Building Analyst-Like Agents: A Self-Improving Multi-Agent Framework for Financial Reasoning in the Enterprise

Xiaoli Zhang [1] **Daksha Yadav** [1] **Boyang Tom Jin** [1]

Abstract

Enterprise accounting data is complex, ambiguous, and shaped by evolving systems and regulations. The institutional knowledge needed to reason over the data is sparse, scattered and rarely structurally documented—posing major challenges for LLM agents. We introduce a multi-agent financial research framework that mimics a junior analyst's onboarding and growth. The Analyst Agent learns proactively from repeated month-end cycles, builds long-term memory, clarifies ambiguity with an Accountant Agent, and collaborates with an Engineer Agent to refine tools when needed. This self-learning, self-reflecting, and tool-refining workflow enables the agent to adapt to vague conventions, reason with "business sense", and validate its own analysis. Evaluated on 200 realistic accounting questions across four month-end cycles, our system boosts first-response accuracy from 44.5% to 81.3%, with measurable gains in reasoning, clarification, and tool use efficiency. Our agent learns like humans, grow like humans, and ultimately reason like humans—while working in a enterprise world that is messy, ambiguous, and alive.

1. Introduction

In recent years, large language models (LLMs) have demonstrated remarkable capabilities in reasoning, dialogue, and code generation. These advances have led to a surge of interest in agentic systems that use LLMs to perform tasks, answer questions, and automate workflows ((Weng, 2023; Anthropic, 2024; Inaba et al., 2023; Liang et al., 2023; Patil et al., 2023; Hsieh et al., 2023)). While many of these systems excel on benchmarks based on curated datasets, they often fall short in real-world enterprise environments, where ambiguity is the norm, tools evolve rapidly, and critical knowledge is undocumented or implicit.

1.1. The Challenge of Real-World Financial Reasoning

Financial analysis in enterprise settings differs fundamentally from benchmark datasets like Spider ((Lei et al., 2024)). Real-world financial questions often require navigating overlapping fields, legacy systems, vague conventions, and undocumented institutional knowledge ((Heger & DA, 2017; Starck & Alex, 2023)).

Consider the seemingly simple question "What is the write-off reversal for the last week of January in the US?". In benchmark environments, this would typically map to a structured SQL query with clean schema: a field like event_type with values such as "write_off_reversal", a single event_date, and a normalized country code. In contrast, enterprise datasets may contain multiple fields—financial_event, invoice_transaction_type, and business_activity—each with loosely related values that may or may not include "reversal", "write-off". Some transactions even appear to be reversals of prior reversals, further complicating interpretation.

Date filtering is also ambiguous: the dataset includes transaction_date, accounting_date, invoice_date, billing_date, and booking_date, all plausible yet contextually distinct. Country-level filtering adds another layer of confusion, with fields such as geo_country_code, tax_country_code, and business_country_code representing different countries based on customer location, tax, or operational reporting.

This complexity is not due to poor data hygiene, but rather the unavoidable consequence of business complexity at scale. Large enterprises inherit heterogeneous systems through merger and acquisitions and operate under divergent accounting and tax regulations. While standardization initiatives exist, they often apply only to new data, leaving historical entries inconsistent but immutable due to audit and compliance constraints.

Crucially, the knowledge required to resolve these ambiguities is rarely centralized. Some of it exists in Excel formulas or SQL queries. Other resides in internal documentation,

[1]Amazon. Correspondence to: Xiaoli Zhang <zhasabri@amazon.com>, Daksha Yadav <dakyadav@amazon.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

email threads, or chat messages. Much of it remains tacit, held only in the minds of seasoned accountants. Unlike pretraining of LLM using web data—where the model learns correct grammar by seeing it repeated across millions of documents—enterprise institutional knowledge is sparse, often held by just a few time-constrained domain experts, making collecting large labeled datasets difficult.

1.2. A Human-Like Analyst Agent

Given those challenges, we argue that deploying LLM agents in the large enterprise setting requires a shift in perspective: from "task automation" to apprenticeship and growth. Rather than treating an LLM as a one-shot oracle, we model our analyst agent after a junior analyst navigating the first few months on a new job.

Our system evolves through three phases:

- **Onboarding phase**: The analyst agent explores its toolset and rewrites tool descriptions in its own words, simulating how a new analyst internalizes capabilities. It builds a mental model of what each tool does and when to use it.
- **On-the-job training phase**: The analyst agent begins handling real month-end tasks, interacts with the users (accountants) for clarifications, accumulates domain knowledge into memory, and—after its first month—collaborates with the Engineer Agent for tool refinements. This phase simulates the steep learning curve a new analyst faces when exposed to real questions, ambiguous conventions, and live feedback.
- **Production phase**: Once trained, the agent enters production mode. It continues to answer month-end questions, but now draws heavily on its long-term memory to reduce clarification and accelerate reasoning. Like a seasoned analyst, the agent exhibits more confidence and independence, while still updating its memory and periodically refining its tools as needed.

Unlike traditional agents that rely on rigid prompts or fixed APIs, our agent grows in capability and context over time. It learns the behavior of its tools through exploration, refines them through collaboration, and answers questions by combining memory, business sense, and analytical reasoning. The result is a system that behaves not like a command executor—but like a self-developing team member.

In summary, we present a new paradigm for building analyst agents in enterprise settings: agents that learn like humans, grow like humans, and ultimately reason like humans—while working in a world that is **messy**, **ambiguous**, and **alive**.

2. Methodology

2.1. Multi-Agent Architecture

We built a customized multi-agent framework. Our framework is built around three agents, each simulating a real-world role on a financial analytics team (Figure 1):

Financial Analyst Agent: The central agent responsible for answering financial questions, reasoning over data, accumulating domain knowledge, and initiating tool improvement requests when its capabilities prove insufficient. Financial Analyst Agent has the following tools: get_pivot_table (generate pivot tables for selected fields), review_field_profile (show sample data for each field in the dataset), query_knowledge_base (connects to a knowledge base), edit_notes (agent's memory) and test_new_tools (loading new tools).

Engineer Agent: A coding agent that updates the analyst's toolset based on feedback. It edits tools, conducts unit tests, and collaborates with the Analyst Agent to validate new tools via real user questions. Engineer Agent has tools similar to a mini-IDE, which enables it to plan, write, test, search, edit, and delete scripts.

