Can AI Autonomously Build, Operate, and Use the Entire Data Stack ?

Lisa Amini
Distinguished Engineer, Director, Data & AI Platforms
IBM – Cambridge, MA

AI is Changing the Data Landscape

What's Changing

New Types of Data

Unstructured, semi/structured, multimodal & complementary taking center stage

New Workloads

Gen AI workloads (RAG, agents) are being deployed at scale

New User Experiences

Co-pilots are democratizing development and data analytics

New Interactions

AI agent interactions are often more speculative, dynamic, and heterogenous

Learnable components

AI is becoming more capable at exploration and design of software components and configurations

What's Still Critical

Price-Performance

Cost-effective at scale without sacrificing compliance

Open Source

Standards (file formats, catalos, query engines) take the lead

Hybrid data sprawl

Across clouds, systems, applications, and environments

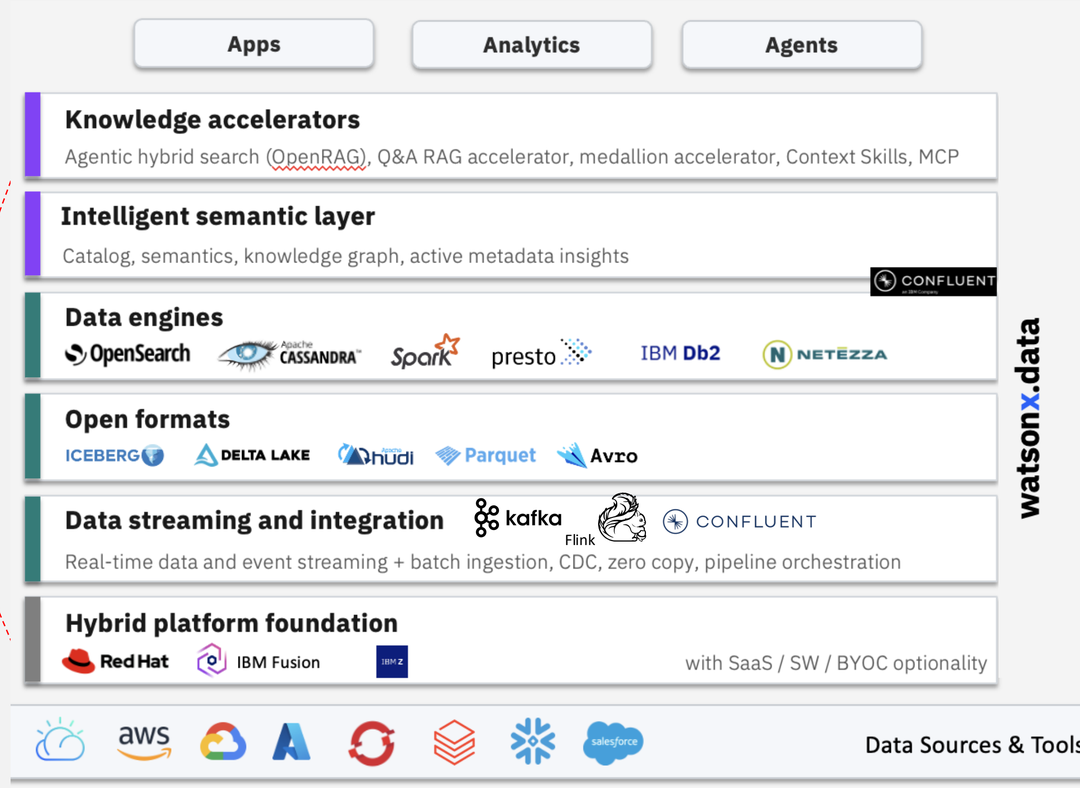
AI for Data Management and Operations

Complementary "stack of tasks" often performed by humans to design and manage Data Systems stacks

- Design and implement data infrastructure
- Update and optimize existing data infrastructure & storage
- Discover data sources and information (*e.g. Search, Enrichment*)
- Create multi-modal data flow (*e.g. Ingestion, Integration, DPH, DQ*)
- Monitor and remediate data flows *across the data life-cycle*
- Ensure governance requirements are enforced
- Uncover actionable insights (*and potential for decision making*)

**

Enterprise Data Stack



**Inversion of layers to illustrate underpinnings (storage, infra, ...) required before upper layers (query engines and interfaces) can be configured and leveraged.*

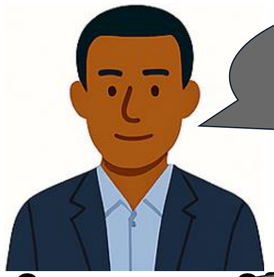
Can AI Autonomously Build, Operate, and Use the Entire Data Stack?

Arvind Agarwal, Lisa Amini, Sameep Mehta, Horst Samulowitz, Kavitha Srinivas
IBM Research
{arvagarw, sameepmehta}@in.ibm.com, {lisa.amini, hsamulo}@us.ibm.com, kavitha.srinivas@ibm.com

Abstract

Enterprise data management is a monumental task. It spans data architecture and systems, integration, quality, governance, and continuous improvement. While AI assistants can help specific persona, such as data engineers and stewards, to navigate and configure the data stack, they fall far short of full automation. However, as AI becomes increasingly capable of tackling tasks that have previously resisted automation due to inherent complexities, we believe there is an imminent opportunity to target fully autonomous data estates.

An illustrative example in the context of creating a data stack for a business use case of financial trading



Compliance anyone? Before we ship anything...

Those ingest pipelines are flaky...

Data Governance

Data Lifecycle Engineer

Ok, prepared a bunch of

Let's add SEC Filings and now got budget for Bloomberg...

Given the available data

Ok, for that type of data I need to update the data infrastructure

Need to request more data...

Only got slightly better...

Data Infrastructure Architect
Designs and builds data infrastructure

Data Acquisition Specialist
Identifies, enriches, evaluates quality of data sources

Data Engineer
Creates multi-modal data flows (e.g., integration)

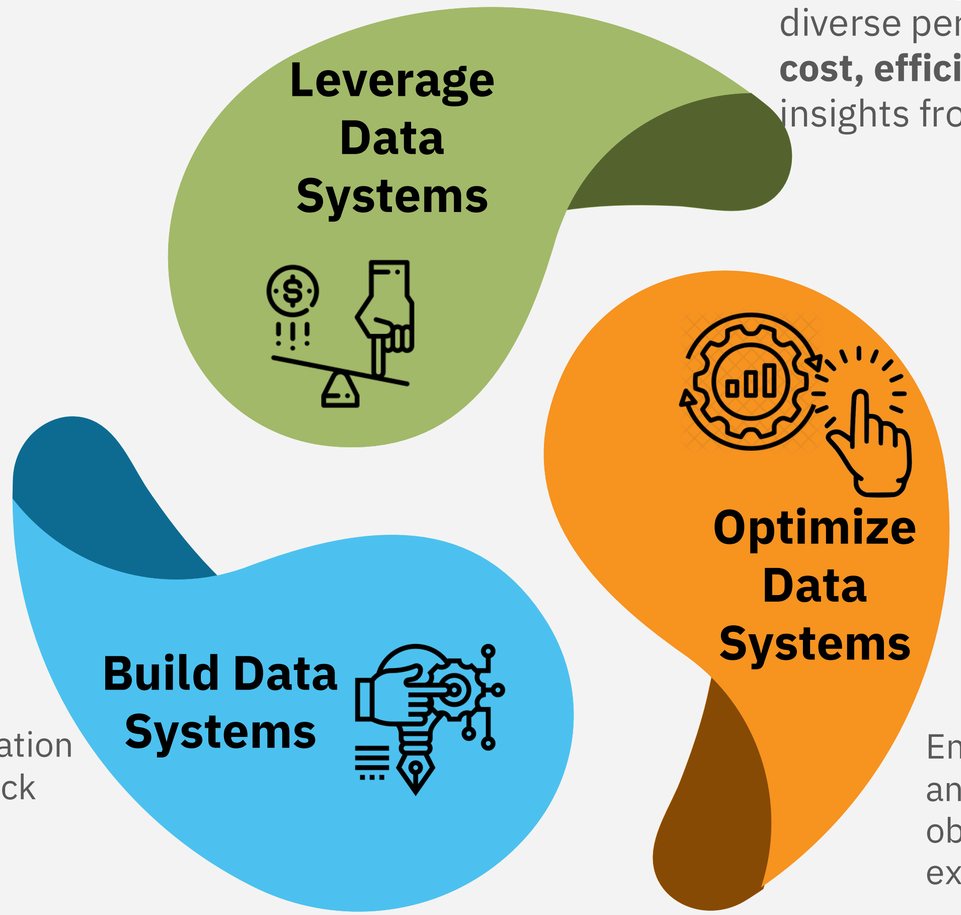
Insight/Decision Analyst
Uncovers actionable Insights

To keep up with the scale of data, let's add some VMs and find better ways to partition and to index the data...

