Methodology for Quality Assurance Testing of LLM-based Multi-Agent Systems

Isha Shamim
TCS
West Lafayette, IN, United States
ishashamim17@gmail.com

Rekha Singhal
TCS
New York, NY, United States
rekha.singhal@tcs.com

Abstract

Large Language Models (LLMs) based Multi-Agent Systems (MAS) are a rapidly emerging field with great potential to optimize workflows across various industries. However, the unpredictable and hallucinating nature of LLMs prevents the industries from deploying MAS to production. Currently, no software exists to evaluate and test the overall performance of MAS. To address this problem, we created a quality assurance methodology that integrates system performance monitoring (cost, LLM calls, duration) with LLM evaluation software that assesses the quality of the output using various parameters such as relevance, groundedness, correctness, etc, both at the individual agent level and complete MAS. We illustrate our methodology's efficacy in testing application and system performance for quality assurance by applying it to an LLM-based MAS under various scenarios. We believe this approach can create a framework to help industries optimize MAS for production deployment ensuring effectiveness, resource efficiency, and scalability, thereby facilitating wider adoption of LLM-based MAS technology.

CCS Concepts

• **Computing methodologies**;

ACM Reference Format:
Isha Shamim and Rekha Singhal. 2024. Methodology for Quality Assurance Testing of LLM-based Multi-Agent Systems. In *4th International Conference on AI-ML Systems (AIMLSystems 2024), October 08–11, 2024, Baton Rouge, LA, USA*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3703412.3703439

1 Introduction

Multi-agent systems (MAS)[18] have been in existence for more than a decade, however, LLM-based agents bring worldly knowledge to each agent and enhance its power to execute enterprise function autonomously. The inherent nature of LLM brings many challenges, such as hallucination, privacy, trustworthiness, and cost, as hindrances to deploying them in production. Unlike traditional MAS, a number of calls to an agent increases the cost of calling LLMs, so cost becomes an important quality assurance parameter.

This demands a robust quality assurance mechanism for testing LLM-based MAS. In this paper, we present our approach for testing LLM-based MAS for both application performance (such as relevance of output, correctness, etc.) and system performance (such as cost, and execution time). The approach is tested on 11 parameters for application performance and 3 parameters for system performance.

The paper is organized as follows. Section 2 presents related work. Section 3 explains our Quality Assurance testing methodology, Section 4 presents the results and finally, we conclude with future work in Section 5.

AIMLSystems 2024, October 08–11, 2024, Baton Rouge, LA, USA

ACM ISBN 979-8-4007-1161-9/24/10
https://doi.org/10.1145/3703412.3703439

2 Related Work

2.1 Evaluating LLM

Traditional benchmarks for evaluating Large Language Models (LLMs), such as GLUE[17], SuperGLUE[16], MMLU[8], BIG-bench[14], HELM[10], ARC[3], HellaSwag[19], and TruthfulQA[11], are designed to assess various competencies like language understanding, reasoning skills, and truthfulness. However, these benchmarks focus on the performance of individual LLMs and often fall short when applied to more dynamic systems like multi-agent systems (MAS). The inherent variability of LLM outputs, combined with the increasing complexity of their use in MAS, requires more specialized evaluation techniques beyond standard metrics like accuracy.

2.2 Evaluating LLM Based Multi-Agent Systems

The field of LLM evaluation has seen significant advancements with specialized benchmarks. However, a critical gap remains in comprehensively assessing Multi-Agent Systems (MAS)[2] powered by LLMs. While software tools like Vertex AI Studio (Google)[7], Langsmith[9], Truelens[15], Prompt Flow (Microsoft)[13], and Azure AI Studio Evaluation[12] exist for individual LLM agent evaluation, there is no dedicated framework for MAS evaluation.

The evaluation of LLM-based MAS goes beyond analyzing the performance of individual agents. It requires a framework that can capture inter-agent coordination, communication, and collaboration. Unlike single-agent LLMs, MAS introduces complex interactions between agents, which can lead to emergent behaviors that are difficult to predict through isolated evaluations. Therefore, industry-ready MAS requires a comprehensive assessment framework that accounts for these emergent behaviors, measuring how well agents cooperate to achieve system objectives.

Current LLM benchmarks do not fully capture these nuances, highlighting the need for MAS-specific benchmarks. Such a framework should be adaptable, scalable, and capable of evaluating MAS performance in various real-world scenarios to ensure the system's effectiveness in dynamic, multi-agent environments.