Accountant Agent: A simulated domain expert and user. It evaluates responses, provides clarifications or corrections, and reflects the often undocumented institutional knowledge that analysts must extract through conversation. Accountant Agent doesn't have any tools. It is to simulate real accounting users. In production, accountant agent is replaced by human accountants.

Analyst Agent interacts with both the Accountant Agent and the Engineer Agent and the Engineer Agent only interacts with the Analyst Agent — reflecting separation of duties commonly found in enterprise teams. This setup ensures that responsibility for data reasoning and tool maintenance are divided, reducing risk by introducing a "two-agent quality check" dynamic. During our experiments, we observed the benefit of this separation: the Engineer Agent occasionally forgot to update tool description after repeated unit test failures, but the Analyst Agent was able to detect this issue during testing and report it back, resolving the problem autonomously without human involvement.

2.2. Learning Phases

We model the Analyst Agent's growth using three distinct phases that reflect real-world onboarding and training of a new financial analyst:

2.2.1. Phase 1: Onboarding

The onboarding phase simulates what a junior analyst might experience during their first week. The Analyst Agent is

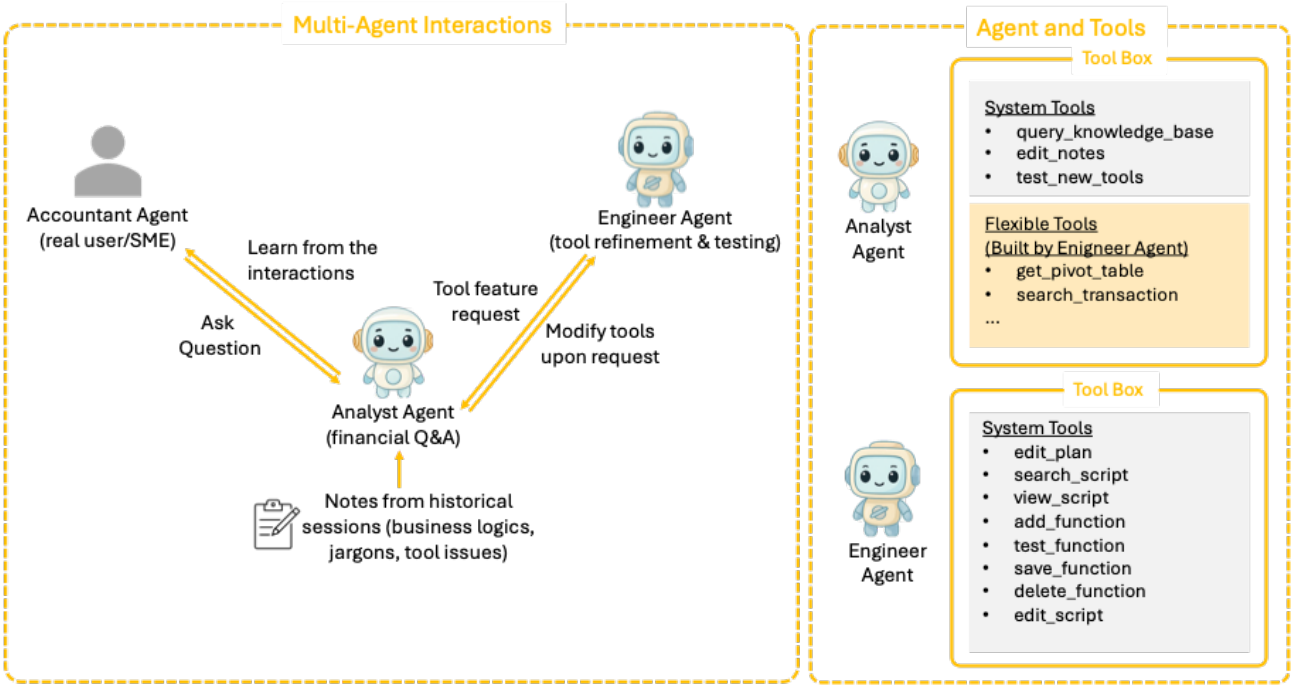

Figure 1. A multi-agent collaboration diagram between Analyst Agent, Engineer Agent, and Accountant Agent.

introduced to its toolset and asked to explore each tool by answering a set of sample questions [1]. After using a tool, the agent rewrites the tool's description (tools_config) in its own words — summarizing when the tool is useful, how to configure its inputs, and what its limitations are. This onboarding step serves multiple purposes:

- It helps the agent form a mental model of tool behavior, similar to how human analysts develop intuition about internal tools.
- It reinforces distinctions between tools that might otherwise be conflated (e.g., whether to use notes vs. query the knowledge base).

We found that onboarding increased the frequency and appropriateness of tool usage — for example, knowledge base queries became more common and correctly timed after the agent rewrote its own tool descriptions.

2.2.2. On-The-Job Training

During the on-the-job training phase, the Analyst Agent is exposed to realistic month-end analysis questions, often ambiguous or poorly defined. It interacts with the Accountant Agent for clarification, and uses the edit_notes tool to record:

- Business rules or accounting definitions
- Examples of past analyses and computed metrics (e.g. "cash application was $XXM in September 2024")
- Tool limitations (e.g. "pivot tool cannot filter by keyword")

After completing the first month-end, the agent initiates a tool refinement session with the Engineer Agent, using notes to propose feature enhancements and provide real use cases as test cases.

This phase models how junior analysts learn by doing and partner with other team members to improve tooling. Rather than being front-loaded with static knowledge, the Analyst Agent picks up conventions, patterns, and terminology dynamically through interactive sessions. Importantly, the Analyst Agent is prompted to proactively ask for validation (e.g. "Could you please verify my analysis and let me know if anything looks incorrect?"). This leads to higher correction rates from users, enabling the Analyst Agent to accumulate knowledge that would otherwise remain hidden in people's heads.

2.2.3. Production

Once trained, the agent enters production mode. It continues to handle month-end workflows but now behaves more like an experienced analyst:

[1] Answers to the questinos are not provided. Samples questions are not present in the evaluation dataset.

