Beyond Text: Advanced Retrieval Augmented Generation for Complex and Multimodal Data

Liang Zhao (Emory) and Chao Huang (HKU)

Retrieval Augmented Generation

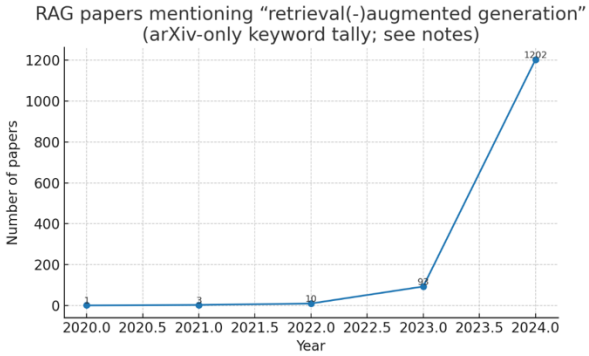
Limits of Large Foundation Models:

- Staleness: weights can’t reflect post-training facts or fast-changing data.
- Coverage: long-tail entities, niche jargon, private docs rarely memorized.
- Precision: numbers, tables, and step-by-step procedures are brittle when recalled from memory.
- Verifiability: hard to cite sources or trace reasoning to evidence.
- Hallucinations and noisy/contradictory sources.

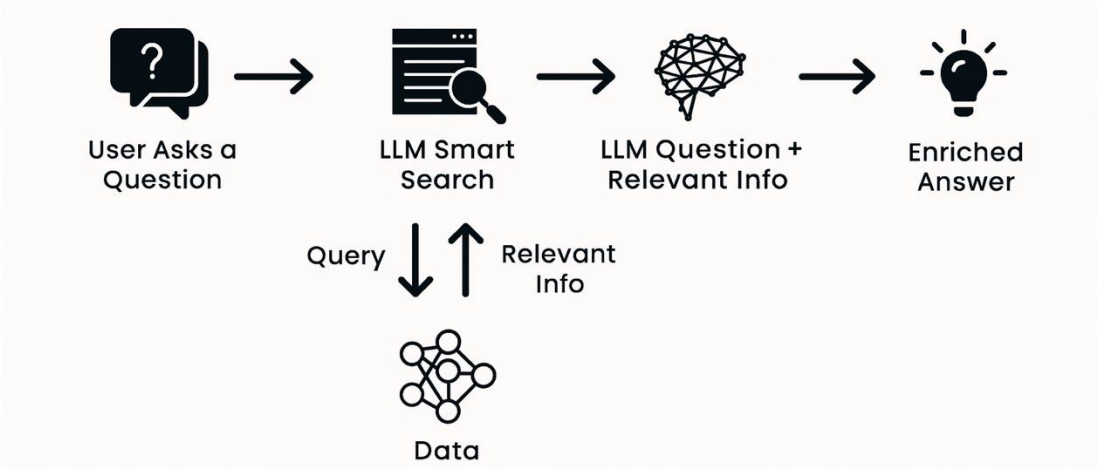
Retrieval Augmented Generation:

A method where a retriever fetches relevant external evidence at inference time and a generator (LLM) conditions on it to produce grounded, factual answers.

Trend of Research Papers in Arxiv



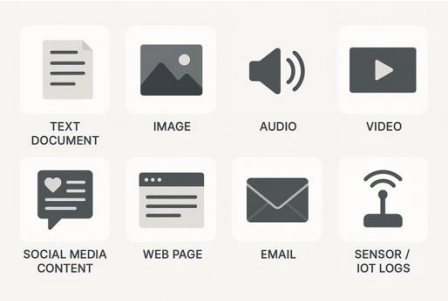
Naive Retrieval Augmented Generation Schematic

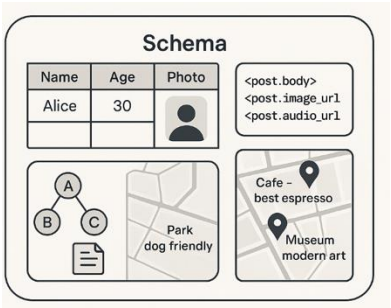


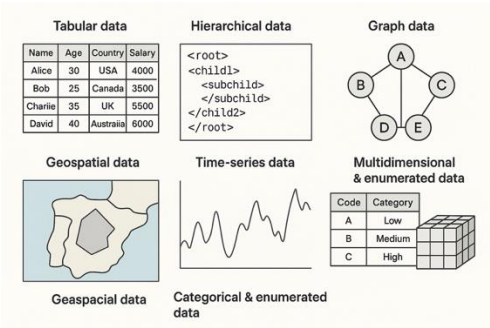
- Information is **flattened text** where proximity ≈ relevance.
- **Single-hop** retrieval suffices (no joins, no aggregations, no structure).
- Top-k chunks are **self-contained** and order doesn't matter.
- …

Challenges in Handling Complex Data

The reality is the external data can be:
unstructured, semi-structured, and structured formats from different modalities





