Contextualizing Large Language Models over Evolving Internal Data and Knowledge

Sunita Sarawagi
IIT Bombay
sunita@iitb.ac.in

aiDM, Bengaluru May 31, 2026.

An Enterprise's Internal Data Ecosystem

Document Collections
Policies, reports, manuals, presentations, contracts

Rule Books / Knowledge Bases
Compliance rules, product catalogs, policy rule-books

Relational Databases
ERP, CRM, finance, HR, operations data

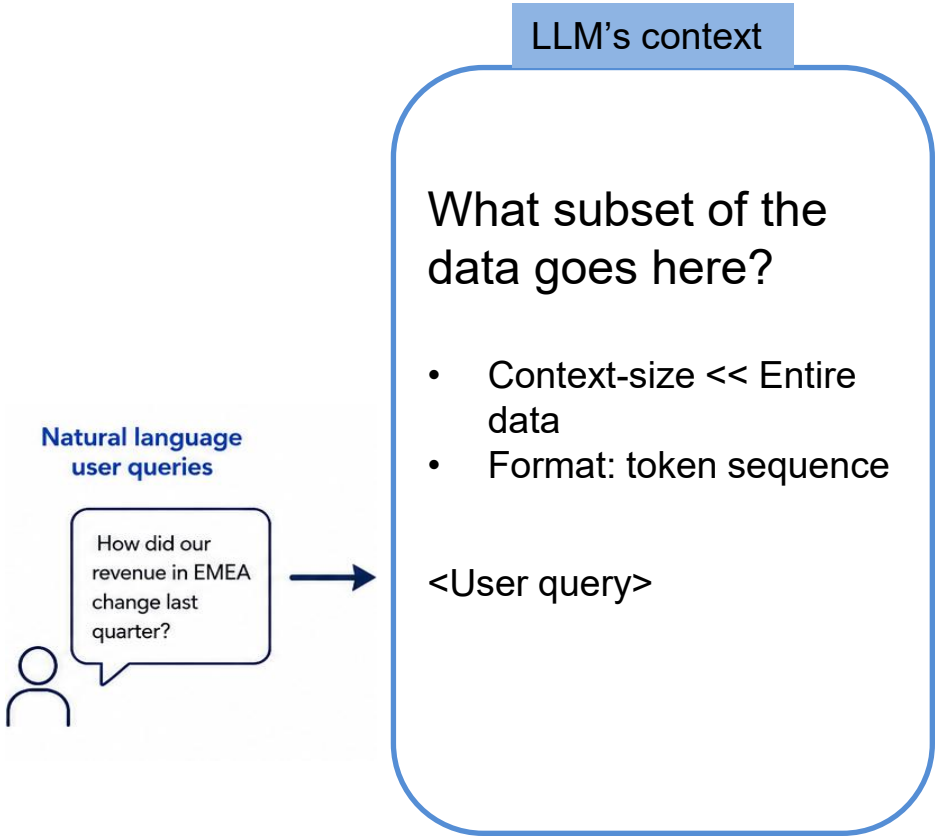
Ontologies / Knowledge Graphs
Concepts, relationships, business semantics

Spreadsheets
Budgets, forecasts, trackers, ad-hoc data

Unstructured & Semi-structured Content
Emails, tickets, chat logs, forms, PDFs, JSON, etc.

Heterogeneous, evolving at different rates, hidden from the external world.

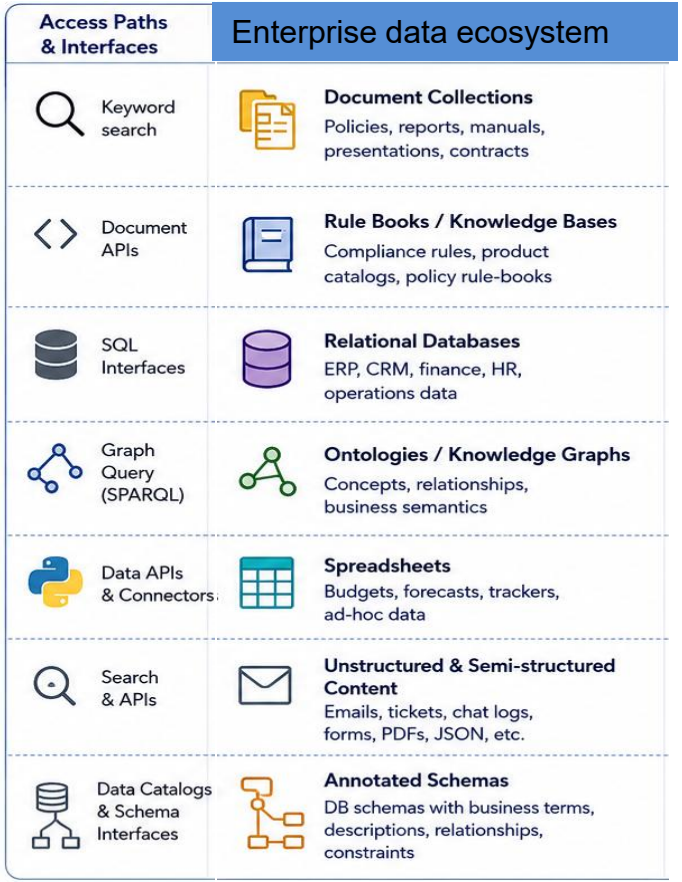
Goal: Natural language queries across the data ecosystem



Goal: Natural language queries across the data ecosystem

- Sparse inverted indices e.g. BM25 or Dense vector indices
- DBMs expose their schema enriched with textual descriptions and typical values
- Ontologies support graph traversal queries like SPARQL
- API access using protocol like MCPs, again significantly enriched with textual descriptions

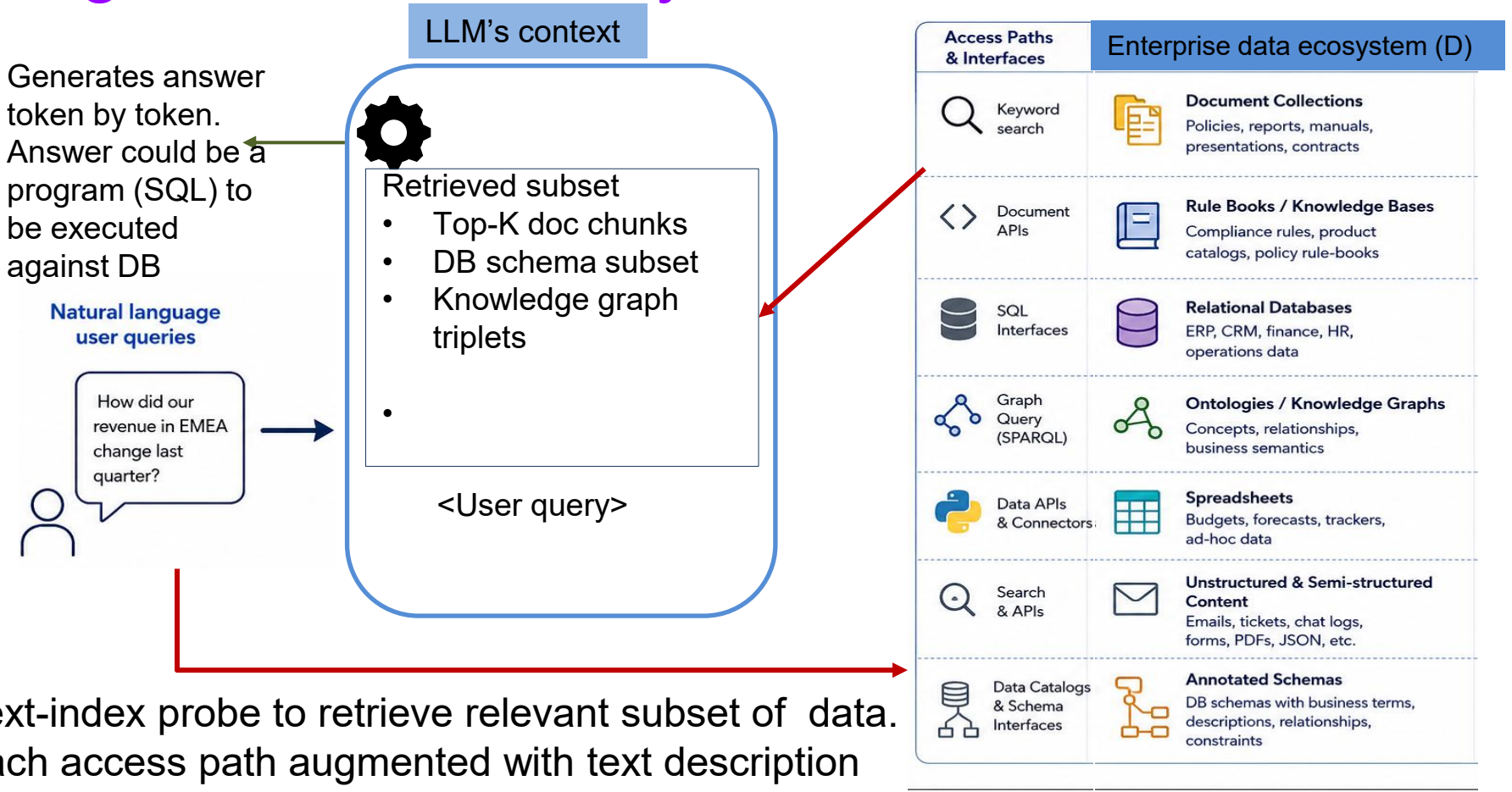
Also examples of how to convert natural language question into access paths that each support.



Retrieval Augmented Answer Generation

- Direct retrieval with keyword index (Plain RAG)
- LLM-assisted query rewrites
- Multi-step reasoning and agentic RAG

First generation RAG systems



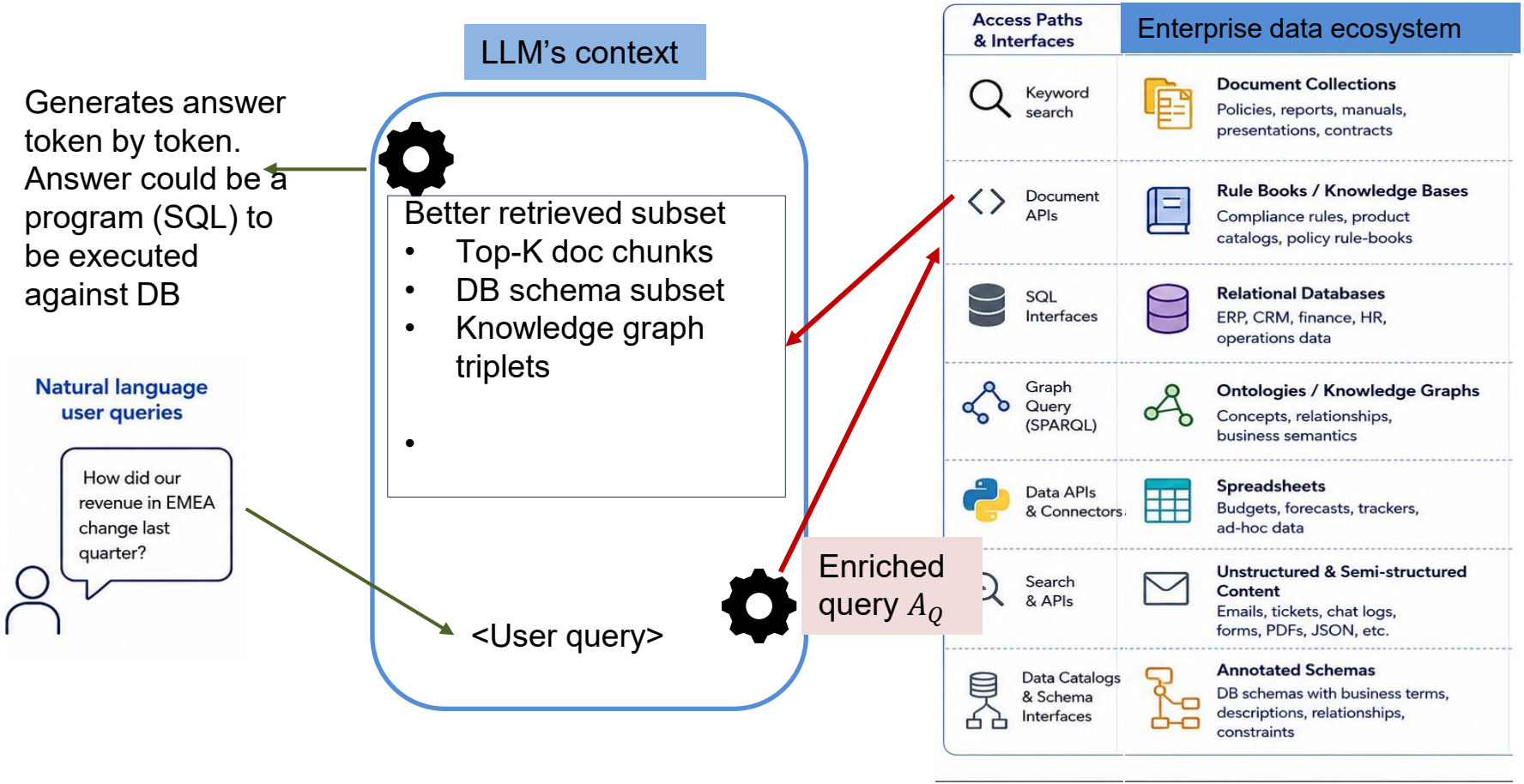
Text-index probe to retrieve relevant subset of data.
Each access path augmented with text description