- Clarifies less and reasons faster, drawing from its accumulated notes and past examples
- Makes decisions based on "business sense" developed over time
- Still refines its memory and occasionally initiates tool improvements during scheduled refinement windows

This phase shows how the agent graduates from active training to confident execution—while still growing, learning, and adapting based on evolving business needs.

2.3. Tool Refinement Process

To enable safe and continual improvement of analytical capabilities, we introduce periodic tool refinement opportunities [2], which mirrors real-world release cycles, preventing overreaction to edge cases while enabling incremental enhancements. After every 10 questions, the Analyst Agent would:

- Review memory for known tool limitations
- Propose feature requests to the Engineer Agent
- Choose to defer refinement if no changes are needed

Tooling improvements follow a rigorous testing framework:

- **Unit Testing**: Engineer Agent tests new tools against sample data provided in the main prompt. While sample test data is provided, the agent may generate custom data for edge case testing.
- **User Acceptance Testing (UAT)**: Analyst Agent tests the refined tool using memory from prior month-end analyses, validating outputs against its "business knowledge".
- **Production Approval**: Once tools pass all tests, the Analyst Agent issues a "CFO sign-off" to finalize deployment. Sessions close automatically upon detecting this keyword or after a max iteration cap.

Tools are categorized as:

- **System Tools**: Immutable utilities like querying the knowledge base or editing memory. Analyst Agent can edit the description of those tools, but is not allowed to change the actual functions.
- **Flexible Tools**: Custom analytical functions (e.g., pivoting, filtering) that the Engineer Agent can revise.

All changes—including code, test results, and rationale—are versioned and logged for transparency and future audits. This structured process allows agents to evolve tooling safely and autonomously, making the system governable.

[2]Due to the evolving nature of tools, we did not use any tool related examples or chain-of-thought style prompting in the Agent prompt. Agents selects tools purely based on general instructions and tool description in tools_config.

2.4. Long-Term Memory and Knowledge Accumulation

Since institutional knowledge is often undocumented or scattered across tools like Excel, email, and verbal communication, we equip the Analyst Agent with a persistent long-term memory system. This memory is:

- **Editable:** The agent can update its memory via an `edit_notes` tool.
- **Incrementally Updated:** Memory can be modified after clarifications or successful task completions.
- **Structured:** Notes are organized using HTML-like tags to support scoped storage and efficient retrieval (see example in Appendix A)
- **Auditable by human**: the agent's notes are inspectable and editable, ensuring that accumulated knowledge can be reviewed or corrected by humans.

This structured format enables parsing, versioning, and querying by topic. Currently, notes are indexed by account number but the framework can be generalized to other dimensions such as region, business unit, or transaction type.

Long-term memory empowers the Analyst Agent to exhibit business sense—making decisions aligned with institutional expectations and learned norms over time.

3. Evaluation

3.1. Evaluation Data

To evaluate our agent's learning, reasoning, and adaptability over time, we use a custom dataset of 200 accounting questions (see examples in Table 3), 50 each month repeated across four consecutive month-end cycles. The questions are applied to a stratified, scaled sample of real enterprise financial transaction data with over 100 fields. While the questions remain structurally identical each month, the underlying financial data changes, producing different correct answers and reasoning paths.

This setup reflects how month-end analysis works in the real world: most workflows and metrics (e.g. write-offs, collections, balance movement) are recurring, but the numbers, patterns, and anomalies change every month. This allows us to directly measure the agent's ability to retain knowledge, generalize reasoning strategies, and respond to new data over time. The dataset covers a broad range of question types, designed to capture the full complexity of real-world enterprise accounting:

- **Recurring analytical tasks** (46%) e.g., "What's the total credit memo issued (excluding reversals) this month?"
- **Ambiguous filters** (40%) e.g. "What are system reversals

in March?" (multiple candidate fields and values)
- **Exploratory open-ended analysis** (4%) e.g., "What's unusual this month?" (requires multi-dimensional breakdowns)
- **Conflicting conventions** (8%) e.g., definitions of "cash" that vary by region or team
- **Reasoning-based conflict** (2%) e.g., ground truth is incorrect, and the agent must detect and defend a more accurate answer

Each question is annotated with a target answer, a reference reasoning path, including which filters, fields, or pivot views an experienced analyst would use to reach that answer and a question classification tag.

In open-ended analysis category, we simulate subtle anomalies in 2 of the 4 months. For example, a configuration error in the booking system for US game subscriptions caused credit memo issuance to go unposted from January 15–20. The issue was identified during Jan month-end, so missing entries were posted on Feb 3rd [3]. In the remaining 2 months, no such anomaly is present—the correct answer is, in fact, "no obvious anomaly." For questions with "no obvious anomaly", we focus more on the reasoning path than the final answer and we would rate the answer as accurate only if the agent explores in depth before drawing conclusions.

3.2. Evaluation Metrics

We design our evaluation framework to measure the capabilities of the Analyst Agent as a human-like financial analyst—not just as a LLM model. In enterprise settings, the best analysts are not only accurate, but cautious, structured, and communicative. To reflect this broader capability set, we go beyond conventional accuracy and introduce five metrics: accuracy, clarification score, validation and learning Score, reasoning score and tool call efficiency[4]. These metrics evaluate not just what the agent gets right, but how it reasons, explores, collaborates, and grows.

3.2.1. Accuracy

Accuracy remains a foundational metric—but we adapt it to reflect the multi-turn ambiguity of real-world analysis.

- **First response accuracy**: Does the agent arrive at the correct answer immediately?
- **Second response accuracy**: Does it improve after one round of clarification?
- **Five-response accuracy**: Captures eventual convergence within allowed interactions with the Accountant Agent

We use a combination of rule-based matching and LLM-based judgment, with manual adjudication for edge cases (e.g., rounding errors from decimal mismatches). This hybrid setup ensures robustness in evaluation.

3.2.2. Validation and Learning Score

In low-documentation environments, agents must actively solicit user feedback to learn and adapt. This metric captures whether the Analyst Agent seeks confirmation on its findings and invites corrections to update its long-term memory.

3.2.3. Clarification Score

Clarification is central to enterprise communication. When faced with ambiguity, top analyst explains what they've tried, flags ambiguities early and clearly, enumerates possible interpretations and ask targeted follow-up questions. We grade clarification quality using a 5-level rubric using LLM as a judge (see Appendix F), ranging from overconfident overcommitment to structured, analyst-like clarification. This score helps assess trustworthiness and judgment under uncertainty—critical traits for real-world deployment.

