JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION
2026, VOL. 121, NO. 553, 29–33: Applications and Case Studies Discussion
https://doi.org/10.1080/01621459.2025.2554757

Discussion of "LAMBDA: Large Model Based Data Agent"

Bang Liu[a], Run Yang[b], and Fan Zhou[b]

[a]Department of Computer Science and Operations Research, University of Montreal, Montreal, Canada; [b]School of Statistics and Data Science, Shanghai University of Finance and Economics, Shanghai, China

Recent advances in large language models (LLMs) (OpenAI 2023; Jaech et al. 2024) have made the long-standing vision of automated data science increasingly achievable. Modern LLM-powered agents are now capable of interpreting natural language problem descriptions, analyzing heterogeneous datasets, and generating executable code-substantially lowering the barrier to entry and democratizing access to advanced data analytics. These capabilities are reshaping how data science workflows are initiated, constructed, and iterated, enabling users with little to no programming background to engage in complex analytical tasks.

As an LLM-powered agent, LAMBDA (Large Model-Based Data Agent) is designed to streamline data analysis tasks using natural language. It facilitates seamless interaction between domain-specific knowledge and advanced AI capabilities, offering tools for domain experts who lack coding expertise. The system includes two core agents: the Programmer, which generates code based on user instructions, and the Inspector, which handles debugging and error correction. Experimental results demonstrate its strong performance across various datasets, showing that it can perform tasks comparable to skilled data analysts. Additionally, LAMBDA is poised to transform data science education by providing interactive, hands-on learning experiences through its user-friendly interface.

LAMBDA introduces two core innovations that distinguish it from prior systems (Weng et al. 2024; Trirat, Jeong, and Hwang 2024; Li et al. 2024). The first is its *Knowledge Integration Mechanism*, which enables dynamic, in-context retrieval of domain-specific code snippets from a curated repository using natural language similarity. This allows the agent to inject analytical knowledge on demand, rather than relying solely on what is encoded in the base LLM. The second innovation is its *multi-agent architecture*, which incorporates a self-correcting feedback loop between the Programmer and Inspector agents. This setup follows the ReAct (Reasoning + Acting) pattern (Yao et al. 2023), allowing LAMBDA to iteratively improve its outputs by coupling reasoning and execution feedback, without the need for gradient-based updates.

Together, these components form a foundation for a new class of robust, agent-driven data science systems. However, they also reveal key limitations and design opportunities. The remainder of this article discusses several forward-looking directions inspired by LAMBDA's architecture and performance. These include: enabling self-evolving agents that can learn from past experiences; expanding agentic roles and workflows to handle more complex analytical pipelines; deepening human-AI collaboration beyond basic correction; improving retrieval via data-aware RAG; building large-scale knowledge bases for dynamic code integration; and framing data analysis environments as reinforcement learning problems to foster statistical reasoning. Through this lens, we outline a broader research agenda for advancing LLM-powered data agents beyond static tool usage toward truly adaptive, collaborative, and intelligent analytic assistants.

1. Toward Self-Evolving Agent Design

The interactive programmer-inspector design in LAMBDA focuses on external feedback to improve the agent's performance. The programmer agent produces an initial solution (code), and the inspector agent provides feedback by catching errors or missteps in that solution. On receiving the inspector's critique, the programmer revises the code, and this loop continues iteratively. In essence, the system performs a form of trial-and-error refinement, guided by an external critic (the inspector) and occasionally by the human user's edits. This design proved effective in practice: LAMBDA's agents were able to converge on correct solutions for many tasks via a few back-and-forth iterations, significantly reducing failures without additional training. The reliance on external feedback (both from the inspector agent and from human corrections) keeps the LLMs themselves fixed (no weight updates), yet still yields improved outcomes through runtime adjustments. This underscores the power of an *agentic learning loop* built around an LLM, where feedback can be injected through dialogue or code edits to compensate for the model's initial limitations.

CONTACT Fan Zhou *zhoufan@mail.shufe.edu.cn* School of Statistics and Data Science, Shanghai University of Finance and Economics, Shanghai, China.

