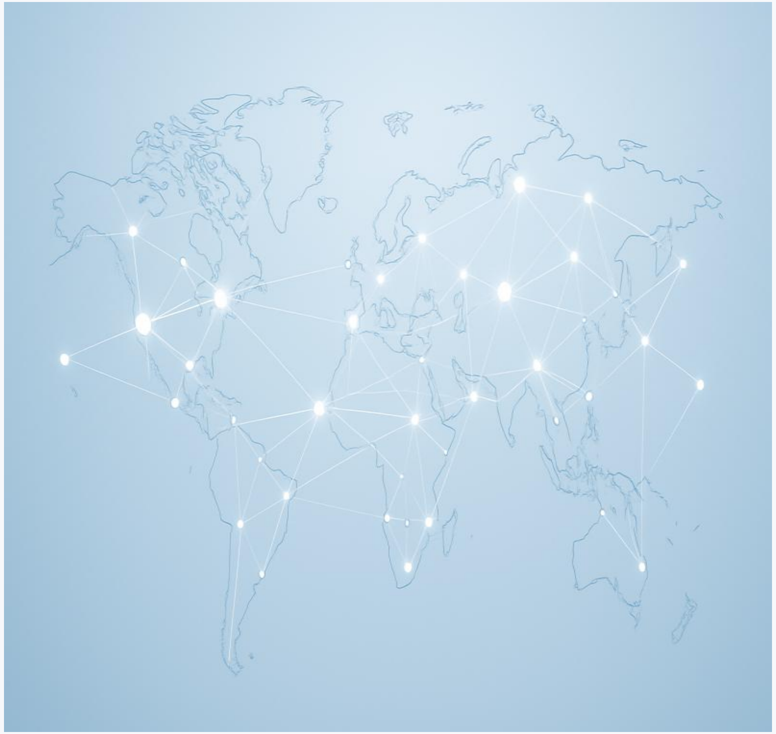
GDELT Knowledge Base

Intelligent Retrieval & RAG Evaluation

Learning to navigate complex events with multi-strategy retrieval

[1]

What is GDELT?



Real-time global record

• GDELT aggregates news reports every 15 minutes, capturing events around the world from news, blogs and social media.
• It records information on people, organizations, locations, themes and emotions, offering a snapshot of the world's events.

Applications

• Event monitoring and crisis detection.
• Risk assessment and predictive analytics.
• Social science research and trend analysis.

[16]

GDELT KG Purpose & Use

From GKG2 to Graph

• GDELT's Global Knowledge Graph (GKG2) is stored in multiple relational tables linking events, mentions and articles.
• Converting these tables into a proper knowledge graph requires an ontology that mirrors the relational schema.

Limitations of Text-Only RAG

• Standard RAG pipelines using unstructured text miss global relationships embedded in the event graph.
• KG-based retrieval lets LLMs ingest relational context, enabling richer reasoning over events and actors.

A Typical Use Case for GDELT Data

Leverage GDELT events and the derived knowledge graph to augment an agentic RAG system of global event data.

By integrating graph-based retrieval strategies, your pipeline captures relationships between actors, locations and themes, providing deeper context for question answering.

[17] [18]

Problem & Audience

Massive & heterogeneous knowledge graph
Millions of global events with varying granularity require flexible retrieval strategies.

Need for adaptable RAG pipelines
Comparing dense, sparse, hybrid & reranked retrievers helps identify optimal approaches.

Audience
AI engineers, data scientists & solution architects exploring knowledge-graph RAG pipelines.

[2] [3]

Solution & Technology Stack

Five-Layer Architecture

Execution: LangGraph server, ingestion & evaluation scripts
Orchestration: LangGraph workflows & state management
Retrieval: Factory of Naive, BM25, Ensemble & Cohere Rerank
Data: Qdrant vector store & HF datasets (38 docs, 12 QA pairs)
Config & External: GPT-4.1 LLM, text-embedding-3-small, Cohere rerank v3.5

Vector Store
Qdrant (Docker)

LLM
OpenAI gpt-4.1-mini

Embeddings
OpenAI text-embedding-3-small

Reranker
Cohere

[4] [5]

