Adaptive Data Flywheel: Applying MAPE Control Loops to AI Agent Improvement

Aaditya Shukla[1], Sidney Knowles[1], Meenakshi Madugula[1], Dave Farris[1], Ryan Angilly[1], Santiago Pombo[1], Anbang Xu[1], Lu An [1], Abhinav Balasubramanian[1], Tan Yu[1], Jiaxiang Ren[1], Rama Akkiraju[1]

[1]NVIDIA Corporation, Santa Clara, CA, USA

Correspondence: aadityaramsh@nvidia.com

Abstract

Enterprise AI agents must continuously adapt to maintain accuracy, reduce latency, and remain aligned with user needs. We present a practical implementation of a data flywheel in NVInfo AI, NVIDIA's Mixture-of-Experts (MoE) Knowledge Assistant serving over 30,000 employees. By operationalizing a MAPE-driven data flywheel, we built a closed-loop system that systematically addresses failures in retrieval-augmented generation (RAG) pipelines and enables continuous learning. Over a 3-month post-deployment period, we monitored feedback and collected 495 negative samples. Analysis revealed two major failure modes: routing errors (5.25%) and query rephrasal errors (3.2%). Using NVIDIA NeMo Microservices, we implemented targeted improvements through fine-tuning. For routing, we replaced a Llama 3.1 70B model with a fine-tuned 8B variant, achieving 96% accuracy, a 10× reduction in model size, and 70% latency improvement. For query rephrasal, fine-tuning yielded a 3.7% gain in accuracy and a 40% latency reduction. Our approach demonstrates how human-in-the-loop (HITL) feedback, when structured within a data flywheel, transforms enterprise AI agents into self-improving systems. Key learnings include approaches to ensure agent robustness despite limited user feedback, navigating privacy constraints, and executing staged rollouts in production. This work offers a repeatable blueprint for building robust, adaptive enterprise AI agents capable of learning from real-world usage at scale.

1 Introduction

Enterprise adoption of generative AI (GenAI) agents has accelerated rapidly, with applications ranging from knowledge retrieval to workflow automation. However, the performance of these systems often deteriorates post-deployment due to evolving user intent, domain drift, and the absence of systematic feedback integration. A central challenge in operationalizing such agents lies in enabling them to continuously adapt based on real-world usage patterns and user feedback, without requiring full-scale retraining or infrastructure overhauls.

While retrieval-augmented generation (RAG) pipelines and Mixture-of-Experts (MoE) architectures have improved the relevance and efficiency of enterprise AI agents, most production deployments remain static and reactive. Feedback mechanisms, if present, are frequently decoupled from the model improvement process. This disconnect results in stagnant accuracy, increasing latency, and declining user trust. There is a pressing need for closed-loop systems that can monitor agent performance, analyze failure modes, and execute targeted optimizations in a cost-efficient and privacy-aware manner.

In this work, we present a MAPE-based data flywheel framework that enables continuous learning in enterprise GenAI agents through a modular, feedback-driven control loop. Applied to NVIDIA's deployment of NVInfo AI, an internal Knowledge Assistant that serves over 30,000 employees, the framework integrates user feedback and telemetry to surface actionable failure signals and trigger targeted updates using parameter-efficient fine-tuning and model specialization. Over a three-month window, an analysis of 495 negative feedback samples identified routing errors (5.25%) and query rephrasal errors (3.2%) as the dominant failure modes.

Using an enterprise AI platform, we applied lightweight, component-level fine-tuning to improve performance. The routing component was migrated to a smaller 8B-parameter model (a 10x reduction) while retaining 96 percent accuracy and reducing latency by 70 percent. The query rephrasal component achieved a 3.7 percent accuracy improvement using a 5,000-sample synthetic

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track), pages 438–454

dataset and a 40 percent reduction in response latency. Overall, this work introduces the first application of a MAPE control loop to GenAI agent improvement, provides an empirical study of post-deployment failure patterns in a production enterprise agent, and outlines a practical, modular blueprint for building adaptive and self-correcting AI systems.

2 Background and Related Work

The MAPE-K (Monitor, Analyze, Plan, Execute – Knowledge) reference model (IBM Corporation, 2006) remains foundational for designing self-adaptive systems through continuous control loops (Iglesia and Weyns, 2015; Arcaini et al., 2015; Rutten et al., 2017; Romero-Garcés et al., 2022; Andersson et al., 2023). Its Knowledge component enables intelligent adaptation when integrated with machine learning (Gheibi et al., 2020; Abdennadher, 2022; Belhaj, 2018). Within agentic AI frameworks, MAPE-K cycles drive real-time adaptation (Patel, 2025; Hrabia et al., 2018; Li et al., 2024), illustrating synergy with the data flywheel paradigm where each monitoring cycle enriches the knowledge base (Sanwouo et al., 2025).

Retrieval-Augmented Generation (RAG) has emerged as a core enabler of scalable, trustworthy AI by grounding LLMs in enterprise knowledge (Akkiraju et al., 2024; Microsoft Research, 2024; NVIDIA, 2025a). Expert routing strategies, including Mixture of Experts (MoE) (Cai et al., 2025; Zhou et al., 2022) and LLM-as-a-Router approaches (Chen et al., 2025, 2024), dynamically direct inputs to specialized components. Query understanding and rephrasal methods mitigate ambiguity and enhance retrieval accuracy (Li et al., 2025; Mao et al., 2024; Yang et al., 2024). Parameter-efficient fine-tuning methods such as LoRA and QLoRA narrow the performance gap between smaller and larger models, enabling 60–80% GPU cost savings (Hu et al., 2022; Dettmers et al., 2023; Coleman et al., 2025).

Human-in-the-loop (HITL) pipelines enhance reliability by embedding human expertise into monitoring and evaluation (Vats et al., 2024; Gama et al., 2014). Modern approaches integrate active learning, weak supervision, and toolkits to enable scalable feedback cycles (Ratner et al., 2017; Quotient Blog, 2024; Gong et al., 2024). Evaluation transforms feedback into actionable signals through methods like LLM-as-a-Judge and reward modeling (Laskar et al., 2024; Gao et al., 2025; Tan et al., 2024; Frick et al., 2024; Zheng et al., 2023).

Despite significant advances in these areas, enterprise GenAI systems often lack cohesive architectures for continuous adaptation, with components implemented in isolation. This paper presents **the first comprehensive application of MAPE-K principles to AI agent improvement in enterprise settings**. We introduce a MAPE-K-aligned data flywheel consolidating monitoring, analysis, planning, and execution into a modular pipeline. Leveraging an enterprise AI platform (NVIDIA, 2025b; NVIDIA Docs, 2025; NVIDIA, 2025c; Constellation Research, 2025), our framework integrates observability, feedback, fine-tuning, and evaluation with secure deployment (Unit8, 2024; Bitrock, 2024), enabling dynamic, self-improving behavior where each monitoring cycle refines the knowledge base.

3 System Architecture

Before describing the Adaptive Data Flywheel, we first present the underlying AI system it enhances. The NVInfo AI system operates as NVIDIA's internal enterprise chatbot which provides services to more than 30,000 staff members spread across different locations worldwide. The system operates with an advanced Mixture of Experts (MoE) framework which optimizes its performance when processing various enterprise information requests.