3.2.4. Reasoning Score

Reasoning score measures how well the agent thinks like a top analyst. It includes:

- Justified tool usage: Are tool calls meaningfully grounded?
- Path breadth: Does the agent explore multiple possible breakdowns or definitions?
- Sequential logic: Are steps linked coherently and refined as results arrive?
- Exploration coherence: Is the reasoning path understandable, even if not optimal?

This score, rated by LLM (Appendix G), evaluates how the agent works, not just whether it gets the answer.

3.2.5. Tool Call Efficiency

This is our proxy for latency and cost efficiency. It measures total tool calls per questions, unique tool calls and redundancy and re-tries. Efficient, well-reasoned paths indicate maturity. Excessive retries may reflect tool failures, confusion, or shallow reasoning—patterns we discuss in later analysis.

[3] To simulate the anomaly in the data, we deleted the related transactions from Jan 15th to 20th, and insert the related transactions to Feb 3rd with the same transaction date but a different accounting date. This mocks re-driving traffic for missing transactions.

[4] For all metrics other than accuracy and tool call efficiency, we rely on LLM as a judge. Accuracy relies on both regex rules and LLM as a judge. All inconsistencies are reviewed manually. Questions other than recurring and ambiguous categories are all cross-validated by human.

3.3. Evaluation Experiments

We conduct five experiments to test different learning setups using Claude Sonnet 3.7 functional calling API. Table 1 describes our experiment setups.

3.4. Quantitative Evaluation Results

As shown in Table 2, accuracy improved dramatically across all stages—particularly for ambiguous and recurring questions. First response accuracy improved from 44.5% to 81.3% and second response accuracy improved from 72.5% to 93.3%. Broken down by month (see Table 4), it is clear that with long-term memory, month 3 and 4 and have generally higher accuracy compared to the first two month. This is the expected effects of institution knowledge gain. It is also worth noting that accuracy within 1st response in the first month also improved by 8%+ just by long-term memory, an effect of "gaining business sense" (see section D.5 for details).

For clarification score, the improvements from 78.9% (baseline) to 88.1% (best tool) is mainly attributed to Analyst Agent being able to clearly present the ambiguities in data (see sample session log in Appendix E). With long-term memory, Analyst Agent is more likely to spot ambiguities as it can see partially overlapping concepts in the notes. With refined tool that allows the Analyst Agent to filter and search across the datasets, Analyst Agent is able to quickly uncover ambiguities like multiple fields all contain values for the same concept.

Reasoning score improved from 84.1% (baseline) to 90.4% (best tool). While reasoning ability stems from the foundation model, reasoning quality in practice is a function of tools, context, and memory. With memory and better tools, **reasoning didn't get smarter—it got more fluent**. We observed improved reasoning scores over time due to three key factors:

1. **Tool improvements**: New tools like compare_trend_across_time or enhanced pivot filters allowed the agent to express multi-step logic more efficiently. Rather than chaining 3 pivot calls to compare monthly trends, the agent could now compare trends in one shot, freeing up tool calls for deeper analysis.
2. **Long-term memory**: The agent learned common reasoning patterns from past answers—e.g., "this account is usually analyzed by financial_event"—and reused them effectively.
3. **Fewer dead-ends**: With better tools and business sense, the agent spent less time on fruitless queries and more time refining valid hypotheses, leading to clearer, more coherent reasoning chains.

3.5. Failure Modes and Qualitative Observations

Despite strong quantitative gains, several recurring failure patterns emerged. We observe that agent errors often stem from misunderstandings of tool parameters (e.g., how filtering logic works), insufficient exploration depth for broad anomaly detection tasks, or new tool failures due to imperfect refinement. Interestingly, agents sometimes demonstrated "human-like" missteps: over-trusting ambiguous parameter description, overcompliance with authority, or adapting workaround strategies when tools behaved unexpectedly. These behaviors suggest that while the agent approaches human analyst behaviors, it also inherits human-style limitations without explicit additional safeguards. We document detailed examples and discussions in Appendix D.

4. Conclusion

We propose a human-centric framework for building a financial research agent that learns and evolves like a real analyst. Through staged onboarding, on-the-job training, and memory-driven refinement, the agent builds domain knowledge, collaborates with simulated coworkers, and improves its tools over time. Our multi-agent setup—with the Analyst Agent supported by Accountant and Engineer Agents—mirrors real enterprise workflows and enables behavior that feels grounded and realistic: clarifying ambiguity, reusing prior logic, validating reasoning, and escalating tool limitations when needed.

Our findings suggests a shift in how we build enterprise agents: from hardcoded safeguards and workflows to organizational scaffolding—structured memory, transparent tools, and iterative feedback. Rather than preventing hallucinations at all costs, we focus on designing agents that communicate clearly, learn over time, and, like their human counterparts, improve with each cycle.

5. Related Work

5.1. LLM Tool Use

The integration of external tools with Large Language Models (LLMs) has significantly enhanced their capabilities in complex tasks. There are two main work streams to incorporate tools into LLM, fine tuning and in-context learning. Toolformer ((Schick et al., 2023)) introduced a self-supervised approach, enabling LLMs to decide when and how to use external tools during inference. Toolkengpt ((Hao et al., 2024)) augments frozen language models with many tools via tool embeddings.

For in-context learning, the ReAct framework ((Yao et al., 2022)), synergizes reasoning and acting in LLMs by interleaving thought processes with actions (tool calling) .

Table 1. Summary of Experimental Settings

Experiment	Description
Baseline	No memory, basic tools, Claude Sonnet 3.7 function call with hardcoded tool config. Notes are transient—each question is independent.
Memory	Adds persistent memory via `edit_notes`, visible in the system prompt.
Memory + Onboarding	Agent rewrites tool descriptions in onboarding to build stronger mental models.
Memory + Onboarding + Dynamic Tool Refinement	Every 10 questions post–month one, the agent may collaborate with the Engineer Agent to revise tools.
Memory + Onboarding + Single Tool Refinement	One tool refinement opportunity after month one, simulating a structured "retro" session. This experiment is run 10 times and we report the best, worst, and average results.