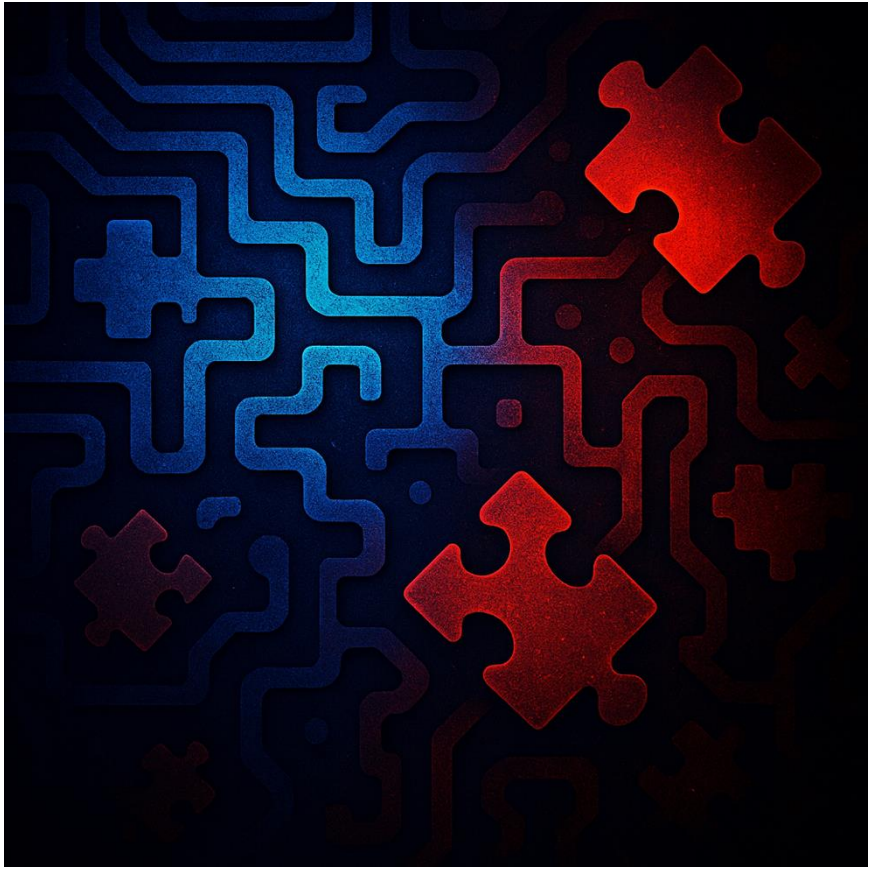


The design of the workflow and components of RAG is highly data-dependent.

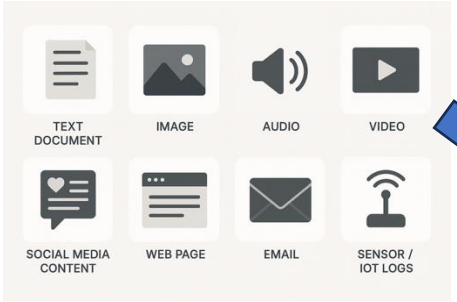
- How to retrieve unstructured vs structured vs different modalities?
- How to fuse multi-modal data?
- How to generate and reason using data under diverse types?
- …

Techniques for Semi-Structured Data Retrieval-Augmented Generation

IJCAI 2025
AI for Semi-Structured Data

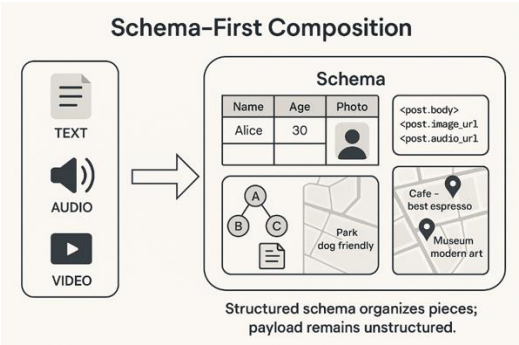


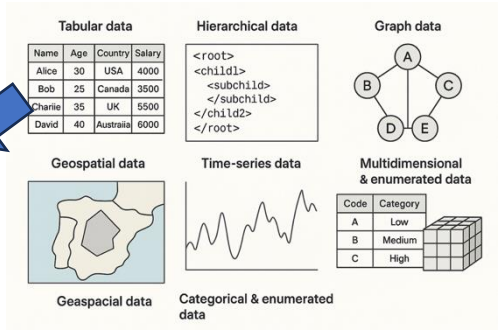
What are semi-structured data



fast & expressive but hard to manage

Wrap content with **minimal, self-describing structure** (IDs, types, timestamps, links) → keep the **payload free-form**.





Definition: data not conform to rigid schemas (like relational tables) yet retains *some* organizational structure through tags, keys, or other markers[1].

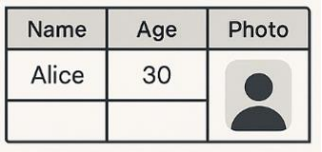
Ubiquitous – raw data is unstructured, to manage which we need to add some structure in some level.