Data Reliability Engineer
Maintains and optimizes existing storage/infrastructure

AI for Data Focus

AI builds, operates, maintains, and leverages data



Enable large scale use of data systems by diverse personas optimizing for **accuracy, cost, efficiency** in retrieval, integration, insights from semi-/un-/structured data

Enable end-2-end Data System creation given user intent, available data stack components, functional and non-functional requirements.

Enable Error Remediaton, Optimization and Redesign of Data System leveraging observability data to improve user experience.

Towards Autonomous Data Management and Operations

Use Data Systems

Given a NL query:

```
List the key metric values and analyst summary for top 5 performing mutual funds by YTD total returns
```

AI generates queries to retrieve data, generates response, potentially with charting

Given a NL request for insights:

```
Show the top 10 brands by revenue with their review sentiment and average rating and build a model to predict gross profit from unit price and quantity
```

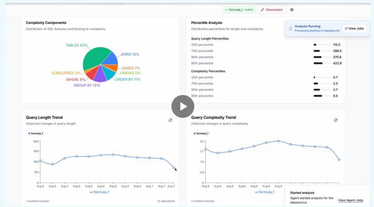
AI generates plan and dataflow over heterogenous data sources, to produce **reliable and robust** retrieval, **integration, and insights**

Build Data Systems

Given a data product topic:

```
Create financial data product that forecasts performance across national and international mutual funds.
```

AI discovers data sources, generates and iteratively refines data product & governance metadata



Given an AI application target:

```
Create a data stack for a RoboTrading App using public sources for market data, news & sentiment, and fundamentals to support agentic workflows for periodic 401k rebalancing.
```

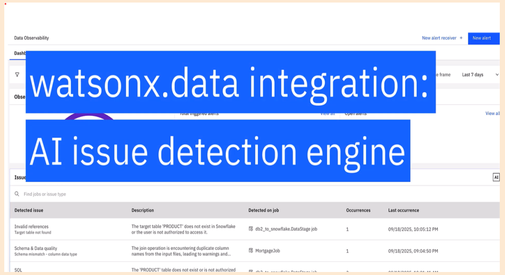
AI selects, configures, deploys appropriate data stack components, and generates data product(s)

Optimize Data Systems

Given a dataflow issue:

```
Target table not found
```

AI classifies issue, generates diagnostics, root cause analysis, affected jobs,…



Given dataflow issue:

```
Excessive timeouts
```

AI generates & executes plan for resolution; generates dataflow script repairs; what-if alternative stack configurations

Project: BEACON

Project: RADAR

Work in Progress

L1
User as an **Operator**
User directs and makes decisions, agent acts

L2
User as a **Collaborator**
User and agent collaboratively plan, delegate, and execute

L3
User as a **Consultant**
Agent takes lead but consults user for expertise/preferences

L4
User as an **Approver**
Agent engages user only n risky or pre-specified scenarios

L5
User as an **Observer**
Agent operates within full autonomy under user monitoring

User Involvement

Agent Autonomy

Levels of Autonomy for AI Agents, arxiv

Autonomous Data Product: Generation & Assessment

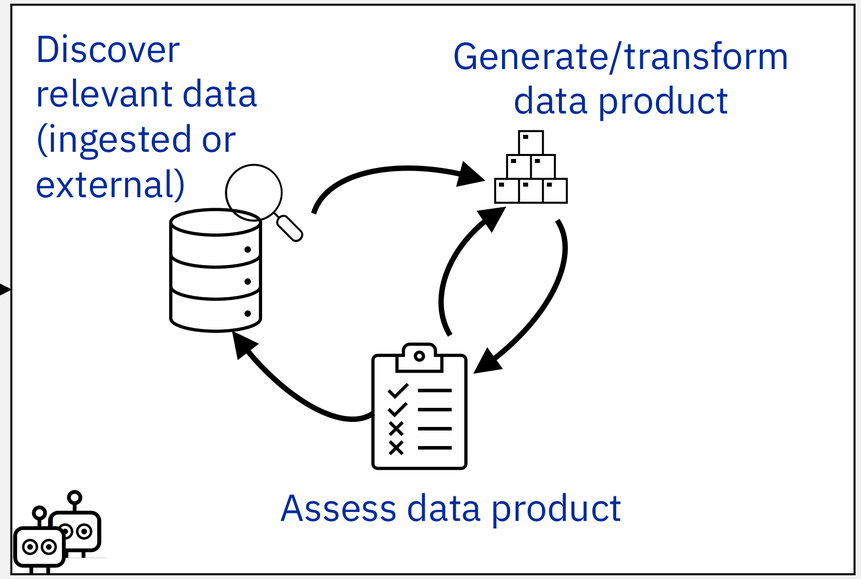
Build Data Systems

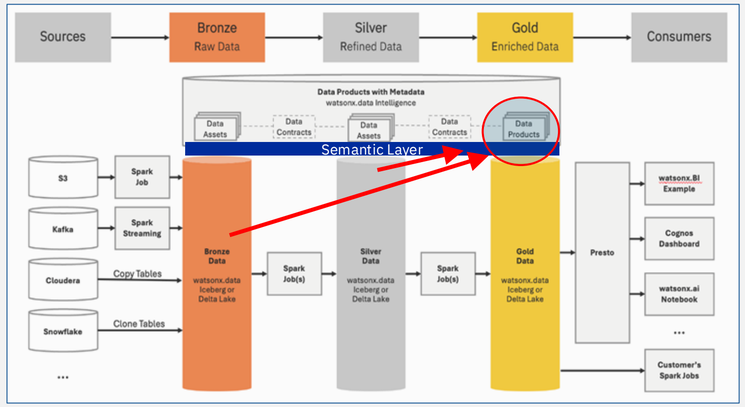
User

Agent

New requests or issue resolution

Create financial data product that forecasts performance across national and international mutual funds. Additional requirements include low latency, GDPR compliance, price-performance, high data quality, restricted user access.





Medallion Architecture

FROM FACTOID QUESTIONS TO DATA PRODUCT REQUESTS: BENCHMARKING DATA PRODUCT DISCOVERY OVER TABLES AND TEXT

Liangliang Zhang
Rensselaer Polytechnic Institute
Troy, NY
zhangl41@rpi.edu

Nandana Mihindukulasooriya
IBM Research
New York, NY
nandana@ibm.com

Niharika S. D'Souza
IBM Research
San Jose, CA
Niharika.DSouza@ibm.com

Sola Shirai
IBM Research
Yorktown Heights, NY
solashirai@ibm.com

Sarthak Dash
IBM Research
New York, NY
sdash@us.ibm.com

Yao Ma
Rensselaer Polytechnic Institute
Troy, NY
may13@rpi.edu

Horst Samulowitz
IBM Research
Yorktown Heights, NY
samulowitz@us.ibm.com

StructText: A Synthetic Text-based Approach for IE Benchmarks from Tabular Data with Multi-Dimensional Evaluation