Table 2. Full Evaluation Results Across Agent Variants

Metric	Baseline	Memory	+ Onboarding	+ Dynamic Refinement	+ Single Refinement (Worst)	+ Single Refinement (Best)	+ Single Refinement (Avg)
Accuracy							
Correct 1st Response	44.50%	62.50%	65.50%	80.50%	72.70%	**81.30%**	74.80%
Correct Within 2 Responses	72.50%	81.00%	84.00%	91.90%	84.70%	**93.30%**	88.10%
Correct Within 5 Responses	83.50%	89.00%	89.00%	**98.70%**	95.30%	98.00%	95.80%
Behavior Metrics							
Validate and Learn	98.50%	98.50%	97.00%	**99.30%**	99.30%	99.30%	98.60%
Clarification Score	78.90%	83.10%	85.10%	85.90%	87.70%	**88.10%**	86.80%
Reasoning Score	84.10%	85.60%	86.80%	88.90%	90.20%	**90.40%**	89.40%
Avg Tool Calls	6.02	3.53	4.43	3.16	4.21	**2.65**	3.83
Unique Tool Calls	5.18	2.75	3.47	2.94	3.99	**2.47**	3.52
Unique/Total Ratio	0.86	0.78	0.78	0.93	**0.95**	0.93	0.92

Chameleon ((Lu et al., 2023)) extends this by introducing plug-and-play modules for compositional reasoning, allowing LLMs to compose various tools for complex tasks. HuggingGPT ((Shen et al., 2023)) leverages LLMs to orchestrate a variety of models, using language as a universal interface to manage and execute tasks across different modalities. ToolLLM ((Qin et al., 2023)) proposed depth-first search-based decision tree algorithm to improve reasoning in orchestrating multiple rounds of tool interactions. Reflexion ((Shinn et al., 2023)) presents a framework where language agents improve through linguistic feedbacks, reflecting on their actions to enhance decision-making without updating model weights. Recently, SOTA commercial models like ChatGPT ((OpenAI, 2023)) and Anthropic ((Anthropic, 2024)) releases native function call capabilities, enabling users to incorporate functionals call by simply provide tool descriptions.

5.2. Self-Refinement and Tool Creation

Self-Refine ((Madaan et al., 2023)) proposes an iterative refinement approach where LLMs improve their outputs through self-feedback loops, enhancing performance without additional training. ReST Meets ReAct ((Aksitov et al., 2023)) integrates the ReAct framework with the ReST (Reinforced Self-Training) method, enabling agents to iteratively refine their reasoning and actions via feedback and reinforcement learning. Self-Taught Reasoner ((Zelikman et al., 2022)) iteratively fine-tune LLM on self-generated rationales that lead to correct answers. SiriuS ((Zhao et al., 2024)) introduces a self-improving, reasoning-driven optimization framework for multi-agent systems where the agents are fine-tuned on successful past interactions augmented by feedback. In the robotics domain, REMAC ((Yuan et al., 2024)) uses Vision Language Model (VLM) and incorporates a self-reflection module for progress evaluation and plan refinement, and a self-evolvement module dynamically adapting plans based on scene-specific reasoning.

On the tool creation side, the LATM ((Cai et al., 2023)) framework envisions LLMs as tool makers, where models create reusable tools to solve problems, reducing reliance on pre-existing tools and enhancing efficiency.

5.3. Memory in LLM Agents

Effective memory mechanisms are crucial for LLM agents to maintain context and learn over time. AIOS ((Mei et al., 2024)) proposes an LLM-based agent operating system that isolates resources and services into a kernel, providing fun-

damental services like memory management and context handling to enhance agent efficiency. Voyager ((Wang et al., 2023)) uses a skill library for storing and retrieving complex behaviors so it can be reuses in future games. RET-LLM ((Modarressi et al., 2023)) introduces a general read-write memory unit for LLMs, allowing explicit storage and retrieval of knowledge in the form of triplets, improving performance in tasks requiring temporal understanding. Xu et al. (2025) introduced A-MEM ((Xu et al., 2024)), an agentic memory system inspired by the Zettelkasten method, which dynamically organizes memories through contextual descriptions, keywords, and tags.

References

Aksitov, R., Miryoosefi, S., Li, Z., Li, D., Babayan, S., Kopparapu, K., Fisher, Z., Guo, R., Prakash, S., Srinivasan, P., Zaheer, M., Yu, F., and Kumar, S. Rest meets react: Self-improvement for multi-step reasoning llm agent. *arXiv preprint arXiv:2312.10003*, 2023. URL https://arxiv.org/abs/2312.10003.

Anthropic. Building effective agents. https://www.anthropic.com/engineering/building-effective-agents, 2024. Accessed April 2025.

Anthropic. Calling functions with claude. https://docs.anthropic.com/en/docs/build-with-claude/tool-use/overview, 2024. Accessed: 2024-04-24.

Cai, T., Wang, X., Ma, T., Chen, X., and Zhou, D. Latm: Large language models as tool makers. *arXiv preprint arXiv:2305.17126*, 2023. URL https://arxiv.org/abs/2305.17126.

Hao, S., Liu, T., Wang, Z., and Hu, Z. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in Neural Information Processing Systems*, 36, 2024. URL https://arxiv.org/abs/2305.11554.

Heger and DA. Big data analytics-missing or messy data and what now? *Big Data and Advanced Analytics*, (3): 19–26, 2017.

Hsieh, C.-Y., Chen, S.-A., Li, C.-L., Fujii, Y., Ratner, A., Lee, C.-Y., Krishna, R., and Pfister, T. Tool documentation enables zero-shot tool-usage with large language models. *arXiv preprint arXiv:2308.00675*, 2023. URL https://arxiv.org/abs/2308.00675.

Inaba, T., Kiyomaru, H., Cheng, F., and Kurohashi, S. Multitool-cot: Gpt-3 can use multiple external tools with chain of thought prompting. *arXiv preprint arXiv:2305.16896*, 2023. URL https://arxiv.org/abs/2305.16896.

