — RAG (Format A: drug→list)** | 86.67% | 84.54% | Qwen +2.13% |
| **Open-book LLM — RAG (Format B: pairs)** | 96.50% | 95.86% | Qwen +0.64% |
| **Open-book LLM — GraphRAG** | 100.00% | 99.96% | Qwen +0.04% |

**Table 3-** LLM accuracy comparison for different architectures.

The results show three trends. First, structured augmentation matters: both RAG formats and GraphRAG substantially outperform a pure LLM. Second, data representation matters: pairwise (Format B) is consistently stronger than list-style (Format A). Third, GraphRAG reaches the deterministic-lookup accuracy ceiling (Qwen-2.5-7B-Instruct: 100.00%; Llama-3.1-8B-Instruct: 99.96%). The tiny deviation from 100% reflects rare LLM mislabels (hallucinations) at the output step, not retrieval failures; locking the binary label to the retrieval outcome would close this gap while retaining the LLM for the explanation.

To further investigate whether the underperformance without data augmentation extends to significantly larger models, we also evaluated ChatGPT 3.5 and ChatGPT 4.0 on a subset of 51 randomly selected drugs in a closed-book setting. We observed a mean accuracy of approximately 55% for ChatGPT 3.5 and 63% for ChatGPT 4. This demonstrates that even advanced, larger language models struggle to accurately identify drug side effects for marketed drugs without specialized augmentation.

Because the binary decision is determined upstream by retrieval/filtering, a rule-basedswitch (match→“YES”, no-match→“NO”) would emit the same labels under this task. In our study, the LLM is retained by design to provide a natural-language interface and to keep a single interface that can scale to more expressive queries. Larger LLMs would be more appropriate for future extensions that require reasoning (e.g., class-level queries, conflict resolution across sources, lay summaries with nuance, severity/frequency synthesis, or multi-hop causal narratives).

**Performance evaluation for reverse queries: from side effect to drug set**

In the previous section, we assessed the forward case: given a *(drug, side-effect)* pair, the system provides a binary YES/NO answer, and we evaluate the correctness of the LLM’s final label. We now turn to the complementary, reverse case: given a side-effect term, return the set of drugs known to be associated with it in SIDER. This shifts both user intent (“Which drugs cause side effect *X*?” vs. “Does drug *D* cause side effect *X*?”) and evaluation: instead of a single label, the output is a set, so we measure precision, recall, and F1 over the returned drug lists, and we track latency because reverse queries can fan out to large result sets. Execution flow is illustrated with an example in **Supplementary Figures 5** and **6**.

To obtain these results, we constructed a stratified benchmark of side-effect terms spanning four tiers: rare (5-19 drugs), small (20-99 drugs), medium (100-499 drugs), and large (500+ drugs) drug sets. We then randomly sample from each group for a total of 121 queries. Ground truth was derived from the same SIDER-based database used in the forward task. We then compared three open-book variants: RAG (Format A: drug to SE list), RAG (Format B: pairs), and GraphRAG (Neo4j graph). For each method, we executed the reverse query end-to-end and computed recall (coverage of true drugs), precision (correctness of returned drugs), F1, and average latency (query start → set materialized). **Table 4** reports macro-averages across tiers; a per-tier breakdown (precision/recall/F1 and latency vs. set size) is provided in **Supplementary Table 1** and specific examples per tier are shown in **Supplementary Figure 7**.

**Table 4** - Values are macro-averaged across small/medium/large tiers.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Architecture** | **Recall** | **Precision** | **F1** | **Avg Latency** | **Throughput** |
| **Open-book LLM — GraphRAG (Neo4j)** | 100.00% | 100.00% | 100.00% | 0.09 s | 11 q/s |
| **Open-book LLM — RAG (Format B: pairs)** | 98.88% | 99.93% | 99.38% | 82.44 s | 0.012 q/s |
| **Open-book LLM — RAG (Format A: drug→list)** | 7.97% | 80.91% | 11.84% | 23.42 s | 0.043 q/s |