Satyananda Kashyap
IBM Research
satyananda.kashyap@ibm.com

Sola Shirai
IBM Research
solashirai@ibm.com

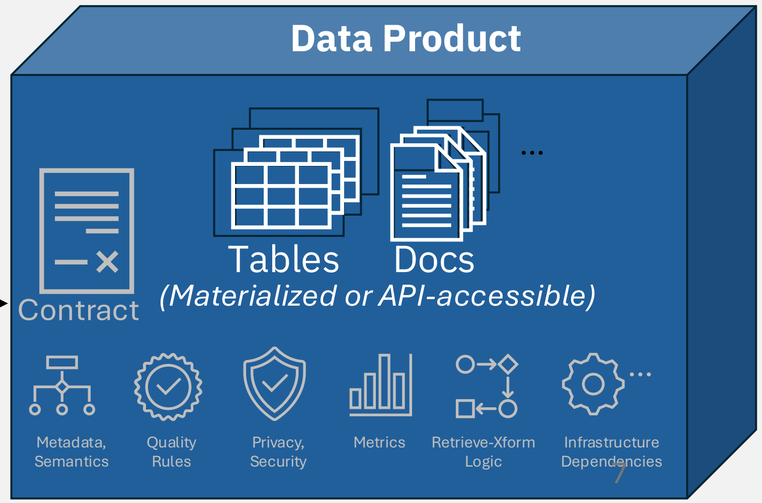
Nandana Mihindukulasooriya
IBM Research
nandana@ibm.com

Horst Samulowitz
IBM Research
samulowitz@us.ibm.com

ABSTRACT

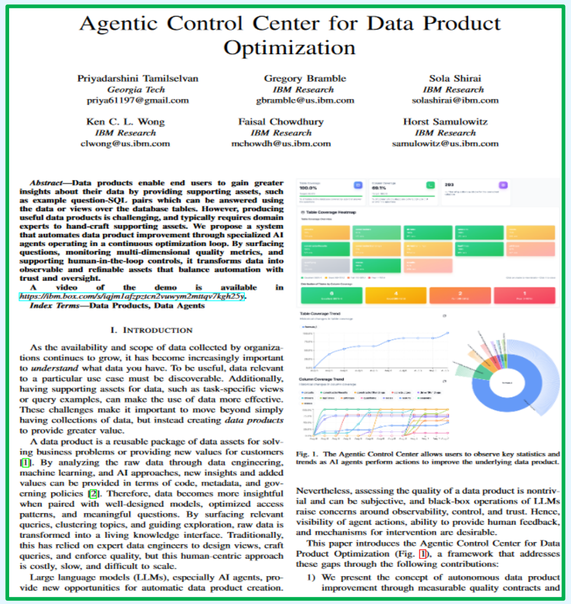
Extracting structured information from text, such as key-value pairs that could augment tabular data, is quite useful in many enterprise

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://huggingface.co/datasets/ibm-research/struct-text.



Data Product Insight Assessment

Tools & Metrics Impact



Agentic Control Center for Data Product Optimization

Priyadarshini Tamilselvan, *Georgia Tech*, priya61197@gmail.com
Gregory Bramble, *IBM Research*, gbramble@us.ibm.com
Sola Shirai, *IBM Research*, solashirai@ibm.com
Ken C. L. Wong, *IBM Research*, clwong@us.ibm.com
Faisal Chowdhury, *IBM Research*, mchowdh@us.ibm.com
Horst Samulowitz, *IBM Research*, samulowitz@us.ibm.com

Abstract—Data products enable end users to gain greater insights about their data by providing supporting assets, such as example question-SQL pairs which can be answered using the data or views over the database tables. However, producing useful data products is challenging, and typically requires domain experts to hand-craft supporting assets. We propose a system that automates data product improvement through specialized AI agents operating in a continuous optimization loop. By surfacing questions, monitoring multi-dimensional quality metrics, and supporting human-in-the-loop controls, it transforms data into observable and refinable assets that balance automation with trust and oversight.

A video of the demo is available in https://ibm.box.com/s/iqim1afzgzcm7swwym2mtiqv7kgh25y.

Index Terms—Data Products, Data Agents

I. Introduction

As the availability and scope of data collected by organizations continues to grow, it has become increasingly important to *understand* what data you have. To be useful, data relevant to a particular use case must be discoverable. Additionally, having supporting assets for data, such as task-specific views or query examples, can make the use of data more effective. These challenges make it important to move beyond simply having collections of data, but instead creating *data products* to provide greater value.

A data product is a reusable package of data assets for solving business problems or providing new values for customers [1]. By analyzing the raw data through data engineering, machine learning, and AI approaches, new insights and added values can be provided in terms of code, metadata, and governing policies [2]. Therefore, data becomes more insightful when paired with well-designed models, optimized access patterns, and meaningful questions. By surfacing relevant queries, clustering topics, and guiding exploration, raw data is transformed into a living knowledge interface. Traditionally, this has relied on expert data engineers to design views, craft queries, and enforce quality, but this human-centric approach is costly, slow, and difficult to scale.

Large language models (LLMs), especially AI agents, provide new opportunities for automatic data product creation.

Fig. 1. The Agentic Control Center allows users to observe key statistics and trends as AI agents perform actions to improve the underlying data product.

Nevertheless, assessing the quality of a data product is nontrivial and can be subjective, and black-box operations of LLMs raise concerns around observability, control, and trust. Hence, visibility of agent actions, ability to provide human feedback, and mechanisms for intervention are desirable.

This paper introduces the Agentic Control Center for Data Product Optimization (Fig. 1), a framework that addresses these gaps through the following contributions:

1) We present the concept of autonomous data product improvement through measurable quality contracts and

ICDM'25 Demo

Tools

Question Generation
Automatically generates relevant questions

Follow-up Question Generation
Creates contextual follow-up questions

SQL Generation
Converts questions to SQL queries

View Creation
Creates optimized database views

Question Clustering / Topic Mapper
Groups similar questions in topics

Predictive Analytics
Group potential predictive targets

Metrics

Question Count
Total number of questions generated

Table Coverage
Percentages of database tables in data product utilized by generated questions

Column Coverage
Percentage of columns accessed in data product by generated questions

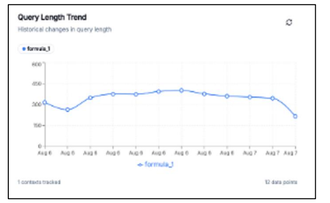
SQL Query Length
Average length of generated SQL (lower is better)

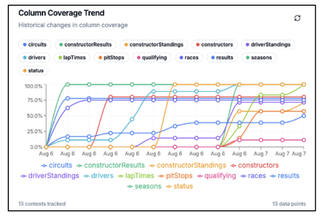
SQL Query Complexity
Complexity score of SQL queries (lower is better)

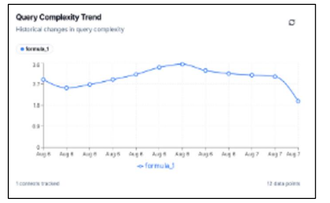
Query Speed
Average query execution time

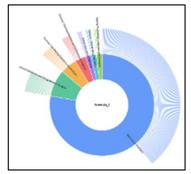
Topics Generated
Number of identified topics