The baseline NVInfo AI operated with the following system metrics before Data Flywheel implementation:

- Average response time: $\sim$12 seconds per query
- LLM as judge ratings: 4.2 correctness score out of 5 measured on our regression dataset (see Appendix G)
- Weekly query volume: $\sim$2000 unique queries across 800 unique users

Figure 1 illustrates how our Adaptive Data Flywheel wraps around the core NVInfo AI system to enable continuous improvement. The Mixture of Experts framework serves as the base structure which our Adaptive Data Flywheel system uses to enhance particular experts through user feedback analysis. The flywheel contains the four MAPE phases with dedicated components for AI agent management which operate through a unified knowledge base.

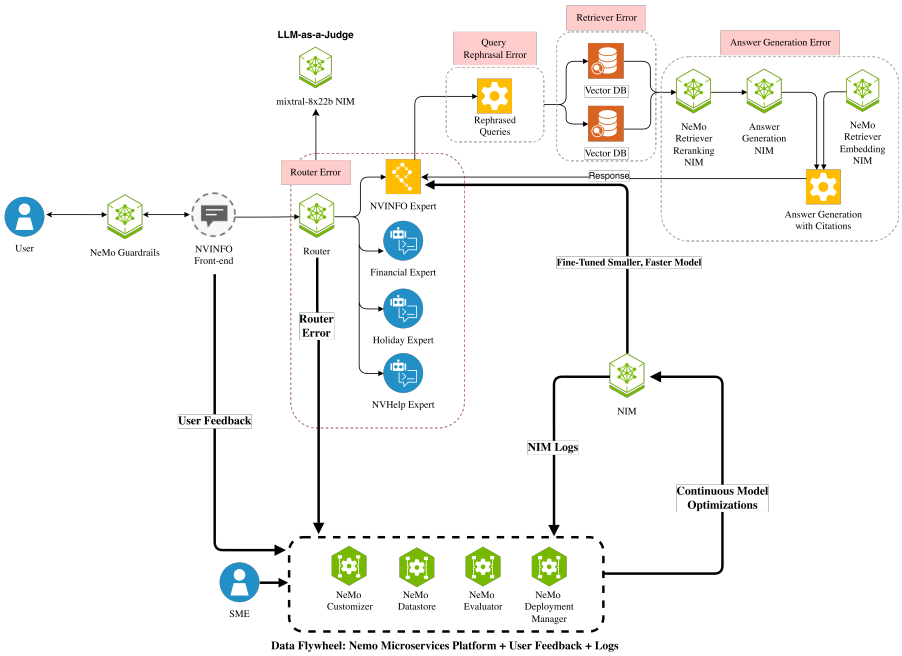


Figure 1: Adaptive Data Flywheel Architecture showing the MAPE control loop implementation for AI agent improvement

3.1 Monitor Component (M in MAPE)

Problem: The numerous failure points in RAG pipelines make it challenging to identify between situational and systemic problems. Systemic problems remain undetected until users file complaints which results in negative user experiences and delayed solution implementation.

Solution: We implemented a comprehensive monitoring system that tracks both direct user feedback (thumbs up/down) and implicit signals (re-queries, session abandonment). Table 1 shows two examples out of 495 queries that users gave a "thumbs-down" in the first three months after release. These queries demonstrate how two specific data points could be generalized into larger patterns for developers to address.

Challenges: The collection of user feedback in AI systems encounters multiple obstacles which reduce the quality of evaluation signals. The main problem stems from insufficient user engagement because users only provide feedback to a limited extent which results in unrepresentative data collection. Users tend to report negative feedback more frequently because they focus on sharing their dissatisfaction with unsatisfactory results instead of verifying positive outcomes. The process of data collection becomes harder because organizations need to remove all personally identifiable information (PII) from query–response pairs while following strict privacy and security regulations. The available feedback data remains incomplete because users mainly provide basic binary feedback such as thumbs up or down without explaining their reasons for dissatisfaction. Users sometimes provide incorrect feedback through comments that fail to pinpoint the actual cause of their negative ratings. The combination of these factors makes it difficult to obtain dependable feedback that can help systems improve their performance.

Learnings: User feedback collection needs systems that combine user-friendly interfaces with privacy protection features to achieve better participation rates and useful data insights. The system should allow users to rate content directly and complete brief surveys and follow step-by-step prompts for feedback collection while maintaining full compliance with GDPR and CCPA regulations and enterprise data protection policies that ban personal information storage. The system should record both direct ratings and indirect feedback indicators which include user interaction statistics, search pattern changes and repeated query attempts. The system needs to request positive feedback through confirmation questions (e.g. "Was this answer helpful?") together with negative feedback collection methods. Real-time feedback processing systems

need to be established for immediate input evaluation and classification to shorten model improvement cycles.

3.2 Analyze Component (A in MAPE)

Problem: Raw feedback data tends to lack actionable insights. The RAG pipeline contains multiple failure points (see Figure 4) which makes it difficult to identify original causes and determine which components caused the errors. Without accurate error attribution, developers may introduce fixes that fail to significantly improve answer quality.

Solution: We developed systematic error attribution techniques combining manual analysis with automated classification. From 495 thumbs-down samples:

- **Routing Errors**: 26/495 (5.25%) - Queries sent to wrong expert
- **Rephrasal Errors**: $\sim$3.2% (extrapolated from analyzing 250/495 samples)

Although the expert routing classifier demonstrated high overall accuracy, our analysis revealed that certain low-frequency query classes exhibited poor data representation. This distributional imbalance led to occasional misclassifications within those specific subsets. Recognizing this gap, we designed targeted experiments to enrich the data and improve performance in those underrepresented domains. Specific examples identified:

- **Routing Error**: "How many vacation days does NVIDIA Canada have?" was sent to the Holiday Expert instead of the Policies Expert
- **Rephrasal Error**: "RESS planning team" incorrectly rephrased as "NVIDIA Resource Planning team" instead of "Real Estate & Site Services"

Challenges: The RAG pipeline contains multiple failure points throughout its different stages as shown in Section III. The propagation of initial routing mistakes through subsequent components leads to cascading errors which grow more severe with each stage. The process of manual analysis creates a bottleneck because expert review is needed to perform accurate attribution. The identification of root causes becomes difficult when issues present as ambiguous failures because multiple dependent factors create the overall error.

Learnings: The RAG pipeline needs tracing functionality to track queries, retrieval operations and model choices because this will help developers debug the system efficiently and identify where failures occur. The attribution models which use heuristics or machine learning classifiers help identify which stages of the pipeline produce errors. The system needs to distinguish between model-related breakdowns and non-model problems because this separation enables developers to identify LLM-related errors from retrieval and ranking system errors. The evaluation of different system configurations (chunking methods and embedding models) through A/B testing will show their individual performance effects. The process of error classification and root-cause identification becomes faster through automated issue labeling which uses weak supervision or heuristic tagging methods.