An exciting direction is to complement such external-feedback loops with mechanisms for the agent to *learn from its own experiences* over time. In the current LAMBDA system, each task is approached anew, and any improvements (or mistakes) made during the iterative process do not persist beyond that single session. One can imagine instead a self-evolving agent that accumulates knowledge as it solves more tasks, gradually enhancing its capabilities. A concrete example comes from the prior work Voyager (Wang et al. 2023), which developed an LLM-driven embodied agent in a Minecraft environment that continuously learns new skills. Voyager stores its successful solutions (code "skills" for accomplishing sub-tasks) in a growing library and can retrieve and reuse them on future tasks. This leads to compound growth in abilities, as each new challenge can leverage a repertoire of prior solutions. Translating this idea to a data-analysis agent, we might equip LAMBDA with a long-term memory or repository of past problem-solving episodes. Over time, the agent could build up a library of useful code snippets, data transformations, and analytical techniques that it discovered or received as feedback. Faced with a new task, the agent could retrieve relevant past cases (or code) from its memory to guide the current solution. In addition to skill reuse, the agent could maintain an *episodic memory* of mistakes and fixes, enabling it to avoid repeating errors. Techniques like Reflexion (Shinn et al. 2023) have demonstrated the power of an agent generating self-reflective feedback and storing it in memory to improve later decisions. By internally recording what went wrong and why (in natural language or symbolic form), an LLM agent can adjust its reasoning on subsequent attempts without external intervention. Incorporating these ideas, a future LAMBDA-like agent could become increasingly autonomous and proficient the more it is used. It would start to exhibit continual learning: refining its internal "knowledge base" of coding patterns, analytical methods, and domain insights with each user interaction. Such a self-evolving design moves closer to an AI data scientist that, like a human, grows in expertise through experience.

2. Domain Knowledge Integration and RAG

Retrieval-Augmented Generation (RAG) has become a core component in modern large language model (LLM) systems, addressing a significant limitation of LLMs the inability to access up-to-date or domain-specific knowledge without retraining (Mallen et al. 2022). A typical RAG pipeline consists of a query rewriting module, a retrieval module, and a reranking module (Hong et al. 2023; Feng et al. 2024). The query rewriting stage refines user input to better reflect intent and improve retrieval relevance, while the reranking module selects the most relevant documents, ensuring high-quality, context-aware knowledge injection.

These components have proven effective for answering factual questions, as demonstrated in datasets like HotpotQA (Yang et al. 2018) and WikiMultiHopQA (Ho et al. 2020). However, the application of RAG to automated data science introduces new challenges. Unlike fact-based retrieval or multi-hop reasoning, data science tasks require the retrieval and integration of procedural and analytical knowledge that aligns with the distribution of data.

The *Knowledge Integration Mechanism* in LAMBDA can be seen as a domain-specialized extension of RAG, designed specifically for automated data analysis. In its current implementation, user instructions are embedded using Sentence-BERT and matched against a curated key-value repository of analytical routines. Similarity-based retrieval determines which code modules are injected into the prompt, with two integration modes: *Full*, which embeds the entire implementation inline, and *Core*, which provides minimal usage examples and delegates execution to a back-end runtime.

2.1. Improved RAG-based Knowledge Integration

While the *Knowledge Integration Mechanism* design of LAMBDA represents an important step toward bringing RAG principles into the domain of statistical computing. However, the current design is still limited in several important aspects. Most notably, the retrieval process relies solely on natural language instructions provided by the user and lacks any direct conditioning on the dataset itself or intermediate analytical outputs (e.g., summary statistics or EDA findings). As a result, the system may fail to surface relevant analytical strategies that are well-suited to the statistical characteristics of the data at hand.

This limitation is particularly important in empirical data analysis, where it is widely accepted that datasets with similar statistical structures often benefit from similar modeling pipelines. For instance, time series datasets typically share characteristics such as autocorrelation, seasonality, and missing data patterns-regardless of the application domain (e.g., rainfall prediction vs. wind forecasting). Techniques like lag feature construction, rolling-window statistics, and spectral decomposition are frequently applied across such tasks. However, a purely language-driven retrieval mechanism may not recognize the shared structure if these similarities are not explicitly described in the prompt.