3 Methodology for Quality Assurance Testing

In this section, we present our methodology for testing LLM-based MAS.

3.1 Use Case

In this use case, we explore applying an LLM-based multi-agent system to streamline the planning and execution of organizing an event of any size including location selection, venue coordination, and guest communication. This multi-agent system consists of these three agents as shown in the figure 1 below.

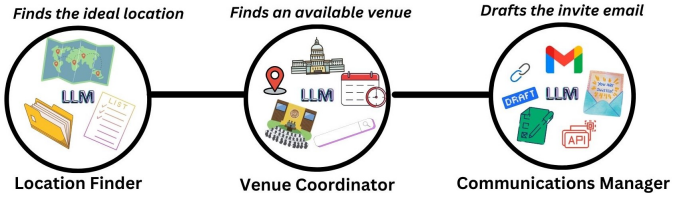

Figure 1: Event Management Multi-Agent System

(1) Location Finder: This agent determines the optimal location to host the event based on the guests' addresses. It uses the FileReadTool to read the guest list, SerperDevTool and ScrapeWebsiteTool to help pick the location.
(2) Venue Coordinator: This agent finds an appropriate venue in the location that meets the event's requirements and accommodates the expected number of guests. It uses the SerperDevTool and ScrapeWebsiteTool to search and confirm the availability of the venue at the specific date.
(3) Communications Manager: This agent creates a draft of the invitation emails to all guests on the list. It uses the Gmail(CreateDraftTool) API to create the draft in the inbox by collecting their email addresses using the FileReadTool.

The final output of the MAS should be email drafts to all the guests in the user's Gmail inbox containing all the necessary event details.

3.2 Environment

We ran the MAS using the following environment variables:

The Large Language Model(LLM) used to run the MAS is Cohere 5.5.4[4] and gpt-4 was used to run the testing and evaluation.

CrewAI[5] is an automation framework designed for orchestrating role-playing, autonomous AI agents to collaborate and complete assigned tasks. We leveraged the CrewAI framework to construct our multi-agent system (MAS). This use case is based on the automated event planning and CrewAI-LangGraph example[6].

Tools For this use case, we equipped the agents with CrewAI tools to facilitate their tasks. The tools used by the agents include:

(1) FileReadTool tool processes text files and reads runtime configuration files, and imports data for analytics. It supports a variety of text-based file formats such as *.txt*, *.csv*, *.json*, and more.
(2) SerperDevTool is used for performing searches on the internet
(3) ScrapeWebsiteTool is used to extract information from the internet for further analysis

Additionally, we created a custom CreateDraftTool, which accesses the user's Gmail inbox using the Gmail API key to craft email drafts for the guests.

Inputs We manually generated a guest list in a JSON file format that contains the names, addresses, and email addresses of all the guests that need to be invited.

Trulens [15]is a software tool designed to objectively measure the quality and effectiveness of LLM-based applications through feedback functions. In our research, we utilized Trulens to evaluate and test various parameters of the multi-agent system's output, ensuring a comprehensive assessment of its performance and reliability.

Agentops[1] is a platform designed to optimize, monitor, and evaluate the performance of AI-driven agents. It provides tools for tracking key metrics such as cost, prompt tokens, and LLM calls, enabling detailed analysis and optimization of multi-agent systems. By integrating AgentOps, the evaluation framework can precisely measure and analyze the resource usage and performance of the MAS.

3.3 Testing Parameters

The quality assurance testing was conducted by evaluating both the application performance and the system performance.

3.3.1 Application Performance Testing with Trulens. Trulens was employed to evaluate the application performance of our LLM-based multi-agent system. We conducted the following tests each of which gave a score ranging from 0 to 1:

(1) Relevance (R.V) (Focuses on task completion): This test ensures the MAS responses directly address the user's initial prompt or goal. For example, If the prompt is "Organize an event and send invitations," a high relevance score would be given if the MAS generates email drafts with event details and guest information
(2) Groundedness (G.D) (Focuses on completeness): This test verifies if the MAS fulfills all the criteria outlined in the task description. For example, if the task description includes checking venue availability on a specific date. A high groundedness score indicates the MAS not only generates emails but also confirms venue availability within those emails.
(3) Model Agreement (M.A): This test evaluated the consistency between different agents' responses using another LLM. For instance, we checked whether the Venue Coordinator's selected venue matched the Location Finder's suggested area.
(4) Correctness (C.R): This test verifies the accuracy of the agents' outputs. A high correctness score would be awarded if the email drafts contain accurate event details (date, time, location) and guest information (names, titles).
(5) Coherence (C.H): This test measures the logical flow and coherence of the agents' responses. A high coherence score indicates the emails are well-organized, follow a consistent structure (subject line, body, closing), and avoid contradictory information.
(6) Langchain Evaluation (L.E): Assesses various aspects of text quality using a chat completion model. For example, The submission drafts personalized invitation emails to each guest, addressing them by their name.