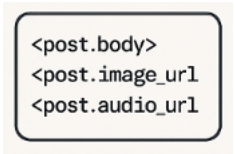
Key characteristics: (1) **Flexibility** – different records may have varying sets of fields or structure; (2) **Implicit schema** – the structure is defined by the data itself (through tags or positions) rather than an external schema; (3) **Partial consistency** – some constraints or regularities exist (e.g. certain tags always present) but not to the extent of a strict schema[6][7].

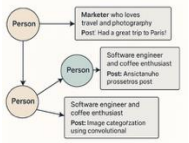
Motivation for Using Semi-Structured Data in RAG

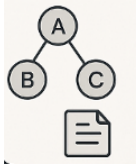
Aspect	Purely Structured RAG	Semi-Structured RAG	Purely Unstructured RAG
Schema Flexibility	Rigid schema; high precision, low adaptability	Flexible schema; balances structure and flexibility	No schema; very flexible but less precise
Ease of Query	High via SQL or API calls	Moderate—keys, tags, or JSON-aware parsing often needed	Low—requires embedding and semantic search
Context & Narrative	Limited to table fields	Rich context from embedded text alongside structured data	Excellent context; all narrative, but lacks facts
Handling Complex Queries	Strong for aggregations/joins, weak for context	Balanced—can support analytics and narrative together	Weak support for accurate numeric or relational data
Ingestion Overhead	Low for native databases, but rigid structure	Moderate—may need lightweight parsing	High—requires crawling, chunking, embedding
Enterprise Integration	Strong—internal database-friendly	Very strong—handles logs, API outputs, embedded tables	Moderate—needs full documents indexed first

Popular Semi-structured Data and their RAGs









- **Table-based** (Relational, Time Event, JSONB, etc.)
- **Structured-document based** (HTML/DOM, XML/JSON, PDF, etc.)
- **Text Graph-based** (Citation/Web, Social Networks, E-commerce Net, etc.)
- **Hybrid-based** (structured data sources + unstructured data sources)

Table-Based Retrieval-Augmented Generation

1. End-to-End Neural Retrieval + Reading

These combine a *dense table retriever* with a *neural reader/generator*, trained jointly so retrieval is optimized for downstream QA.

- **T-RAG (Pan et al. 2022)** — Jointly fine-tunes a dense retriever and a seq2seq generator (BART) so that retrieved tables maximize answer correctness.