These results highlight a clear pattern. GraphRAG performs a single indexed Cypher expansion to enumerate all connected drugs and therefore achieves exact coverage with near-instant latency, effectively matching a deterministic lookup ceiling on this structured graph database. RAG (Format B) approaches perfect F1 but slows dramatically as the number of matching drugs increases due to retrieving and aggregating many pairwise snippets (**Supplementary Table 1**). RAG (Format A) under-retrieves because list-style chunks are vulnerable to windowing/chunking limits. Practically, for prompts like *“Which drugs cause [SE]?”* we recommend GraphRAG as the default reverse-query backend on SIDER-derived graphs, with text-RAG reserved for scenarios involving unstructured evidence or multi-source fusion.

**Misspelling Robustness and LLM-Assisted Normalization**

In practical applications, user queries may contain misspelled drug names (e.g., floxetine for fluoxetine). Our current open-book pipelines are brittle under such misspellings, because the input string does not match a canonical name in our databases. To keep the core architectures unchanged and preserve their strengths, we introduce a lightweight LLM-based entity-normalization layer that operates between the user query and retrieval.

Concretely, upon receiving a user question, a compact LLM performs drug name correction/normalization to a canonical SIDER entry before any retrieval step. The normalized *(drug, side-effect)* is then passed to the same back-end used in the forward or reverse tasks (open-book RAG or GraphRAG). This simple guardrail resolves spelling noise while preserving closed-book/open-book behavior and avoiding changes to downstream logic.

To test this, we built a small database of 10 commonly misspelled drug names previously reported in the literature (**Supplementary Table 2**). Our results indicate that our Qwen 7B spell correction module achieves 80% accuracy in this small set. We then run our whole evaluation procedure for 9 out of the 10 drugs that we could map to our database. The results are shown in **Table 5**. Our Qwen 7B spell correction achieved ~88% recovery for both architectures, transforming a catastrophic 100% degradation into a manageable 12% degradation. This could be improved using larger LLMs.

|  |  |  |
| --- | --- | --- |
| **Condition** | **RAG Format B (F1 score)** | **GraphRAG**  **(F1 score)** |
| Original architecture + no misspelled drugs | 0.9474 | 1.0000 |
| Original architecture + misspelled drugs | 0.0000 | 0.0000 |
| LLM-based Spelling normalization + misspelled drugs | 0.8333 | **0.8750** |

**Table 5** – Comparison of different model/conditions to handle drug name misspelling.

**Data availability**

All data generated or analyzed during this study are available in the Github link <https://github.com/apicurius/drugRAG/tree/main/data/processed>

**Code availability**

The data and code used in our study are available here: <https://github.com/apicurius/drugRAG/tree/main>

**Funding**

This project was supported by Virtual Hipster Corporation.

**Acknowledgments**

This project was supported by Virtual Hipster Corporation and Kwaai.

**Author Contribution**

D.G. conceived the study and supervised the project. S.N, O.E. and D.G. designed the RAG frameworks. S.N. and O.E. implemented software architecture and ran experiments. P.A., A.D., and A.F. provided critical review and guidance on the medical and clinical aspects of the study. R.R. provided methods for analysis. D.G., P.A. and A.D. prepared the draft of the manuscript. All authors reviewed, edited, and approved the final manuscript.

**Discussion**

We studied two ways of injecting curated knowledge from SIDER 4.1 into LLM workflows: a text‑based RAG pipeline and a graph‑based variant (GraphRAG). Using a large, balanced benchmark derived from SIDER, GraphRAG consistently reached the empirical ceiling for yes/no questions about catalogued drug–side‑effect pairs, and in reverse queries (side effect to drug set) it enumerated the exact set. Text RAG performed well but became slower as the number of matching drugs increased. Together, these results indicate that structured knowledge—particularly a graph representation—provides a reliable path to accurate, interactive retrieval when a curated reference resource is available.