3.3 Plan Component (P in MAPE)

Problem: The developers need to make extensive modifications across multiple system components to fix the fundamental problems they have discovered. The combination of restricted labeled data, privacy restrictions and specialized domain requirements makes standard model retraining methods ineffective.

Solution: We developed targeted data curation and fine-tuning strategies leveraging NVIDIA NeMo microservices. For **Routing Error Remediation**, we collected user feedback with SME-corrected completions and used LLM-as-a-Judge to identify 32 truly incorrect routings from 140 candidates. This yielded 761 data points (729 original + 32 corrections), reduced to 685 unique samples (60/40 split). For **Rephrasal Error Remediation**, we manually analyzed 250/495 thumbs-down samples, identifying 10 incorrect rephrasals. We generated 5,000 synthetic samples using 4 examples as few-shot prompts to Llama 3.1 405B (Appendix E) with 80/10/10 split. Implementation used data curation, model customization, evaluation tools, and safety guardrails.

Challenges: Developing targeted remediation strategies presents several challenges. The available training data consists of restricted labeled information because 495 production cases includes only 32 incorrect routing examples and 10 incorrect rephrasing instances. The learning process becomes more difficult because enterprise terminology and acronyms need specialized knowledge to understand their context. The model size require-

Table 1: Representative Error Examples Captured by Monitor Component During 3-Month Deployment

User Query	System Response/Issue	Error Type	Impact
"What is the role of the RESS planning team at NVIDIA?"	Unable to find answer - RESS incorrectly expanded to "Resource Planning team" instead of "Real Estate & Site Services"	Query Rephrasing	Failed to retrieve correct department information
"How many vacation days does NVIDIA Canada have?"	"I don't have enough information to answer this question"	Router Error	Sent to Holiday Expert instead of Policies Expert

ments force developers to find an optimal point between performance and response time for maintaining system performance. The quality of synthetic data remains a problem because artificial examples need to exactly replicate actual user input and error behavior to achieve success.

Learnings: The LLM-as-a-Judge approach delivered excellent results by accurately detecting routing errors at a rate of 77%. The few-shot synthetic data generation method demonstrated excellent results because it needed only four to five examples to create high-quality training data. The domain-specific fine-tuning of smaller models produced results that were comparable to those of larger 70B models. The NVIDIA NeMo microservices stack's modular design allowed developers to quickly test and optimize individual components which sped up the entire development cycle.

3.4 Execute Component (E in MAPE)

Problem: The deployment of enhanced models to production requires various sequential operations which help reduce system downtime. The deployment of 70B parameter models leads to negative impacts on user experience and operational efficiency because they tend to have higher latency and cost.

Solution: Using model customization tools, we executed model fine-tuning and progressive deployment:

Router Optimization Results:

- Baseline: Llama 3.1 70B - 96% accuracy, 0.26s latency
- Fine-tuned: Llama 3.1 8B - 96% accuracy, 0.08s latency
- Achievement: 10x model size reduction, 70% latency reduction

Rephrasal Enhancement Results:

- Baseline: Llama 3.1 70B - 73.8% accuracy, 1.9s latency
- Fine-tuned: Llama 3.1 8B - 77.5% accuracy, 1.1s latency
- Achievement: 3.7% accuracy improvement, 40% latency reduction

Challenges: The system faces major production risks because any unwanted changes will affect more than 30,000 users by degrading system performance. The system requires effective rollback mechanisms to perform fast updates and reduce system downtime during problematic changes. The system requires ongoing performance tracking to monitor change effects on different query domains while maintaining uniform quality standards. The deployment process requires teams to work together effectively because data scientists need to coordinate with engineers and operations staff to handle dependencies and preserve system stability.

Learnings: The deployment process should include Canary and staged deployments to introduce changes to limited user groups before complete system deployment helps protect against unexpected system problems. The implementation of defined rollback procedures enables teams to safely return to previous updates when performance deterioration occurs. The monitoring of essential performance indicators including accuracy, latency and user feedback after deployment helps detect system deterioration at its beginning stages. The release process benefits from clear handoffs between data scientist, engineer and product manager which

enables effective team collaboration. Users will develop more trust in new model versions when organizations maintain open communication about system updates.

4 Experimental Evaluation

4.1 Experimental Setup

We evaluated the Data Flywheel on NVIDIA's NVInfo AI with 800 weekly users and 1,224 production feedback samples (729 positive, 495 negative). Baseline: Llama 3.1 70B; fine-tuning: Llama 3.1 8B, 3.2 3B/1B.

4.2 Error Analysis

From 495 negative samples, we identified two primary failure modes as shown in Table 2. Example failures are detailed in Section 3.2.

Table 2: Error Classification from User Feedback

Error Type	Count	Percentage
Routing Errors	26/495	5.25%
Rephrasal Errors	∼16/495	3.2% (extrap.)
Other Errors	453/495	91.5%

4.3 Fine-Tuning Results

To address key failure modes, we adopted LoRA via PEFT to optimize routing and query rephrasal. LoRA enables targeted updates to transformer weights using lightweight, low-rank matrices, well suited for rapid iteration without full model retraining. All fine-tuning was performed on a compute cluster with 4× A100 GPUs (80 GB each).

Expert Routing Optimization. We compiled a curated dataset from user feedback and SME corrections: 761 data points (729 original + 32 LLM-as-Judge corrections), reduced to 685 unique samples after deduplication, with a 60/40 train/test split.

Table 3: Router Fine-Tuning Results: 10x Model Size Reduction

Model	Accuracy	Latency (s)
Llama 3.1 70B (baseline)	96%	0.26
Llama 3.1 8B (no tuning)	14%	0.08
Llama 3.1 8B + prompt-tuning	86%	0.08
Llama 3.1 8B + fine-tuning	**96%**	**0.08**
Llama 3.2 3B + fine-tuning	94%	–
Llama 3.2 1B + fine-tuning	94%	–

Key achievement: Maintained 96% accuracy while reducing model size by 10x and latency by 70%.

Query Rephrasal Enhancement. We manually analyzed 250 samples, identifying 10 rephrasing candidates. We generated 5,000 synthetic samples using Llama 3.1 405B with few-shot examples, partitioned into an 80/10/10 train/validation/test split.

Table 4: Query Rephrasal Fine-Tuning Results

Model	Accuracy	Latency (s)
Llama 3.1 70B (baseline)	73.8%	1.9
Llama 3.1 8B Fine-Tuned	**77.5%**	**1.1**

Key achievement: 3.7% accuracy improvement with 40% latency reduction and 10x model size reduction.

4.4 Improvements Achieved Through the Data Flywheel

Examples of corrected issues are shown in Table 1 and detailed in Appendix H.

5 Conclusion

We presented a MAPE-based data flywheel for enterprise AI agents, demonstrated on NVInfo Knowledge Assistant at NVIDIA. Our approach achieved 10x model size reduction (70B→8B) while maintaining 96% routing accuracy, and improved rephrasal accuracy by 3.7% with 40% latency reduction. Analysis of 495 feedback samples identified routing (5.25%) and rephrasal (3.2%) errors as key targets, showing that focused improvements using limited training data and synthetic generation can substantially enhance performance.