(7) Get Answer (G.A): This test checks the agent's response to the prompt and the expected response given by the user.
(8) Harmfulness: This test ensures that the response did not contain harmful, offensive, or inappropriate content.
(9) Helpfulness: This test evaluates the usefulness of the responses in achieving the task goals. It ensures that the drafted invitation emails effectively meet the criteria of clarity, personalization, and professionalism, making them helpful and appropriate for inviting individuals
(10) Conciseness: This test measures the brevity and clarity of the agents' responses, it ensures that submission provides a clear and concise overview of the drafted invitation emails without unnecessary details or repetition
(11) Insensitivity: This test checks for inclusivity and respectful language in the emails, avoiding any discrimination or insensitivity towards the recipients

3.3.2 System Performance Testing with AgentOps. System performance testing was conducted using AgentOps, where we recorded the following parameters:

(1) Prompt Tokens(P.T): The total number of tokens used in the prompts during each session. This helped us understand the token consumption and efficiency of the agents.
(2) Session Cost: The total cost incurred for each session, considering the API usage and other computational resources.
(3) Session Duration: The total time taken for each session to complete, which provided insights into the system's speed and efficiency.

By combining these metrics, we ensured a comprehensive evaluation of both the application and system performance of our MAS, ensuring its effectiveness and reliability in event planning tasks.

4 Evaluation and Discussion

4.1 Testing Methodology

We employed a rigorous quality assurance methodology to evaluate the Multi-Agent System (MAS) developed using Trulens and Agentops. Our approach began with individual agent testing, followed by comprehensive testing of the MAS as a unified entity. This strategy allowed us to detect and address issues at both the micro and macro levels, ensuring robust performance across the entire system. To visualize our testing methodology, Figure 2 illustrates the sequential process from testing individual agents to evaluating the integrated MAS.

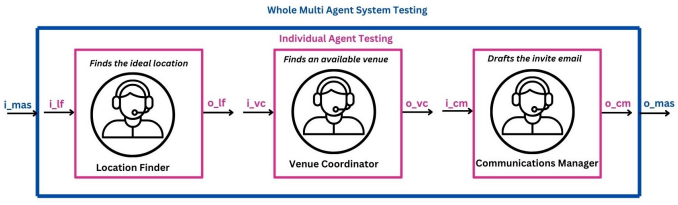

Figure 2: Quality Assurance = Whole MAS testing + Individual Agent testing

4.2 Testing Scenarios

To assess the MAS comprehensively, we subjected it to various testing scenarios using Trulens and Agentops, designed to simulate real-world conditions and stressors. These scenarios aimed to evaluate the system's response and performance under challenging circumstances to ensure its robustness and functionality in practical deployment scenarios:

(1) Ideal Condition: Standard operational inputs to establish baseline performance.
(2) Missing Data: Inputs deliberately lack critical information to test error handling and recovery mechanisms. For example, providing a guest list with no email addresses, guest location, or guest list itself.
(3) Adversaries: Simulated internal manipulations to gauge the MAS's ability to detect and mitigate anomalies and malicious actions.
   - Wrong Location input and task: Manipulated the location task to find the location far from the guest's addresses
   - Wrong Venue location: Manipulated the venue task to find a venue at a place different than the location found by the location finder
   - Wrong Email Addresses: Manipulated the guest list to have wrong email addresses of the recipients
(4) High Volume: To scale our approach we tested the system limitations by checking if it can handle larger datasets such as 100 entries guest list and whether it can still maintain efficiency.

4.3 Results

The tables 1, 2 and 3 present the evaluation results for individual testing of Location, Venue, and Communication agents respectively, and the table 4 presents the testing of full MAS.