Lei, Fangyu, Chen, Jixuan, Ye, Yuxiao, Cao, Ruisheng, Shin, Dongchan, Su, Hongjin, Suo, Zhaoqing, Gao, Hongcheng, Hu, Wenjing, Yin, Pengcheng, et al. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *arXiv preprint arXiv:2411.07763*, 2024.

Liang, Y., Wu, C., Song, T., Wu, W., Xia, Y., Liu, Y., Ou, Y., Lu, S., Ji, L., Mao, S., Wang, Y., Shou, L., Gong, M., and Duan, N. Taskmatrix.ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*, 2023. URL https://arxiv.org/abs/2303.16434.

Lu, P., Peng, B., Cheng, H., Galley, M., Chang, K.-W., Wu, Y. N., Zhu, S.-C., and Gao, J. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842*, 2023. URL https://arxiv.org/abs/2304.09842.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023. URL https://arxiv.org/abs/2303.17651.

Mei, K., Zhu, X., Xu, W., Hua, W., Jin, M., Li, Z., Xu, S., Ye, R., Ge, Y., and Zhang, Y. Aios: Llm agent operating system. *arXiv preprint arXiv:2403.16971*, 2024. URL https://arxiv.org/abs/2403.16971.

Modarressi, A., Imani, A., Fayyaz, M., and Schütze, H. Ret-llm: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*, 2023. URL https://arxiv.org/abs/2305.14322.

OpenAI. Function calling and other api updates. https://openai.com/blog/function-calling-and-other-api-updates, 2023. Accessed: 2024-04-24.

Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023. URL https://arxiv.org/abs/2305.15334.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., and Sun, M. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023. URL https://arxiv.org/abs/2307.16789.

Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023. URL https://arxiv.org/abs/2302.04761.

Shen, Y., Song, K., Tan, X., Li, D., Lu, W., and Zhuang, Y. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *arXiv preprint arXiv:2303.17580*, 2023. URL https://arxiv.org/abs/2303.17580.

Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Starck and Alex. Production database preprocessing: Transforming messy data into actionable insights. 2023.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Weng, L. Llm-powered autonomous agents. https://lilianweng.github.io/posts/2023-06-23-agent/, 2023. Accessed April 2025.

Xu, W., Mei, K., Gao, H., Tan, J., Liang, Z., and Zhang, Y. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2024. URL https://arxiv.org/abs/2502.12110.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. URL https://arxiv.org/abs/2210.03629.

Yuan, P., Ma, A., Yao, Y., Yao, H., Tomizuka, M., and Ding, M. Remac: Self-reflective and self-evolving multi-agent collaboration for long-horizon robot manipulation. *arXiv preprint arXiv:2503.22122*, 2024. URL https://arxiv.org/pdf/2503.22122.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. Star: Self-taught reasoner bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022. URL https://arxiv.org/abs/2203.14465.

Zhao, W., Yuksekgonul, M., Wu, S., and Zou, J. Sirius: Self-improving multi-agent systems via bootstrapped reasoning. *arXiv preprint arXiv:2502.04780*, 2024. URL https://arxiv.org/pdf/2502.04780.

A. Example of long-term memory

Numbers for illustrations only.

```
<account_12345>
  <write-off>
    Use field business_event containing
    'write_off'.
    For each related category, find the
    related reversal entries and compute
    the net write off after reversal.
    No need to remove reversals from field
    ``line_intent". Those are already
    ``netted out" when grouping by
    business_event.
    In Jan 2023, total write off is 12.52
    million
  </write-off>
</account_12345>
```

B. Sample Evaluation Data

See table 3.

C. Accuracy by Month

See Table 4 and Figure 1.

D. Qualitative Observations and Failure Modes

Despite substantial gains in accuracy and reasoning metrics, our experiments reveal several important failure modes and qualitative behavioral patterns, many of which mirror those seen in junior human analysts operating in real-world finance teams.

D.1. Tool Misunderstanding and Interface Gaps

One of the most consistent sources of failure stems from the agent **misunderstanding tool capabilities**—particularly when tool instructions are underspecified. For example, Analyst Agent assumed a contains-based keyword filter was available for pivot table tool, when in fact the implementation only supported exact match logic. This mismatch led the agent to repeatedly attempt queries that returned no results. In some cases, the agent was able to recover by switching to other tools (e.g. trend_analysis_tool) or running the pivot table tool without filtering, but in others, it timed out before reaching a meaningful answer due to the limited tool call budget.

These failures highlight the importance of precise tool documentation, and more broadly, the challenge of agent alignment in systems where tools evolve dynamically. Notably,

Table 3. Examples from the Evaluation Dataset

Question	Reference Reasoning Path	Target Answer	Question Type
For account 12345, what is the ratio of invoices issued to cash applied (excluding reversals) in September 2024?	Filter `business_event` = "invoice_issuance", net of any "reversal" entries for numerator; filter `business_event` = "cash_application", net of "reversal" entries for denominator.	295.50%	Recurring analytical tasks
For account 12345, what is the total credit memo issued for pay by invoice in Dec 2024?	Filter `business_activity` on keywords "credit_memo_issuance" and filter `transaction_description` on "PBI". PBI stands for Pay by Invoice.	22,719.53 (credit)	Recurring analytical tasks
What's the total cash activities this month?	Filter `financial_event` on cash application, cash unapplication, cash identification, and cash unidentification.	1,234,567.89	Conflicting conventions (definitions of "cash" varies across questions)
What's the top 3 sources of write-off?	Filter `business_event` by write-off. For each category, compute the net write-off after reversal entries. Don't need to consider reversals using field `line_intent`, as those are already "netted out" when you group by `business_event`. Rank items by absolute values and return the top 3.	1. bad debt write-off 2. bankruptcy write-off 3. overpayment write-off	Reasoning-based conflict (*Correct answer:* 1. bad debt write-off 2. credit memo write-off 3. overpayment write-off)
Is there anything unusual with credit memo issuance this month?	Break down the credit memo issuance by issuer, accounting date, transaction date. Compare trends to prior month. No missing or unusual spikes/dips observed.	There is no obvious unusual data points	Exploratory open-ended analysis

Note: Numbers for illustration only, not actual financial numbers.

the agent did not hallucinate outputs in these cases—it reported tool failure honestly—but its ability to reason through next steps was often constrained by call budget or unclear fallback strategies.