Data Ingestion & Preparation

12-page PDF
Raw GDELT paper

38 Documents
Page-level chunks

HF Datasets
Sources & testset

12 QA Pairs
Synthetic evaluation set

[6] [7]

Layered Architecture

Execution
Ingestion & evaluation scripts, LangGraph Platform server, LangChain Document loaders

Orchestration
LangGraph builders & state schemas

Retrieval
Factory: Naive, BM25, Ensemble, Cohere Rerank

Data
Qdrant store, HF datasets & manifest

Config & External
LLM, embeddings, reranker & environment

[8] [9]

Retrieval Strategies & Factory Pattern

Strategy	Approach	Details
Naive	Dense vector search	Embeddings + cosine similarity (k=5)
BM25	Sparse keyword	TF-IDF & in-memory index (k=5)
Ensemble	Hybrid 50/50	Weighted merge of dense & sparse
Cohere Rerank	Contextual compression	Wide retrieval (k=20) → rerank top-3

Factory Pattern Example

```
def create_retrievers(documents, vector_store, k=5):

return {

"naive": vector_store.as_retriever(search_kwargs={"k": k}),

"bm25": BM25Retriever.from_documents(documents, k=k),

"ensemble": EnsembleRetriever(
            retrievers
=[vector_store.as_retriever(search_kwargs={"k": k}),
BM25Retriever.from_documents(documents, k=k)],
            weights
=[0.5, 0.5],

),

"cohere_rerank": ContextualCompressionRetriever(
            base_retriever
=vector_store.as_retriever(search_kwargs={"k": 20}),
            compressor
=CohereRerankCompressor(model="rerank-v3.5", top_n=3),

),

}
```

[10] [11]

Evaluation Pipeline & Metrics

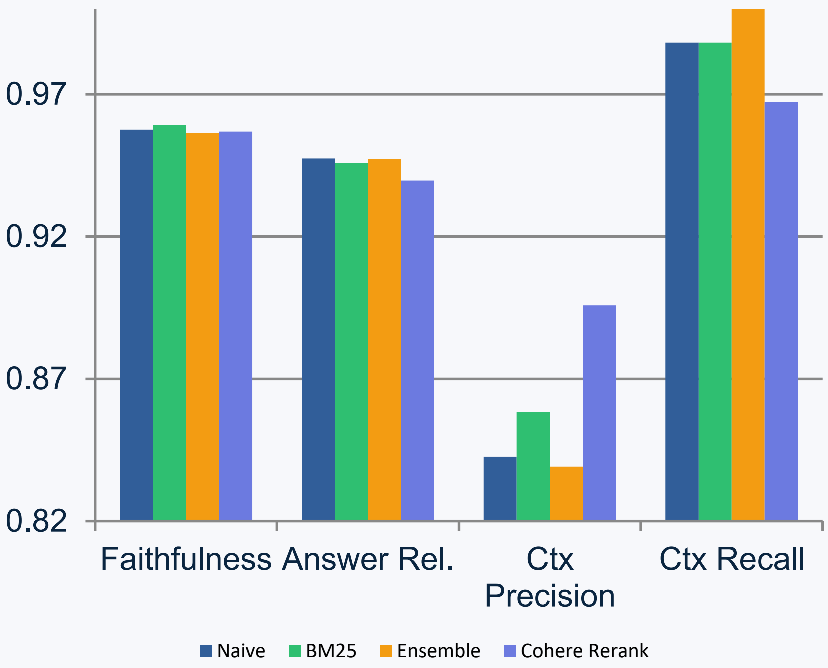
Data Loading
Load source docs & golden test set from HuggingFace

Build RAG Stack
Vector store → retrievers → graphs

Inference
Run 12 questions across 4 retrievers

Evaluation
Compute four RAGAS metrics per run

Analysis
Aggregate scores & rank strategies



[12] [13]

