

Credo: Declarative Control of LLM Pipelines via Beliefs and Policies

Duo Lu

Andrew Crotty

Uğur Çetintemel



BROWN

Northwestern

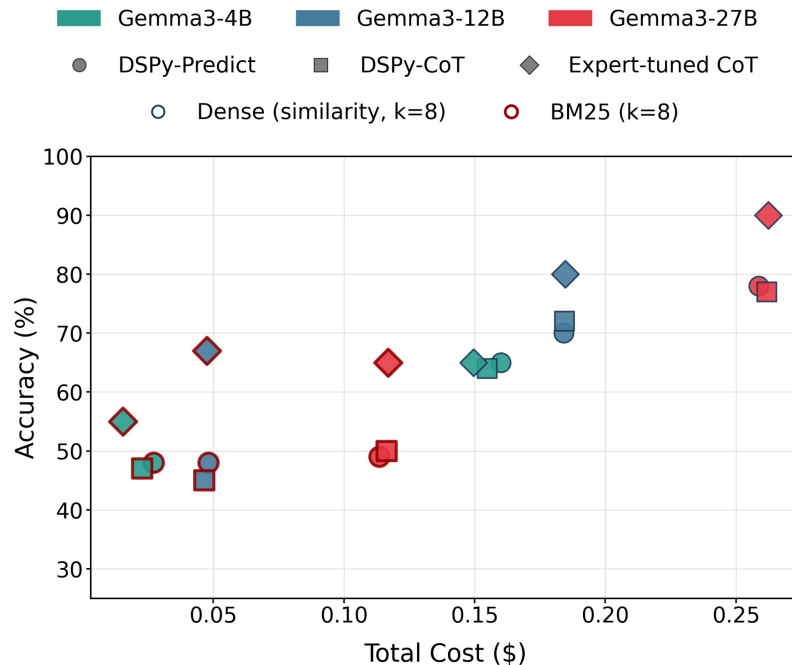
Motivation

One Size Does Not Fit All

- LLM data pipelines face a brutal cost-accuracy tradeoff

The Dilemma:

- *Simple Query?* Deep retrieval and large models are a waste of money
- *Complex Query?* Lightweight pipelines quietly fail on accuracy



We need a system that observes, thinks, and adapts during execution

Background

Declarative but Static (e.g., LOTUS, DocETL)

- Excellent for specifying what to compute, but the execution plan is frozen at compile-time

Adaptive but Opaque (e.g., ReAct, AutoGen):

- Highly flexible, but the logic is “black-boxed” inside prompts. It’s impossible to audit, verify, or tune without breaking the prompt

Can we bring database-style declarative control to agentic adaptivity?

Analogy: Separate the Logical Pipeline (What) from the Physical Policy (How)

Credo System Overview

A Database-Backed Semantic Control Plane

- Observations (Beliefs): Persistent semantic signals (e.g., “query is complex”)
- Decisions (Policies): Declarative WHEN-THEN rules (e.g., WHEN complexity > 0.8 THEN use 27B model)
- Execution (Actions): A reactive engine that performs *Operator Rewrites* (changing parameters) or *Pipeline Rewrites* (corrective re-execution)