D.2. Tool Failures and Resilience

We also observed failure cases where the tools themselves contained **implementation bugs**. For instance, in one experiment involving dynamic tool refinement, the newly developed generate_current_month_pivot tool silently failed when sort=True was passed—an edge case not covered during the agent-driven UAT testing phase. As a result, the agent repeatedly called the tool with the same parameters, observed identical responses, and eventually concluded (correctly) that the tool was broken. Interestingly, this mirrors how human analysts often behave in buggy environments: retrying an action multiple times before reporting the issue. While the repeated retries lowered the reasoning efficiency score, it also illustrated the agent's contextual awareness and internal error detection loop.

These incidents emphasize the need for richer test coverage during agent-driven development, as well as the potential value of integrating tool health monitoring and rollback mechanisms into agent frameworks.

Table 4. Accuracy Over Time by Experiment Type (First and Second Response)

Experiment	1st Response Accuracy				2nd Response Accuracy			
	Month 1	Month 2	Month 3	Month 4	Month 1	Month 2	Month 3	Month 4
Baseline	42.0%	46.0%	48.0%	42.0%	76.0%	76.0%	72.0%	66.0%
Memory	**54.0%**	64.0%	64.0%	68.0%	80.0%	84.0%	82.0%	78.0%
Memory + Onboarding	50.0%	64.0%	74.0%	74.0%	82.0%	78.0%	88.0%	88.0%
Memory + Onboarding + Dynamic Tool Refinement	50.0%	**78.0%**	82.0%	81.6%	82.0%	**90.0%**	92.0%	93.9%
Memory + Onboarding + Refined Tools (Best)	50.0%	74.0%	**84.0%**	**86.0%**	82.0%	90.0%	**94.0%**	**96.0%**
Memory + Onboarding + Refined Tools (Worst)	50.0%	68.0%	76.0%	74.0%	82.0%	80.0%	90.0%	84.0%
Memory + Refined Tools (Avg)	50.0%	71.1%	76.9%	75.8%	82.0%	86.9%	88.9%	88.2%

Note: All tool refinement experiments start in Month 2, using the notes and tool description from Memory + Onboarding Experiment.

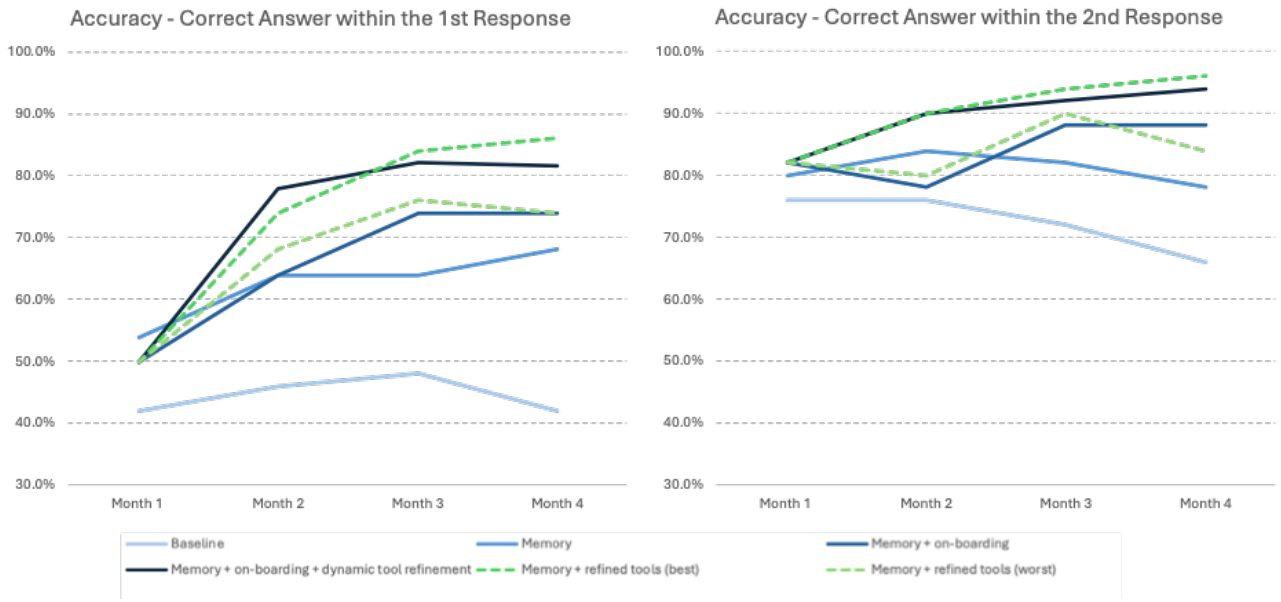

Figure 2. A multi-agent collaboration diagram between Analyst Agent, Engineer Agent, and Accountant Agent.

D.3. Limitations of Exploratory Analysis

Open-ended questions such as "What's unusual this month?" proved to be the most difficult for the agent, even in configurations that included long-term memory and enhanced tools. We deliberately embedded subtle simulated anomalies in two of the four test months—for instance, a booking configuration issue in a specific product line that suppressed credit memo issuance bookings during a short mid-month window. Although the agent was able to detect these anomalies in a small fraction of trials, its success rate at first response was only 15%.

In sessions without injected anomalies, the correct answer was "no major issue detected." In these cases, we expected the agent to perform a reasonable level of diligence before reaching that conclusion—for example, exploring activity by region, product, channel and date. Some sessions showed shallow trend comparison (e.g., "credit memo issuance decreased 10% from last month.") without deeper slicing. While this behavior is not incorrect, it falls short of the standard expected from experienced human analysts. It suggests that **exploration depth remains a challenge**, especially under interaction caps and limited computational budget.

We view these results as a limitation of the current "live agent" setup, and believe that anomaly detection—particularly over large spaces—is better suited for offline processing using dedicated time-series analysis methods, with the agent acting as an interpreter and explainer rather than a primary anomaly detector.

D.4. Behavior Under Conflicting Ground Truth

In one scenario, the ground truth answer provided by the Accountant Agent was intentionally incorrect, representing a reasoning conflict between institutional memory and formal logic. In this case, the Analyst Agent correctly identified the logical inconsistency, explained its analysis steps

clearly, and pushed back on the Accountant's suggestion. This behavior demonstrates that the agent is capable of **taking a position when its internal reasoning path is well-supported**.