Predictive Performance
Averaged predictive performance across all targets



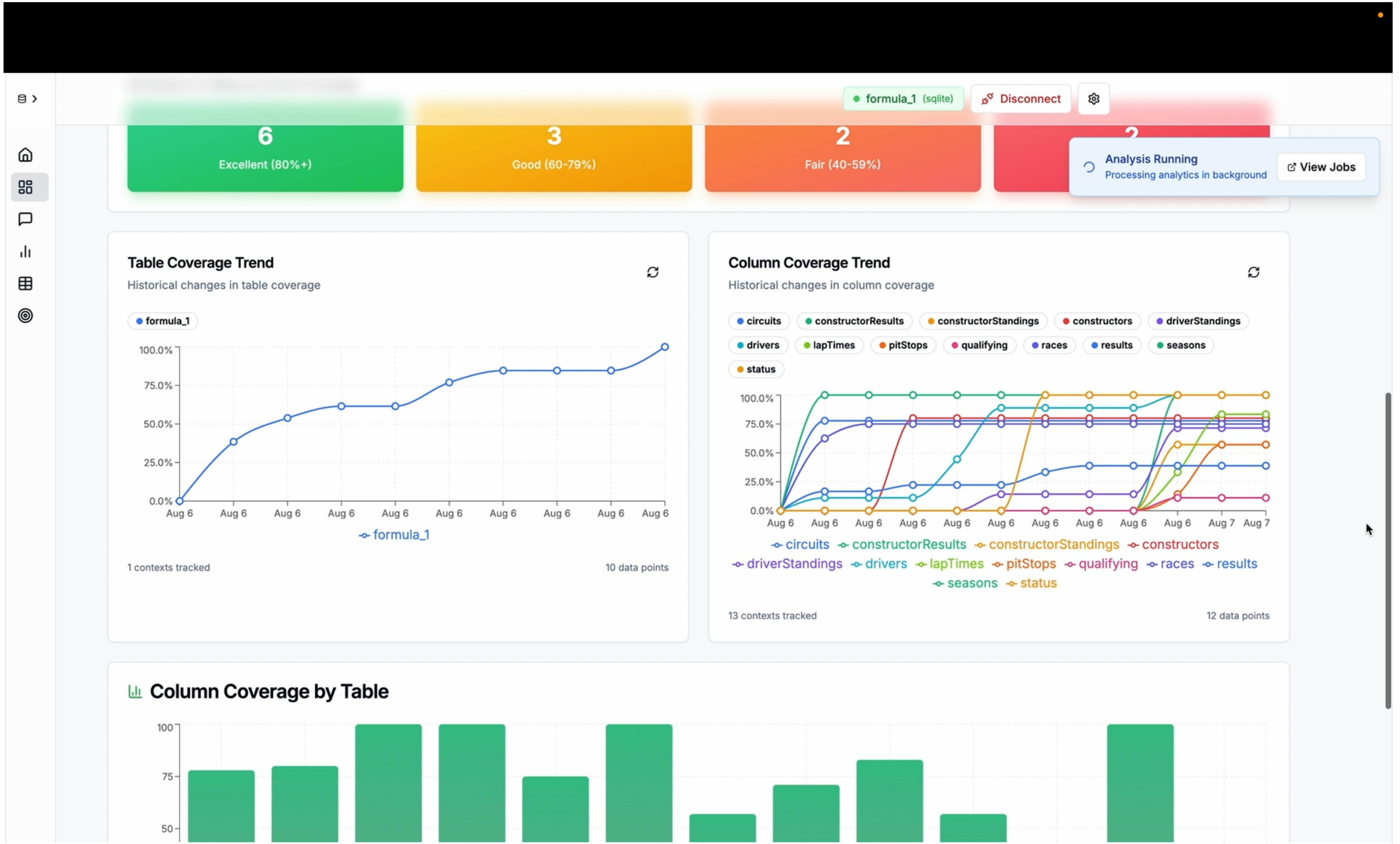






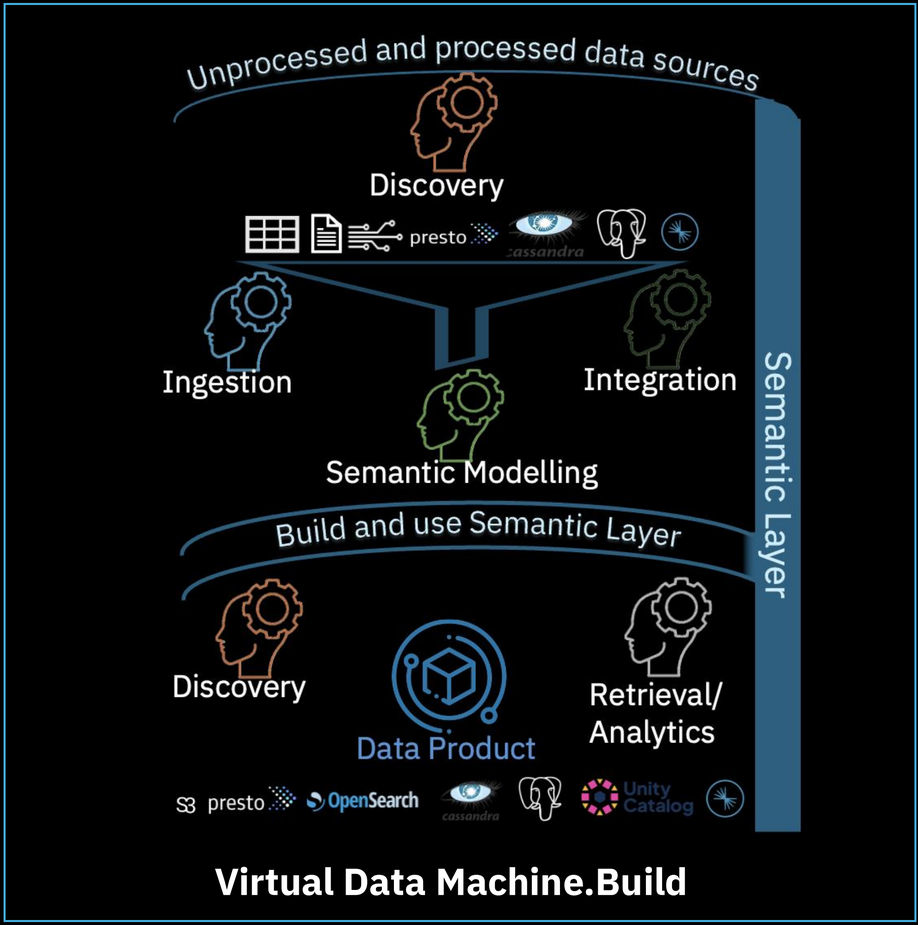
Legend
Positive impact
Streamlines query

Autonomous Data Product Refinement



Virtual Data Machine.Build

"I want to build an app that helps to predict 'time-to-fill' for job roles in our organization."



Virtual Data Machine.Build

Event-based Multi-Agent Orchestration

Discovery Agent

- Autonomously performs search for relevant data
- Use of either **intrinsic knowledge or search** to determine as many relevant data sources as possible

Produces Database/schema/table names, URLs, APIs

Ingestion Agent

- Autonomously perform **code generation** and **skill use** to collect data from sources
- Run **data sampling** and inspect file contents to **validate ingestion results** against request
- Store **data and code artifacts** for traceability and down-stream use

Produces data files, ingestion code, samples, tests, …

Semantic Modelling + Indexing Agent

- Agent-driven **storage and indexing strategies** based on **user intent and data properties**
- **Reuses** existing tools for processing data
- **Extracts metadata** (entities, concept hierarchy, topics) useful for downstream tasks
  - Data Discovery / Efficient Semantic Operators

Produces stored and indexed data, extracted metadata, concept hierarchies, topics, entities, reusable code for data prep

Architect Agent

- Agent-driven **data system design** based on **user intent and requirements**
- Uses **data characterization and related information** from other agents to inform design
- Performs **Open-ended** System Design with **Steer** as a service