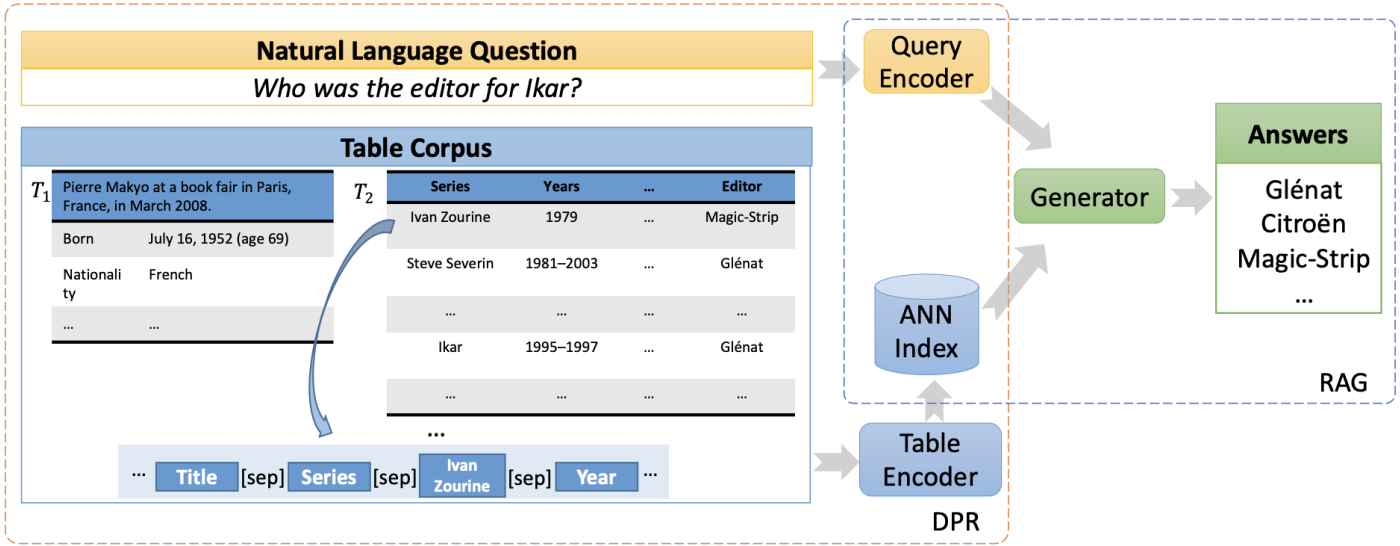


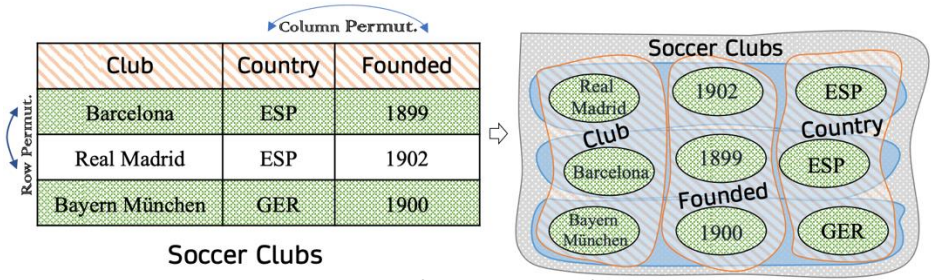
Table-Based Retrieval-Augmented Generation

2. **Structure-Aware Table Encoding**

Type 1: Flattened + Positional Embeddings:

•Example: **TAPAS** — Linearizes the table into a sequence and uses special embeddings for row/column IDs to let transformers "see" table structure.

TAPAS (Herzig et al., 2020)
TaBERT (Yin et al., 2020)
TAPEX (Liu et al., 2021)



HyTREL (Chen et al.)

Type 2: Graph/Hypergraph Encodings:

•Example: **HyTREL** — Represents a table as a hypergraph (cells as nodes, hyperedges for row/column/table groupings) and uses GNNs to produce permutation-invariant embeddings.

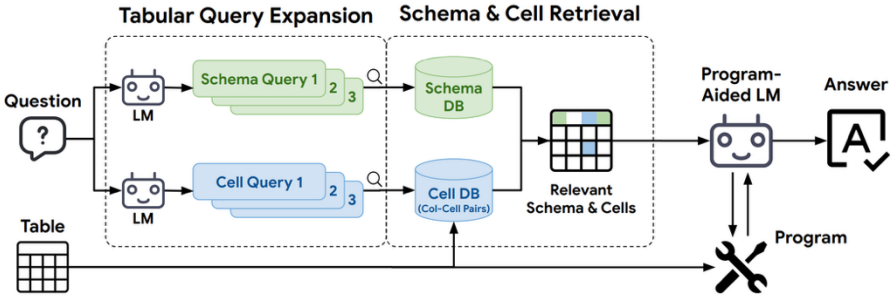
HyTREL (Chen et al., NeurIPS 2023)
TURL (Deng et al., 2020) — relational table representation

Table-Based Retrieval-Augmented Generation

3. Programmatic / Tool-Augmented

Rely on an LLM's reasoning capability to *generate programs or queries* that operate on structured data, rather than reading tables directly as text.

- **SQL Generation and Execution:**
  - Example: **LOTUS** (Patel et al., 2024) — LLM produces a SQL query for a database, executes it, and integrates results into the final answer.
- **API- or Tool-Driven Reasoning:**
  - Example: **TableRAG** (Chen et al. 2024) - LLM calls a table API to fetch specific rows/columns, then reasons over the returned subset.



TableRAG

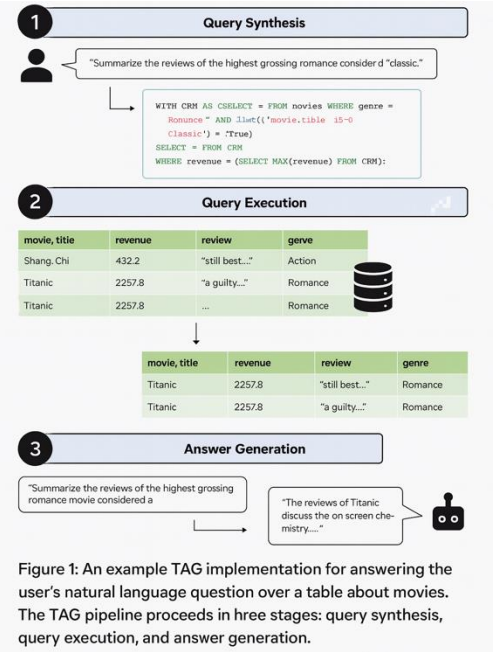


Figure 1: An example TAG implementation for answering the user's natural language question over a table about movies. The TAG pipeline proceeds in hree stages: query synthesis, query execution, and answer generation.

LOTUS (Patel et al., 2024)