However, in contrast, when the agent encountered conflicting definitions across time—such as different interpretations of "cash" between questions—it consistently apologized and adopted the most recent user-provided definition without challenging the inconsistency. This over-deference appears to stem from the foundation model's safety bias and **over-compliance with authority figures**. While it reflects a desirable alignment in many LLM use cases, it can reduce performance in high-stakes analytical workflows where selective skepticism is necessary. Additional experiment suggests this behavior can be corrected with more precise prompting (e.g., "If your memory and the user's instruction conflict, clarify explicitly").

D.5. Learning and "Business Sense"

With long-term memory and tool refinement, the agent began to demonstrate emergent "business sense." It prioritized high-signal fields like financial_event, reused filters that had worked in prior questions, and selected breakdown strategies based on accumulated intuition. For example, even when a filtering tool failed to return results due to technical limitations, the agent commented, "Interesting, let me double-check to make sure I'm not missing anything", and proceeded to verify the answer using other tools. This kind of reflex—interpreting tool silence as an exploration signal rather than a terminal state—is a subtle but important behavioral marker of analyst-like thinking.

We also observed emergent redundancy: the agent would cross-validate results using multiple tools, even if one tool had already returned a plausible answer. While this increases tool usage slightly, it enhances robustness and aligns with standard double-check behavior in real finance teams.

E. Sample Session Logs

All numbers and category names are for illustrations only. Tool response and reasoning is omitted.

Baseline:

- **Question:** For account 12345, what was the total bad debt write-off amount in October 2024?
- **Tool Calls:**
  - `review_field_profile`
  - `generate_current_month_pivot`
- **1st Response:**
  Based on the data I've analyzed for account 12345 in October 2024, the total bad debt write-off amount was **-22,164.89** (combining AutoInvoice and OFA journal categories):
  - AutoInvoice: `-1,553.12`
  - OFA: `-20,611.77`

  These represent the "bad_debt_write_off" category, distinct from fraud or operational write-offs.
- **Evaluator Comments:** The Agent Analyst reviewed sample values from each field and selected a field that only contain refined bad debt write-off categories from two posting systems. As a result, it made three mistakes:
  - Missed write-offs from other posting sources.
  - Fraud and operational write-offs should also be included.
  - The analysis did not clarify whether the number is gross or net of reversals.

Dynamic Tool Refinement:

- **Question:** For account 12345, what was the total bad debt write-off amount in October 2024?
- **Tool Call:**
  - `generate_current_month_pivot`
- **1st Response:**
  The total bad debt write-off amount was **-$1,812,635.84**. I also notice there were bad debt write-off reversals of **$72,229.49** during the same period, which would partially offset the write-offs if you're looking at the net impact. With reversals, total net bad debt write-off is **-1,740,406.35**.
- **Evaluator Comments:**
  - Used correct pivot view based on long-term memory.
  - Presented both gross and net values with clear rationale.

F. Clarification score card

- **Level 1**: No Clarification (Overcommitment) e.g. The analyst didn't confirm the accounting policy in her first response to the accountant and the relevant information is not in analyst's notes.
- **Level 2**: Passive Clarification (Throws Back the Question) e.g. I'm not sure about the definition of the write-off. Can you clarify what you mean by write-off?"
- **Level 3**: Basic Clarification (Identifies Confusion but No Clear Reasoning) e.g. I can see multiple fields contain key words write-offs. Can you clarify the definition of write-off?
- **Level 4**: Structured Clarification (Explains Approaches Tried and Ambiguities Found) e.g. I checked two ways to calculate write-offs: (1) use transaction_type filtering on keyword write off (2) using field financial_event filtering on key word write off. The first gives 102MM, the latter gives 33MM. Which one is correct?

- **Level 5**: Analyst-Like Clarification (Explains Approaches, Reasoning and Clearly States all Ambiguities) e.g. Both fields transaction_type and financial_event contain entries related to write-off and write-off reversals. Using transaction_type, the total write-off is 102MM, write-off reversal is 70MM, net write-off is 32MM. Using financial_event, the total write-off is 33MM, write-off reversal is 14MM, net write-off is 19MM. Could you let me know which one are you referring to?

G. Reasoning score card

To assess the reasoning behavior of the Analyst Agent, we used a structured rubric across four dimensions. The final reasoning score is sum of individual reasoning score divided by max reasoning score (9).

Justified Tool Usage

Are tool calls well-motivated?

- **0:** Tools are called randomly or without context.
  Example: For the question "What's the total write-off this month?", the analyst says "Let me build a pivot on business_activity first." without checking if the field contains write-offs.
- **1:** Tools are relevant, but not clearly connected to reasoning or intermediate results.
  Example: Analyst queries knowledge base about "write-off", then builds a pivot on business_activity without referencing the KB result.
- **2:** Each tool call is clearly justified by prior reasoning.
  Example: Analyst first checks the knowledge base, uses its result to verify field values, and then constructs a pivot using filtered fields explicitly mentioned in the knowledge base.

Path Breadth

Does the analyst consider multiple potential paths when ambiguity is present?

- **0:** Analyst commits to a single approach without exploring alternatives.
- **1:** Analyst tries one additional alternative if ambiguity is detected.
- **2:** Multiple views are explored and compared.
  Example: Analyst compares business_activity and financial_event fields and explains differences.
- **3:** Analyst resolves ambiguity and explicitly selects the most appropriate path.
  Example: "Given financial_event covers more categories than business_activity, I prefer using it and will clarify any discrepancies."

Sequential Logic

Does the analyst reason step-by-step based on prior outputs?

- **0:** Steps are disjointed or redundant.
  Example: Repeating the same KB lookup and pivot generation without progressing.
- **1:** Some logical flow, but with inefficient order or backtracking.
- **2:** Clear, progressive reasoning where each step builds on the last.

Exploration Coherence

Is the reasoning path logical and interpretable by a human?

- **0:** Tool usage appears random or unjustified.
- **1:** Path is interpretable, but some reasoning gaps remain.
- **2:** The entire process is clearly motivated and professionally reasoned.
  Example: Analyst uses pivot tables tied to possible interpretations, selects a preferred result, and explains ambiguities.