AIDM '26 · SIGMOD WORKSHOP · BENGALURU

Recall Is Not Enough

Token-Centric Metrics for Agentic Schema Access

Ioana Giurgiu · **Michael Nidd**

IBM **Research** Europe

MOTIVATION

The Agent–Data Mismatch

Agents iterate. Databases were optimized for one-shot queries. The bottleneck moved from disk I/O to token cost.

Metadata vs. Context

10k-table catalogs overflow context windows and trigger hallucination.

I/O vs. Tokens

Optimizers minimize CPU + disk. Agents pay per token — linearly, every step.

Execution vs. Exploration

Agents need semantic context to pick the next action, not exact result sets.

THE PROBLEM

Why Recall Alone Is Misaligned

100%

recall under traditional evaluation

...can be the WORST option once you count tokens.

1 **Non-monotonic usefulness**

More context ≠ better reasoning — irrelevant tokens dilute attention.

2 **The full-schema baseline is dominated**

It maximizes B_N and κ for the same recall an aware method approaches.

3 **Single-step evaluation hides compounding**

Per-step gaps scale linearly across N reasoning steps.

OUR ARGUMENT

Two Intertwined Contributions

01

A Reporting Standard

Three token-centric metrics — η, B_N, κ — that compose existing quantities (recall, tokens, API price) into a comparable template.

→ Reverses method rankings

02

A Prototype Pruning Layer

Bi-encoder retrieval + cross-attention + schema graph closure. Lightweight, per-step, no LLM calls at pruning time.

→ Shows the metrics can guide design

CONTRIBUTION 1

Three Token-Centric Metrics

η

Recall-per-Token

$$\eta = Recall(C') \,/\, Tokens(C')$$

CAPTURES

Information density

Penalizes high-recall methods that retain excessive metadata.

B_N

Cumulative Burden

$$B\_N = \Sigma\ Tokens(C'_i) + Tokens(result_i)$$

CAPTURES

Trajectory cost

Per-step savings compound across an N-step ReAct cycle.

κ

Cost per Step

$$\kappa = p_{in} \cdot tokens + p_{out} \cdot gen + c\_prune$$

CAPTURES

Deployment cost

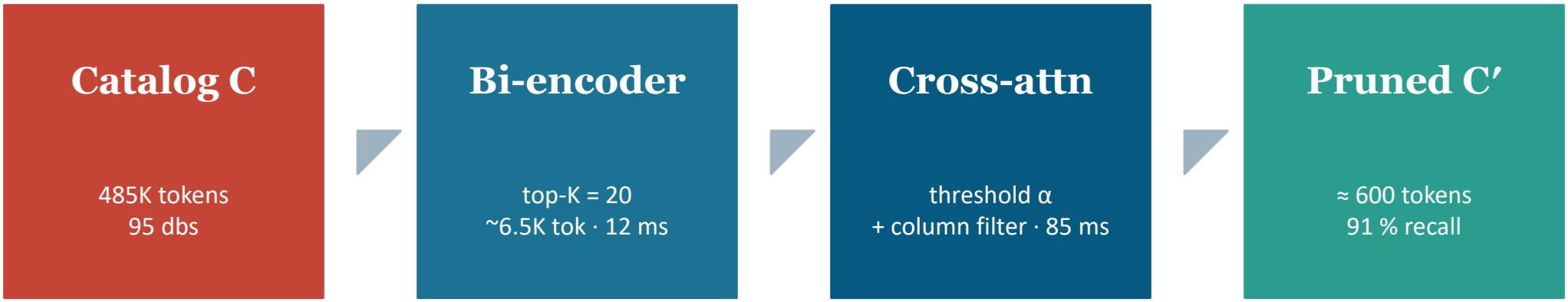
Real $/step under API pricing, including pruning overhead.

Each axis is independently optimizable. A method must score well on all three.

CONTRIBUTION 2

A Token-Aware Pruning Layer

Coarse-to-fine: cheap recall first, then deep alignment on the survivors.



No LLM calls at pruning time

No fine-tuning off-the-shelf models

≈ 100 ms total overhead

Per-step primitive

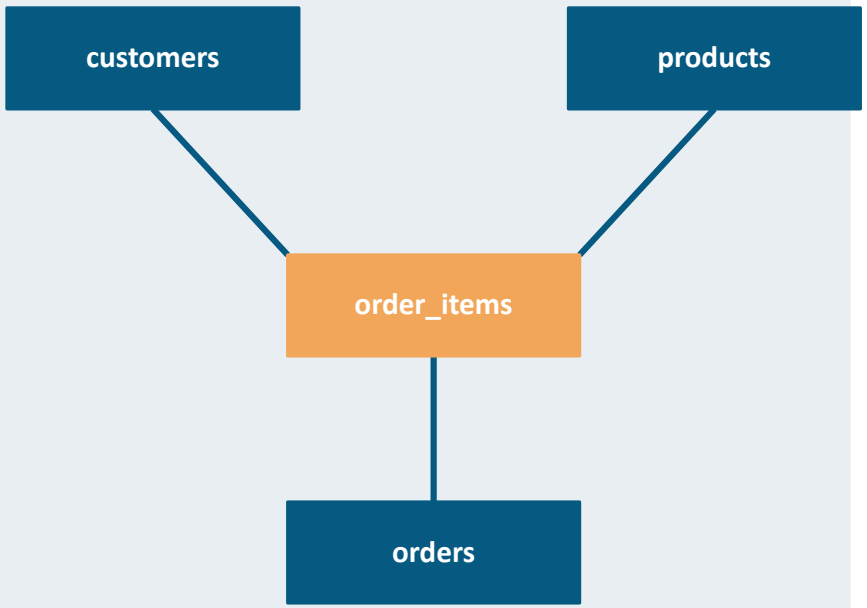
WHY SEMANTICS ALONE FAIL

Schema Graph Closure

Semantic similarity misses bridge tables.