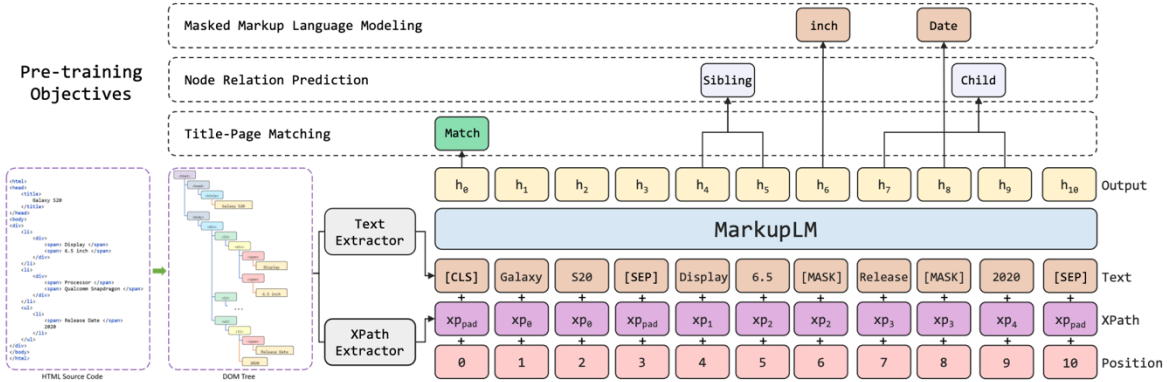
Structured Document-based: Dense Retrieval

Structure-Preserving Representation (at the encoding stage)
Encode hierarchy via linearization with structural markers or via graph/tree encoders that follow DOM/JSON/XML relationships.

Examples:

- **MarkupLM**

(Li et al., 2022)



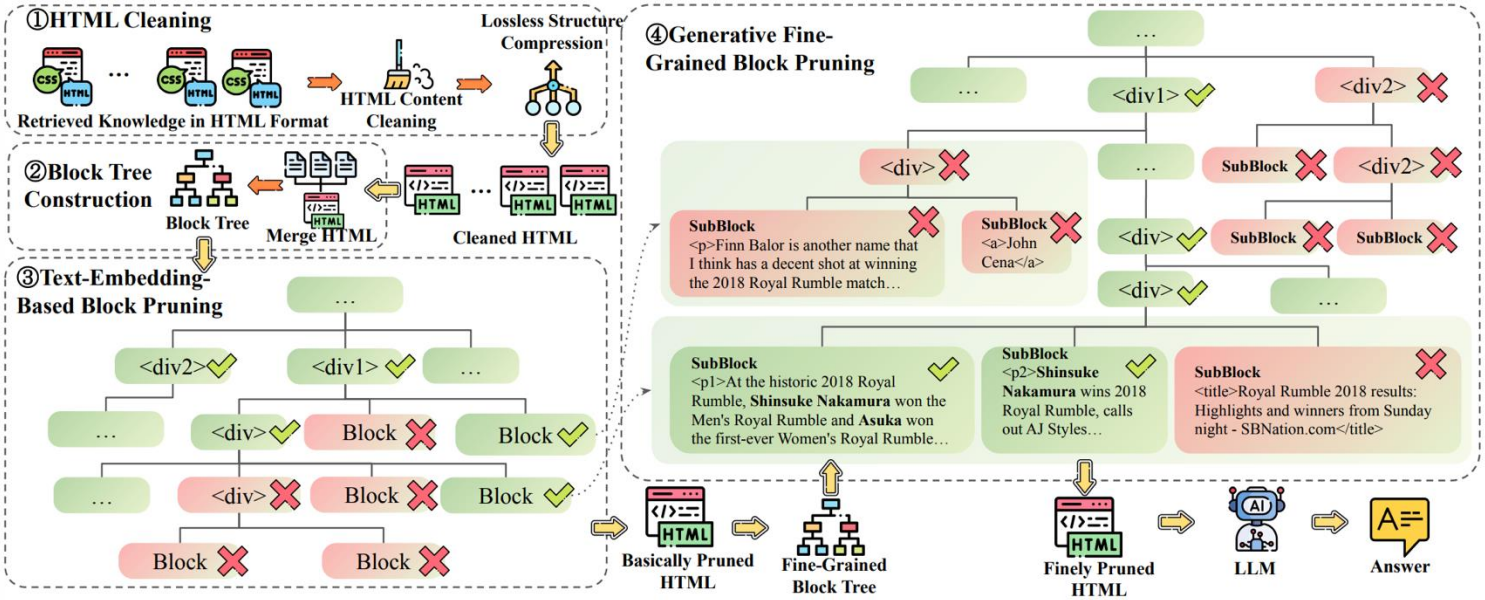
- **DOM-LM (Deng et al., 2022)** – encoding both text and DOM tree structure with a transformer-based encoder and learning generalizable representations for HTML documents.
- **Gur et al. (2023)** – LLM fine-tuning on HTML linearization, leveraging latent structure learned from pretraining.
- **StructFormer (Wang et al., 2022)** – Structure-aware transformer encoding tree-structured data.

Structured Document-based: Sparse Retrieval

Structure-Aware Retrieval (at the Retrieval Stage)

Index at schema-aligned units (fields, DOM elements, XML subtrees) and combine semantic search with symbolic filtering (e.g., city="Atlanta").