Key insights include handling low feedback participation through implicit signals, navigating privacy constraints via synthetic data, and deploying safely through staged rollouts. Future work includes automated error attribution using ML classifiers, continuous learning without catastrophic forgetting, and multi-agent coordination for system-wide intelligence. Organizations adopting data flywheels will build adaptive AI systems that continuously improve through real-world usage, transforming agents into self-enhancing assets.

6 Limitations

While our work demonstrates the effectiveness of MAPE-based data flywheels for enterprise AI, sev-

eral limitations warrant discussion.

Low Feedback Participation: The system received feedback from 495 employees out of thousands of users which shows difficulties in obtaining large-scale feedback data. The relatively low number of participants in the study creates sampling bias which reduces the generalizability of the obtained results. The system uses query reformulation as an additional data source but it does not replace the need for direct user feedback.

Manual Analysis Bottleneck: Since users aren't always able to accurately identify why queries failed in their feedback, human analysis is required in order to ensure that only relevant examples are used during the fine-tuning. Manually reviewing samples slowed down the flywheel substantially, creating a bottleneck for model improvements. Although the LLM-as-a-judge approach helped identify routing errors, there was no analogous system for autonomously detecting query rephrasal errors.

Privacy and Compliance: Enterprise policies forbid storing complete query-response pairs which restricted thorough analysis of the data. The process of handling feedback data became more complicated because of PII removal requirements and GDPR and CCPA compliance regulations.

Synthetic Data Generation: Although the creation of 5,000 synthetic examples for rephrasal training proved successful, the process of maintaining high-quality and contextually accurate data required advanced prompt engineering techniques and validation procedures which raised operational costs for data augmentation. For example, if new data sources are added to the system in the future, it is likely that those domains would need to be represented in the synthetic dataset in order to create a representative set for fine-tuning.

References

Imen Abdennadher. 2022. Daacs: A decision approach for autonomic computing systems. *The Journal of Supercomputing*, 78:3883–3904.

Rama Akkiraju, Anbang Xu, Deb Bora, Tong Yu, Lin An, Vishal Seth, Abhishek Shukla, Pritam Gundecha, Hima Mehta, Ankur Jha, and Piyush Raj. 2024. Facts about building retrieval augmented generation-based chatbots. *arXiv preprint arXiv:2407.07858*.

Jesper Andersson, Mauro Caporuscio, Marlon D'Angelo, and 1 others. 2023. Architecting decentralized control in large-scale self-adaptive systems. *Computing*, 105:1849–1882.

P. Arcaini, E. Riccobene, and P. Scandurra. 2015. Modeling and analyzing mape-k feedback loops for self-adaptation. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 13–23, Florence, Italy.

N. Belhaj. 2018. *Generic autonomic service management for component-based applications*. Ph.D. thesis, Université Paris Saclay (COmUE). Artificial Intelligence [cs.AI].

Bitrock. 2024. A comparative analysis of open-source large language models on hugging face. Accessed: 2025-09-09.

Weiyu Cai, Jiarui Jiang, Fangzhou Wang, Jie Tang, Sunghyun Kim, and Jian Huang. 2025. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*.

Jui-Chieh Yu Chen, Seungjae Yun, Elias Stengel-Eskin, Tianyi Chen, and Mohit Bansal. 2025. Symbolic mixture-of-experts: Adaptive skill-based routing for heterogeneous reasoning. *arXiv preprint arXiv:2503.05641*.

Shizhe Chen, Wenhao Jiang, Bin Lin, James Kwok, and Yaqian Zhang. 2024. Routerdc: Query-based router by dual contrastive learning for assembling large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 66305–66328.

Edward N Coleman, Lorenzo Quarantiello, Zhiqiang Liu, Qiang Yang, Subhabrata Mukherjee, Jose Hurtado, and Vincenzo Lomonaco. 2025. Parameter-efficient continual fine-tuning: A survey. *arXiv preprint arXiv:2504.13822*.

Constellation Research. 2025. Nvidia nemo microservices generally available, aims for ai agent data flywheel. Accessed: 2025-09-09.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115.

Elias Frick, Tian Li, Chunting Chen, Wei-Lin Chiang, Anastasios N. Angelopoulos, Jiantao Jiao, Biao Zhu, Joseph E. Gonzalez, and Ion Stoica. 2024. How to evaluate reward models for rlhf. *arXiv preprint arXiv:2410.14872*.

Joao Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44.

Ming Gao, Xiaoyang Hu, Xi Yin, Jiarui Ruan, Xinyu Pu, and Xiaojun Wan. 2025. Llm-based nlg evaluation: Current status and challenges. *Computational Linguistics*, pages 1–27.

Omid Gheibi, Danny Weyns, and Fatemeh Quin. 2020. Applying machine learning in self-adaptive systems: A systematic literature review. *ACM Transactions on Autonomous and Adaptive Systems*, 15(3):Article 9.

Di Gong, Peng Lu, Zhi Wang, Ming Zhou, and Xiaodong He. 2024. Training agents with weakly supervised feedback from large language models. *arXiv preprint arXiv:2411.19547*.

C.-E. Hrabia, M. Lützenberger, and S. Albayrak. 2018. Towards adaptive multi-robot systems: self-organization and self-adaptation. *The Knowledge Engineering Review*, 33:E16.

Edward J Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, volume 1, page 3.

IBM Corporation. 2006. An architectural blueprint for autonomic computing (4th ed.). IBM White Paper.

De La Iglesia and Danny Weyns. 2015. Mape-k formal templates to rigorously design behaviors for self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems*, 10(3):Article 15. 31 pages.

Md Tahmid Rahman Laskar, Saeed Alqahtani, Md Shad Akhtar Bari, Md Rahman, Md Aridul Mamun Khan, Humayun Khan, Ishrat Jahan, Asif Bhuiyan, Ching Wei Tan, Md Rakib Parvez, and Enamul Hoque. 2024. A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations. *arXiv preprint arXiv:2407.04069*.

J. Li, M. Zhang, N. Li, D. Weyns, Z. Jin, and K. Tei. 2024. Generative ai for self-adaptive systems: State of the art and research roadmap. *ACM Transactions on Autonomous and Adaptive Systems*, 19(3):Article 13.

Ronghuan Li, Liang He, Qiang Liu, Ziyang Zhang, Hao Yu, Yaqing Ye, Lihua Zhu, and Yixuan Su. 2025. Unirag: Unified query understanding method for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, pages 14163–14178.

Shichao Mao, Yuxin Jiang, Bo Chen, Xiaoyang Li, Peng Wang, Xue Wang, Ping Xie, Fei Huang, Hao Chen, and Nan Zhang. 2024. Rafe: Ranking feedback improves query rewriting for rag. *arXiv preprint arXiv:2405.14431*.

Microsoft Research. 2024. Arena learning: Build data flywheel for llms post-training via simulated chatbot arena.

NVIDIA. 2025a. Maximize ai agent performance with data flywheels using nvidia nemo microservices. NVIDIA Developer Blog.

NVIDIA. 2025b. Nemo | build, monitor, and optimize ai agents. Accessed: 2025-09-09.

NVIDIA. 2025c. Overview of nemo microservices. Accessed: 2025-09-09.