Our aim was not to introduce RAG or Graph‑augmented LLMs de novo but to adapt them to pharmacovigilance and to compare representation formats under the same tasks and prompts. Framing the problem this way makes clear that performance is governed primarily by how knowledge is represented and retrieved, not by parametric scale alone. The graph schema, with explicit drug and side‑effect nodes linked by a single relation, reduces ambiguity and enables exact membership checks and set expansion with a simple query, while preserving a natural‑language interface for end users.

In the present design, correctness is determined upstream by retrieval and filtering; the LLM is retained to standardize the output and provide a concise, evidence‑grounded rationale. This separation keeps the interface flexible for richer interactions (e.g., reverse queries now, class‑level or multi‑source questions later) while allowing a “label‑locking” guardrail when absolute fidelity is required. We used compact models to support exhaustive evaluation at low cost and latency; larger models can be swapped in when queries demand more reasoning or synthesis.

Our scope is intentionally narrow: we evaluate retrieval of previously catalogued associations within a closed set. The work does not address discovery of new adverse events, causal inference, or bias correction in spontaneous reporting10,27,28. We also focus on single‑drug forward queries and their reverse; class‑based questions require additional ontology integration and dedicated benchmarks. Finally, exact‑match stages are brittle to misspellings and brand–generic variation; a lightweight pre‑retrieval normalization layer mitigates this without altering downstream logic. We did not include a fine‑tuned classifier baseline (e.g., BioBERT/RoBERTa trained on 141k SIDER pairs), prioritizing retrieval‑based methods that leverage a complete knowledge store without memorization. While a supervised classifier could perform well on frequent pairs, it would likely trail RAG/GraphRAG on coverage and maintainability; we leave a systematic comparison to future work.

Looking ahead, the same interface can be extended along two axes. First, query breadth: class‑level prompts, ranked effect sets by frequency or severity, and multi‑hop graph explanations. Second, data breadth: integrating spontaneous reporting systems and literature via vector indices alongside the graph to surface emerging or rare signals. These extensions keep deterministic retrieval for fidelity and use the LLM to present evidence in language clinicians and patients can use.

**Methods**

**Overview**

We developed and evaluated two open-book architectures for drug–side-effect retrieval with large language models (LLMs): (1) text RAG over SIDER-derived documents and (2) GraphRAG over a Neo4j knowledge graph. We also include a closed-book LLM (no retrieval) baseline. Unless specified, all LLM inference was zero-shot (no fine-tuning) and served using vLLM.

Terminology. “Closed-book LLM (no retrieval)” = model answers from parametric memory only. “Open-book LLM” = model receives retrieved evidence at inference time (RAG or GraphRAG).

**Data source and preparation**

We used the Side Effect Resource (SIDER) 4.125, which compiles adverse events from randomized trials and post-marketing surveillance. We retained drugs with an Anatomical, Therapeutic, and Chemical (ATC) code and side effects mapped to MedDRA Preferred Terms (PTs), yielding 141,209 associations linking 1,106 marketed drugs to 4,073 PTs.

**Evaluation subset (forward YES/NO)**

For forward binary evaluation, we constructed a balanced set as follows: for each drug with ≥10 PTs we sampled 10 positives (documented PTs) and 10 negatives (PTs not linked to that drug absent from the knowledgebase). This produced 19,520 pairs spanning 976 drugs and 3,851 PTs. The full SIDER-derived graph (141,209 edges) was used for reverse-query ground truth.

**Knowledge representations**

We derived three representations of SIDER for retrieval (Fig. 1b, Fig. 2a):

1. **Format A (drug→list).** A per-drug narrative listing all PTs.
2. **Format B (pairs).** One sentence per (drug, PT) pair.
3. **Graph.** A bipartite Neo4j graph with nodes = {Drug, SideEffect(PT)} and directed edges :may\_cause\_side\_effect.

**Entity recognition module**

For the architectures, we extract drug and side effect names from the retrieved context using a two-stage procedure: LLM-based extraction with temperature of 0.1 followed by regex-based parsing (to remove prefixes, parse comma-separated, parse lists, filter and remove duplicates if needed).