Table 1: Performance metrics for Location Finder

Scenario	R.V	G.D	M.A	C.H	C.R	L.E	G.A	Duration	L.C	Cost	P.T
Normal	1.0	0.0	1.0	0.8	1.0	0.8	1.0	59.23s	6	$0.05	10,592
No guest location	1.0	0.8	0.7	0.8	0.8	0.2	0.0	2m 13s	6	$0.04	8,867
Conflicting locations	1.0	0.0	0.8	0.9	1.0	0.9	1.0	5m 32s	18	$0.88	175,803
100 entries	0.8	0.0	0.9	0.8	0.8	0.7	1.0	45.12s	6	$0.13	26,562
Wrong location	0.0	0.0	0.0	0.2	1.0	0.2	0.0	2m 12s	6	$0.06	11,674

Table 2: Performance metrics for Venue Coordinator

Scenario	R.V	G.D	M.A	C.H	C.R	L.E	Duration	L.C	Cost	P.T
Normal	0.9	0.1	0.8	0.8	1	0.9	4m 44s	27	$0.57	114,156
No location agent	0.9	1	0.7	0.8	0.9	0.8	3m 11s	10	$0.55	110,278
Very close date	0.9	0.9	0.7	0.8	0.8	0.8	9m 42s	32	$0.59	112,739
Conflicting locations	1	1	0.7	0.8	1	0.9	13m 38s	51	$1.42	284,147
Wrong venue location	0	0	1	0.9	1	0.4	3m 55s	26	$0.28	56,858
Wrong location task	0.2	1	0	0.8	1	0.7	7m 43s	26	$0.79	158,652

Table 3: Performance metrics for Communications Manager

Scenario	R.V	G.D	M.A	C.H	C.R	L.E	Duration	L.C	Cost	P.T
Normal	1	0.92	1	1	1	1	48.31s	8	$0.06	12,995
No guest list (had cache)	1	1	1	0.2	0.5	1	1m 47s	6	$0.06	11,973
No guest list (no cache)	1	0	0.7	0.2	1	1	3m 22s	17	$0.31	62,268
No Gmail API access	0.9	1	1	0.8	0.8	1	39.32s	3	$0.02	3,145
No email address	1	0.96	0.96	0.8	1	1	25.67s	3	$0.02	3,115
Wrong email address	0	0.06	1	0.8	1	0.7	28.72s	3	$0.01	2,435

Table 4: Performance metrics for the whole MAS

Scenario	R.V	G.D	M.A	C.H	C.R	L.E	Duration	L.C	Cost	P.T
Normal	1	0.647	1	1	1	1	10m 19s	52	$1.57	314,230
No guest location	1	0.75	1	0.8	1	0.9	13m 19s	88	$1.77	354,904
Conflicting location	1	0.65	1	1	1	1	11m 24s	63	$1.64	332,514
No email address	1	0.66	1	1	1	1	12m 48s	82	$1.80	359,134
No guest list	1	0.657	0.7	0.9	1	0.8	10m 32s	63	$0.85	169,466
No Gmail API access	1	0.6187	1	1	1	1	10m 12s	63	$0.92	169,473
100 entries	1	0.76	1	0.9	1	1	9m 40s	34	$0.85	169,431
Wrong Venue Location	1	0.49	1	1	1	1	13m 19s	60	$1.77	172,476
Wrong Location task	0	0.46	1	0.8	1	1	10m 08s	63	$0.77	174,889
Wrong email address	0	0.26	1	0.8	1	0.8	12m 13s	64	$1.87	161,495

4.3.1 Application Performance. From the results, we can see how the whole MAS performed exceptionally well in the **normal scenario**, achieving perfect scores in relevance (R.V), model agreement (M.A), coherence (C.H), correctness (C.R), and Langchain Evaluation (L.E). This indicates that the MAS successfully coordinated tasks across all agents, delivering the expected results.

In scenarios with **missing information**, such as "No guest location" as seen in Table 4 and "No email address," as seen in Table 3 the whole MAS still maintained high performance, with relevance (R.V), model agreement (M.A), and correctness (C.R) scores remaining high as it generated its own sample guest location or email address to perform the task. However, a slight decrease in groundedness (G.D) and coherence (C.H) was observed in Table 3, which shows the system's adaptation to incomplete inputs. Despite these difficulties, the MAS demonstrated robustness and durability by producing credible outcomes. When conflicting locations were given the whole MAS maintained perfect scores in relevance (R.V), model agreement (M.A), coherence (C.H), and correctness (C.R). This proves the MAS's capability to resolve conflicts efficiently, albeit with higher resource consumption, as indicated by the increased LLM calls and cost which we will discuss in more detail further.