Limitations of Basic RAG

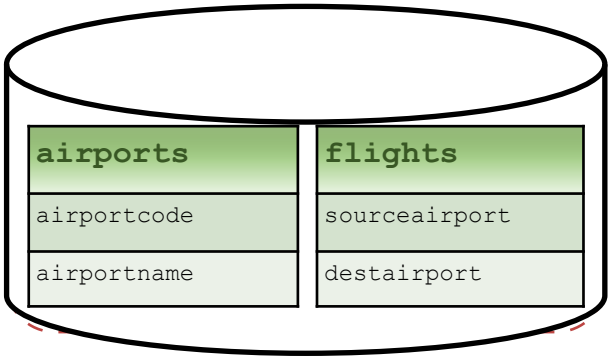
- Structural mismatch between query and local-content
  - Query: natural language.
  - Content: A database schema or a knowledge graph

One interesting idea: LLM rewrites query to bring query closer to the source

LLM Rewrites the Query before Probing



Converting Natural Language Queries to SQL Programs to Retrieve Relevant Structured databases



D: Schema of a Local Structured Database

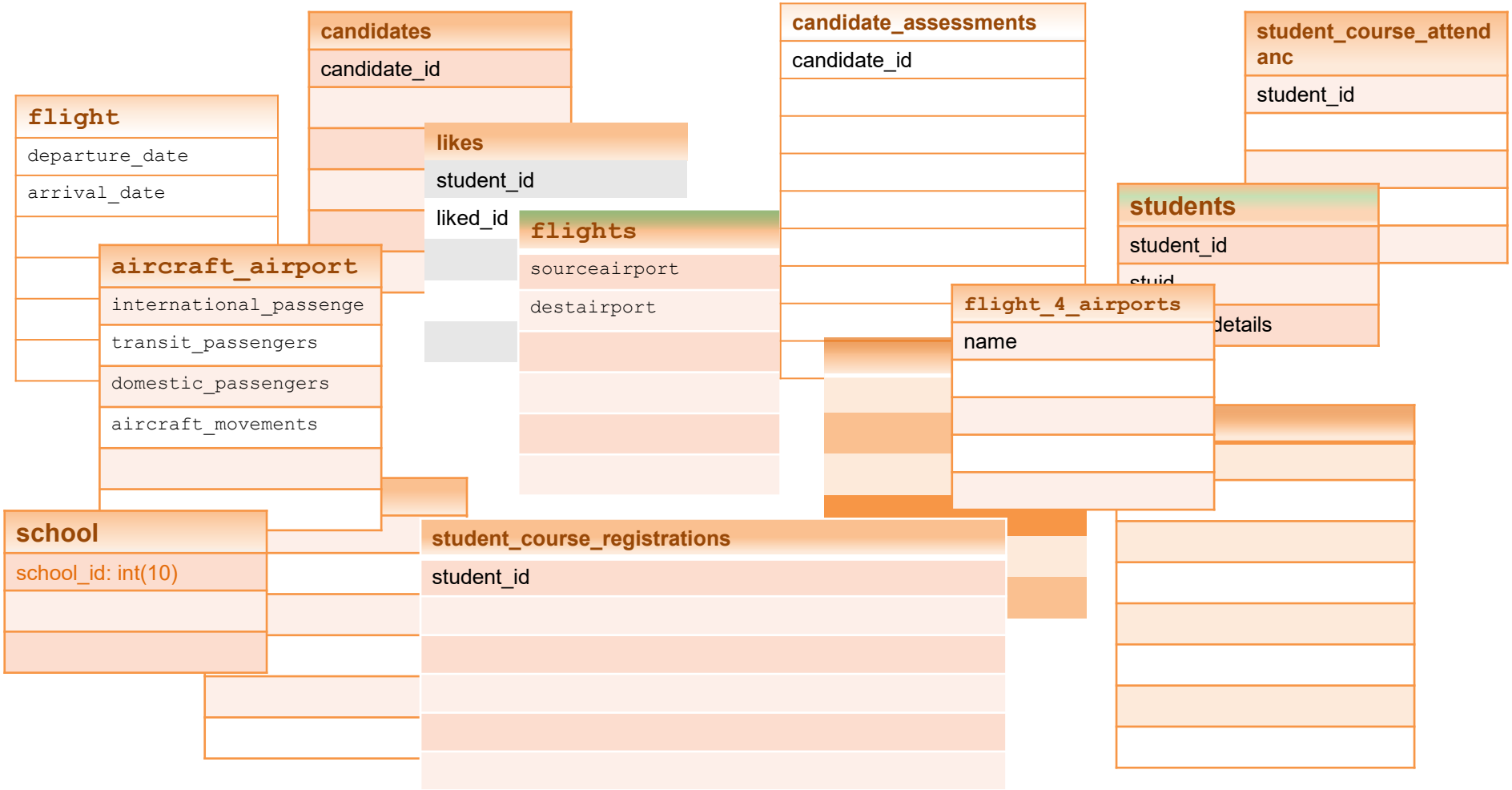
Which airports do not have departing or arriving flights?

User query: Q

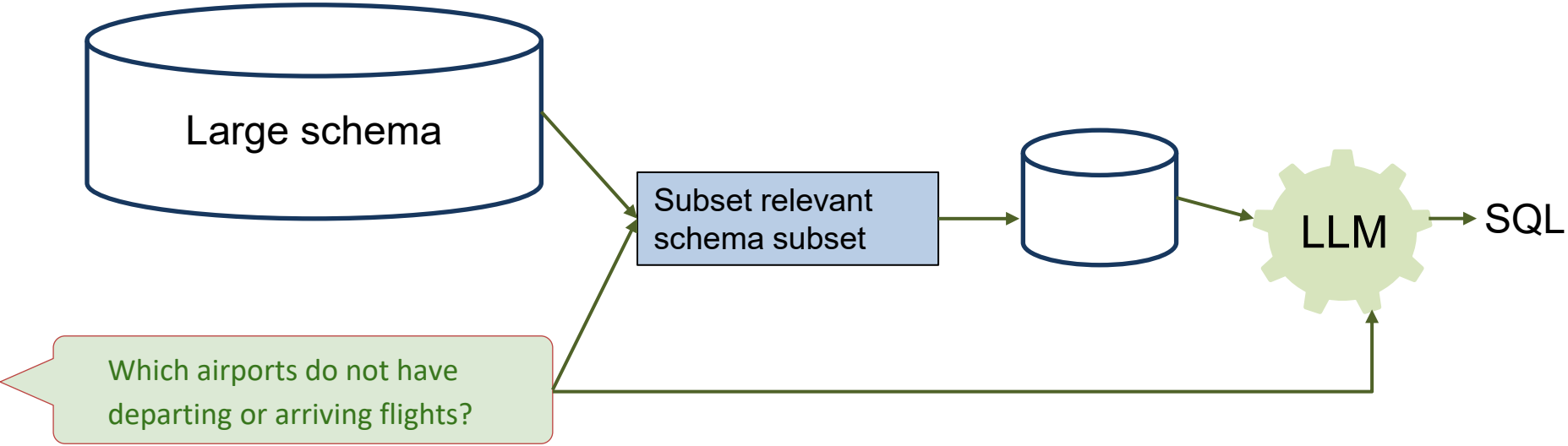
Please write SQL for {question} without any explanation based on the following SQLite DDL:

```
CREATE TABLE airport(
        airportcode INTEGER,
        airportname INTEGER,
        ……,
        ……
);
CREATE TABLE flights(
        sourceairport INTEGER,
        destairport INTEGER,
        ….,
        …..
);
…
Question: {question}
SQL:
```

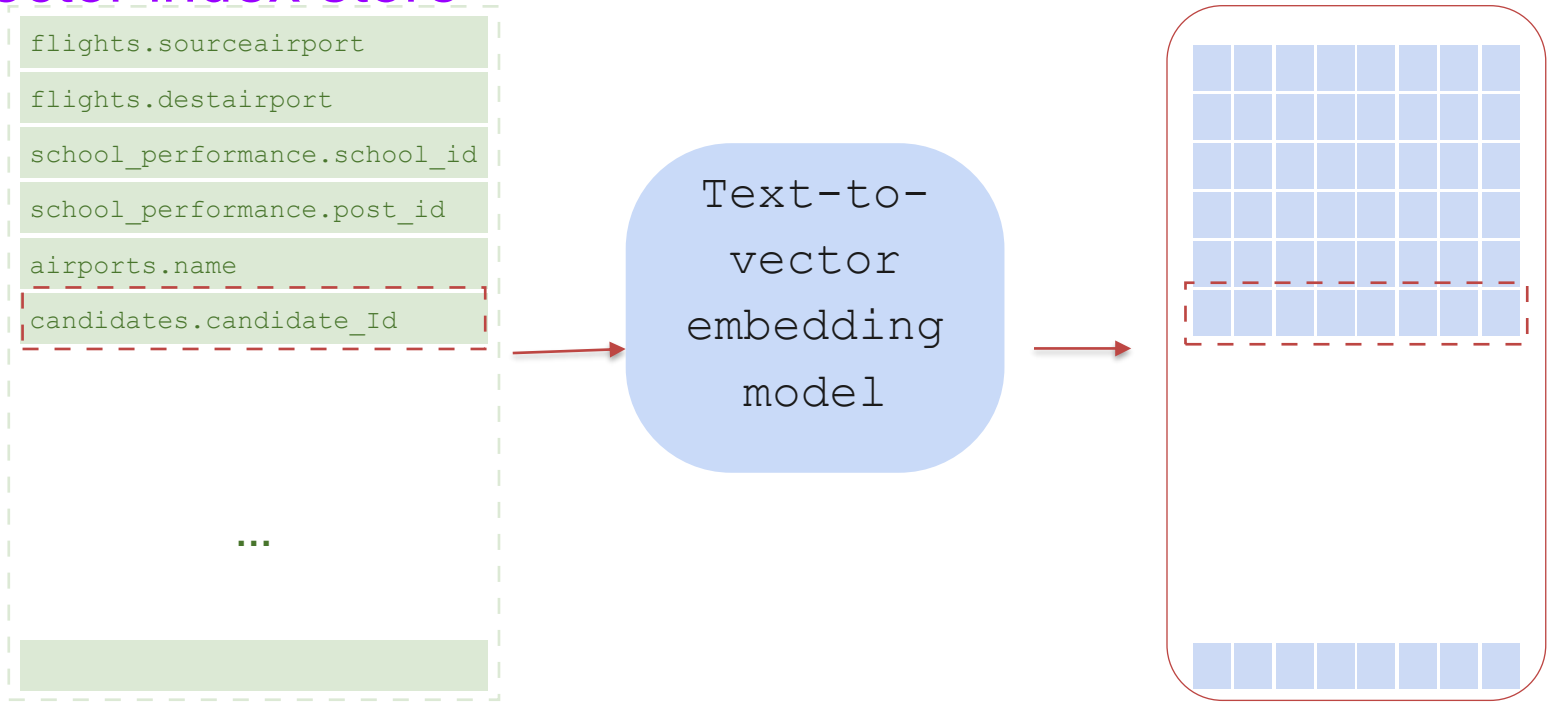
```
SELECT airportname
FROM airports
WHERE airportcode NOT IN
(
   SELECT sourceairport
   FROM flights
   UNION
   SELECT destairport
   FROM flights
)
```


candidates
candidate_id
flight
departure_date
arrival_date
likes
student_id
liked_id
candidate_assessments
candidate_id
student_course_attend
anc
student_id
students
student_id
aircraft_airport
international_passenge
transit_passengers
domestic_passengers
aircraft_movements
flights
sourceairport
destairport
flight_4_airports
name
school
school_id: int(10)
student_course_registrations
student_id

RAG to retrieve relevant schema subset



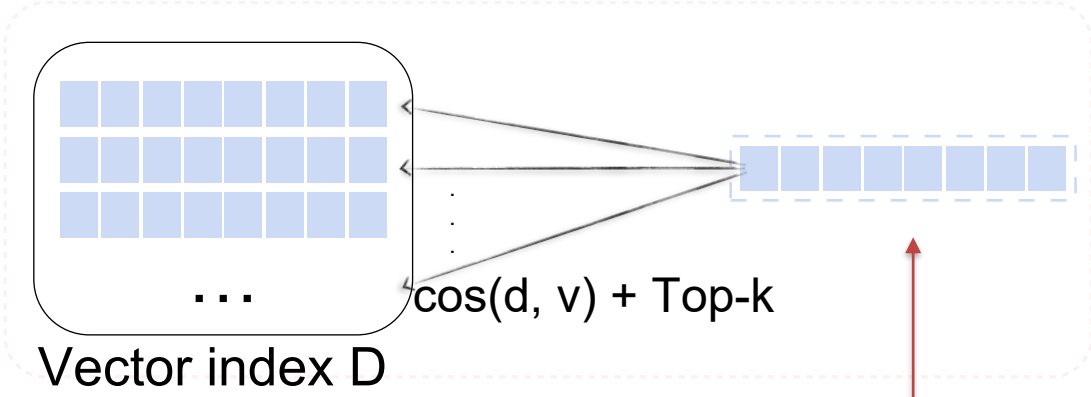
Standard RAG pipeline: Offline encode Schema elements into a Vector index store



Schema elements as table.column name

Vector Index Score

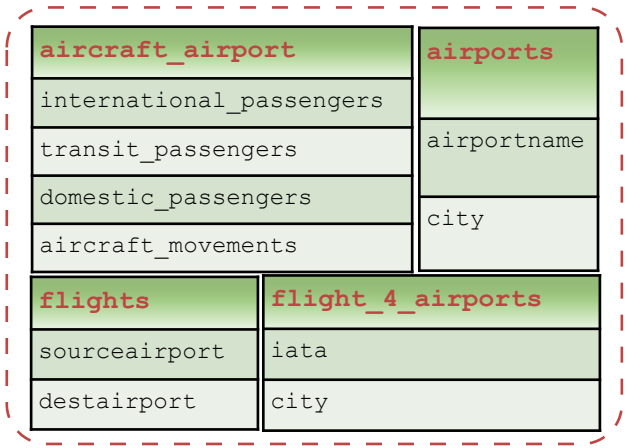
Standard RAG pipeline: Query-time convert query to vector and probe index.