**Text RAG framework**

**Indexing.** Format A documents were chunked at newlines and embedded (OpenAI text-embedding-ada-002, 1,536-D); embeddings were stored in Pinecone for similarity search. We also indexed Format B sentences.

**Querying.** A user question (e.g., “Is urticaria an adverse effect of aspirin?”) is embedded and top-k results are retrieved. A lightweight entity recognizer extracts (drug, PT) from the question. A filter checks whether the extracted pair appears in the retrieved items.

**Prompting.** Based on that check, we create a concise instruction/context stating either “the drug is known to be associated with the PT” or “is not known …”, then ask the LLM for a binary YES/NO plus a one-sentence rationale grounded in the retrieved text.

**GraphRAG framework**

**Graph database.** We loaded the SIDER bipartite graph into Neo4j.

**Querying.** For an input (drug, PT) we execute an exact Cypher match, e.g.:

**MATCH (d:Drug {name:$drug})-[:may\_cause\_side\_effect]->(s:SideEffect {name:$pt}) RETURN count(\*)>0 AS exists.**

**Prompting.** The result conditions a short instruction (“known/not known”), and the LLM outputs a YES/NO plus a one-sentence rationale.

**Decision layer and role of the LLM**

The binary label is determined by retrieval (text filter or Cypher result). The LLM formats the final YES/NO and concise explanation. We constrained outputs to binary for evaluation comparability across representations. Where absolute fidelity is required, we can lock the label to the retrieval outcome and let the LLM generate only the rationale.

**Models and inference setup (vLLM)**

We used two public instruction-tuned small models without any additional training:

* Llama-3.1-8B-Instruct (Meta; 8B) — used as-is via vLLM.
* Qwen-2.5-7B-Instruct (Alibaba; 7B) — used as-is via vLLM.

**Serving.** All inference ran on vLLM (https://github.com/vllm-project/vllm), configured with PagedAttention for high-throughput serving. We used temperature = 0.0, top-p = 1.0; max output tokens were 32 for forward YES/NO and 2,000 for list extraction experiments. Decoding parameters and prompts are provided in the code repository (Code availability). For the misspelling study, Qwen-2.5-7B-Instruct also served as a pre-retrieval normalizer (spell/alias correction) with a compact JSON output schema; still zero-shot.

**Performance evaluation (forward YES/NO)**

We measured whether the LLM’s final YES/NO matched the SIDER-derived ground truth on the 19,520 balanced pairs. This binary framing is in line with prior work on drug side‑effect prediction22, 23.We compared: (i) Closed-book LLM (no retrieval), (ii) Open-book RAG (Format A), (iii) Open-book RAG (Format B), and (iv) Open-book GraphRAG. As a theoretical upper bound, a deterministic lookup over SIDER (or the graph) yields 100% if entities are canonicalized; residual errors in open-book runs arise from the generative step when not label-locked.

**Metrics.** Performance was measured across standard binary classification metrics. For a binary classification task, these metrics are defined using the following terms:

* **True Positives (TP):** Correctly predicted positive instances.
* **True Negatives (TN):** Correctly predicted negative instances.
* **False Positives (FP):** Incorrectly predicted positive instances (Type I error).
* **False Negatives (FN):** Incorrectly predicted negative instances (Type II error).

The formulas for each metric are as follows:

* **Accuracy:** Measures the proportion of correctly classified instances out of the total instances.

​

* **F1 Score:** The harmonic mean of precision and sensitivity, providing a balance between them.

​

* **Precision:** Measures the proportion of true positive predictions among all positive predictions. It answers: "Of all instances predicted as positive, how many are actually positive?"
* **Sensitivity (Recall):** Measures the proportion of true positive predictions among all actual positive instances. It answers: "Of all actual positive instances, how many were correctly identified?"
* **Specificity:** Measures the proportion of true negative predictions among all actual negative instances. It answers: "Of all actual negative instances, how many were correctly identified?" Specificity=TN+FPTN​

**Reverse-query benchmark (PT → drug set)**

We evaluated prompts of the form “Which drugs cause [PT]?” using the full SIDER-derived graph as ground truth for each PT. For each architecture, we computed precision/recall/F1 over the returned drug set and latency (query start → set materialized).