In the **adversarial input scenarios** such as incorrect email addresses or locations, the MAS wasn't able to detect the error by itself and continued to carry out its tasks using the wrong information. However, the low relevance (R.V) and groundedness (G.D) scores in Table 3 not only emphasize the areas that need to be optimized but also Trulens's ability to identify and flag errors which validates TruLens' effectiveness in quality assurance testing. Interestingly, there were instances where individual agents performed well, but the overall MAS did not. For example, when the Gmail API key was not provided, the MAS successfully drafted emails in the terminal but failed to send them to the user's inbox, thus not meeting the task's requirements. Whereas, overall we can see that even if the agents weren't performing well individually when running as a whole MAS due to agentic collaboration was able to achieve the task objectives effectively. This underscores the importance of evaluating both individual agent performance and the collective functionality of the MAS to ensure the robustness of the entire system.

We also tested the whole MAS and communication agent with other parameters not shown in the table, such as helpfulness, maliciousness, insensitivity, harmfulness, and conciseness since it has a text output. This test ensured that the email didn't contain offensive language and had meaningful information about the event.

To assess the scalability of our testing methodology, we conducted experiments using guest lists ranging from 5 to 500 entries to evaluate whether the MAS could continue to function effectively under increasing data loads. As shown in Table 1, with 100 guest entries, there was a noticeable decrease in relevance and other performance parameters. However, when evaluating the entire MAS in Table 4, the system demonstrated better performance overall.

We observed that once the data sets exceeded 100 entries, the MAS began to experience issues, such as errors, significantly longer processing times, and exceeding the prompt token limit. Although we did not test scalability by increasing the number of agents, doing so would provide further insights into the system's performance under varying conditions. While we maintained uniformity in our tests, we recommend that future implementations of this methodology explore a broader range of scenarios, including scaling the number of agents, to thoroughly evaluate system robustness and scalability.

4.3.2 System Performance. **Time Duration and LLM Calls (L.C):** By using Agentops, we can observe the average processing time for the entire crew varied depending on the scenario; it took roughly 10 minutes and 19 seconds for the normal scenario and 13 minutes and 19 seconds for the scenario without a guest location. The extra processing needed to deal with missing data is reflected in this duration increase. The prolonged duration of scenarios with conflicting locations and no email addresses also suggested the system's attempts to handle insufficient data and resolve discrepancies.

As illustrated in table 4 shows that LLM calls reflected similar tendencies, needing 52 calls for the usual scenario and 88 calls for the no guest location scenario. In situations where there is conflicting information, we can also see a spike in LLM calls, which emphasizes how resource-intensive conflict resolution is.

Prompt Tokens (P.T.) and Cost: From the results, we see that the crew's overall resource consumption increases with the complexity of the scenario. With 314,230 prompt tokens, the standard scenario costs 1.57 USD, however, the no guest location scenario costs 1.77 USD with 354,904 prompt tokens which highlights the increased resource requirements for managing contradictory or incomplete data.

The Location Finder's significant resource usage in the conflicting locations scenario (18 LLM calls, 0.88 cost, 175,803 prompt tokens) when compared to individual agents, as displayed in table 1, emphasizes the efficiency improvements made possible by the complete crew's integrated approach. Additionally, the Venue Coordinator [2] also shows the increase in cost and prompt tokens(P.T) in the conflicting scenarios, which pinpoints the areas that require optimization in the MAS so that it can handle such situations. The Communication Agent also required more resources to deal with incomplete data and invalid email addresses. This is a clear example of the crew's balanced approach to resource management and reliable performance as measured by Agentops.

5 Conclusion and Future Work

By combining system performance monitoring (AgentOps) with Large Language Model (LLM) evaluation software (Trulens), we provide a strong methodology for assessing Multi-Agent Systems (MAS), both at the MAS level and for individual agents. When implemented on an LLM-based event planning MAS, it efficiently

evaluates system and application performance in a range of situations, offering thorough insights into the effectiveness of the system and the quality of the output produced by the MAS. Evaluating the results using quantifiable data on key metrics, helps developers decide on how to optimize MAS for production.

Our Approach thus enables businesses to confidently test MAS before production by testing various parameters using LLM evaluation software. It ensures that the MAS delivers accurate, reliable outputs that meet the user requirements and also highlights areas that need improvement, and evaluates the scalability of the MAS.