Performance & Next Steps

Naive	**0.926**
BM25	**0.932**
Ensemble	**0.923**
Cohere Rerank	**0.944**

Findings

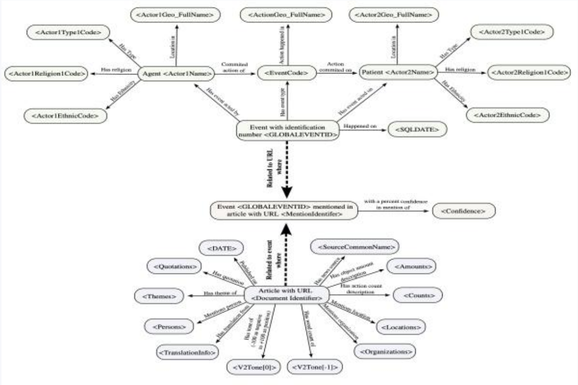
- Cohere Rerank achieves the highest average due to enhanced context precision.
- BM25 improves precision over Naive but slightly lowers answer relevancy.
- Ensemble balances dense & sparse but does not surpass extremes.

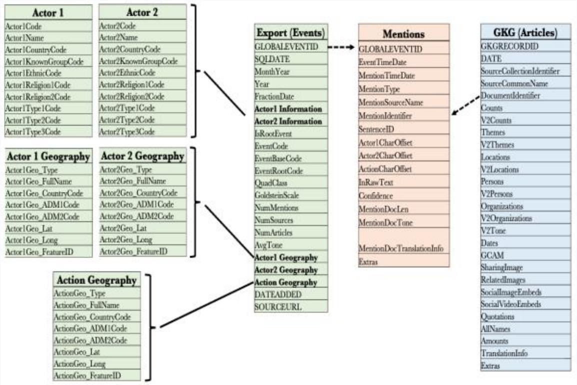
Next Steps

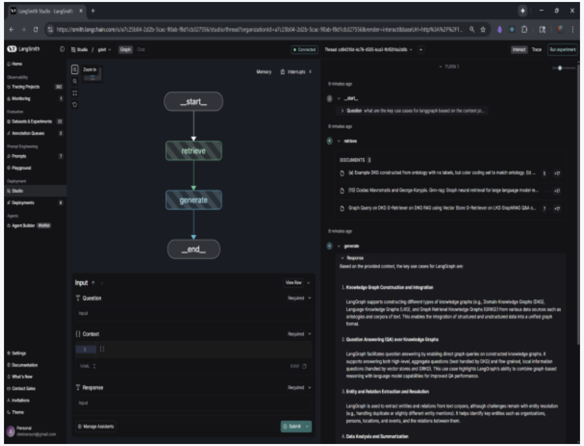
- Dynamic weighting & meta-learning to adapt retrieval strategy per query.
- Graph-based retrieval exploiting event relations for deeper context.
- Integrate graph neural rankers & cross-lingual embeddings.
- Evaluate on additional domains to stress-test generalisability.

[14]

Anchors & Transparency







Ontology (not integrated currently)

Mapping relational tables to entities & relationships enables KG-based retrieval with richer context.

Schema (not integrated currently)

Relational structure of actors, events, geographies, mentions & articles forms the basis for graph conversion.

Interactive UI

LangGraph Studio visualises retrieval pipelines and encourages experimentation via chat and live panels.

Datasets: sources | testset | eval metrics | eval inputs
Reproducible workflow:
make process executes modular steps; manifests track provenance & checksums, ensuring transparency. Explore metrics in the interactive Hugging Face Datasets SQL console.

[19] [20]

Conclusions & Reflections

Key Takeaways

- Multi-layer architecture promotes modularity & separation of concerns.
- Retriever factory enables experimentation & easy comparison of dense, sparse, hybrid & reranked methods.
- RAGAS metrics (faithfulness, answer relevancy, context precision & recall) provide holistic evaluation.
- Manifest-driven reproducibility ensures deterministic results & data lineage.

Reflection

This study showcases a production-grade RAG system built atop GDELT documentation and papers. The architecture and evaluation methodologies can generalise to learning how to more effectively use other knowledge graphs, fostering rigorous experimentation and observability. Future work may explore Semantic Chunking, Parent Document retrieval methods, GraphRAG techniques, dynamic retrieval selection and broader multi-lingual datasets.

For a deeper dive:

- deliverables
- architecture
- Teaching Claude Code how to `make` it from scratch
- GDELT papers and documentation