* **GraphRAG:** single Cypher expansion enumerates all connected drugs (exact coverage; sub-second latency).
* **Text RAG (Format B):** retrieves and aggregates many pairwise snippets (high F1; latency grows with set size).
* **Text RAG (Format A):** prone to under-retrieval due to chunking/window limits.

**Misspelling robustness and LLM-assisted normalization**

Because user inputs can contain spelling variants, we inserted a lightweight LLM-based spell-correction/normalization step before retrieval. A compact instruct model is prompted to map the drug mention to the canonical SIDER drug name (generic form) and the corrected query is then passed unchanged into the existing RAG or GraphRAG pipelines. We evaluated this by re-running the forward YES/NO benchmark on a misspelling set derived from our main evaluation and comparing raw misspelled versus normalized inputs for each architecture. This normalization is zero-shot (no fine-tuning) and leaves all downstream components and decision rules unchanged, while substantially restoring performance for exact-match stages.

**Orchestration and systems**

* **Vector search:** Pinecone (Format A/B embeddings).
* **Graph store:** Neo4j (Cypher endpoints).
* **LLM serving:** vLLM for all models; API layer implemented with a lightweight service (e.g., FastAPI) coordinating retrieval → decision → LLM explanation.
* **Reproducibility:** fixed prompts, decoding params (temperature=0.0, top-p=1.0), and logged seeds where applicable.

**Statistical analysis and reporting**

We report point metrics on the balanced forward set and macro-averages for reverse queries; tier-wise breakdowns and per-query examples are provided in the Supplementary. Confidence intervals are omitted because evaluation is deterministic given fixed prompts/decoding and static ground truth; variability stems mainly from retrieval recalls (text RAG) and output length constraints in list extraction settings.

**References**

1. Naghavi, M. *et al.* Global, regional, and national age-sex specific mortality for 264 causes of death, 1980–2016: a systematic analysis for the Global Burden of Disease Study 2016. *The Lancet* **390**, 1151–1210 (2017).

2. Sunshine, J. E. *et al.* Association of adverse effects of medical treatment with mortality in the United States: A secondary analysis of the Global Burden of Diseases, Injuries, and Risk Factors Study. *JAMA network open* **2**, e187041–e187041 (2019).

3. Haerdtlein, A. et al. Which Adverse Events and Which Drugs Are Implicated in Drug‑Related Hospital Admissions? A Systematic Review and Meta‑Analysis. *J. Clin. Med.* 12, 1320 (2023).

4. Ernst, F. R. & Grizzle, A. J. Drug-related morbidity and mortality: updating the cost-of-illness model. *J Am Pharm Assoc (Wash)* **41**, 192–199 (2001).

5. Ziegler, D. K., Mosier, M. C., Buenaver, M. & Okuyemi, K. How much information about adverse effects of medication do patients want from physicians? *Arch Intern Med* **161**, 706–713 (2001).

6. Rothenberger, D. A. Physician Burnout and Well-Being: A Systematic Review and Framework for Action. *Dis Colon Rectum* **60**, 567–576 (2017).

7. *Handbook of Clinical Drug Data*. (McGrawHill, Medical Publ. Division, New York, 2002).

8. Tayefi, M. *et al.* Challenges and opportunities beyond structured data in analysis of electronic health records. *WIREs Computational Statistics* **13**, e1549 (2021).

9. Montastruc, J.-L. Pharmacovigilance and drug safety: Fair prescribing and clinical research. *Therapie* **77**, 261–263 (2022).

10. Hazell, L. & Shakir, S. A. W. Under-reporting of adverse drug reactions : a systematic review. *Drug Saf* **29**, 385–396 (2006).

11. Mao, J. J. *et al.* Online discussion of drug side effects and discontinuation among breast cancer survivors. *Pharmacoepidemiol Drug Saf* **22**, 256–262 (2013).