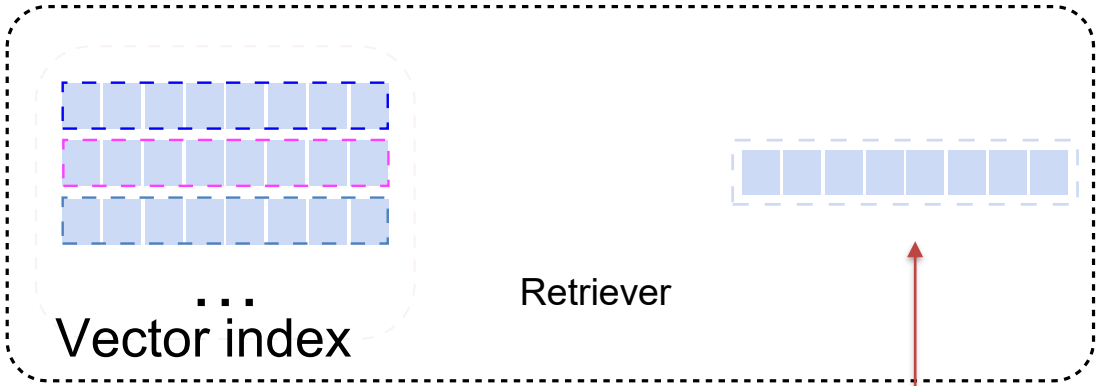


Which airports do not have departing or arriving flights?

Text-to-vector embedding model

Retrieve top-K matches





Which airports do not have departing or arriving flights?

Text-to-vector embedding model

Fix by LLM-based hallucinations

Given question, ask LLM to make-up a minimal database schema that can answer the question (A_Q) without seeing D

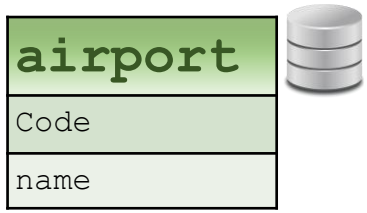
Which airports do not have departing or arriving flights?

Hallucinate a minimal schema of a relational database that can be used to answer the natural language question.

Which airports do not have departing or arriving flights?

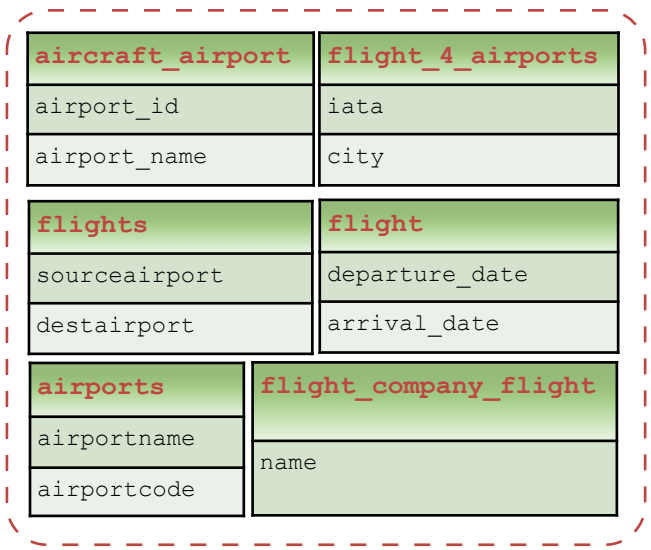
LLM

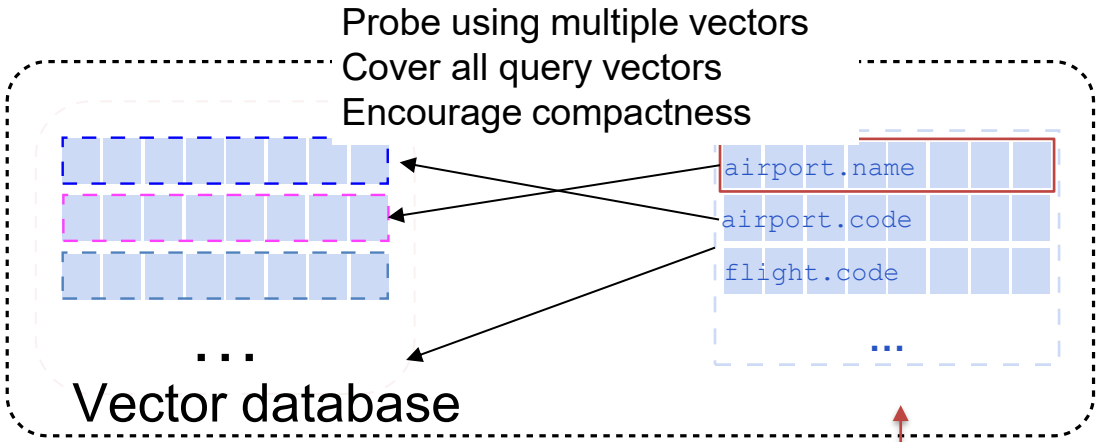
Prompt to LLM

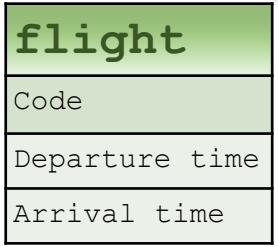


A_Q: Hallucinated schema, structured like the local sto

RAG using A_Q: Structured matches with multiple keywords







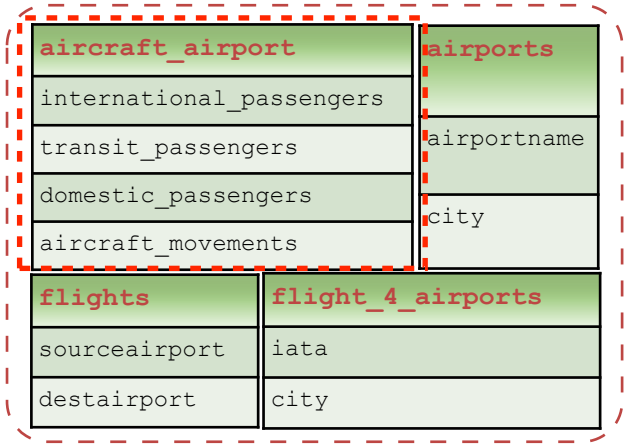
Text-to-vector embedding model

Which airports do not have departing or arriving flights?

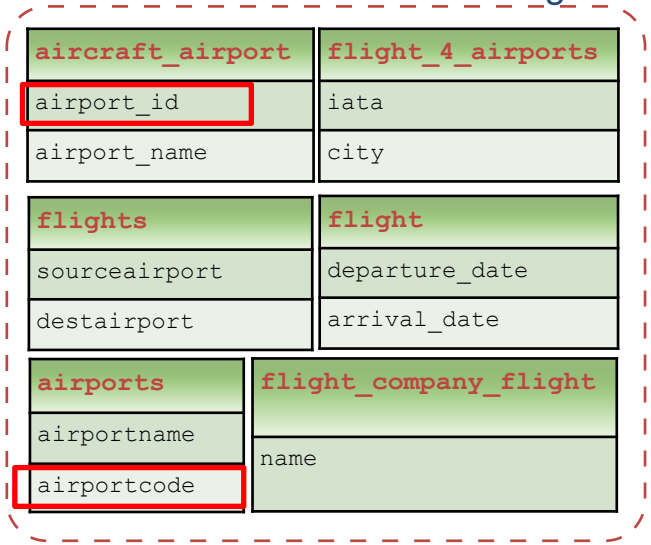
LLM

Comparing the two retrievers

Single vector



Collective with hallucinated schema for segmentation



Coverage of columns like airportcode

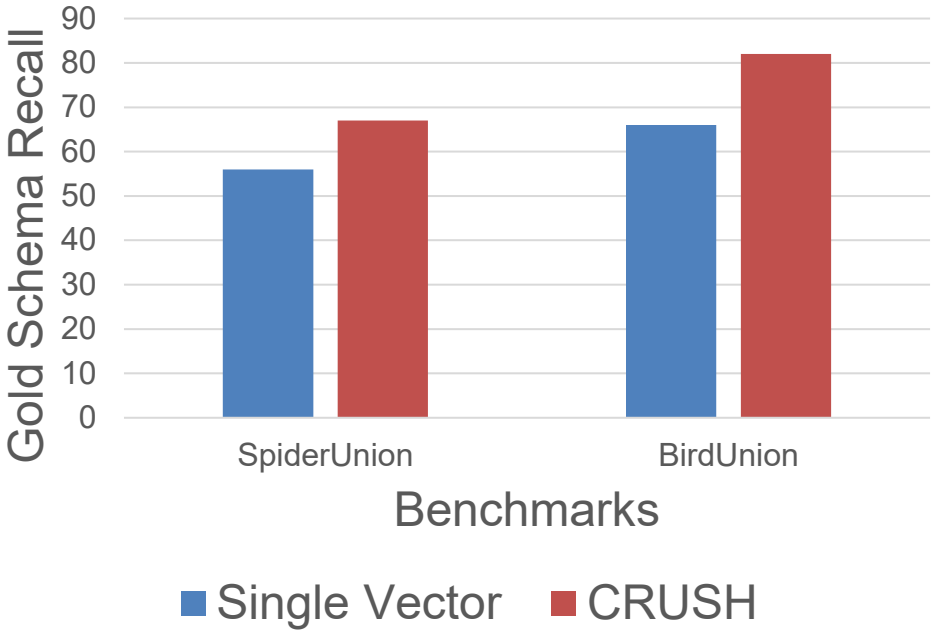
Which airports do not have departing or arriving flights?

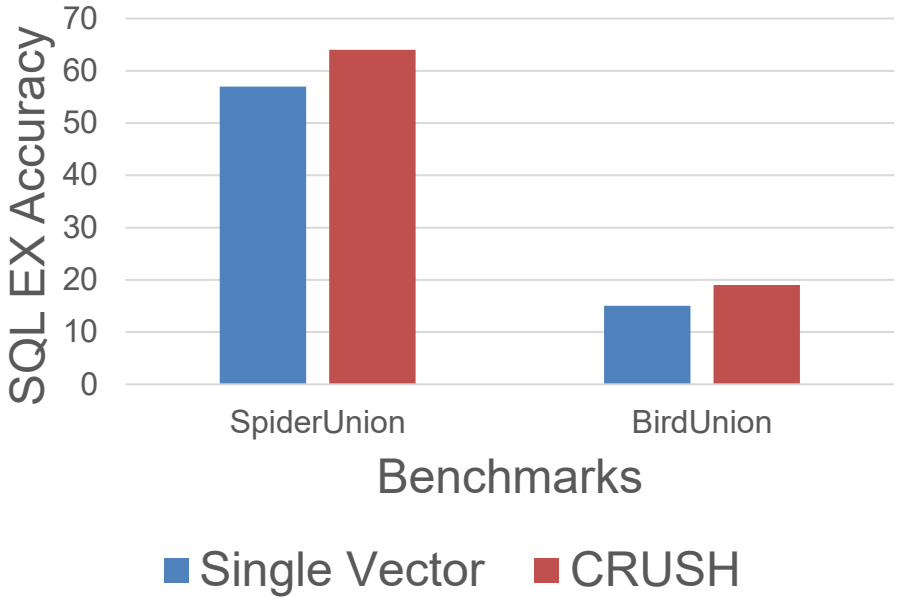
SQL

```
SELECT airportname
FROM airports
WHERE airportcode NOT IN
( SELECT sourceairport
    FROM flights
    UNION
    SELECT destairport
    FROM flights)
```

Results

Budget=20





Similar ideas useful across many other types of searches.

CRUSH4SQL…Kothari, Sarawagi, Chakrabarti. EMNLP 2023.

Query Expansion in the Age of Pre-trained and Large Language Models: A Comprehensive Survey

Limitations of Basic RAG

- Structural mismatch between query and local-content
  - Query: natural language.
  - Content: A database schema or a knowledge graph

  Solution: LLM rewrites query using its world knowledge

- Cannot handle complex information needs requiring stitching together information from multiple sources

Harness “reasoning” power of LLMs to orchestrate multi-step processing

LLM reasoning

Answer.