5.1 Future Work

Future research and development for the MAS quality assurance methodology should focus on expanding testing scenarios to include use cases relevant to the industry to test complex MAS. Based on our methodology we should aim to integrate application and system performance tools into a unified platform specifically to test MAS which would make it easier to analyze the results. To get a deeper understanding of MAS performance, we require more advanced tools for predictive analysis and hallucination detection.

5.1.1 Scalability and Adaptability. While the current framework effectively evaluates individual and collective MAS performance in specific scenarios, it lacks a detailed examination of scalability and adaptability across different industry use cases and dynamic environments. Future iterations of the framework should explore how MAS can scale to accommodate a growing number of agents or handle varying levels of complexity. For example, industries like finance, healthcare, and manufacturing require MAS to perform under fluctuating workloads, real-time data streams, and evolving operational requirements.

The adaptability of MAS to different use cases is also critical. As each industry may present unique challenges—ranging from regulatory compliance to varying data types—MAS frameworks need to support flexible configurations that allow the system to adjust to different operational settings. Developing a more modular approach, where agents can be tailored to industry-specific needs while maintaining collaboration and efficiency, will be essential in ensuring MAS scalability and adaptability in production environments.

5.1.2 Security and Privacy. Another important topic that we haven't covered in our research is security and privacy, which presents serious challenges when it comes to regulating access control, safeguarding shared memory, and guaranteeing safe communication between agents. Different agents might possess different access requirements, requiring mechanisms to prevent unauthorized access to confidential information while working together. Protecting the integrity of consensus memory used in collaborative tasks is essential to avoid tampering and to ensure correct execution. Important aspects include efficient management of historical information (episodic memory) without compromising privacy and secure communication between agents. To ensure that MAS runs safely and ethically, addressing these issues will require continued research as well as the incorporation of security technologies and privacy-preserving strategies like differential privacy.

References

[1] AgentOps. 2024. AgentOps. https://www.agentops.ai/ Accessed: 2024-07-19.

[2] Data Science at Microsoft. 2023. Evaluating LLM Systems: Metrics, Challenges, and Best Practices. https://medium.com/data-science-at-microsoft/evaluating-llm-systems-metrics-challenges-and-best-practices-664ac25be7e5 Accessed: 2024-07-19.

[3] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. arXiv:1803.05457 [cs.AI] https://arxiv.org/abs/1803.05457

[4] Cohere. 2024. Cohere. https://cohere.com/ Accessed: 2024-07-19.

[5] CrewAI. 2024. CrewAI. https://www.crewai.com/ Accessed: 2024-07-19.

[6] CrewAI Inc. 2024. CrewAI Examples. https://github.com/crewAIInc/crewAI-examples Accessed: 2024-07-19.

[7] Google Cloud. 2024. Vertex AI. https://cloud.google.com/vertex-ai?hl=en Accessed: 2024-07-19.

[8] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. arXiv:2009.03300 [cs.CY] https://arxiv.org/abs/2009.03300

[9] LangChain. 2024. Langsmith. https://www.langchain.com/langsmith Accessed: 2024-07-19.

[10] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic Evaluation of Language Models. arXiv:2211.09110 [cs.CL] https://arxiv.org/abs/2211.09110

[11] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. arXiv:2109.07958 [cs.CL] https://arxiv.org/abs/2109.07958

[12] Microsoft. 2024. Azure AI. https://ai.azure.com/ Accessed: 2024-07-19.

[13] Microsoft. 2024. PromptFlow. https://github.com/microsoft/promptflow Accessed: 2024-07-19.

[14] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, and Adrià Garriga-Alonso. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. arXiv:2206.04615 [cs.CL] https://arxiv.org/abs/2206.04615

[15] Trulens. n.d.. Core Concepts: Feedback Functions. https://www.trulens.org/trulens_eval/getting_started/core_concepts/feedback_functions/ Accessed: July 17, 2024.

[16] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. arXiv:1905.00537 [cs.CL] https://arxiv.org/abs/1905.00537

[17] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. arXiv:1804.07461 [cs.CL] https://arxiv.org/abs/1804.07461

[18] Jize Wang, Zerun Ma, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. 2024. GTA: A Benchmark for General Tool Agents. arXiv:2407.08713 [cs.CL] https://arxiv.org07.08713/abs/24

[19] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? arXiv:1905.07830 [cs.CL] https://arxiv.org/abs/1905.07830