- **Example: HtmlRAG (Tan et al., 2025)**



Structured Document-based: Generation

3. Tool-Augmented Structured Navigation *(at the reasoning stage)*

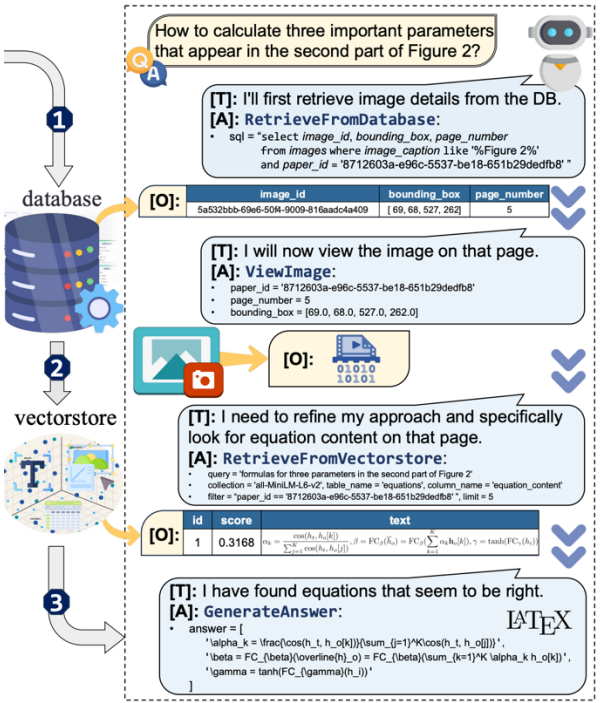
Exploit structure **after retrieval**, during reasoning, by explicitly traversing or querying the document.

- **How it works:** LLM generates structured queries (XPath, JSONPath, SQL) and calls deterministic parsers or search functions.
- **Example:**

NeuSym-RAG (Cao et al., 2025): organizes semi-structured PDF content into both the relational database and vector store, enabling LLM agents to iteratively gather context until sufficient to generate answers.

Contrato360 (Seabra et al., 2024)

WebAgent (Gur et al., 2024).

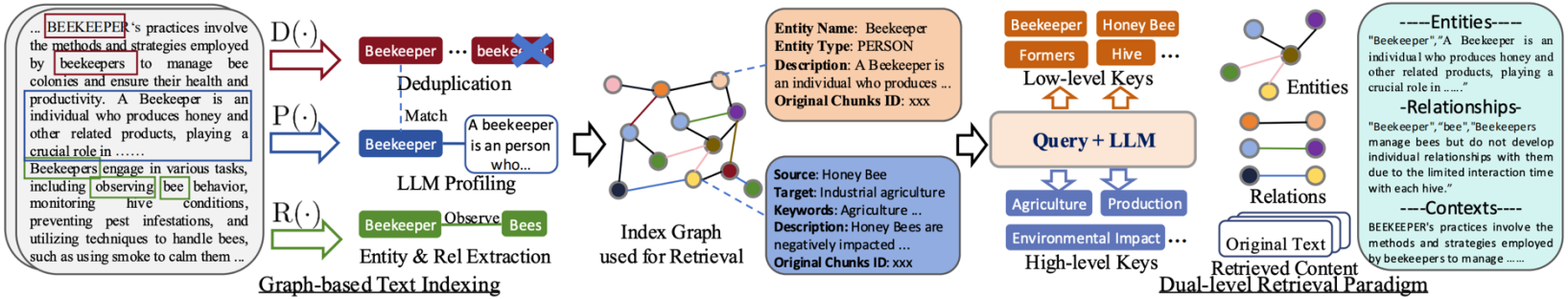


Text Graph: Graph-indexed Texts

First build an indexing graph from documents, and then use the graph to guide document retrieval.

Examples:

1. **GraphRAG (Edge et al., 2024)**
2. **LightRAG (Guo et al., 2025)**



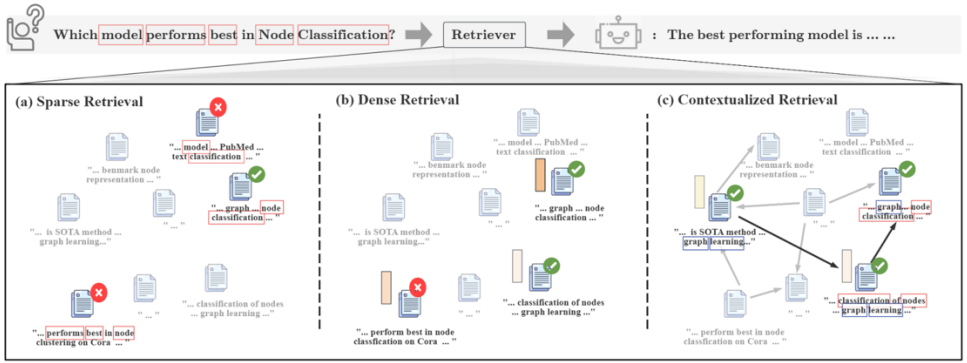
Text Graph: Text-attributed Graphs: Retriever Side