LLM decides done. Could take 100s of steps

Retrieved subset

LLM generates tool call 2

Retrieved subset

LLM generates tool call 1

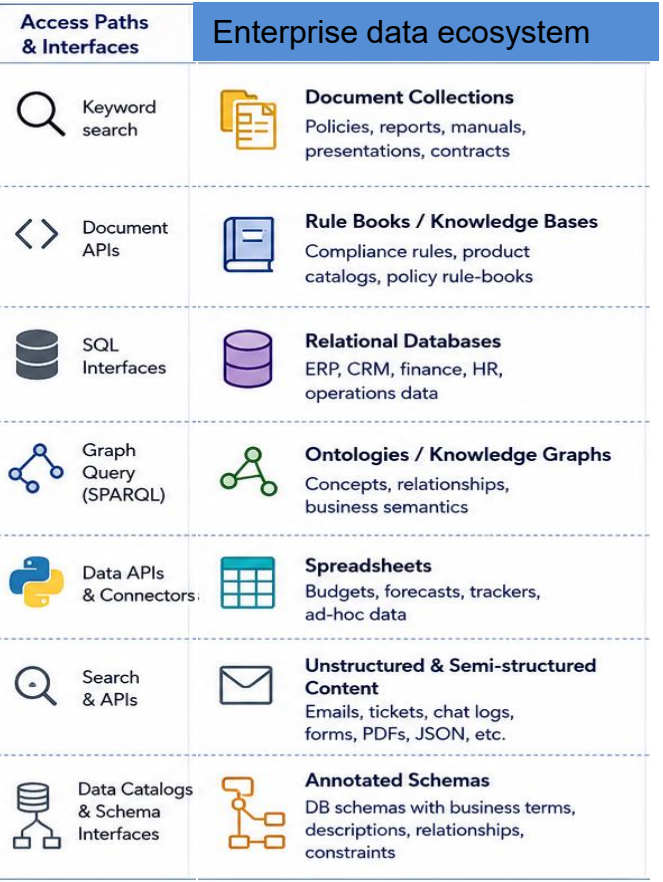
<User query>

List of tool with description:
1. Search Docs:….
2. Query DB:…
3. Graph AI:…..
4. ..
5. ..
100s of such tools.

LLM's context

Natural language user queries

How did our revenue in EMEA change last quarter?



Limitations of reasoning systems

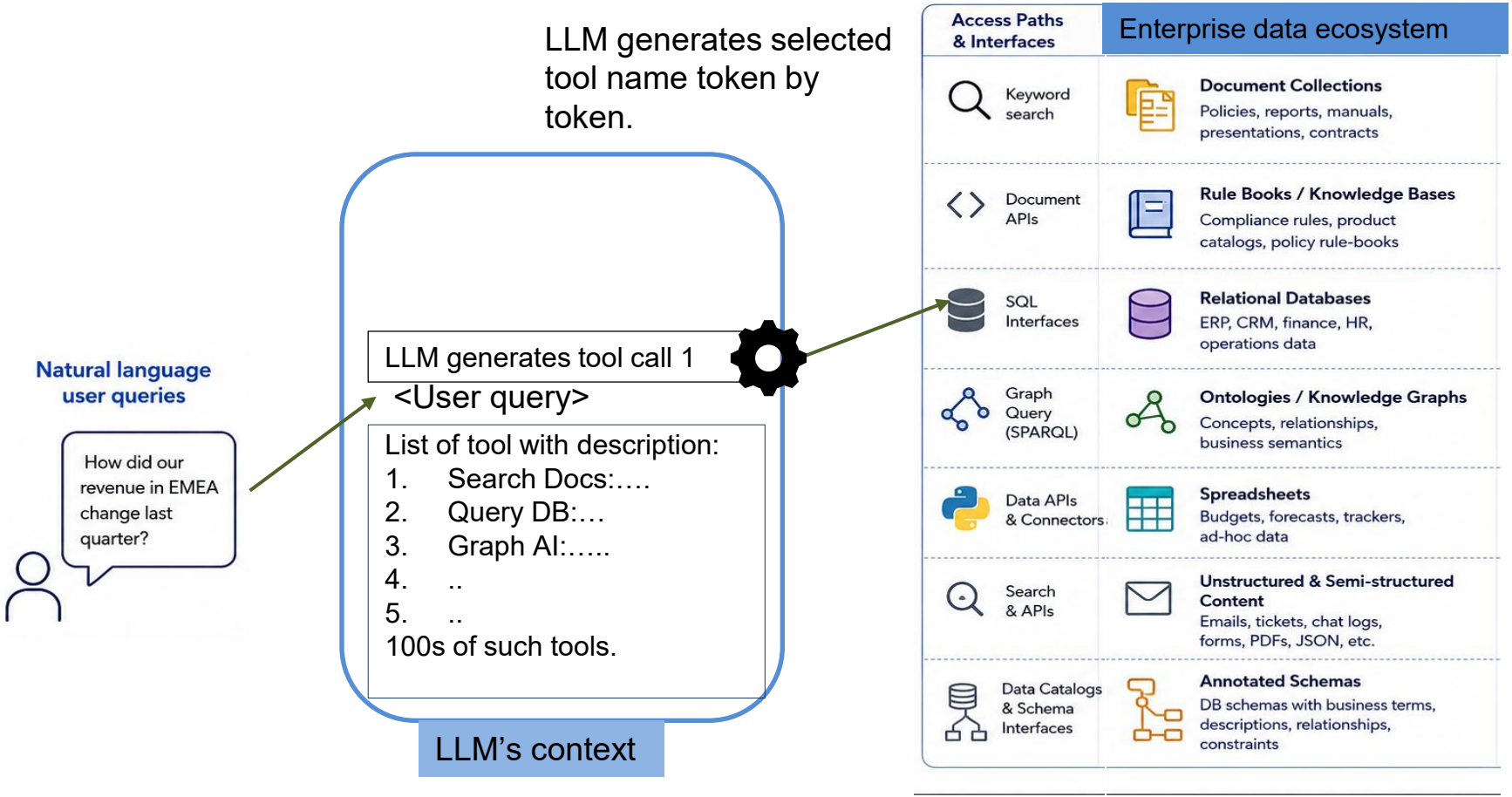
- Reasoning is expensive!
  - 100s of LLM calls for single queries
- As context length increases performance degrades
  - Particularly for cost-effective smaller models.

What can we do to make reasoning both more accurate and efficient?

Short-cut reasoning: Retrieve, not Generate

- Don’t treat LLM as just a black-box token generator
- LLM’s state provides rich semantic representation of tokens in the context.
- With query as “probe” and attention to various parts of the context as relevance score, we can short-cut different parts of the reasoning process.
- Two examples:
  - Tool calling
  - Graph traversal

LLM Generates tool name to call



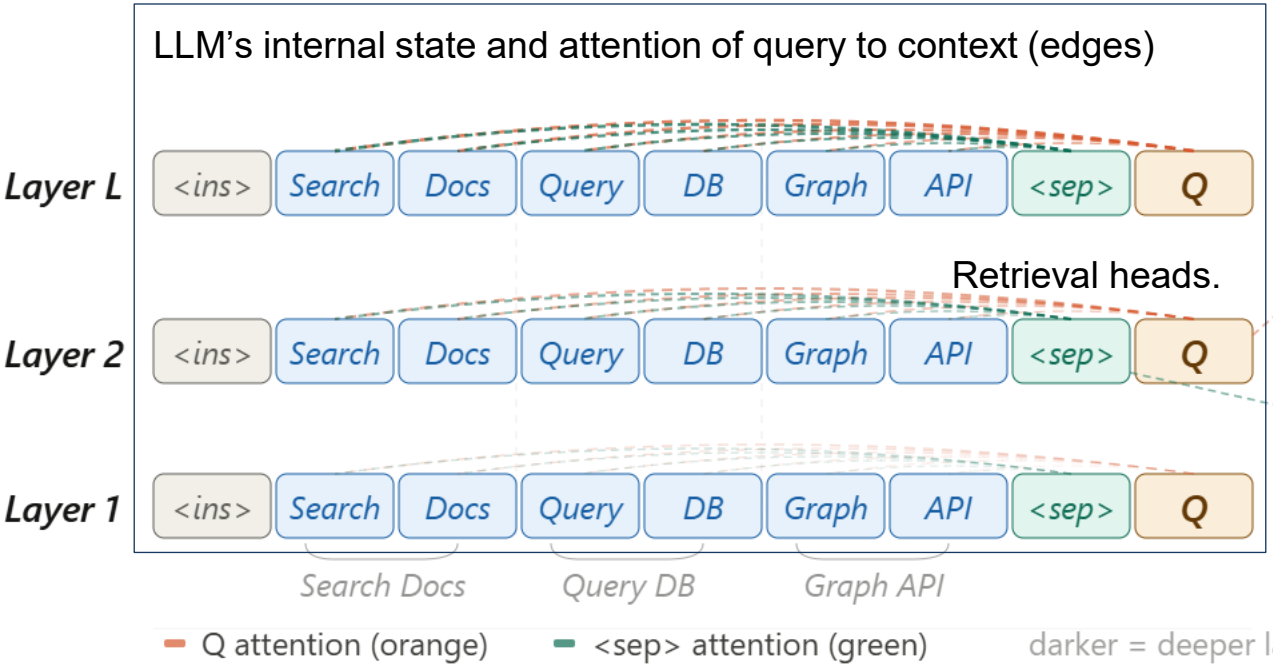
Tool selection via default LLM generative path is brittle

- Sensitive to item ordering, identifier format (FinanceTool vs [5]), and identifier placement

- Suffers from positional biases (recency/attention sink bias)

We show that direct tool selection from context is more based on relevance, and not so much on prompt formatting

Directly Retrieve from LLM's Context: No generation.



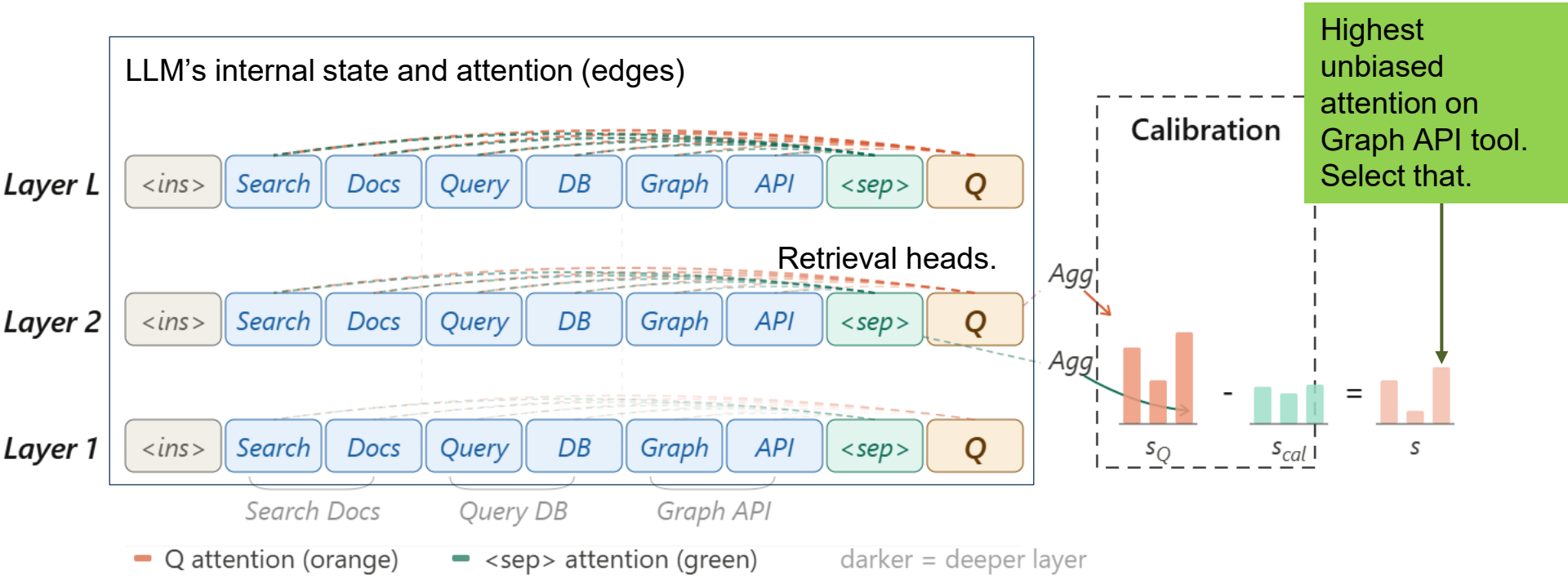
- Expect more attention on relevant tool names
- Not all attention heads follow that pattern
- Subset of heads take the role of linking to distant relevant tokens → called retriever heads

Discovering and Calibrating Retriever heads.

- Use a few task-specific examples in-context (5—10 suffice) to rank heads based on highest relative attention to correct tool name.

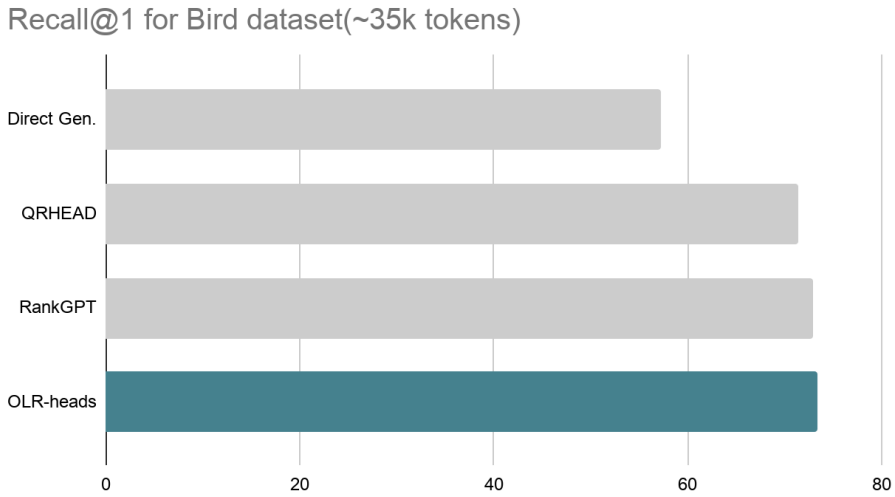
- LLMs have bias for tool-names near the query or near the starting instruction token.
  - De-bias by subtracting attention of query-neutral token.

Directly Retrieve from LLM's-Context: No generation.