Moreover, many useful patterns in data science practice are not tied to specific task names or domains, but instead emerge from data-driven considerations, such as skewness, multimodality, sparsity, or high cardinality. A retrieval system unaware of these properties risks retrieving generic or suboptimal routines, limiting the system's ability to generalize effectively across tasks. This underscores the need for retrieval mechanisms that are not only linguistically informed but also *statistically aware*, enabling the agent to reason jointly over both user intent and data context.

Thus, while LAMBDA's current mechanism demonstrates a practical application of RAG in data science, it also highlights the research gap in developing retrieval strategies that can incorporate structured signals from the dataset itself. Addressing this challenge will be crucial for building more adaptive and intelligent data science agents capable of aligning analytical procedures with both the semantic and statistical dimensions of a given task.

2.2. Building a Dynamic Knowledge Base

Another promising direction for future research is to build a large, continuously updated knowledge repository that houses a wide range of statistical methods and data science techniques, as

well as common problems and solutions. This repository would serve as a rich source of knowledge, seamlessly integrated into the RAG pipeline. By embedding this dynamic knowledge base within the RAG framework, LAMBDA could retrieve the most up-to-date and relevant methodologies, enhancing its ability to generate actionable insights across diverse data science tasks.

Such a repository could contain not only foundational methods like regression and classification algorithms but also domain-specific techniques such as time-series forecasting, anomaly detection, and deep learning-based approaches. It could also store real-world challenges and datasets from competitions, along with corresponding solutions and best practices. Integrating this knowledge base would ensure that the system remains relevant, timely, and capable of addressing the rapidly evolving landscape of data science.

The richness of the knowledge base would provide a robust foundation for statistical thinking. As datasets become more complex, the ability to draw from a broad array of techniques will allow the system to make better-informed decisions about the most appropriate methods for a given problem. Furthermore, a self-updating knowledge base, leveraging automated literature reviews and integration with open-source repositories, could continually expand its scope, ensuring that the system remains equipped with cutting-edge tools.

However, building and maintaining such a knowledge repository presents several challenges. Ensuring the quality, diversity, and accuracy of the methods included is crucial, as is keeping the repository updated. Additionally, efficient and context-aware retrieval remains an open challenge, particularly when integrating the repository with the RAG framework.

3. Enhance Statistical Reasoning via Reinforcement Learning

Recent breakthroughs in language model reasoning, such as DeepSeek-R1 (Guo et al. 2025), demonstrate that reinforcement learning (RL) can substantially enhance Chain-of-Thought (CoT) capabilities (Wei et al. 2022). By optimizing outcome-based reward signals, these models are able to develop generalizable reasoning strategies that extend beyond the limitations of traditional supervised fine-tuning. Additionally, recent works like SPIRAL (Liu et al. 2025) and Play to Generalize (Xie et al. 2025) illustrate how game-like training environments can effectively scaffold the learning of transferable reasoning skills.

Building on these insights, an exciting direction for future research is to conceptualize automated data analysis as a reinforcement learning environment, where the agent learns to reason *through* data analysis, rather than merely operating over it. This framework provides a principled opportunity to investigate whether large language models (LLMs) can acquire *statistical thinking*-including capabilities such as hypothesis formulation, uncertainty-aware decision making, and adaptive inference-by interacting with diverse data workflows and receiving feedback based on task outcomes.

LAMBDA's existing modular architecture, which incorporates sub-agents, provides an excellent foundation for building such a reinforcement learning-based environment. In this setup, agent actions could include invoking specific data transformations, revising prompts or code, or triggering self-correction routines. The environment states would evolve to reflect the changing analytic context, including updates to data transformations or the analytic model itself. Rewards could be defined based on various outcomes, such as predictive performance, efficiency metrics (e.g., minimal retries or iterations), or intermediate objectives like interpretability and statistical coherence.