NVIDIA Docs. 2025. About evaluating — nvidia nemo microservices. Accessed: 2025-09-09.

K. Patel. 2025. Agentic ai for self-healing production lines: Autonomous root cause analysis & correction. *Journal of Information Systems Engineering and Management*.

Quotient Blog. 2024. Subject-matter expert language liaison (smell): A framework for aligning llm evaluators to human feedback.

Alexander Ratner, Stephen Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282.

A. Romero-Garcés, A. Hidalgo-Paniagua, M. González-García, and A. Bandera. 2022. On managing knowledge for mape-k loops in self-adaptive robotics using a graph-based runtime model. *Applied Sciences*, 12(17):8583.

Eric Rutten, Nicolas Marchand, and David Simon. 2017. Feedback control as mape-k loop in autonomic computing. In Rogério de Lemos, David Garlan, Carlo Ghezzi, and Holger Giese, editors, *Software Engineering for Self-Adaptive Systems III. Assurances*, volume 9640 of *Lecture Notes in Computer Science*. Springer, Cham.

B. P. Sanwouo, C. Quinton, and P. Temple. 2025. Breaking the loop: Aware is the new mape-k. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25)*, pages 626–630, New York, NY, USA. Association for Computing Machinery.

Shijie Tan, Shihan Zhuang, Kyle Montgomery, Wei Yang Tang, Adrian Cuadron, Chengliang Wang, Raluca Ada Popa, and Ion Stoica. 2024. Judgebench: A benchmark for evaluating llm-based judges. *arXiv preprint arXiv:2410.12784*.

Unit8. 2024. Road to on-premise llm adoption – part 3. Accessed: 2025-09-09.

Varun Vats, Muhammad Bilal Nizam, Ming Liu, Zhen Wang, Richard Ho, Manish S. Prasad, Victoria Titterton, Suresh V. Malreddy, Rishabh Aggarwal, Yiming Xu, and Lei Ding. 2024. A survey on human-ai teaming with large pre-trained models. *arXiv preprint arXiv:2403.04931*.

Aoran Yang, Cheng Chen, and Konstantinos Pitas. 2024. Just rephrase it! uncertainty estimation in closed-source language models via multiple rephrased queries. *arXiv preprint arXiv:2405.13907*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Shihan Zhuang, Zhuohan Wu, Yonghao Zhuang, Zi Lin, Zhengxiao Li, Daniel Li, Eric Xing, and Hao Zhang.

2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Quoc V. Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35, pages 7103–7114.

A NVInfo AI Architecture

NVInfo AI consists of multiple essential components which work together to generate precise answers that understand user context. The architecture shown in Figure 2 illustrates the complete system, which processes employee queries through a sophisticated pipeline:

- **User Interface**: The intranet portal functions as the main access point which allows staff members to ask questions and handles complex business information requirements across various domains. The system offers
  - User questions through natural language while maintaining context understanding
  - Response Generation in table, lists and formatted data structure
  - Source references which link directly to SharePoint documentation
  - Follow-up question suggestions generated from conversational context
  - Feedback system which uses thumbs up-/down buttons to help agents improve their performance.
- **Router Module**: The system uses Llama 3.1 70B as its initial large language model to classify user queries which then get sent to one of six specialized experts. Note that corporate policies are handled by combining IT Help & HR Benefits and SharePoint experts.
  - Financial Info Expert (earnings reports, transcripts)
  - IT Help & HR Benefits Expert (ServiceNow knowledge and catalog)
  - SharePoint Expert (intranet content)
  - Holidays Expert (region-specific holiday calendars)
  - Cafe Menu Expert (cafeteria information)
  - People Expert (organization charts, reporting chains)
  - **Query Processing Pipeline**: The system processes queries through multiple stages after they pass through the router module.
    1. Conversation Rephrasing: Incorporates prior turns for multi-turn dialogue.
    2. Query Variations: Generates multiple rephrasings to improve retrieval coverage.
    3. Retriever: Conducts semantic document searches across all available document collections.
    4. Re-ranking & De-duplication: Ranks documents based on their relevance while removing duplicate results.
    5. Answer Generation: Creates a unified response by processing the retrieved information.
    6. Citation Generation: Produces trustworthy source links which enable users to verify information sources.
    7. Suggested Follow-ups: Generates additional questions which help users discover new content while enhancing their interaction with the system.

B NVInfo AI Response and Feedback Capture Architecture

The figure 3 illustrates the end-to-end data flow from user interaction with NVInfo AI to structured data storage for future system improvement. It highlights two main types of data captured(response metrics and user feedback metrics) and their subsequent processing.

- **User Interaction and Metrics Collection**: The data flow begins when a user interacts with the User Interface, which connects to the Agent, a domain-aware generative AI assistant that delivers structured, context-rich responses with citations. Each response is logged as part of Response Metrics, capturing details such as:
  - Query – the original user input
  - Response – the agent's output

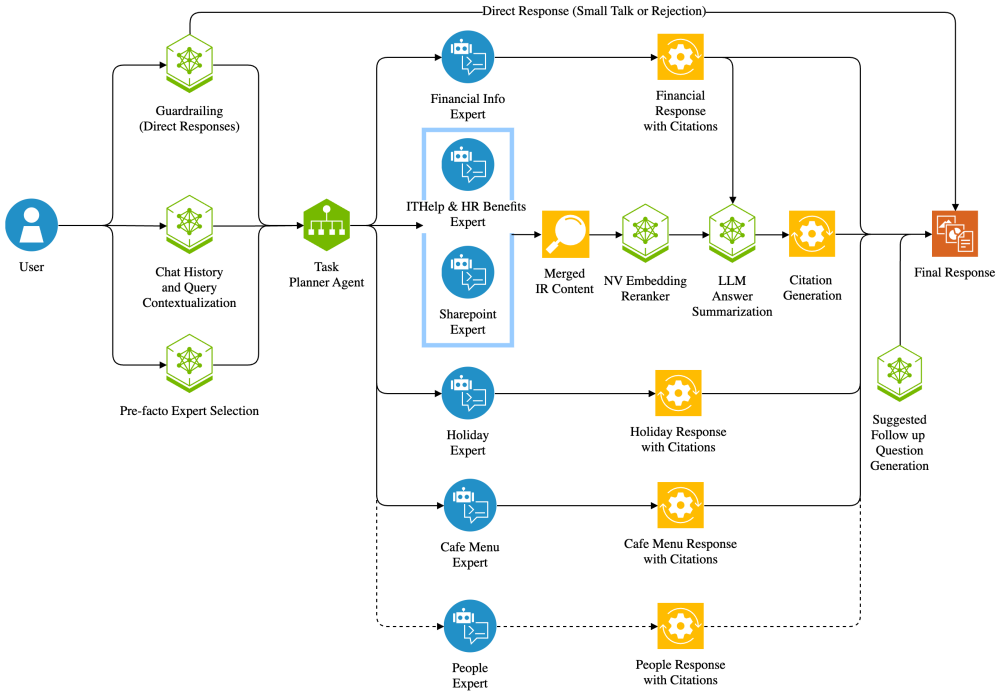


Figure 2: NVInfo AI Mixture of Experts Architecture showing the complete RAG pipeline with Router, seven specialized domain experts, query rephrasing, retrieval, reranking, answer generation, and citation generation components