Produces system design specification, refinement of individual components

AI agents are powerful, but expensive, black boxes

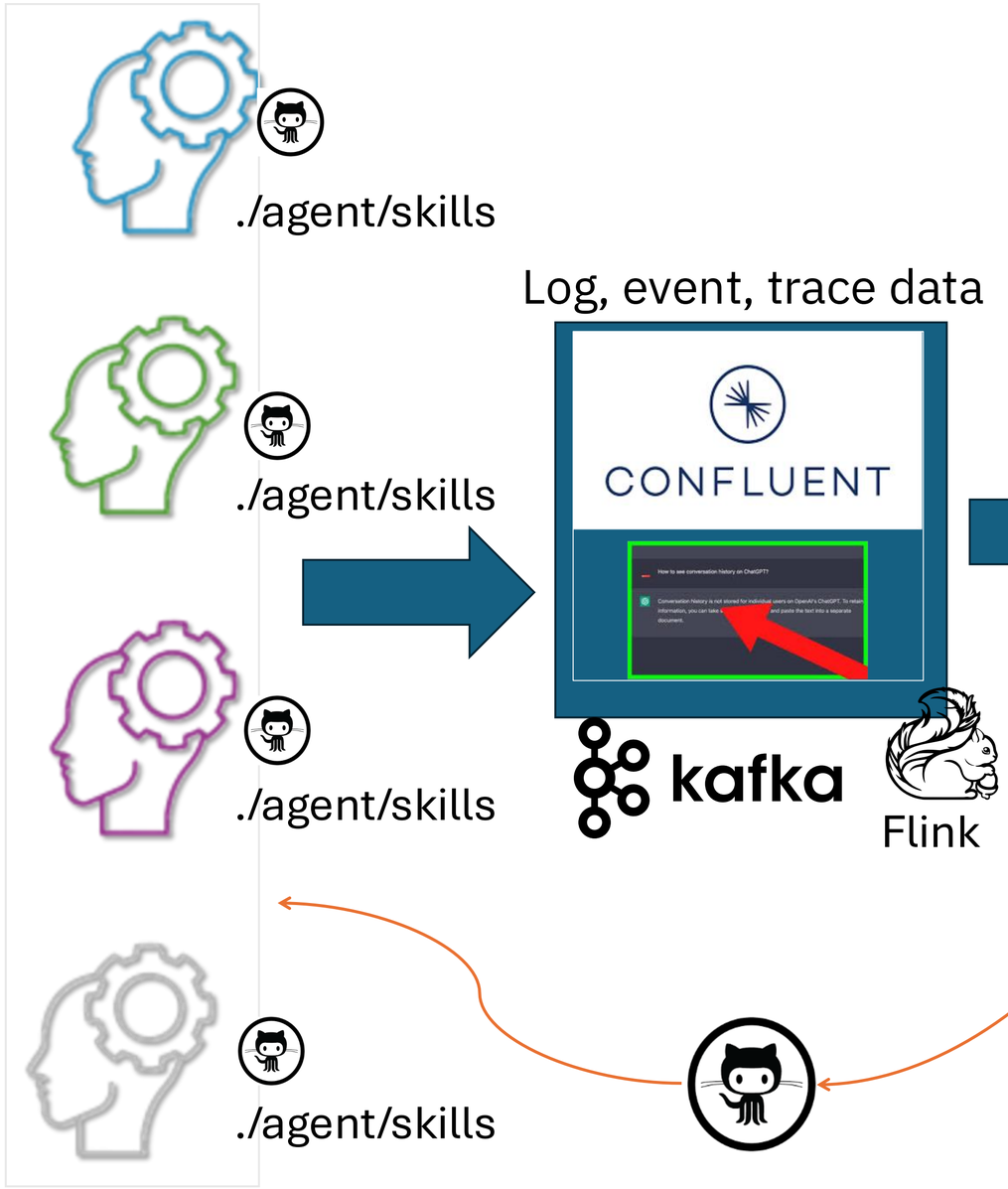
- Enterprises running agents at scale burn through millions of tokens per month
- Learning from agent interactions, at scale, is challenging
- Every agent session starts cold, and rediscovers skills, tools, trajectories, …
- The best models, tools, and skills for a given task are often discovered in production, instead of imagined by developers

Current Issues and Open Challenges with Agent Skills

Challenge Area	Description	Impact
Skill selection	Incorrect choice of skills can be a problem	Wrong or incomplete outcomes
Skill creation	Largely manual right now, with any extraction being mostly targeting 'prompt' extraction	Inefficient and not scalable
Less reliance on skills as code (MCP is not portable)	Skills as prompts can still produce nondeterminism	Limited value if not reliable
Observability and testing	Hard-to-trace if skill is used	Difficult debugging and evaluation
Token Cost	Large skills registry balloon costs	Skills loaded but never used

Key idea: **Generate skills as code, from agent logs.** Create skills across agents, monitor skill use, refactor skills.

Continual Skill Learning



Log Ingestion and Evaluation Flow

Decompose code and context into skills

Skills creation

Test generation (data skills need specialized tests)

Evaluate push

Monitor skill use across agents

Target skills for refactoring

Generalize skills

Continuous Improvement Pipeline

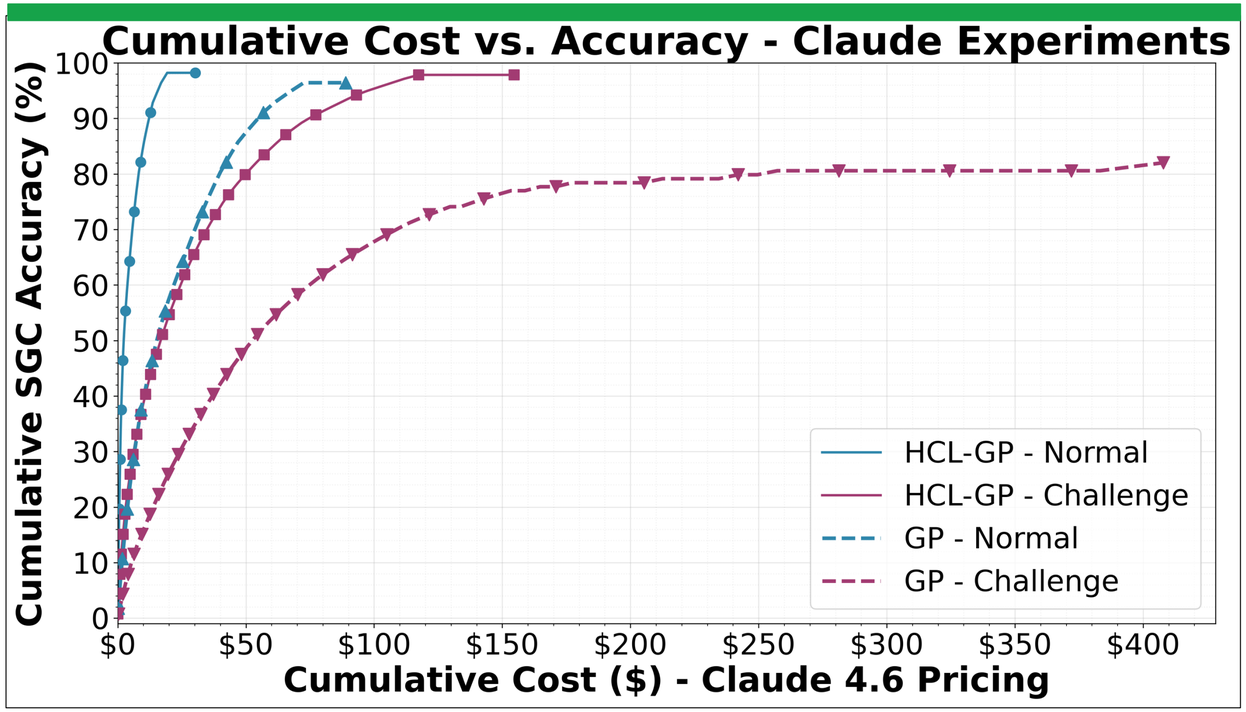
Kavitha Srinivas, Shirin Sohrabi, Michael Katz, Michael Glass, Harsha Kokel, Haritha Ananthakrishnan, Horst Samuolwitz

Evidence of technical effect from AppWorld Benchmark

Reported benchmark results

Method	Model	Normal		Challenge	
		TGC	SGC	TGC	SGC
Claude Code	Claude Opus 4.5	66.0	56.6	–	–
Smolagents Code	Claude Opus 4.5	70.0	60.4	–	–
OpenAI Solo	Claude Opus 4.5	68.0	56.6	–	–
ReAct	Claude Opus 4.5	61.0	52.8	–	–
ReAct Short	Claude Opus 4.5	64.0	54.7	–	–
Alibaba AgentRL	Qwen3-14B	86.9	80.4	67.6	50.4
IBM CUGA	GPT-4.1	73.2	62.5	57.6	48.2
LOOP	Qwen2.5-32B	72.6	53.6	47.2	28.8
GP (our baseline)	GPT-OSS 120B	0.0	0.0	0.7	0.7
GP (our baseline)	Claude Sonnet 4.6	96.4	96.4	83.2	82.0
HCL-GP	GPT-OSS 120B	62.5	62.5	41.0	41.0
HCL-GP	Claude Sonnet 4.6	**98.2**	**98.2**	**98.3**	**97.8**

Results submitted to leaderboard, not yet posted.