A query about "customer lifetime value" needs order_items as a join bridge — even though it has zero semantic overlap with the intent.

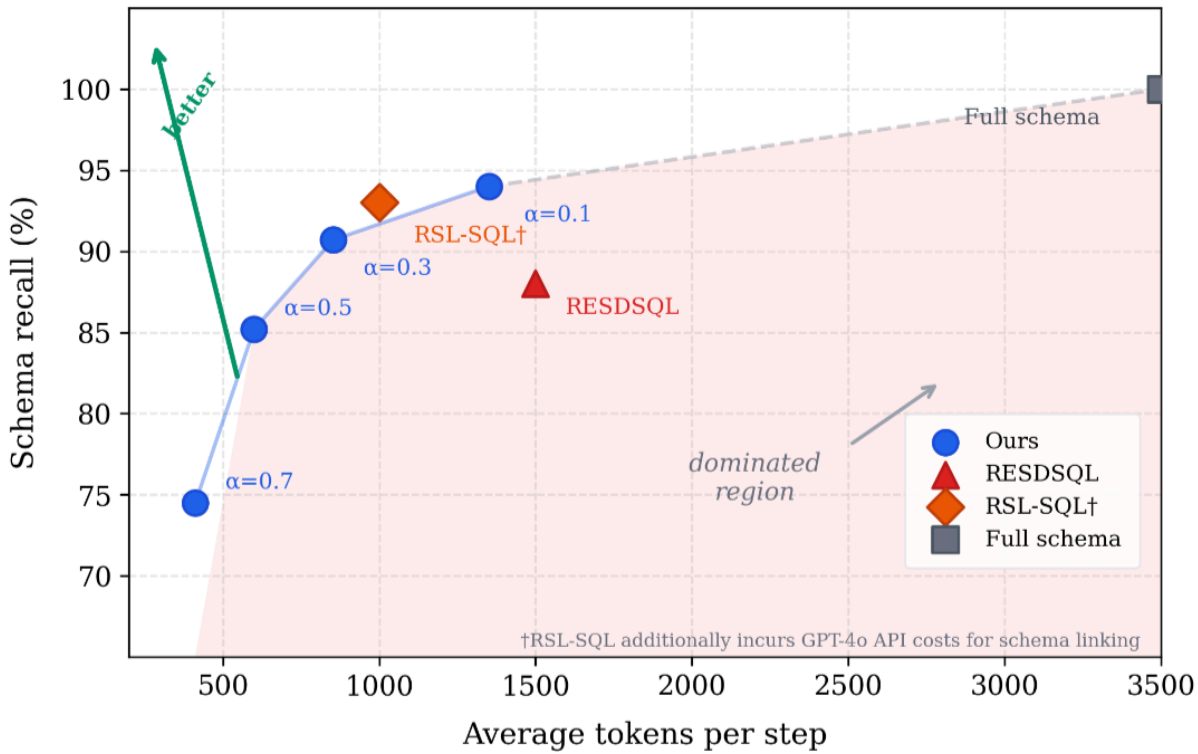
1. After cross-attention, take semantically scored tables T_s
2. Compute Steiner tree over the FK graph connecting T_s
3. Promote any bridge table on the tree — regardless of its score
4. Keep PK / FK columns + columns above relaxed α_bridge



order_items has no semantic overlap — promoted by graph closure

RESULTS ON BIRD-SQL (1,000 QUERIES)

The Ranking Reverses



Ranking under each metric

By RECALL alone

Full ▶ RSL ▶ RESDSQL ▶ Ours

Full-schema looks optimal

but maximizes B_N and κ

By η (recall-per-token)

Ours ▶ RSL ▶ RESDSQL ▶ Full

2.4–7× advantage in η

2.5–8.3× lower $\hat{B}_{10}$ over a ReAct loop

VALIDATION

η Predicts Cost-Adjusted Accuracy

End-to-end SQL on BIRD-SQL with Claude. The method with the highest η also delivers the highest execution-accuracy-per-token.

Condition	acc (%)	η (%/tok)	η_{acc} (%/tok)	Tokens (avg.)
Full schema	57.3	0.020–0.050	0.016–0.029	2K–5K
$\alpha = 0.3$	58.8	0.106	0.069	853
$\alpha = 0.5$	55.1	0.142	0.094	600

Table 3: Downstream SQL execution accuracy using Claude across three pruning conditions (without evidence). η_{acc}: execution-accuracy-per-token.

Raw accuracy gap ≈ 3.7% → η_acc gap = 3–6 × · The token-aware ranking is the correct one.

TAKEAWAYS

Recall is not enough.

01 Token-centric metrics reverse rankings

η, B_N, κ expose what recall hides — adopt them as a reporting standard.

02 Pruning is feasible without LLM calls

Bi-encoder + cross-attention + graph closure: 3.3–8.3× compression, ≈100 ms, no training.

03 η predicts η_acc

Methods optimal under η also win on cost-adjusted SQL accuracy.

Thank you · Questions?