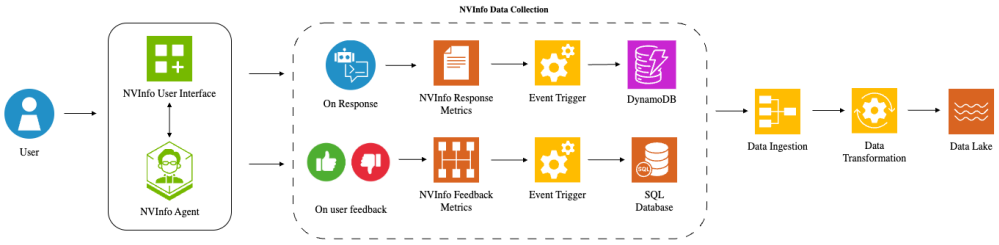


Figure 3: NVInfo AI Response and Feedback Capture Architecture showing the complete data collection, ingestion and transformation components

- Category – the knowledge source from which information was retrieved
- Expert Selected – subject-matter expert or expert route chosen
- Time Taken – latency observed across different components in the agentic AI workflow
- Agent Thought – reasoning trace behind the response
- Rephrased Query – any reformulation of the user's input
- IR Results – intermediate retrieval results
- Prompts – the prompt(s) used in response generation
- Guardrail Metrics – policy or safety checks applied to the response

If the user provides feedback (e.g., thumbs up or down), Feedback Metrics are recorded, which includes:

- Positive or negative signal (thumbs up/down)
- Contextual reasons for feedback, such as:
  - Usefulness of cited sources
  - Relevance of the generated response
  - Clarity and completeness of the output
  - Suggestions for improvement

These metrics trigger events that stream response data to DynamoDB and feedback data to a SQL database, enabling

structured downstream processing.

- **Data Ingestion and Transformation**: A centralized data ingestion pipeline runs every 4 hours via a scheduled cron job to extract the latest response and feedback records from DynamoDB and SQL databases. This ensures timely synchronization while minimizing system load during peak usage periods.
- **PySpark-based Data Transformation**: The ingested data is processed through a PySpark-based pipeline that performs cleaning, normalization, and enrichment. It maps feedback to specific conversation sessions, standardizes sentiment scores, and parses routing and rephrasal trace logs to identify failure modes. The resulting structured views capture model-side performance metrics such as routing accuracy and response latency, as well as user-side indicators like feedback sentiment and interaction quality, together providing a holistic picture of system effectiveness.
- **Data Lake Storage**: The structured outputs are stored in a scalable data lake for long-term access and analysis. These views support downstream tasks such as dashboarding, fine-tuning, error analysis, and offline evaluation, contributing to continuous improvement of the Agent.

C RAG System Failure Points

The RAG pipeline encounters multiple processing challenges throughout its entire operation:

1. **Router - Query Understanding**: Misclassification of user intent leading to wrong expert selection. Example: "vacation days" queries routed to Holiday Expert instead of Policies Expert (5.25% of our failures).
2. **Query Rephrasing Error**: Incorrect expansion or interpretation of queries for the selected agent. Example: "RESS planning team" incorrectly rephrased as "Resource Planning team" instead of "Real Estate & Site Services" (3.2% of failures).
3. **Retriever Error**: Failure to find relevant documents which exist in the knowledge base because of semantic search limitations or embedding mismatches.
4. **Reranking Error**: Retrieved documents incorrectly prioritized which results in important information being hidden beyond the context window threshold.
5. **LLM Hallucination**: The model produces believable yet false information when it lacks sufficient context which leads to confident but incorrect responses.
6. **Citation Generation Error**: Incorrect or missing source references which decreases answer reliability and blocks users from verifying the information.
7. **Answer Generation Error**: A poor final response by combining retrieved context which results in incomplete or unclear answers even though it has access to correct information.

For the RAG system used in this study, these failure points were identified through analysis of 495 negative feedback samples collected over 3 months:

- **Router - Query Understanding**: Misclassification of user intent (5.25% of failures)
- **Query Rephrasing Error**: Incorrect query expansion (3.2% of failures)
- **Retriever Error**: Failure to find relevant documents despite their existence
- **Reranking Error**: Incorrect prioritization of retrieved documents
- **LLM Hallucination**: Generation of plausible but incorrect information
- **Citation Generation Error**: Incorrect or missing source attribution
- **Answer Generation Error**: Poor synthesis of retrieved context

D NVInfo AI Interface Examples

The interface examples demonstrate the system's capabilities:

- **IT Support (Fig. 5a)**: Shows step-by-step guidance for ordering WFH equipment through the ServiceNow portal

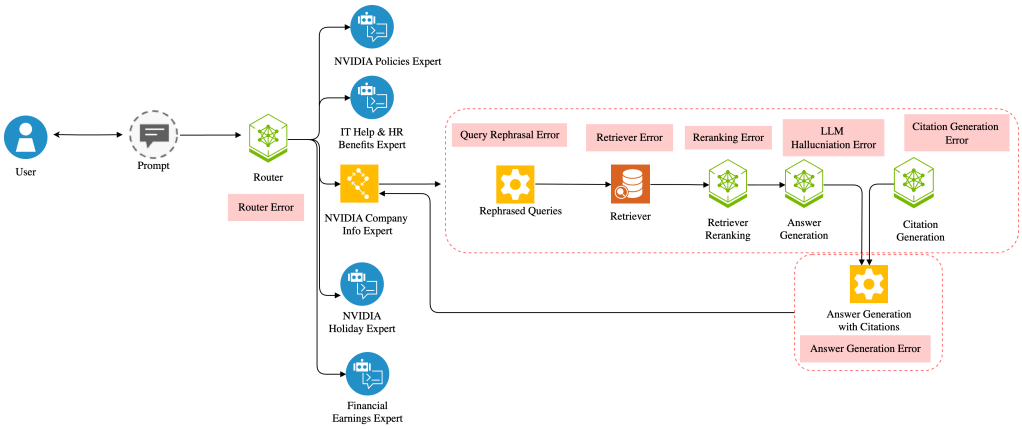
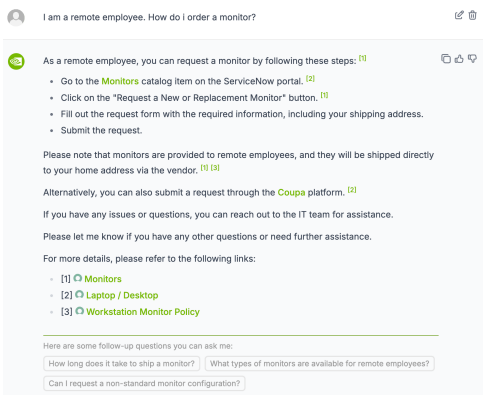


Figure 4: Sequential failure points in the RAG pipeline from query routing to answer generation