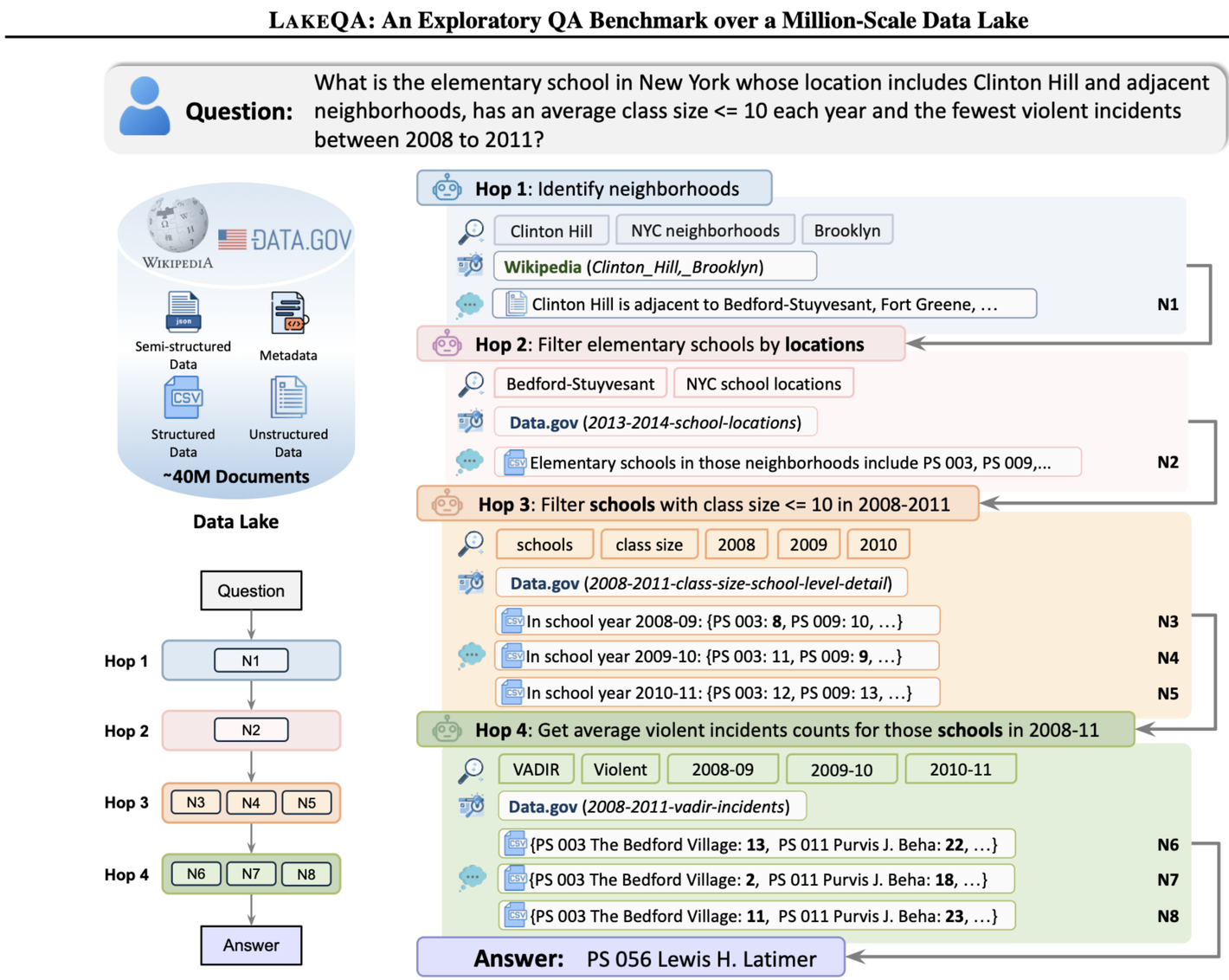
Learning and Reusing Policy Decompositions for Hierarchical Generalized Planning with LLM Agents

Shirin Sohrabi Haritha Ananthakrishnan Harsha Kokel Kavitha Srinivas Michael Katz
IBM

Hierarchical Component Learning for Generalized Policies (IBM HCL-GP)

https://arxiv.org/pdf/2605.06957v1

Current Focus: Synthesis and Refine Data Skills leveraging a variety of open & enterprise data



LakeQA (Columbia University)

- 9.5TB data lake
- Multi hop reasoning over tables and text
- Search as an API over Wikipedia/structured content

ICML'26 LakeQA: A benchmark for Complex Exploratory QA over a Million-scale Data Lake, Haonan Wang, et al (Eugene Wu)

https://arxiv.org/abs/2606.10460

Derived Functions

1. build_alert_breakdown_query()

Purpose: Build SQL queries to aggregate alerts by category and status for any alert source table within a date range

EVIDENCE FROM LOGS:

506a9fea-a082-4a94-b605-23ef1db40486.json — Lines 9397-9587

```
SELECT 'Quality Notifications' as alert_source, qmart as category, stat as status, co
```

Pattern Identified: Multiple alert source queries follow same pattern: aggregate by category and status with date filter

506a9fea-a082-4a94-b605-23ef1db40486.json — Lines 27301-27587

```
SELECT 'Production Failures' as alert_source, zfailure_reason as category, stat as st
```

Generated Code

```
"""
Cluster 4: Supply chain root cause analysis and quality failure

Functions to support multi-step root cause analysis workflows on
data, including quality notification analysis, production failur
carrier incident correlation, and QC inspection parameter failur
"""

import json
from typing import Any
from datetime import datetime, timedelta


# Alert source tables and their key fields
ALERT_SOURCES = {
    "quality_notifications": {"table": "qmel", "date_field": "er
    "quality_holds": {"table": "zquality_hold", "date_field": "e
    "carrier_incidents": {"table": "zcarrier_report", "date_fiel
    "production_failures": {"table": "afko", "date_field": "gstr
    "qc_inspection_failures": {"table": "qals", "date_field": "e
    "credit_blocks": {"table": "zcredit_block", "date_field": "z
    "pod_rejections": {"table": "zpod", "date_field": "lfdat", "
}

# Quality notification types
QMART_TYPES = {
    "Q1": "Customer Complaint",
    "Q5": "Audit Finding",
}


def build_alert_breakdown_query(source_name: str, date_from: str
    """
    Build a SQL query to get alert breakdown by category and sta
    for a specific alert source within a given date range.

    Returns a SQL string that aggregates counts by category and
```

Example Generated Data Skill

Example: Generalizing skills to maximize code usage

Generalize code skills

Ingestion agent

restCountries/main.py: Type conversion for population, area, coordinates
nominatim_geocoding/main.py: Safe type conversion with error handling
open_meteo_ingestion/main.py: Numeric type handling

Semantic modeling agent

type-inference/scripts/infer_types.py: Name-based and value-based type inference
csv-analysis/scripts/analyze_csv.py: Pattern-based type detection

Generalized Component: **UniversalTypeInferencer**
Pattern-based type detection (regex for dates, numbers, UUIDs)Safe type conversion with error handling
SQL type mapping

```
class UniversalTypeInferencer:
    """
    Universal type inference for any data source.
    Combines name-based heuristics with value-based analysis.
    """

    # Shared regex patterns
    ISO_DATE = re.compile(r"^\d{4}-\d{2}-\d{2}$")
    ISO_DATETIME = re.compile(r"^\d{4}-\d{2}-\d{2}[T ]\d{2}:\d{2}")
    UUID_PATTERN = re.compile(r"^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$", re.I)
    INTEGER_PATTERN = re.compile(r"^-?\d+$")
    DECIMAL_PATTERN = re.compile(r"^-?\d+\.\d+$")

    # Name-based rules (from type-inference)
    NAME_RULES = [
        (re.compile(r".*_(id|uuid|key|code)$", re.I), "VARCHAR(255)"),
        (re.compile(r"^(is|has|can|should)_", re.I), "BOOLEAN"),
        (re.compile(r".*_datetime$", re.I), "TIMESTAMP"),
        (re.compile(r".*_(amount|price|total|cost)$", re.I), "DECIMAL(15,2)"),
        # ... more rules
    ]

    def infer_type(self, field_name: str, values: List[Any]) -> Dict[str, Any]:
        """Infer SQL type using both name and values"""
        # Try name-based first
        name_result = self._infer_from_name(field_name)
        if name_result:
            sql_type, rule = name_result
            return {
                "sql_type": sql_type,
                "rule": rule,
```

Virtual Data Machine

AI-powered data task assistant

New Task | Restart Services | Haritha Ananthakrishnan | Logout

What data task do you need help with?