This RL formulation offers several unique advantages, including:

- *Outcome-supervised learning.* Rewards can be defined based on downstream task success-such as final model accuracy, stability, or robustness-encouraging the agent to explore effective reasoning paths that directly correlate with real-world success.
- *Scalability through dataset diversity.* By varying the input datasets, the environment can expose the agent to a wide range of distributional patterns. This diversity helps the agent generalize reasoning strategies across domains, enabling it to apply learned techniques to a variety of problem contexts, from healthcare analytics to financial forecasting.
- *Rich state and action space.* Unlike static, tool-use settings, data analysis involves complex, interdependent steps that require the agent to make adaptive decisions at each stage. This dynamic environment fosters the emergence of structured reasoning and decision-making strategies, which are essential for solving complex real-world problems.

Framing data analysis as an RL environment thus presents not only a systems-oriented opportunity but also a principled setting for studying how reasoning abilities can emerge, specialize, and transfer within agent-driven language models. By training models in such environments, we can explore how statistical reasoning evolves through interaction with diverse datasets, further enhancing the capability of LLMs in data science and beyond.

4. Expanding Agentic Roles and Workflows

LAMBDA adopts a streamlined multi-agent architecture with two primary roles (programmer and inspector) cooperating in a fixed iterative workflow. This design choice emphasizes simplicity and clarity: one agent is responsible for solution generation and the other for solution checking. It proved sufficient for the scope of tasks considered (mostly classical ML and data analysis problems in controlled settings). The two-agent loop resembles a common real-world workflow where a junior analyst drafts an analysis and a senior reviewer checks and requests revisions. By showing that even just two coordinated LLM agents can handle nontrivial data science tasks end-to-end, LAMBDA provides a baseline architecture for agentic AI in this domain. Importantly, it also incorporates a flexible knowledge integration step where external domain experts or models can be consulted. Overall, the system design is modular and extensible, inviting exploration of alternative agent roles or additional components in the loop.

When moving to more complex and open-ended analytical projects, a richer assembly of agent roles and a more adaptive workflow may be required. One direction is to increase the *granularity of specialization* among AI agents. Instead of a

single "programmer" handling all aspects of code generation, we could assign different sub-agents to different aspects of the analysis. For example, we might have a Data Engineer agent focused on data cleaning and preprocessing, a Model Builder agent to select and train machine learning models, a Visualizer/Reporter agent to produce figures and interpret results, etc. By dividing responsibilities, each agent can operate with prompts and internal knowledge tailored to its specific subtask, potentially improving efficiency and outcomes. The challenge is then to orchestrate these specialists effectively. How should they communicate, in what order should their actions occur, and who coordinates the overall process? This is where dynamic or learned workflow design becomes crucial. Manually hard-coding a complex workflow (especially one that might need to adapt on the fly) is difficult. Exciting recent research has started to tackle automated optimization of agentic workflows. For instance, AFlow (Zhang et al. n.d.) is a framework that uses search algorithms to automatically discover effective sequences of LLM agent actions for a given problem. Similarly, the concept of Automated Design of Agentic Systems (ADAS) (Hu, Lu, and Clune n.d.) proposes a meta-learning approach: a "meta-agent" can experiment with different multi-agent organizational patterns (varying the number of agents, their roles, and their interaction protocols) and identify configurations that perform best on certain tasks. These methods open the door to AI systems that can configure *themselves*. In the context of data analysis, an adaptive agentic system might, for example, decide to spawn an extra debugging agent if a problem is particularly tricky, or to re-order the workflow (e.g., perform an exploratory data analysis step before modeling if the data is unfamiliar). In the future, we anticipate libraries of modular agents that an AI orchestrator could draw upon as needed, analogous to how a project manager assembles a team of experts for a complex project. By advancing beyond the fixed two-agent loop and enabling more flexible, complex workflows, such systems could take on far more ambitious data science tasks, potentially handling projects that involve many interdependent steps and expertises (from data wrangling to causal inference to narrative report generation).

5. Deepening Human-in-the-Loop Collaboration

The authors of LAMBDA rightly recognize the importance of human expertise in the analytical loop. The system is designed to be "user-in-the-loop": users initiate the process by describing their task and providing data, and they remain able to intervene at any point by inspecting outputs or injecting feedback. In particular, LAMBDA's interface allows a user to directly edit the code produced by the AI if an error or inefficiency is noticed. This direct hand-off between the AI agent and human user is a valuable safety mechanism; it ensures that the analysis can be corrected or steered by a knowledgeable person when the AI falters. Additionally, by letting the user save code, models, and results, LAMBDA supports transparency and reproducibility-users are not "locked out" of the analysis process, but rather are active supervisors. Such human-in-the-loop elements increase trust in the system and make it a true collaborative tool rather than a fully autonomous black-box.