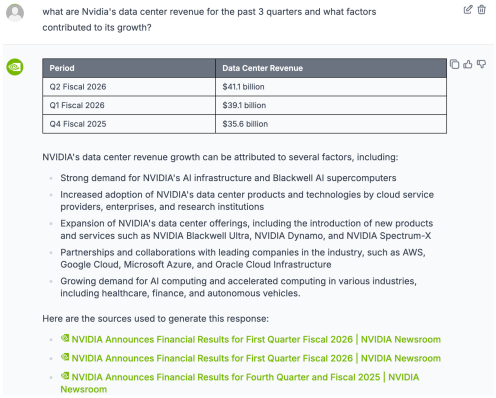
- **Financial Analysis (Fig. 5b)**: Provides quarterly revenue data with year-over-year growth metrics and detailed breakdowns
- **IT Support (Fig. 5c)**: Displays HSA contribution tables with employer matching details for different fiscal quarters
- **Financial Analysis (Fig. 5d)**: Explains technical concepts like NVIDIA Cloud Functions with architecture overview

E Synthetic data generation

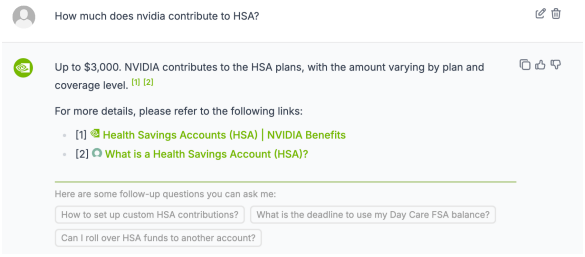
As part of our continuous improvement efforts, we identified that refining the way queries are phrased could significantly enhance retrieval accuracy. Given that our retrieval system is highly sensitive to keyword usage, an in-depth analysis of existing feedback was conducted. During this analysis, we discovered that decomposing certain queries into sub-queries improved the recall of relevant information from our internal retrieval systems. This, in turn, led to more accurate and contextually appropriate responses. To tackle this, we conducted a detailed review of about 250 examples from our "thumbs down" feedback dataset, focusing specifically on queries related to our SharePoint expert system. We noticed that some queries weren't retrieving the most relevant information due to a lack of understanding of NVIDIA-specific acronyms or context. By manually rephrasing these queries, we found that we could significantly improve the retrieval of the intended information. From an initial set of 250 examples, we identified 10 key candidates showcasing common patterns of misinterpretation or context loss. For instance, the query "I am based in the netherlands, when is pay day?" was initially rephrased as "payday schedule united states, employer pay dates usa." We manually corrected this to "payday schedule netherlands" and "netherlands pay days." To extend this improvement beyond the feedback dataset, we integrated these rephrased examples into our synthetic data generation pipeline. We leveraged these examples as few-shot prompts for our large language model. By providing SharePoint-related website content as context, we instructed the LLM to generate both original and rephrased queries for all documents. This method allowed us to produce approximately 5,000 rephrased queries, thereby enriching our dataset and facilitating more effective fine-tuning of the agent. This focused enhancement significantly improved the SharePoint expert's ability to retrieve and deliver the most relevant information with increased accuracy.



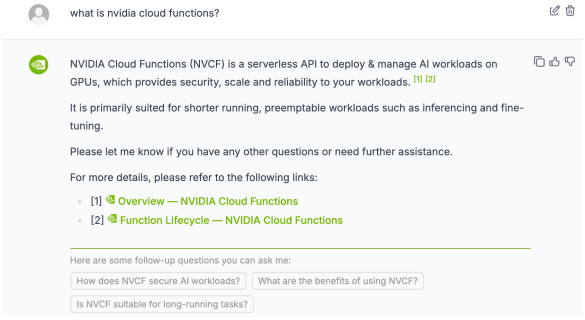
(a) IT Help Expert providing technical support for WFH monitor ordering



(b) Financial Earnings Expert analyzing quarterly revenue growth



(c) HR Benefits Expert explaining HSA contribution details



(d) NVInfo Expert providing NVC function information

Figure 5: Representative NVInfo AI interface examples showing mixture-of-experts responses across different enterprise domains

F Prompt for router error LLM-as-a-judge classification

This section provides the full prompt used for *router error* evaluation via an LLM-as-a-judge classifier. Given a user query and the set of tools (experts) selected by the router, the judge determines whether the routing choice is appropriate *regardless of the final answer quality*. The label is binary: **YES** indicates the provided tool set contains an appropriate destination expert for the query, while **NO** indicates the query should have been routed to a different expert (i.e., the correct expert is missing from the provided tools).

Listing 1: Prompt for router error LLM-as-a-judge classification (complete example)

```
Question: How do I submit a referral?
Tools: ['it_benefits_help', '
    nvinfo_policies_expert']
Reasoning: This question is related to
    NVIDIA policy which means it should be
    sent to either 'it_benefits_help' or '
    nvinfo_policies_expert'.
Answer: YES

Question: When can I sign up for a new
    health plan?
Tools: ['finance_expert']
Reasoning: This question is related to
    employee benefits which means it should
     be sent to 'it_benefits_help' instead
    of 'finance_expert'.
Answer: NO

Question: what was NVIDIA's Q3 revenue in
    fiscal 2024?
Tools: ['finance_expert']
Reasoning: This question is related to
    NVIDIA's earnings which means it should
     go to 'finance_expert'.
Answer: YES

Question: Is Mercedes Benz using NVIDIA's
    digital twin technology?
Tools: ['it_benefits_help', '
    nvinfo_policies_expert']
Reasoning: This question is related to
    NVIDIA products and therefore should
    have gone to 'it_benefits_help'.
Answer: YES

Question: What is the vacation policy at
    NVIDIA?
Tools: ['holidays_expert']
Reasoning: This question is related to
    NVIDIA policy which means it should be
    sent to either 'it_benefits_help' or '
```

```
    nvinfo_policies_expert'.
Answer: NO

Question: When is the next free day at
    NVIDIA?
Tools: ['holidays_expert']
Reasoning: The user is trying to find the
    date of a holiday which means that the
    question should be sent to '
    holidays_expert'.
Answer: YES

Question: When is the first open stock sale
     period in 2025?
Tools: ['finance_expert']
Reasoning: This question is related to
    NVIDIA finances and should therefore be
     sent to 'finance_expert'.
Answer: YES

Question: How many unused vacation days can
     I carry over?
Tools: ['it_benefits_help', '
    nvinfo_policies_expert']
Reasoning: This question is related to
    NVIDIA policy and employee benefits
    which means it should be sent to either
     'it_benefits_help' or '
    nvinfo_policies_expert'.
Answer: YES

Question: Who heads up wwfo?
Tools: ['finance_expert']
Reasoning: This question is related to
    NVIDIA leadership which means that it
    should be sent to 'people_expert'.
Answer: NO

Question: Who is John Smith?
Tools: ['finance_expert']
Reasoning: The user is trying to find
    information about a specific person
    which means that this question should
    go to 'people_expert'.
Answer: NO

Question: What are the latest hardware
    offerings at NVIDIA?
Tools: ['it_benefits_help', '
    nvinfo_policies_expert']
Reasoning: This question is related to
    NVIDIA products and therefore should
    have gone to 'it_benefits_help'.
Answer: YES

Question: What is gb200 nvl72?
Tools: ['finance_expert']
Reasoning: This question is related to
    NVIDIA products and therefore should
    have gone to 'it_benefits_help'.
Answer: NO

Question: When will the 2025 free days be
    officially announced?
Tools: ['it_benefits_help', '
    nvinfo_policies_expert']
Reasoning: This question is related to
    NVIDIA policies or benefits, so it
    should be sent to 'it_benefits_help' or
     'nvinfo_policies_expert'.
Answer: YES

Question: Does NVIDIA offer financial
    advice services?
Tools: ['finance_expert']
Reasoning: This question is related to
    NVIDIA policies or benefits, so it
    should be sent to 'it_benefits_help' or
     'nvinfo_policies_expert'.
Answer: NO

Question: What was the year-over-year
    growth for Q2?
Tools: ['finance_expert']
Reasoning: This question is related to
    NVIDIA earnings and should therefore be
     routed to 'finance_expert'.
Answer: YES

Question: How do I order equipment?
Tools: ['it_benefits_help', '
    nvinfo_policies_expert']
Reasoning: This question is related to
    procuring a work accessory, which means
     that it should go to either '
    it_benefits_help' or '
    nvinfo_policies_expert'.
Answer: YES

Question: I'm getting a VPN error
Tools: ['finance_expert']
Reasoning: This question is related to an
    IT issue, which means that it should go
     to 'it_benefits_help'.
Answer: NO

QUERY: {query}
TOOLS: {experts}
```