Start typing to begin a new task

I want to build an app that helps to predict "time-to-fill" in our organization.

Send

Lets start with an open ended user request

Tasks

Live updates active

Task_1782483199281_849fd871
0 notifications

I want to build an app that helps ...
0 notifications

Event Log

Connected

67 events logged

TOOL_INVOKED: transfer_to_agent 14:39:58.379
invoked by: architect_agent
▶ View details

AGENT_STARTED: architect_agent 14:39:56.193
▶ View details

TOOL_INVOKED: report_semantic_modelling_completed_tool 14:39:
invoked by: semantic_modelling_indexing_agent
▶ View details

AGENT_COMPLETED: semantic_modelling_indexing_agent 14:39:45
▶ View details

TOOL_INVOKED: opensearch_indexer_agent 14:39:15.287
invoked by: semantic_modelling_indexing_agent
▶ View details

TOOL_INVOKED: report_topic_extraction_status_tool 14:38:50.458
invoked by: semantic_modelling_indexing_agent
▶ View details

TOOL_INVOKED: insert_topics_to_db_function 14:37:57.630
invoked by:
▶ View details

Monitoring: agent-events, tool-events

Why we built STEER

Inspired by the Karpathy-style “autoresearch” loop — agents proposing, running, and analyzing experiments — our first internal campaign proved both sides of the story:

Agents are excellent at search
They run broad campaigns, reuse harnesses, compare variants, and grind through spaces humans would not exhaust.

Agents are weak at research taste
They chase ghost parameters, overfit noise, miss obvious literature, and spend budget on directions a human would kill early.

STEER keeps the useful part: **humans choose the axis; agents execute transparently and preserve the evidence.**

STEER: three layers, always running, all auditable

Mirrors a research lab: PI sets direction, experts own domains, execution does the work. Needs an evaluation metric.

LAYER 1	**Humans/Agents — GitHub Issues** Researchers open issues with directions. Multiple people steer the same agent. The issue thread is the lab notebook.
LAYER 2	**Research Coordinator (PI) + Expert Agents** Agents accumulate knowledge across sessions — every experiment result, every dead end, every cross-project insight is written to persistent memory (as markdown) and available. PI cross-pollinates findings between projects.
LAYER 3	**Execution** Agents build on prior code — harnesses, evaluation scripts, and pipelines are reused and extended across experiments, not regenerated from scratch each run. Delegate heavy implementation to Codex / Claude Code. All jobs in named tmux sessions auditable by humans.

Built on OpenClaw (substrate). Every experiment tied to an issue · Agents never go dark > 2 hrs ·

The methodology is the operational discipline — not the substrate.

Satyananda Kashyap, Niharika D'Souza, Nanadana Mihindukulasooriya, Horst Samulowitz

Initial Motivation:

Our Research: How to efficiently represent structured & unstructured data for heterogeneous querying?

Pipeline complexity	The search space is combinatorial — stages × models × retriever params — not a simple hyperparameter sweep.
Firehose of research	Hot research area. Novel approaches appearing daily/weekly.
Example Compute	AutoSchemaKG ATLAS-Wiki: 78,400 GPU hours, 900M+ nodes, 5.9B edges.

IMPOSSIBLE TO KEEP UP. SOLUTIONS for DATA & AI ARE PROHIBITIVELY EXPENSIVE.

Our question: can we get the same (or better) quality at a fraction of the cost?

Starting point - AutoSchemaKG (Bai et al., 2025): fully autonomous KG construction. 92% schema alignment. 12-18% improvement on multi-hop QA.

- Benchmark: MuSiQue multi-hop QA (2-4 hop questions across paragraphs)
- Evaluation: closed-loop — KG built from MuSiQue's own context paragraphs (matching Table 5)
- Metrics: Exact Match (EM) and F1 on answer extraction
- Cost metric: total pipeline LLM tokens (Stages 1-3)

The approach: reproduce the baseline, measure token cost per stage, then work backward toward efficiency.

PRE-STEER: The autoresearch experiment — where we started.

155 experiments in 58 hours. Karpathy-style loop. Strong results — and a clear map of where autonomous agents hit their ceiling.

- F1 vs pipeline tokens (cost)
- F1 vs wall clock time (speed)

Strong Pareto position: beat the paper's F1 at fraction of its token cost. Also surfaced where autonomous agents need humans — **32% of experiments were wasted - the agent couldn't tell signal from noise.**

What the autonomous run got right, and what it missed

Systematic search excels. Knowing what's worth searching for is the gap.

✓ GOT RIGHT

(what autonomous search is good at)

Entity-only wins

Skip schema induction. Save 98% tokens. TERAG reached this independently.

Maverick 17B reader

Best reader model. +10.7 F1 over 70B baseline.

KG is irreplaceable

Dense retrieval loses 8.2 F1 — even with oracle decomposition.

Systematic coverage

155 runs, brute-force breadth humans wouldn't have patience for.

✗ MISSED

(what autonomous can't do alone)

Ghost parameters

30 runs tuning knobs HippoRAG2 doesn't use. A human with grep catches it in 5 seconds.

Noise as signal

F1 deltas < 0.01 within noise — agent locked one in as a "critical finding."

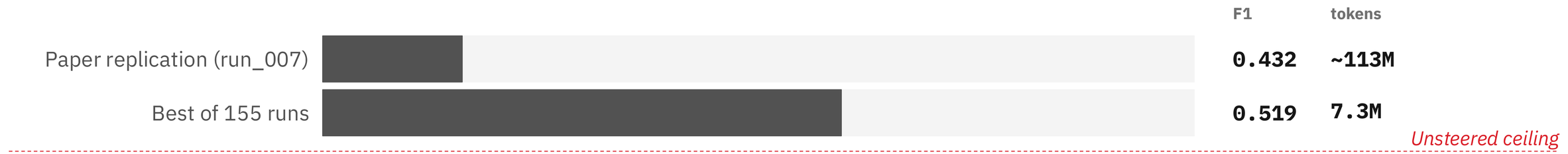
Missed literature

TERAG answered our exact question 6 months earlier. Agent never checked.

STEER: How a human steered direction helped

Issue #8 — Niharika: "Have you seen GEPA's optimize_anything?" The agent had searched retrieval exhaustively. She pointed to prompt optimization — a dimension autonomous had never explored.

UNSTEERED — 155 runs, 58 hrs (retrieval hyperparameter search)



HUMAN STEERING — Issue #8,(prompt optimization axis)

Niharika: "Have you seen GEPA's optimize_anything?" → Agent integrates GEPA but uses proxy metric.
Niharika: "That's not the real thing." → Agent re-wires to full pipeline evaluation.
PI Agent: "Session stalled — pick up." → 20-candidate GEPA run proceeds autonomously.

STEERED — prompt axis

What just happened with Issue #8 generalizes: autonomous covers breadth within an axis; humans recognize which axes are worth opening. STEER is the operational form of that division of labor.

Case study 2: KG construction / graph memory

Original question

For multi-hop QA over text, how much symbolic structure should we build before retrieval?