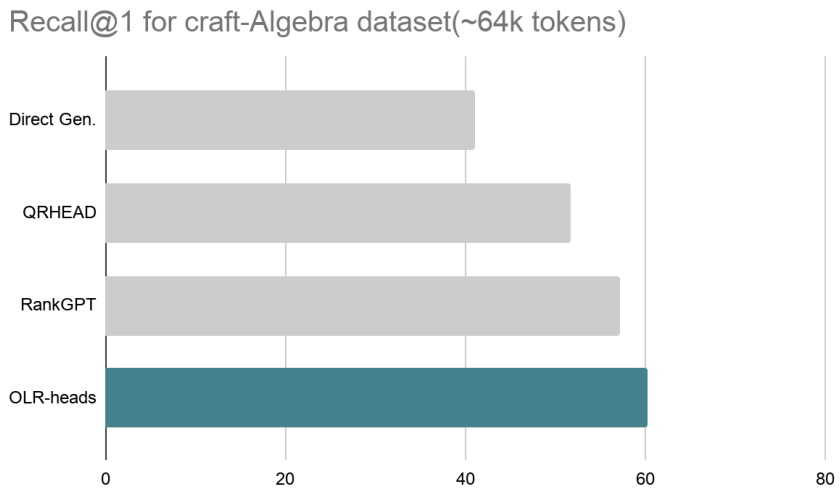


Accuracy comparison

Database Routing among 80 DBs



Algebra tool selection from 280 tools



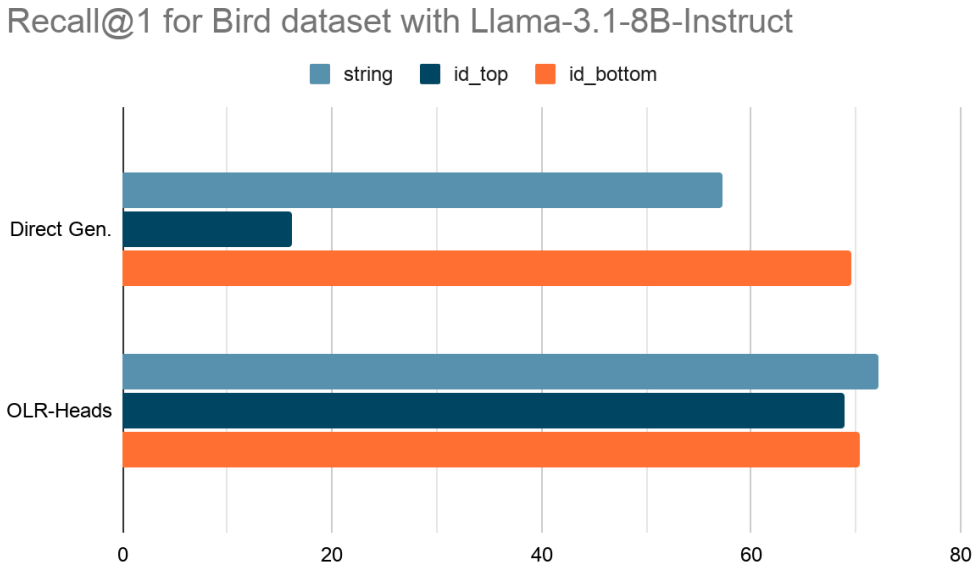
Context-retrieval provides better Recall@1 on Bird and craft-Algebra dataset while requiring only a single forward pass on a Llama-3.1-8B-Instruct model

Robust to Identifier Representation Changes

Variant: String
tool_id: WebsiteTool
tool description: Quickly create and deploy websites, and publish content on them.

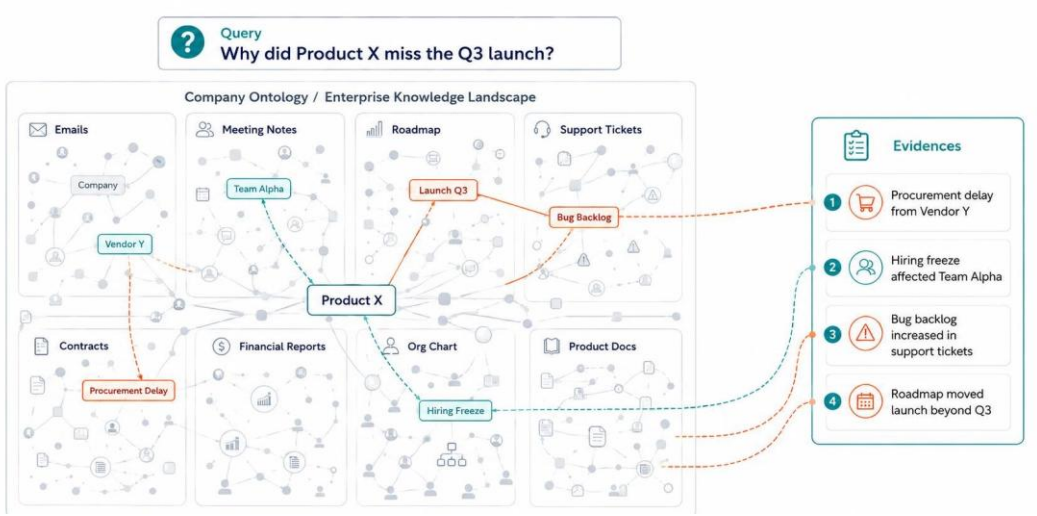
Variant: id_top
tool_id: [5]
tool name: WebsiteTool
tool description: Quickly create and deploy websites, and publish content on them.

Variant: id_bottom
tool name: WebsiteTool
tool description: Quickly create and deploy websites, and publish content on them.
tool_id: [5]



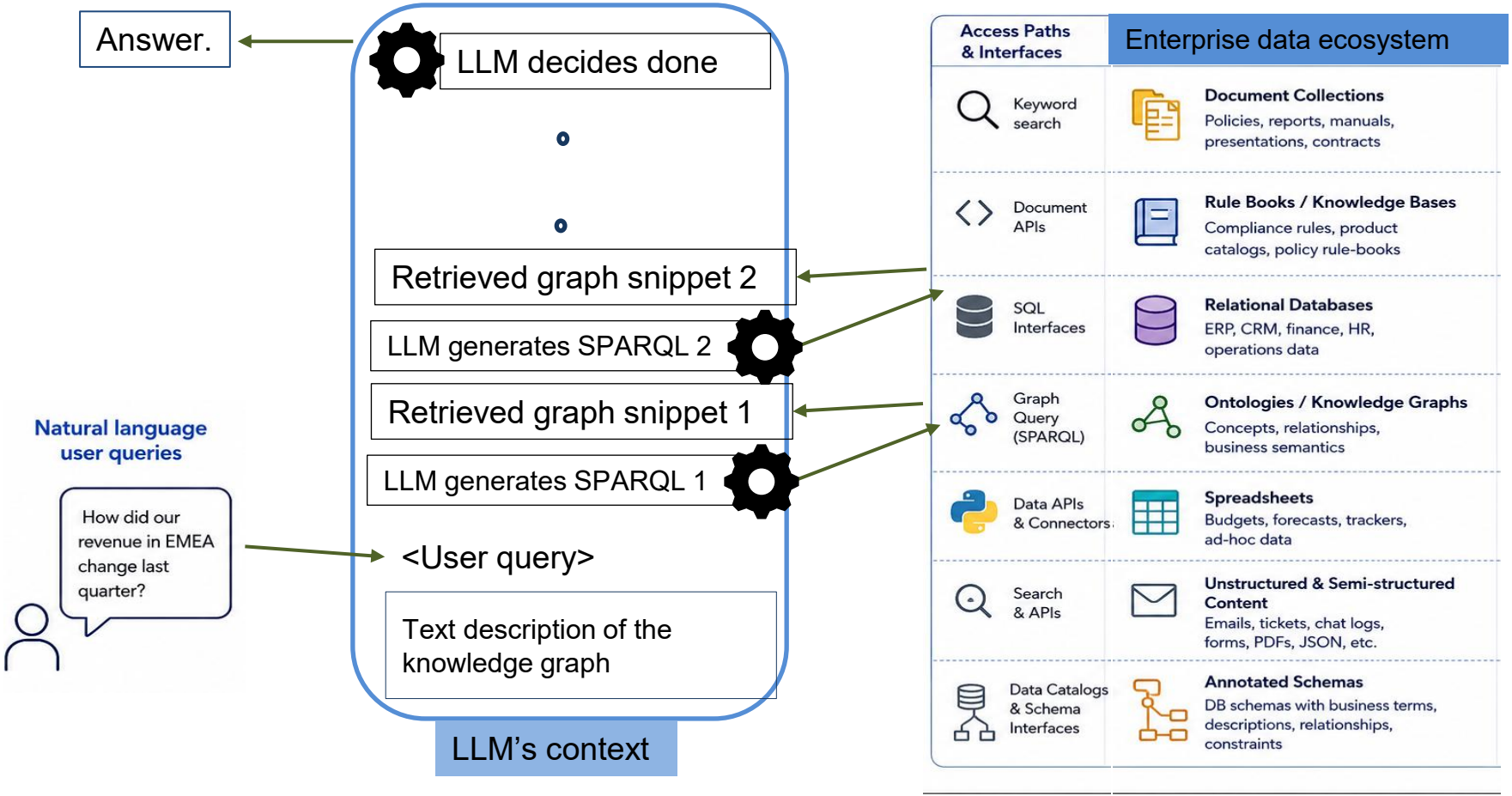
Context Retrieval to Short-Cut reasoning for Multi-hop Queries

A case study: path queries on knowledge graphs/ontologies. Such queries are multi-hop, and cannot be handled with a single RAG retrieval



One query → a chain of evidence across many heterogeneous sources. Each link surfaces the next: procurement delay → hiring freeze → bug backlog → roadmap slip.

LLM reasoning for graph traversal



But why limit the LLM’s access through tiny peepholes?

Current LLMs support large contexts

LLM long-context reasoning

Answer.

LLM decides done

LLM generates token-by-token to reason….

Natural language user queries

How did our revenue in EMEA change last quarter?

<User query>

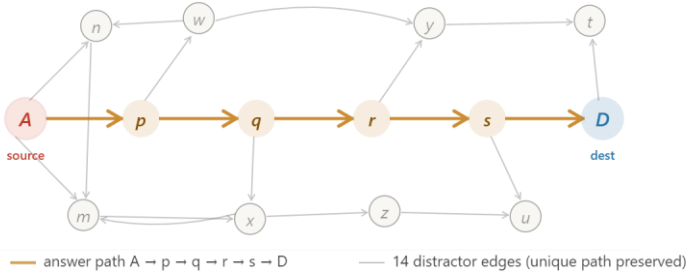
Entire graph in context.

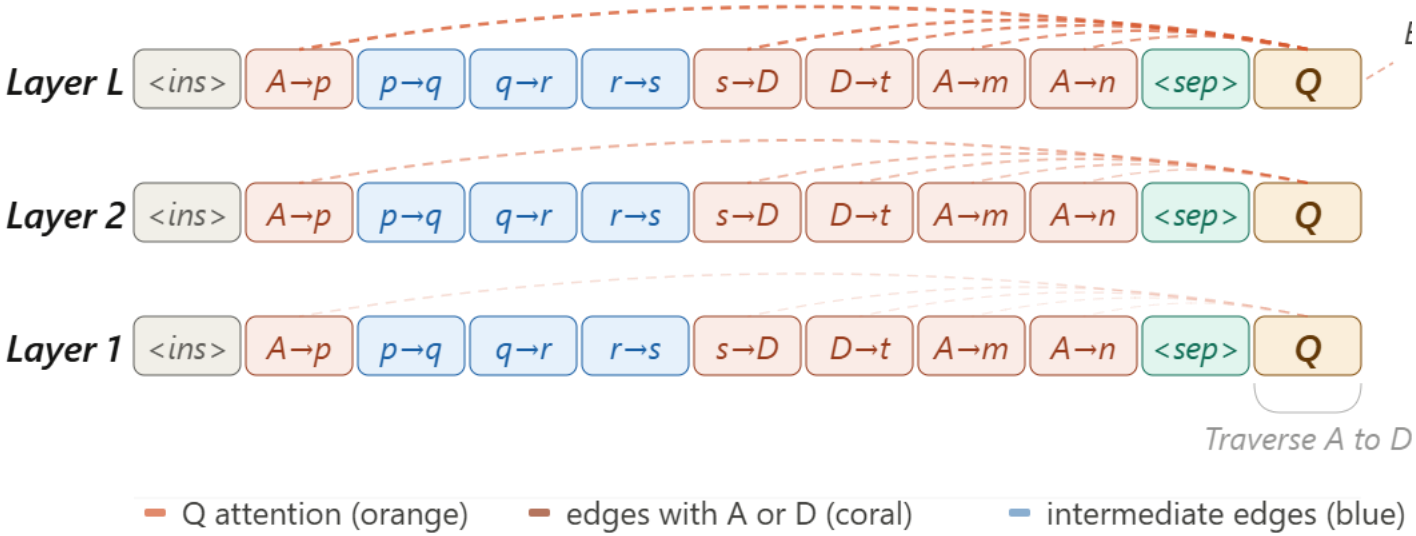
LLM's context

Brittle, particularly for large graphs, Expensive.