12. Chang, Y. *et al.* A Survey on Evaluation of Large Language Models. *ACM Trans. Intell. Syst. Technol.* **15**, 39:1-39:45 (2024).

13. Thirunavukarasu, A. J. *et al.* Large language models in medicine. *Nat Med* **29**, 1930–1940 (2023).

14. Landman, R. *et al.* Using large language models for safety-related table summarization in clinical study reports. *JAMIA Open* **7**, ooae043 (2024).

15. Singhal, K. *et al.* Large language models encode clinical knowledge. *Nature* **620**, 172–180 (2023).

16. Golan, R., Reddy, R. & Ramasamy, R. The rise of artificial intelligence-driven health communication. *Transl Androl Urol* **13**, 356–358 (2024).

17. Van Veen, D. *et al.* Adapted large language models can outperform medical experts in clinical text summarization. *Nat Med* **30**, 1134–1142 (2024).

18. Low, Y. S. et al. Answering real‑world clinical questions using large language model retrieval‑augmented generation and agentic systems. *Digital Health* 11, 20552076251348850 (2025).

19. Wang, X., Xu, X., Liu, Z. & Tong, W. Bidirectional Encoder Representations from Transformers-like large language models in patient safety and pharmacovigilance: A comprehensive assessment of causal inference implications. *Exp Biol Med (Maywood)* **248**, 1908–1917 (2023).

20. Liu, Z. *et al.* AI-based language models powering drug discovery and development. *Drug Discov Today* **26**, 2593–2607 (2021).

21. Sun, Z., Pergola, G., Wallace, B. C. & He, Y. Leveraging ChatGPT in Pharmacovigilance Event Extraction: An Empirical Study. Preprint at https://doi.org/10.48550/arXiv.2402.15663 (2024).

22. Galeano, D., Li, S., Gerstein, M. & Paccanaro, A. Predicting the frequencies of drug side effects. *Nat Commun* **11**, 4575 (2020).

23. Galeano, D. & Paccanaro, A. Machine learning prediction of side effects for drugs in clinical trials. *Cell Rep Methods* **2**, 100358 (2022).

24. Martin, L., Hutchens, M., Hawkins, C. & Radnov, A. *How Much Do Clinical Trials Cost?* (Nature Publishing Group, 2017).

25. Kuhn, M., Letunic, I., Jensen, L. J. & Bork, P. The SIDER database of drugs and side effects. *Nucleic acids research* **44**, D1075–D1079 (2015).

26. Ke, Y. *et al.* Development and Testing of Retrieval Augmented Generation in Large Language Models -- A Case Study Report. Preprint at https://doi.org/10.48550/arXiv.2402.01733 (2024).

27. Tandon, V. R., Mahajan, V., Khajuria, V. & Gillani, Z. Under-reporting of adverse drug reactions: a challenge for pharmacovigilance in India. *Indian J Pharmacol* **47**, 65–71 (2015).

28. García-Abeijon, P. *et al.* Factors Associated with Underreporting of Adverse Drug Reactions by Health Care Professionals: A Systematic Review Update. *Drug Saf* **46**, 625–636 (2023).

29. Onakpoya, I. J., Heneghan, C. J. & Aronson, J. K. Post-marketing withdrawal of 462 medicinal products because of adverse drug reactions: a systematic review of the world literature. *BMC medicine* **14**, 10 (2016).

30. Li, M., Kilicoglu, H., Xu, H. & Zhang, R. BiomedRAG: A retrieval augmented large language model for biomedicine. *J Biomed Inform* **162**, 104769 (2025).

31. Soman, K. *et al.* Biomedical knowledge graph-optimized prompt generation for large language models. *Bioinformatics* **40**, btad614 (2024).

32. Sohn, J. *et al.* Rationale-Guided Retrieval Augmented Generation for Medical Question Answering. Preprint at <https://arxiv.org/abs/2411.00300> (2025).

33. Shi, Y. *et al.* MKRAG: Medical Knowledge Retrieval Augmented Generation for Medical Question Answering. Preprint at <https://arxiv.org/abs/2309.16035> (2024).