The intuitive answer was: more structure should help.

STEER turned that into a measurable frontier:

- quality: EM / F1 / Recall@5,
- cost: LLM tokens,
- wall-clock / operational burden,
- and robustness across hop counts.

Steering mattered

A naive autonomous agent can keep adding schema, events, concepts, prompts, and graph variants.

The steering question was sharper:
Where is the structure boundary?
What must be materialized in the graph, and what should be deferred to retrieval/traversal/reader reasoning?

KG: updated result frontier

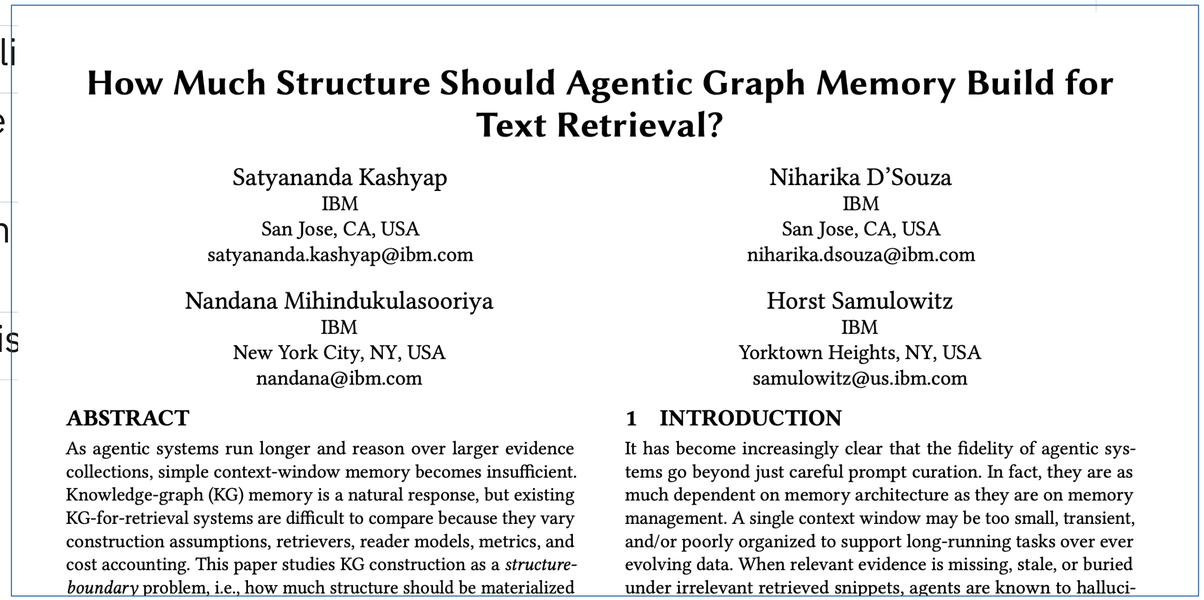
F1

43.25
full KG baseline 113.4M tokens

52.80
entity graph + GEPA + Maverick~13.6M tokens

59.68
same lightweight graph + stronger reader~13.8M tokens

Result	Interpretation
Entity-only graph beat full construction	More symboli
GEPA/prompt optimization helped after graph simplification	Optimize the
Stronger reader improved same graph	Some “graph problems.
No-graph baselines still failed	The answer is



How Much Structure Should Agentic Graph Memory Build for Text Retrieval?

Satyananda Kashyap
IBM
San Jose, CA, USA
satyananda.kashyap@ibm.com

Niharika D’Souza
IBM
San Jose, CA, USA
niharika.dsouza@ibm.com

Nandana Mihindukulasooriya
IBM
New York City, NY, USA
nandana@ibm.com

Horst Samulowitz
IBM
Yorktown Heights, NY, USA
samulowitz@us.ibm.com

ABSTRACT

As agentic systems run longer and reason over larger evidence collections, simple context-window memory becomes insufficient. Knowledge-graph (KG) memory is a natural response, but existing KG-for-retrieval systems are difficult to compare because they vary construction assumptions, retrievers, reader models, metrics, and cost accounting. This paper studies KG construction as a *structure-boundary* problem, i.e., how much structure should be materialized

1 INTRODUCTION

It has become increasingly clear that the fidelity of agentic systems go beyond just careful prompt curation. In fact, they are as much dependent on memory architecture as they are on memory management. A single context window may be too small, transient, and/or poorly organized to support long-running tasks over ever evolving data. When relevant evidence is missing, stale, or buried under irrelevant retrieved snippets, agents are known to halluci-

Case study 3: VDM.build architecture help

Starting point

VDM-style work creates open-ended technical questions that are not just “run an experiment”:

•Which backend architecture is sufficient?
•What should be materialized vs. deferred?
•What data access is missing?
•What is the smallest safe validation step?
•What cost/latency/governance assumptions are driving the decision?

What STEER added

Acted as an architecture agent:
• turned VDM architect agent's asks into reviewable design options,

• separated facts, assumptions, blockers, and recommendations,

• produced decision artifacts humans could critique,

• based on human inputs and added information changed the recommendation

VDM architecture threads: concrete examples

Thread	Human steering	What the agent produced	Why it helped
Time-to-fill predictive app	Design a practical backend for a VDM-derived app	Postgres/RDS-first recommendation, S3 feature zone, Fargate API, batch ML path, cost provenance, Terraform skeleton in PR #7	Humans got an inspectable architecture artifact instead of a vague “use cloud services” answer.
Cost-benefit provenance	“Show how cost numbers were produced”	Durable workflow rule: workload facts + pricing source + arithmetic + confidence/exclusions + decision impact	Improved future architecture recommendations; prevented hand-wavy cost claims.
VDM data-grounding loop	Reassess the VDM architect plan against real enterprise data + S3 evidence	Initial static recommendation was made without full data access; human steering pointed to available credentials; agent inspected Postgres/S3 and revised the architecture	Shows the real STEER pattern: human correction turned an incomplete-evidence recommendation into a data-grounded Postgres/RDS-first plan.

Related Publications

- “SemStruct: Contextualizing Semantic Embeddings with Structural Information for Schema Matching”, KDD 2026
- “RAG vs. GraphRAG: A Systematic Evaluation and Key Insights”, KDD 2026
- "Data in the Age of AI", Keynote at LLM+Vector @ ICDE 2026
- "Autonomous DataOps", Invited Talk to IBM-Lockheed
- “HGR-TabE: Universal Tabular Embeddings via Maximal Correlation Alignment”, FMSD @ ICML 2026
- “Data Understanding for Agents using Schema Grounding”, DEEM @ SIGMOD 2026
- “A Comprehensive Survey on Data Augmentation.”, IEEE Transactions on Knowledge and Data Engineering, 2026
- “ADP-MA: An Interactive System for Autonomous Data Processing using Meta-Agents”, Demo @ IJCAI 2026
- “Can a single tabular embedding model service different tasks?”, Tada @ VLDB 2026
- “Towards Budget-Aware Dense Retrieval for Tables: Trade-offs, Alternatives and Future Directions”, Tada @ VLDB 2026
- “Predicting Table Joinability in Data Lakes using a Metadata Knowledge Graph”, Tada @ VLDB 2026
- “New paradigms for knowledge discovery in the age of large language models”, ACM Transactions on Knowledge Discovery from Data, 2026
- “Towards Universal Tabular Embeddings: A Benchmark Across Data Tasks”, Under Revision @ VLDB 2027
- “Autonomous Data Processing using Meta-Agents”, Under Review
- “Data Understanding for Automated Data Analysis: A Survey”, Under Review
- “Zero-LLM Table Retrieval via Rule-Based Question Embeddings”, Under Review
- “DPDisc: From Factoid Questions to Data Product Requests for Open-World Data Product Discovery over Tables and Text”, Under Review
- “Bridging Business Intent and Data: A Benchmark for Automatic Relational Data Product Generation”, Under Review
- "How Much Structure Should Agentic Graph Memory Build for Text Retrieval?", Under Review

Thank you!

Any questions?