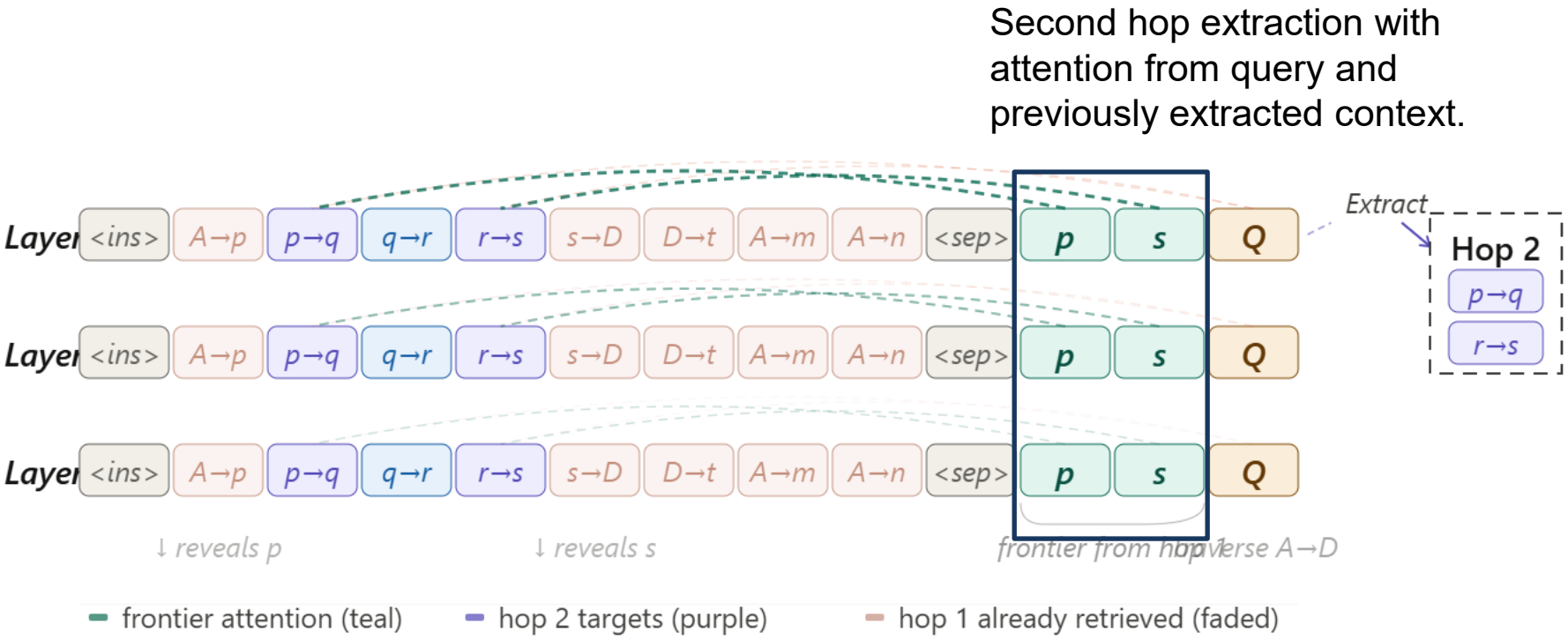
Access Paths & Interfaces	Enterprise data ecosystem
Keyword search	**Document Collections** Policies, reports, manuals, presentations, contracts
Document APIs	**Rule Books / Knowledge Bases** Compliance rules, product catalogs, policy rule-books
SQL Interfaces	**Relational Databases** ERP, CRM, finance, HR, operations data
Graph Query (SPARQL)	**Ontologies / Knowledge Graphs** Concepts, relationships, business semantics
Data APIs & Connectors	**Spreadsheets** Budgets, forecasts, trackers, ad-hoc data
Search & APIs	**Unstructured & Semi-structured Content** Emails, tickets, chat logs, forms, PDFs, JSON, etc.
Data Catalogs & Schema Interfaces	**Annotated Schemas** DB schemas with business terms, descriptions, relationships, constraints

Chained-Context-Retrieval: Short-cut Long-Context Reasoning





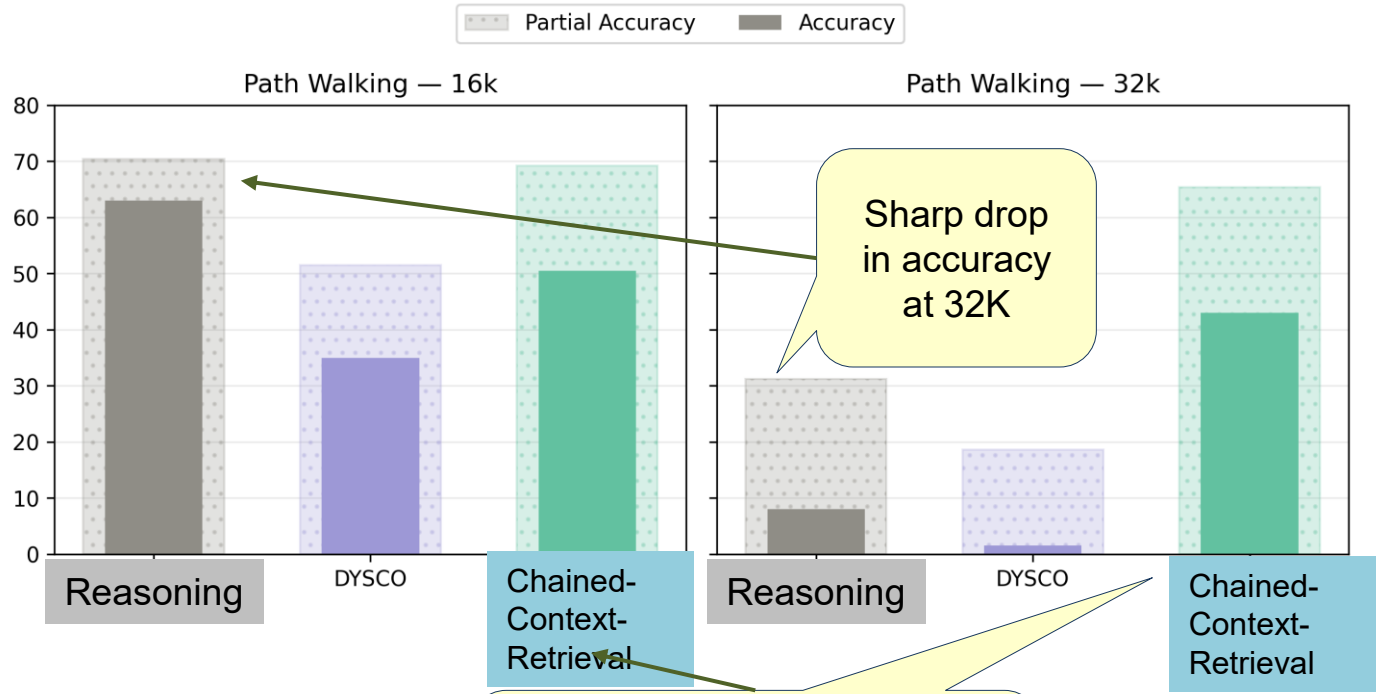
Chained-Context-Retrieval: Short-cut Long-Context Reasoning



Accuracy for two context sizes: 16K, 32K

Graph with 200 edges

Qwen3-8B



Limitations of reasoning systems

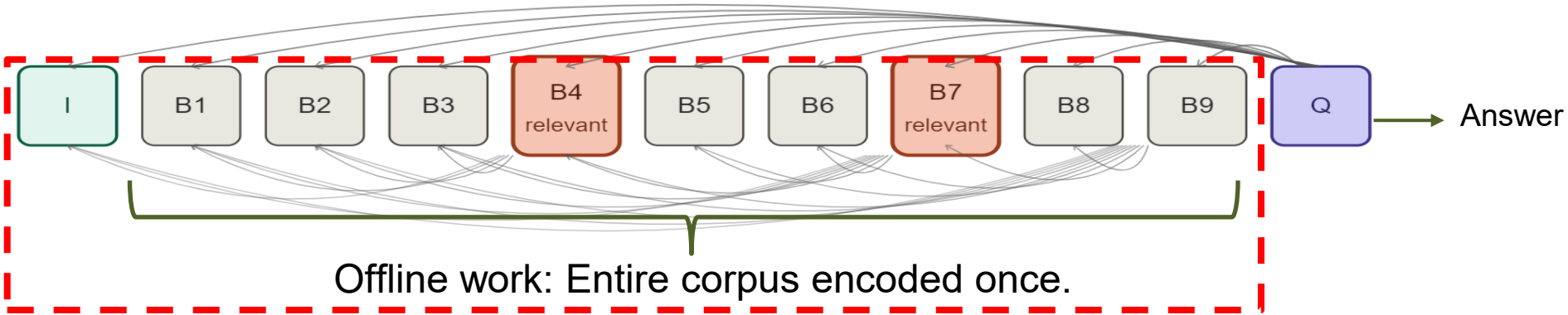
- Reasoning is expensive!
  - 100s of LLM calls for single queries
- As context length increases performance degrades
  - Particularly for cost-effective smaller models.

Solution: Do not always generate, Directly retrieve from long-context

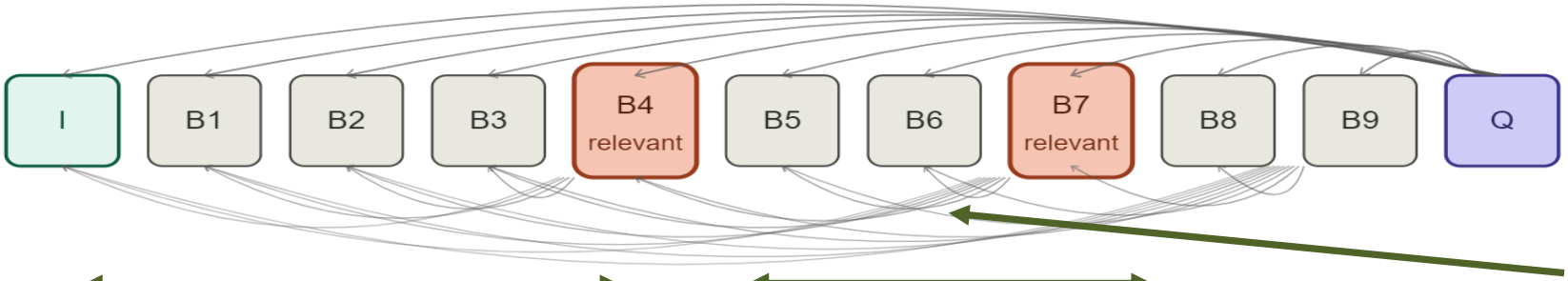
- Wasteful to repeatedly encode different data subsets via the LLM.

Caching LLM Encodings of Documents for Reuse

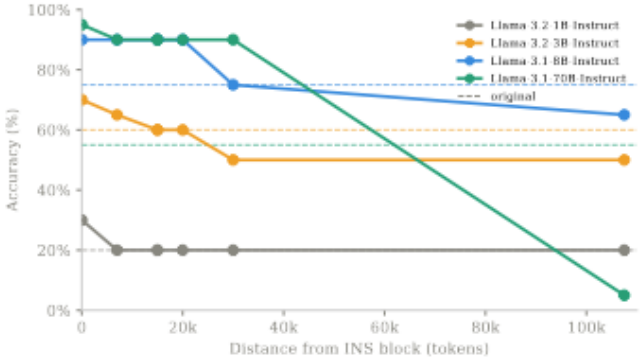
- Offline: Feed local content D to LLM, store the KV cache states
- Query-time: No separate RAG. Query processed by the LLM with the KV cache as state.



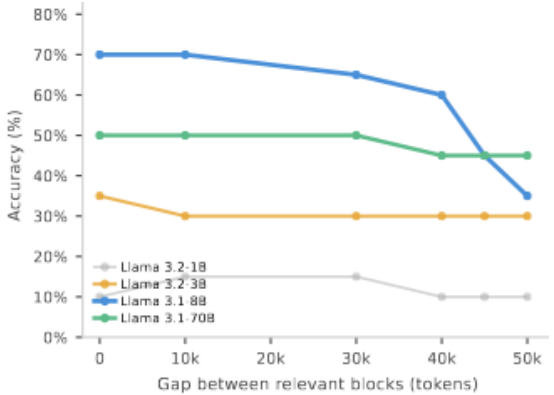
Challenges for full corpus KV cache reuse



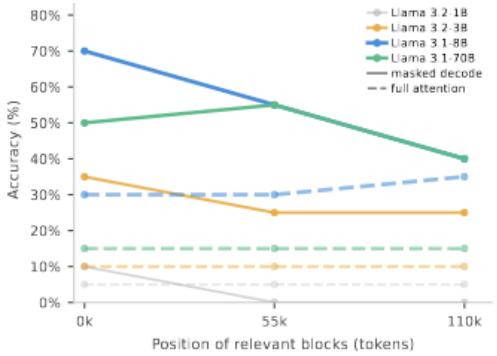
Accuracy drops as distance from instruction blocks increases



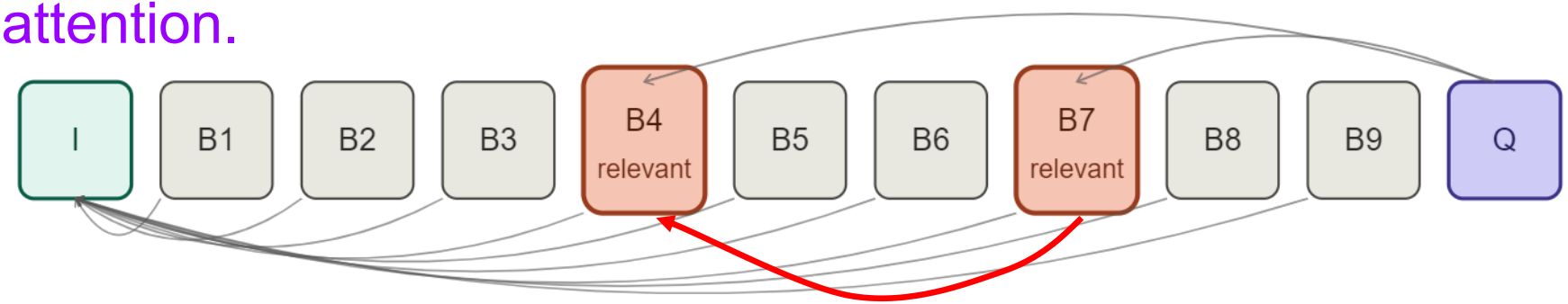
Accuracy drops as gap between relevant blocks increases



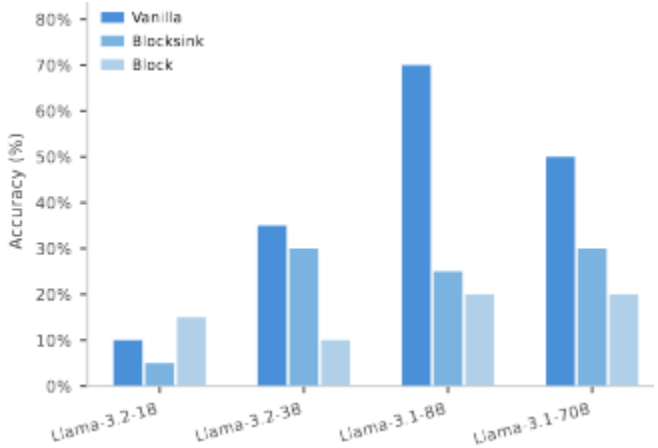
Accuracy drops if relevant blocks encoded with attention to several previous irrelevant blocks → “context rot”



Independently encoding blocks misses critical inter-block attention.