We believe there is further potential to amplify human insight and oversight in agentic AI systems, moving toward a deeply collaborative human-AI partnership. In LAMBDA, the primary mode of human feedback is via direct code edits or re-prompts after the AI's attempt. Future systems could allow more nuanced and conversational interactions between the human expert and the AI agents. For example, instead of (or in addition to) editing code, a user might give natural-language feedback: "The approach you used to impute missing data is too simplistic-consider using a regression imputation or consult domain knowledge on likely values." The AI agent(s) would then interpret this instruction and adjust their plan accordingly.

Another possibility is enabling the AI to ask the human clarifying questions during the process (e.g., "Do you trust the outliers in this dataset, or should I remove them?") or to present interim results for guidance. For instance, the agent might say: "Here is a preliminary model; would you like to prioritize interpretability or predictive accuracy?" Or in a model selection scenario: "Between a random forest and a logistic regression, the former gives better predictive performance while the latter is more interpretable-shall I proceed with one or compare both?"

By incorporating richer two-way communication, the human can intervene at a higher level of abstraction, injecting expertise or preferences throughout the analysis, rather than only catching mistakes post hoc. The ultimate vision is an "amplified analyst" paradigm, where human and AI form a tightly integrated team. Recent work on AI collaborators in scientific research, such as the SciSciGPT prototype (Shao et al. 2025), points in this direction. In SciSciGPT, an LLM-based agent handles many routine analytic tasks but keeps a human researcher in the loop for critical decisions, and the framework explicitly aims to balance human and AI contributions.

In a similar spirit, we argue that the goal of agentic systems like LAMBDA should not be to replace the human analyst, but to enhance them, allowing the human to focus on creative and judgment-intensive aspects while the AI handles laborious computations and explorations. Achieving this will require careful interface design (so that interacting with the AI is intuitive and does not itself become a burden) and perhaps new training paradigms for the AI agents (so that they learn to defer to human input and to explain their reasoning in understandable terms). If successful, such human-AI collaboration could accelerate discovery in data-rich fields: the human provides contextual understanding and ethical oversight, while the AI provides speed, breadth of knowledge, and tireless exploration. We see LAMBDA's current human-in-loop feature as a foundational step in this journey. By expanding the ways in which humans can guide and partner with the AI, future systems will further amplify human value in the analytical process, ensuring that AI serves as a force multiplier for human intelligence rather than a mere replacement.

6. Summary

LAMBDA represents a significant step toward realizing the vision of intelligent, collaborative, and accessible data science systems powered by large language models. By combining a modular multi-agent design with a domain-adapted knowledge integration mechanism, it demonstrates that LLMs can

effectively support complex analytical workflows through natural language interaction, even without fine-tuning.

At the same time, LAMBDA also reveals key limitations and inspires a broader research agenda. We have outlined several promising directions, including enabling self-evolving agents through memory and reflection, expanding agentic roles and workflows, integrating richer forms of human feedback, advancing RAG with data-aware retrieval, and framing data analysis as an RL environment to promote statistical reasoning.

Taken together, these directions point toward a future in which LLM-based agents not only assist in data science tasks but learn and grow through experience, collaborate fluidly with human experts, and reason adaptively in complex, uncertain environments. LAMBDA provides a foundational architecture and proof of concept for this future-an AI partner that augments human insight and accelerates the practice of data science.

Acknowledgments

The authors thank the Editor for the suggestions and helpful feedback which improved the article significantly.

Disclosure Statement

The authors report there are no competing interests to declare.