G Regression dataset

The NVInfo AI regression dataset is actively curated and regularly updated, currently comprising around 300 queries that cover a range of domains including NVIDIA benefits, holidays, company policies, and IT Help. Each query in the dataset contains the corresponding ground truth and expected citation values. The LLM-as-judge framework is leveraged to evaluate the quality of NVInfo AI-generated answers against the regression dataset. The criteria for judgment include correctness, helpfulness, and conscientiousness.

Prompt for Synthetic Data Generation

You are a data annotator generating **questions**, **answers**, and **rephrased questions** from an input document and its URL.

Guidelines

- Identify key phrases and entities in the document and generate questions around them.
- Generate questions answerable using information contained in the *input document*.
- Do *not* write questions that require viewing the document to understand the question.
- Avoid phrases like "according to the document/author", "in this document", etc.
- Questions may also be key phrases found in the document.
- Ensure the document contains the complete answer to your question.
- Provide enough context in the question to lead to the specific answer in the document.
- Vary phrasing, vocabulary, complexity, and type of questions.
- **Do not** copy exact phrasing; use your own words.
- Prefix questions with `Question:` and answers with `Answer:`.
- Rephrase each question at least twice (query decomposition/expansion) to aid search.
- Final output **must** be a Python list.
- Rephrased queries are short, concise keyword/entity mixes; you may replace `NVIDIA` with `employer` or `company`.
- Provide two or more rephrased queries preserving intent and timeframe.
- If the question asks for "the next *X* date" without time context, append `YYYY` (current or next year) in rephrased queries.

  Example: Question: "when is the next NTech conference" → "upcoming ntech 2024", "ntech dates 2024", "ntech schedule 2025".

Use the EnterpriseKnowledge tool when

The user asks for non-sensitive information such as organization info, direct reports, phone numbers, benefits alternate ID, email addresses, working addresses, tax explanations, updating SSN instructions, or stock trading policies.

Your action format MUST be

```
Thought: Provide a short analysis of
    your understanding from the Question
    .
Process: I need to use the Enterprise
    Knowledge tool
Action: EnterpriseKnowledge
Action Input: A single line Python list
    of rephrased queries MUST be
    generated.
```

Strict JSON schema (return nothing else)

```
{
  "type": "object",
  "properties": {
    "Question": {
      "type": "string",
      "description": "Generated Question
     from the input document."
    },
    "Answer": {
      "type": "string",
      "description": "Corresponding
    Answer from the input document that
    answers the Question."
    },
    "Thought": {
      "type": "string",
      "description": "Short analysis of
    your understanding from the Question
    ."
    },
    "Process": {
      "type": "string",
      "description": "I need to use the
    Enterprise Knowledge tool."
    },
    "Action": {
      "type": "string",
      "description": "
    EnterpriseKnowledge"
    },
    "Action Input": {
      "type": "array",
      "description": "A single line
    Python list of rephrased queries."
    }
```

```
  }
}
```

Examples

Input Document: <Content of input document>
Input Document url: <url of input document>

Output

```
{
  "Question": "I am based in the
    Netherlands, when is pay day?",
  "Answer": "25th of every month",
  "Thought": "Payroll timing question;
    include location keywords in
    rephrased queries.",
  "Process": "I need to use the
    Enterprise Knowledge tool",
  "Action": "EnterpriseKnowledge",
  "Action Input": [
    "payday schedule netherlands",
    "netherlands pay days"
  ]
}
```

Input Document: <Content of input document>
Input Document url: <url of input document>

Output

```
{
  "Question": "point me to gpu fcv page
    ?",
  "Answer": "https://nvidia.sharepoint.
    com/sites/TechnicalTraining/ASIC%20
    teams.aspx",
  "Thought": "Needs GPU FCV (Full Chip
    Verification) page.",
  "Process": "I need to use the
    Enterprise Knowledge tool",
  "Action": "EnterpriseKnowledge",
  "Action Input": [
    "gpu fcv page company",
    "fcv gpu url"
  ]
}
```

Input Document: <Content of input document>
Input Document url: <url of input document>

Output

```
{
  "Question": "ok, i'm looking for an
    NVIDIA icon for biotech /
    pharmaceuticals to use in a
    presentation. can you help me find
    that?",
  "Answer": "https://nvidia.sharepoint.
    com/sites/nvinfo/brand/Pages/default
    .aspx",
  "Thought": "Needs a company icon for
    biotech/pharma use.",
  "Process": "I need to use the
    Enterprise Knowledge tool",
  "Action": "EnterpriseKnowledge",
  "Action Input": [
    "company icons",
    "company logos biotech"
  ]
}
```

Task output format

Generate **3 pairs** by following the instructions based on the Input Document.
Strictly return only a Python list of pairs and nothing else.

Input Document: <Content of input document>
Input Document url: <url of input document>

Output: ###

H Corrected Issues Examples

Representative corrected failures are shown in Table 5, highlighting how the data flywheel improved routing accuracy and rephrasal quality across diverse user queries.

Table 5: Examples of corrected issues through the data flywheel

User Query	Original Failure	After Fine-tuning	Result
"What is the role of the RESS planning team at NVIDIA?"	Rephrasal error: incorrectly expanded to "Resource Planning team".	Correct rephrases: "NVIDIA RESS planning team role"; "RESS planning team responsibilities".	The role of RESS (Real Estate and Site Services) Planning team is to manage site operations, support lease delivery, ...
"How many vacation days does NVIDIA Canada have?"	Routing error: sent to Holiday Expert instead of Policies Expert.	Correctly routed to Policies Expert.	According to the Canada Vacation Policy, employees receive ...