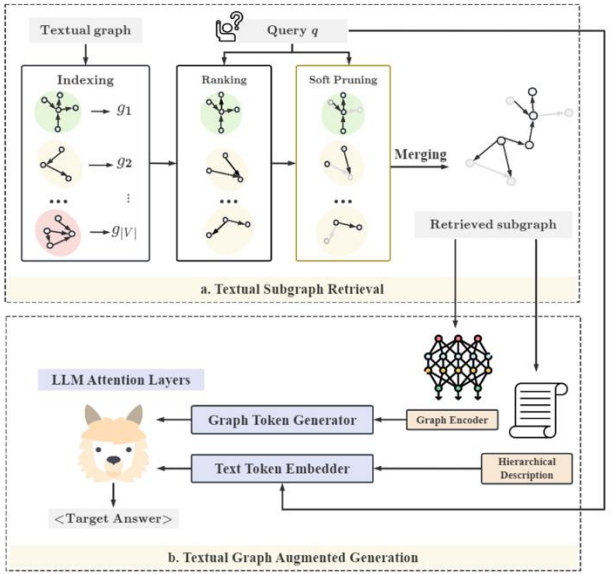
Text-attributed Graphs are graphs whose nodes (and edges) are textual, such as citation graphs, social networks, and e-commerce networks

Nodes (and edges) of the graph are authentically (rich) texts. Use both the graph structure and graph text to decide which part to retrieve.

Examples:

1. **GRAG (Hu et al., 2025)**
2. **CG-RAG (Hu et al., 2025)**



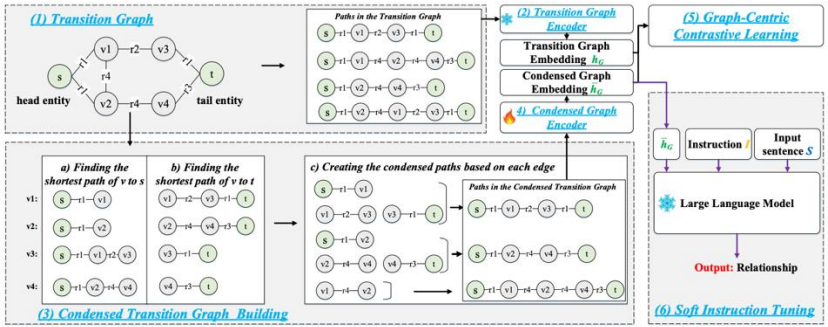


Text Graph: Text-attributed Graphs: Generator Side

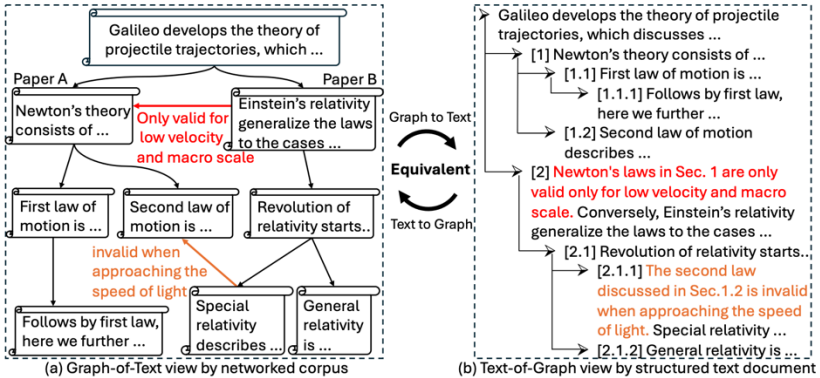
Generator side:

Use the retrieved subgraph to prompt LLMs soft prompt vs hard prompt

1. **Hard prompt:**
   - **TAGA (Zhang et al., 2025)**
2. **Soft prompt:**
   - **CTLP (Li et al., 2024)**



CTLP (Li et al., 2024)

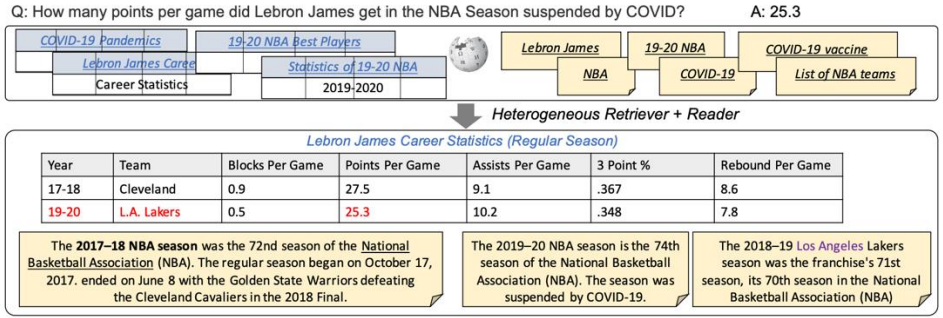


TAGA (Zhang et al., 2025)

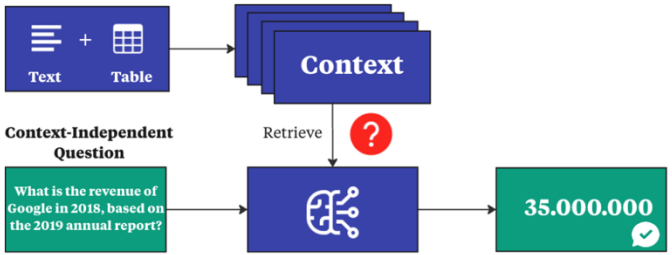
Hybrid: Fusion-in-Retriever

Combine structured and unstructured data *before or during* retrieval so the system returns fused evidence in one shot.