Cross-attention between blocks relevant to a query important for multi-document queries.



Challenge of Corpus-level KV cache reuse for multi-document queries

How to best pre-encode long documents in an LLM's cache so that during query time we are able to provide the illusion of relevant blocks meeting these four conditions:

Compactness

Are relevant blocks placed close to the instruction?

Contiguity

Are relevant blocks assigned adjacent positions in the query-time context?

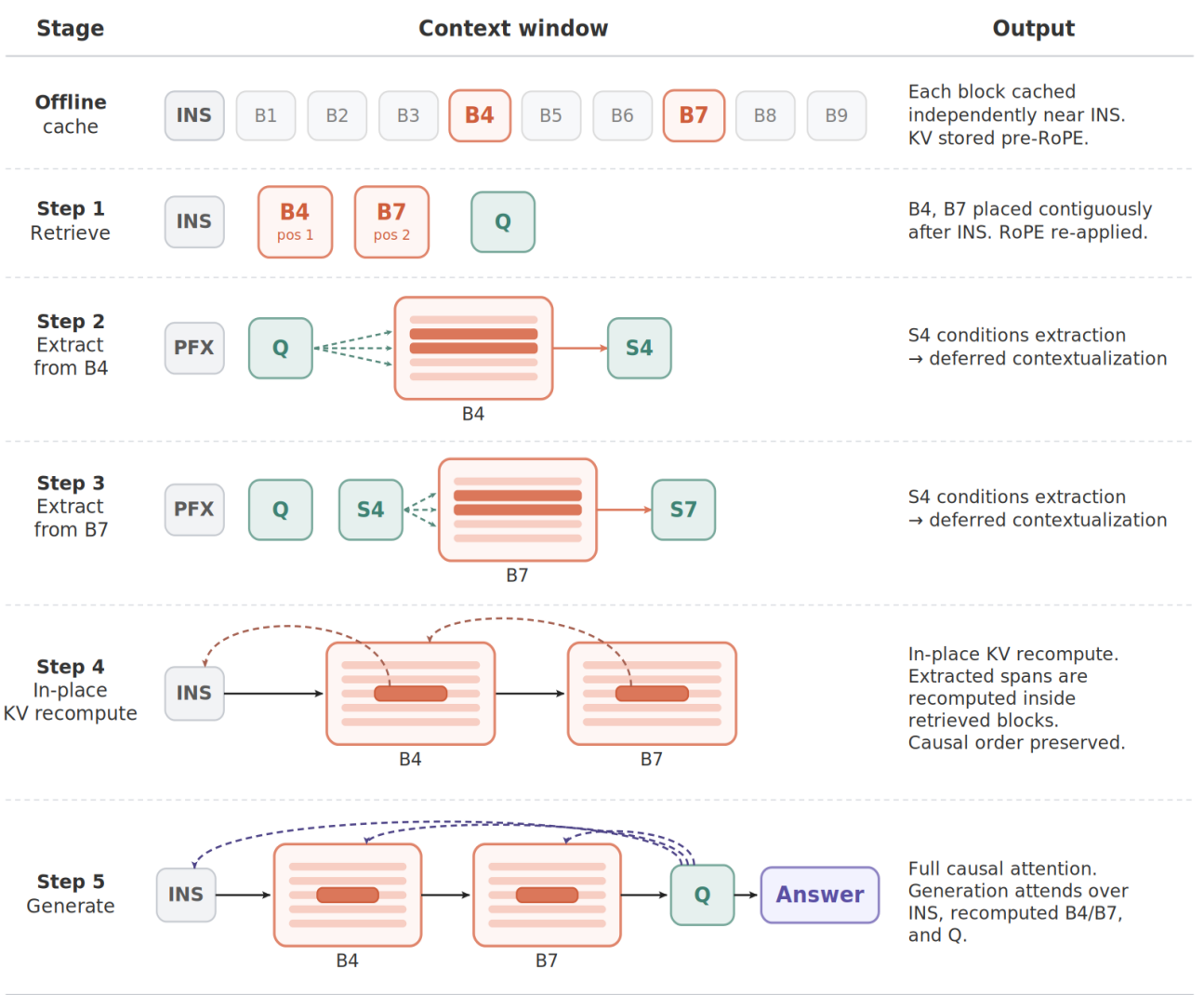
Cleanliness

Are relevant block states formed without attention to irrelevant blocks?

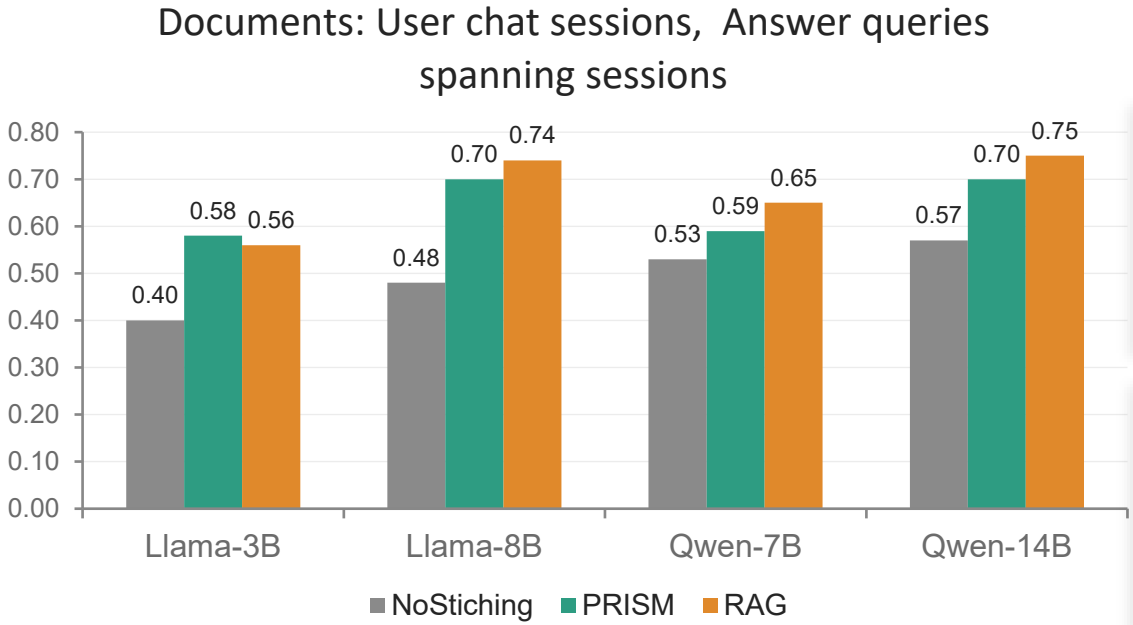
Cross-attention

Are dependencies among relevant blocks reflected in the KV states?

Stitching KV Caches of independently encoded document blocks (PRISM)



Deferred stitching of KV caches almost as accuracy, much faster



Time-to-first-token

~~1.57s~~ → **0.29s**

RAG → KV-cache

total time (L8)

~~12.2s~~ → **2.95s**

Full-Context Reuse → PRISM

PRISM ≈ re-encode quality, and ≫ block-local baselines — the gap is the value of cross-attention. ~5× lower TTFT, no fine-tuning, no full re-encode.

Limitations of reasoning systems

- Reasoning is expensive!
  - 100s of LLM calls for single queries
- As context length increases performance degrades
  - Particularly for cost-effective smaller models.

  Solution: Do not always generate, Directly retrieve from long-context
- Wasteful to repeatedly encode different data subsets via the LLM.

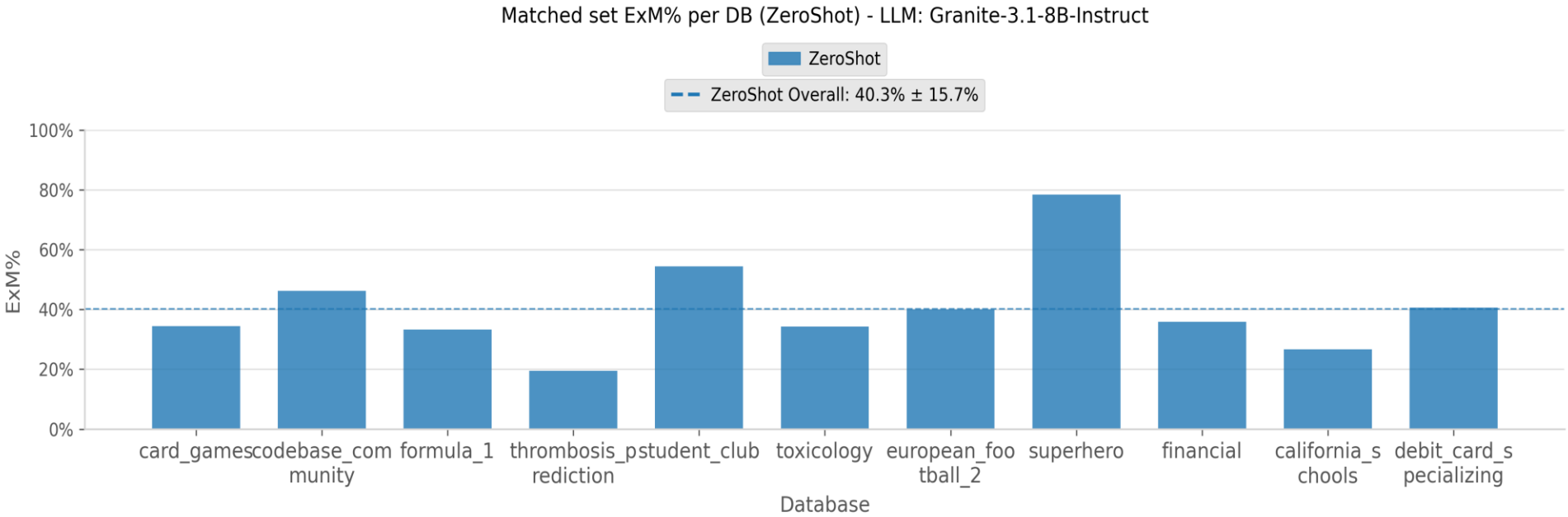
  Solution: Encode once and cache. Reuse with partial stitching.
- KV caches incur huge storage overheads.

  Solution: KV cache compression a hot research topic.
- Why do we treat each query as a fresh memory-less new call?

Learning from Feedback:
A case study: Enterprise Text-to-SQL

Off-the-Shelf Text-to-SQL generation

- Not accurate enough
- High variation in accuracy across schema



Current Customization Options

Fine-tuning

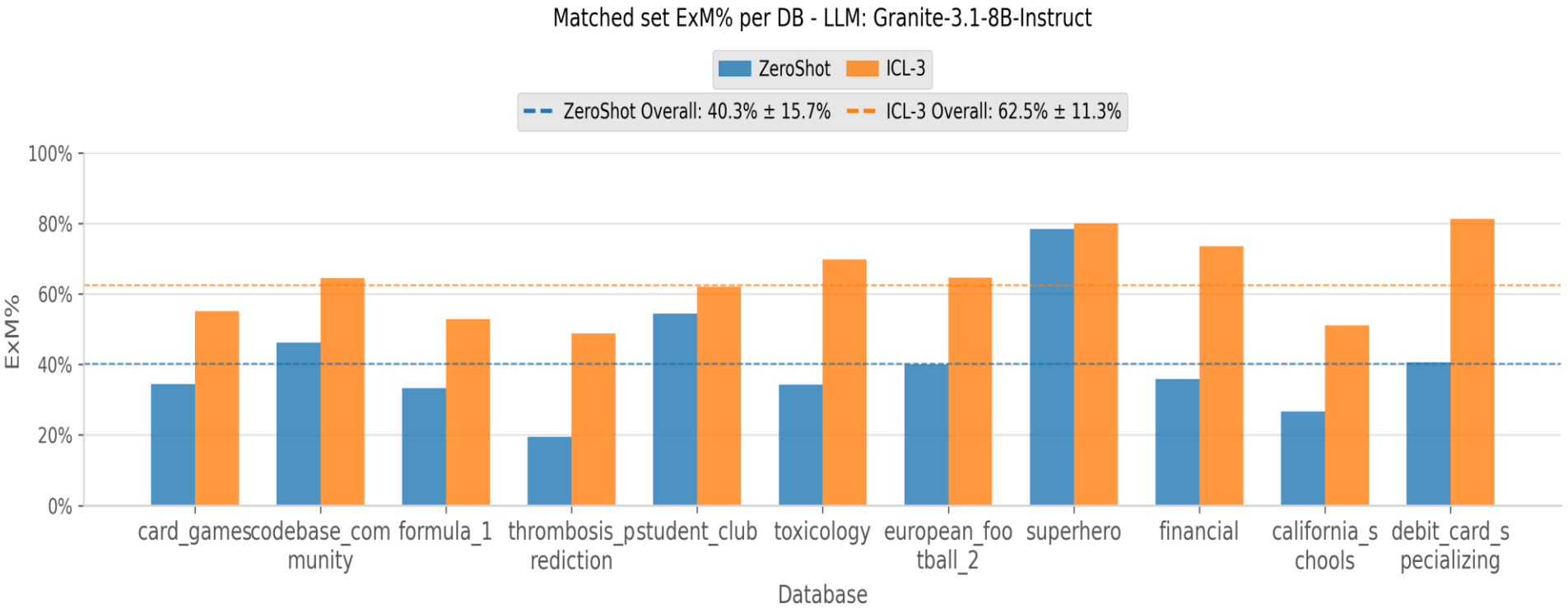
- Requires substantial amount labeled data upfront.
- Costly for smaller enterprises.
- May lead to the model "forgetting" how to perform other tasks.