References

Feng, S., Shi, W., Wang, Y., Ding, W., Balachandran, V., and Tsvetkov, Y. (2024), "Don't Hallucinate, Abstain: Identifying LLM Knowledge Gaps via Multi-LLM Collaboration," arXiv preprint arXiv:2402.00367. [30]

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. (2025), "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," arXiv preprint arXiv:2501.12948. [31]

Ho, X., Nguyen, A.-K. D., Sugawara, S., and Aizawa, A. (2020), "Constructing a Multi-Hop QA Dataset for Comprehensive Evaluation of Reasoning Steps," arXiv preprint arXiv:2011.01060. [30]

Hong, G., Kim, J., Kang, J., Myaeng, S.-H., and Whang, J. J. (2023), "Why So Gullible? Enhancing the Robustness of Retrieval-Augmented Models Against Counterfactual Noise," arXiv preprint arXiv:2305.01579. [30]

Hu, S., Lu, C., and Clune, J. (n.d.), "Automated Design of Agentic Systems," in *The Thirteenth International Conference on Learning Representations*. [32]

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. (2024), "OpenAI o1 System Card," arXiv preprint arXiv:2412.16720. [29]

Li, Z., Zang, Q., Ma, D., Guo, J., Zheng, T., Liu, M., Niu, X., Wang, Y., Yang, J., Liu, J., et al. (2024), "Autokaggle: A Multi-Agent Framework for Autonomous Data Science Competitions," arXiv preprint arXiv:2410.20424. [29]

Liu, B., Guertler, L., Yu, S., Liu, Z., Qi, P., Balcells, D., Liu, M., Tan, C., Shi, W., Lin, M., et al. (2025), "Spiral: Self-Play on Zero-Sum Games Incentivizes Reasoning via Multi-Agent Muti-Turn Reinforcement Learning," arXiv preprint arXiv:2506.24119. [31]

Mallen, A., Asai, A., Zhong, V., Das, R., Khashabi, D., and Hajishirzi, H. (2022), "When Not To Trust Language Models: Investigating Effectiveness of Parametric and Non-parametric Memories," arXiv preprint arXiv:2212.10511. [30]

OpenAI, R. (2023), "GPT-4 Technical Report," arxiv 2303.08774, View in Article 2, 1. [29]

Shao, E., Wang, Y., Qian, Y., Pan, Z., Liu, H., and Wang, D. (2025), "SciGPT: Advancing Human-AI Collaboration in the Science of Science," arXiv preprint arXiv:2504.05559. [32]

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. (2023), "Reflexion: Language Agents with Verbal Reinforcement Learning," in *Advances in Neural Information Processing Systems* (Vol. 36), pp. 8634–8652. [30]

Trirat, P., Jeong, W., and Hwang, S. J. (2024), "Automl-Agent: A Multi-Agent LLM Framework for Full-Pipeline Automl," arXiv preprint arXiv:2410.02958. [29]

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. (2023), "Voyager: An Open-Ended Embodied Agent with Large Language Models," in *Intrinsically-Motivated and Open-Ended Learning Workshop@ NeurIPS2023*. [30]

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022), "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Advances in Neural Information Processing Systems* (Vol. 35), pp. 24824–24837. [31]

Weng, L., Tang, Y., Feng, Y., Chang, Z., Chen, R., Feng, H., Hou, C., Huang, D., Li, Y., Rao, H., et al. (2024), "Datalab: A Unified Platform for LLM-Powered Business Intelligence," arXiv preprint arXiv:2412.02205. [29]

Xie, Y., Ma, Y., Lan, S., Yuille, A., Xiao, J., and Wei, C. (2025), "Play to Generalize: Learning to Reason through Game Play," arXiv preprint arXiv:2506.08011. [31]

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. (2018), "HotpotQA: A Dataset for Diverse, Explainable Multi-Hop Question Answering," arXiv preprint arXiv:1809.09600. [30]

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023), "React: Synergizing Reasoning and Acting in Language Models," in *International Conference on Learning Representations (ICLR)*. [29]

Zhang, J., Xiang, J., Yu, Z., Teng, F., Chen, X.-H., Chen, J., Zhuge, M., Cheng, X., Hong, S., Wang, J., et al. (n.d.), "Aflow: Automating Agentic Workflow Generation," in *The Thirteenth International Conference on Learning Representations*. [32]