- **Document linking:** OTT-QA (Chen et al., 2021) **–** pre-combined table + text chunks into single retrievable units, e.g., fuse a table segment and relevant passages into a group.
- **Hybrid Sparse+Dense Retrieval:** *T2-RAGBench* (2025) – BM25 + dense embeddings for text+table queries.
- **Unified textualization for heterogeneous sources (UniK-QA)** — converts tables/KBs/lists to text so a single retriever can operate across all sources.



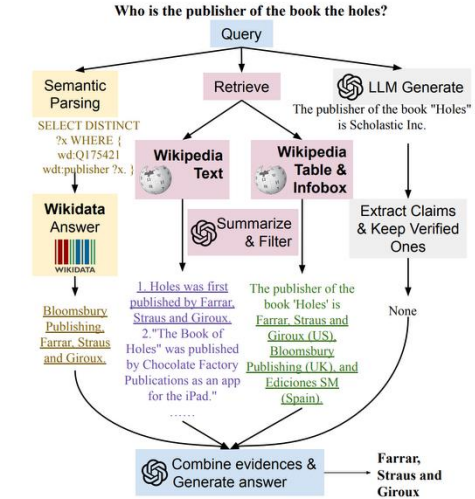
OTT-QA (Chen et al., 2021)



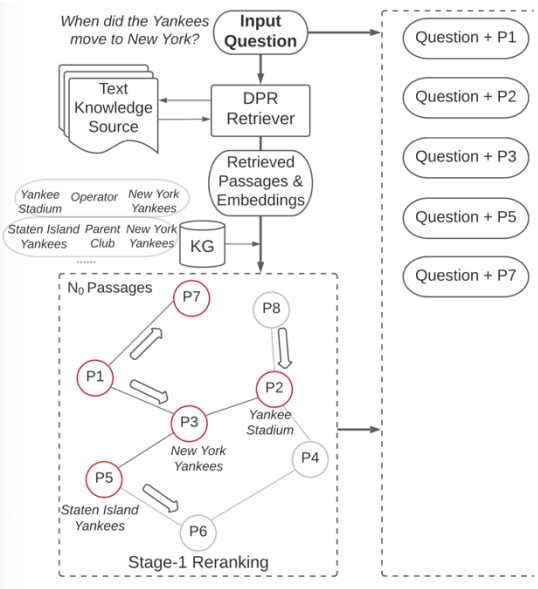
T-RAGBench (2025)

Hybrid: Fusion-in-Generator

•**Concatenate multi-source data** —separately encodes many contexts and fuses only in the decoder; canonical late-fusion reader used widely in hybrid pipelines.
Examples: SPAGHETTI (Zhang et al., 2024), Fusion-in-Decoder (Izacard & Grave, 2020), DuRePa (Dual Reader-Parser) (Li et al., 2021)



SPAGHETTI (Zhang et al., 2024)

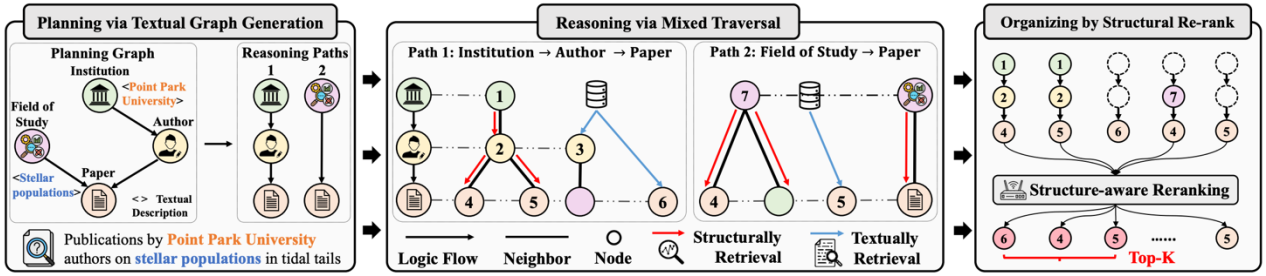


KG-FiD (Yu et al., 2022)

•**Interweave multi-source data** - improves FiD by structuring retrieved evidence with a knowledge graph and re-ranking before decoding; representative of strong late-fusion readers with structural priors.
Examples: REANO (Fang et al., 2024), KG-FiD (Yu et al., 2022)

Hybrid: Iterative

•**Planning&Reasoning - MoR (Lei et al., 2025)** — retrieve textual and structural knowledge by via a Planning-Reasoning-Organizing framework.



•**Sequential (RL/iterative) hybrid retrieval** — OTT-QA (Jose et al., 2024) retrieve information sequentially, where a selected piece of data helps searching for the next piece. A decision-maker (e.g., DRL) alternates *text* and *table* retrieval and then reads.