In-Context Learning (ICL):

- Adapts the LLM for new databases "on-the-fly" by providing a few labeled examples in the prompt.

In-Context Learning helps

Overall accuracy improves from 40% to 62%



ICL Fails to Harness Highly Related Queries

Question: How many K-12 schools in Contra Costa register more than 420 free meals but have free or reduced-priced meals numbering under 610?

```
SELECT COUNT(CDSCode) FROM frpm WHERE `County Name` = '
     Contra Costa' AND `Free Meal Count (K-12)` > 420 AND
      `FRPM Count (K-12)`< 610
```

Question: In Los Angeles how many schools have more than 500 free meals but less than 700 free or reduced price meals for K-12?

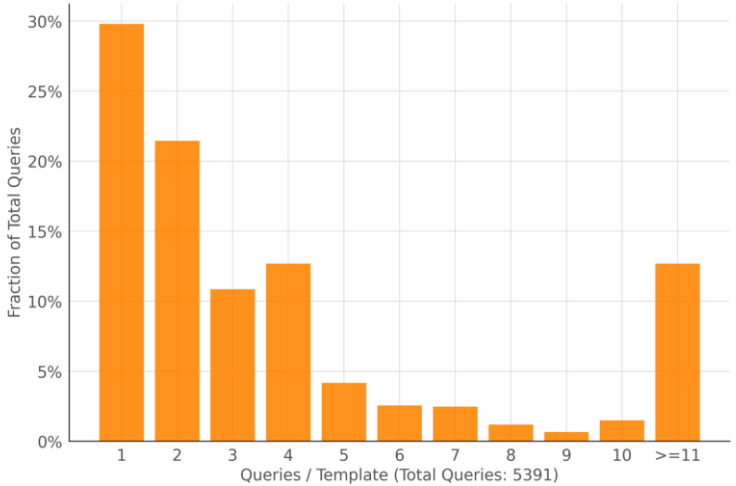
```
SELECT COUNT(CDSCode) FROM frpm WHERE `County Name` = '
     Los Angeles' AND `Free Meal Count (K-12)` > 500 AND
     `Enrollment (K-12)` < 700
```

Enterprise workloads contain repeated SQL patterns

Proprietary query workload from a large bank

More than 50% queries would find a previously seen query with a matching template.

Challenge: Large Natural language variation even for the same SQL



Challenging to infer that these two queries follow the same SQL structure

Question: How many K-12 schools in Contra Costa register more than 420 free meals but have free or reduced-priced meals numbering under 610?

Question: In Los Angeles how many schools have more than 500 free meals but less than 700 free or reduced price meals for K-12?

TeCOD: Reliable Answers for Recurring Questions with Template Constrained Decoding (SIGMOD 2026)

Checkout our talk and poster on June 2nd.

Three-step fix

I: Generalize labelled examples to templates

x^2 : Is molecule TR183 known to be a carcinogen?

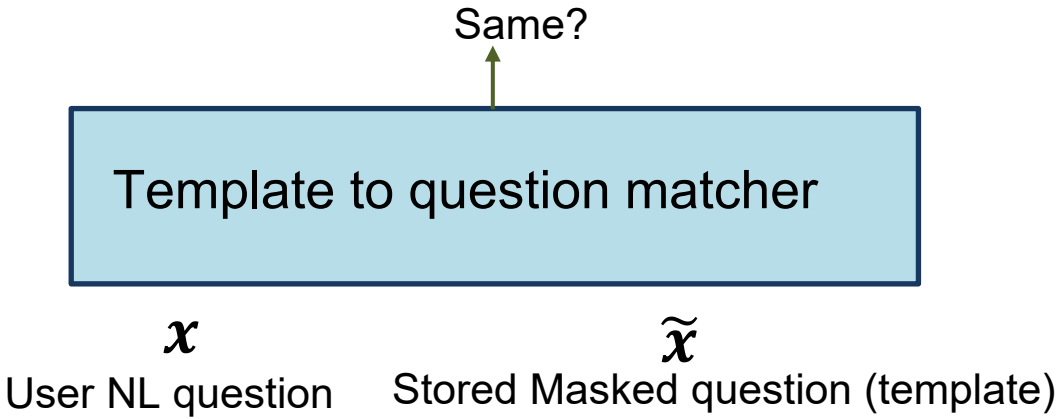
y^2 : SELECT T.label FROM molecule AS T WHERE T.molecule_id = 'TR183'

$\widetilde{y^2}$: SELECT T.label FROM molecule AS T WHERE T.molecule_id = String

$\widetilde{x^2}$: Is molecule String known to be a carcinogen?

Three-step fix

II: Custom matching of test questions with templates



Repurpose a Natural language inference model for the task

III: Template constrained decoding

If close enough template $\tilde{y}$ found, constrain generated SQL to match template

- Requires extending default LLM generation code for template constrained generation.

Encode template as a grammar

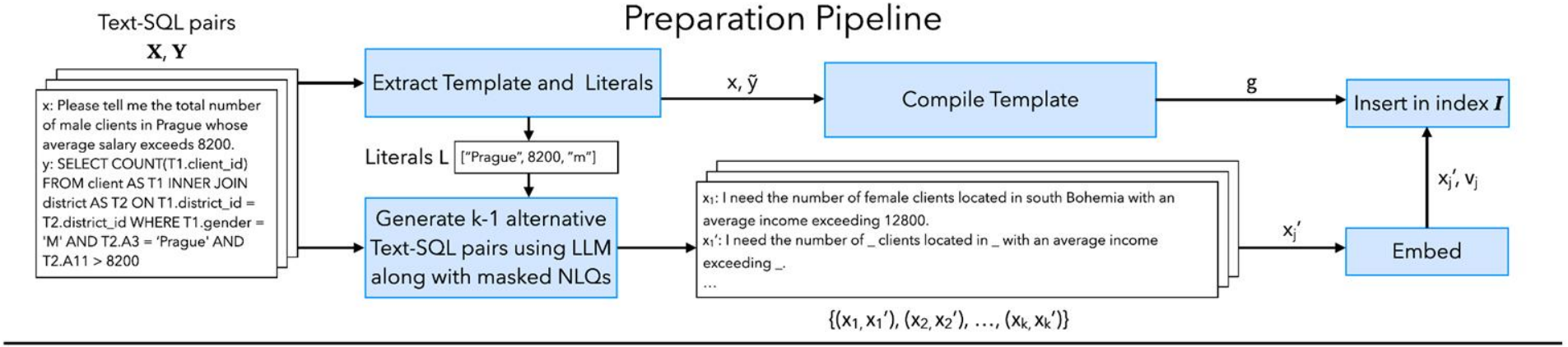
SELECT COUNT (T1.account_id
…..T1.amount < Number”

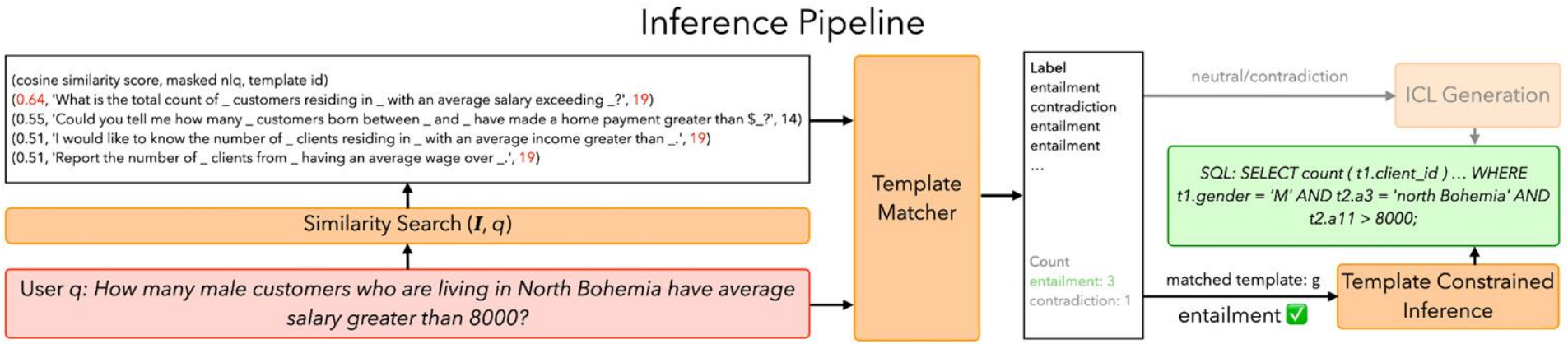
SELECT WS "COUNT" WS? "(" WS?
"T1.account_id" …..”T1.amount" WS? "<"
WS? “[\d]+”

Efficient libraries exist for grammar constrained decoding

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning. EMNLP 2023

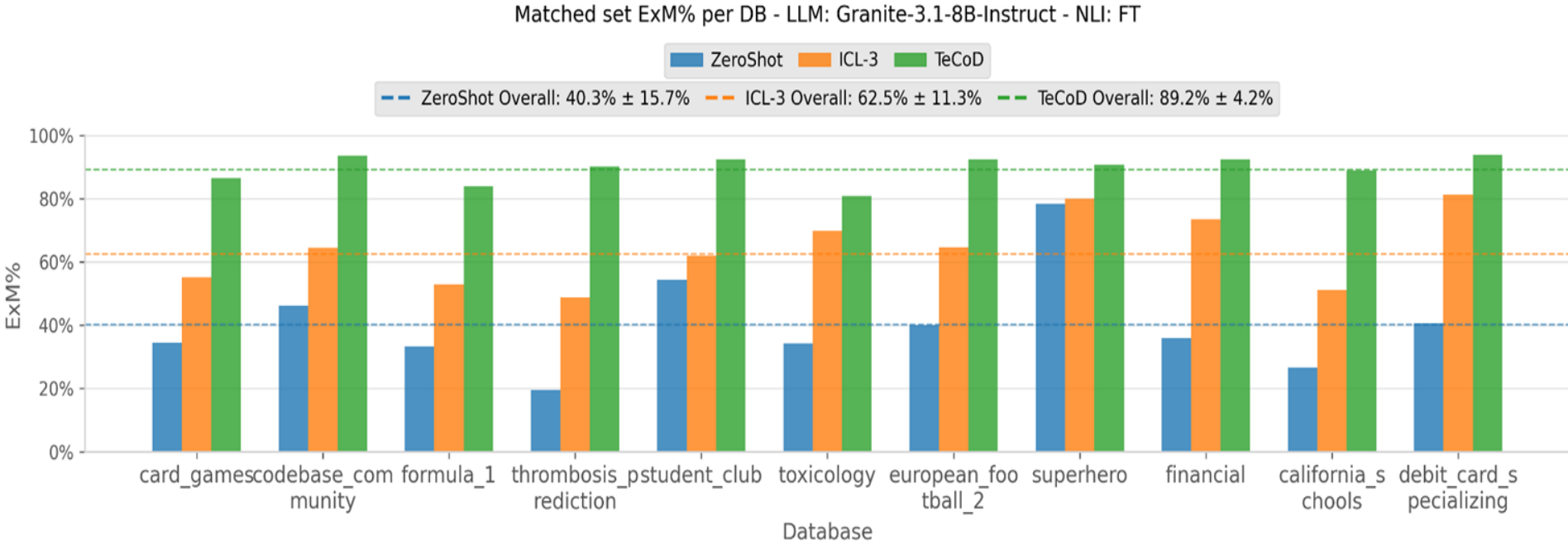
TeCOD for Template Constrained Decoding





Better usage of memory helps much more than default In-Context-Learning.

Huge jump in accuracy reaching almost 90% even with small models



Concluding Remarks

- LLMs not designed for co-existing with enterprise data ecosystem.
- Many attempts at retrofitting them seem to be needlessly contorted leading to huge costs
- An AI model designed natively for embedding in an enterprise would be different…

Towards a better LLM—Data Bridge

- Initiation step: LLM efficiently and thoroughly reads and understands the entire data eco-system in which it is operating
- Data: encoded once as compactly as inverted indices and as effectively as KV caches and directly used by LLM without fresh query-specific encodings.
- LLM retains memory of past explorations and can continuously and immediately absorb user feedback
- Uses memory to amortize on efficiency and reliability.

Thank you all for listening