RCAgent: Cloud Root Cause Analysis by Autonomous Agents with Tool-Augmented Large Language Models

Zefan Wang*
Tsinghua University
Beijing, China
wang-zf20@mails.tsinghua.edu.cn

Zichuan Liu*
Nanjing University
Nanjing, China
zichuanliu@smail.nju.edu.cn

Yingying Zhang†
Alibaba Group
Hangzhou, China
congrong.zyy@alibaba-inc.com

Aoxiao Zhong
Havard Univerrsity
Cambridge, USA
aoxiaozhong@g.harvard.edu

Jihong Wang*
Xi'an Jiaotong University
Xi'an, China
wang1946456505@stu.xjtu.edu.cn

Fengbin Yin
Alibaba Group
Hangzhou, China
yinfengbin.yfb@alibaba-inc.com

Lunting Fan
Alibaba Group
Hangzhou, China
lunting.fan@taobao.com

Lingfei Wu
Anytime AI
New York, USA
lwu@anytime-ai.com

Qingsong Wen†
Squirrel Ai Learning
Bellevue, USA
qingsongedu@gmail.com

ABSTRACT

Large language model (LLM) applications in cloud root cause analysis (RCA) have been actively explored recently. However, current methods are still reliant on manual workflow settings and do not unleash LLMs' decision-making and environment interaction capabilities. We present RCAgent, a tool-augmented LLM autonomous agent framework for practical and privacy-aware industrial RCA usage. Running on an internally deployed model rather than GPT families, RCAgent is capable of free-form data collection and comprehensive analysis with tools. Our framework combines a variety of enhancements, including a unique Self-Consistency for action trajectories, and a suite of methods for context management, stabilization, and importing domain knowledge. Our experiments show RCAgent's evident and consistent superiority over ReAct across all aspects of RCA—predicting root causes, solutions, evidence, and responsibilities—and tasks covered or uncovered by current rules, as validated by both automated metrics and human evaluations. Furthermore, RCAgent has already been integrated into the diagnosis and issue discovery workflow of the Real-time Compute Platform for Apache Flink of Alibaba Cloud.

CCS CONCEPTS

• **Computing methodologies → Natural language processing**;
• **Software and its engineering → Cloud computing**.

KEYWORDS

Root Cause Analysis, Large Language Model, Cloud Systems

1 INTRODUCTION

Cloud computing platforms have been increasingly utilized for application and service deployment in recent years [8, 29]. Anomalies in cloud computing systems, such as unrecoverable failures and hanged jobs, severely impact customer experience and can potentially violate service level agreements[2, 58]. Root Cause Analysis (RCA) [1, 31, 61], a core component of site reliability engineering, is currently receiving ongoing attention from large cloud computing enterprises such as Amazon, Microsoft, Google, and Alibaba. To increase the efficiency of cloud service reliability enhancement, a series of Artificial Intelligence for Operations (AIOps) approaches [9, 49, 59] have been widely adopted in RCA to reduce the MTTR (mean time to resolve). While these typical AIOps aid in automated processes, their application faces challenges such as poor data quality, shifting data distribution, laborious data annotation, and limited generalization for models [11].

The advancements in Large Language Models (LLMs), especially within the GPT [5, 32–34] and LLaMA [45, 46] families, indicate an intriguing future of solving intricate reasoning tasks. Recent works demonstrate the use of LLMs in cloud RCA tasks. [2, 17] fine-tune GPT models for root causes summarization. These works rely heavily on the computationally expensive supervised adaption to cloud system tasks and do not fully utilize the generalization and reasoning abilities of LLMs. One possible solution is to use few-shot RAG [22] on LLMs, with representative methods such as RCACopilot [10], PACE-LM [60], and Xpert[16]. However, these works are

all based on the GPT family and scenarios within Microsoft, not addressing the data privacy concerns associated with using LLMs with cloud system data. Furthermore, none of the above methods leverage the autonomous capabilities of LLMs for information collection, decision-making, and environmental interaction [55].

Tool-augmented autonomous agents, as demonstrated in early experiments [6], further unlock the potential of LLMs in interactive environments. By equipping LLMs with defined tools and associated documentation, and by facilitating tool invocation through mechanisms like function calls or command line inputs, and then executing these tools and returning environmental feedback, LLMs can handle tasks that require extensive expertise and abilities. A representative paradigm within the realm of autonomous agents is ReAct [55], a workflow that embodies a thought-action-observation loop and offers flexibility for extensions [27]. However, the adoption of LLM agents in the AIOps field, especially with noisy and lengthy data, remains limited [21, 28]. The primary challenges are action validity and context length, both of which heighten the demands of LLM-as-agent capabilities [26]. Also, to the best of our knowledge, there is no interactive environment built upon realistic production-level RCA problems for LLM agents to operate on.

To this end, we introduce **RCAgent**, the first practical LLM-based RCA framework within the tool-augmented autonomous agent paradigm. We design an enhanced prompting cycle skeleton and an interactive environment enriched with external knowledge and stabilization techniques, tailored for LLM agents to handle diverse data types. Additionally, we design aggregation methods for action trajectories and text output, combining suboptimal results from LLMs. Unlike ReAct, our approach operates in a trajectory-level zero-shot way, eliminating the need for manual or auto-generated action examples. Furthermore, to facilitate general and secure industrial usage, we forgo the use of powerful external API models and implement this framework on a locally deployed model, underscoring the efficacy of our stabilization method.

The analysis results from RCAgent are being utilized in the Real-time Compute Platform for Apache Flink of Alibaba Cloud to diagnose anomalous stream processing jobs uncovered by current methods. We have incorporated a feedback mechanism in the company to identify issues in the PaaS and IaaS layers of the cloud system, offering insights for development teams.

We summarize our contributions as follows:

- We propose RCAgent, the first tool-augmented agent based on LLM for privacy-aware real-world cloud RCA, unleashing the decision-making ability of LLMs in the AIOps field.
- We introduce a bag of methods to enhance the tool agent, including aspects of prompting framework, tool setting, stabilization, and aggregation methods. These make the agent based on locally deployed LLM a valid solution for complex environments like cloud systems.
- We demonstrate the practical usage of RCAgent with real-world experiments on computing jobs in Alibaba Cloud.

2 CHALLENGE

Though tool-augmented LLM agents provide new possibilities for the cloud RCA task, including autonomous decision-making and
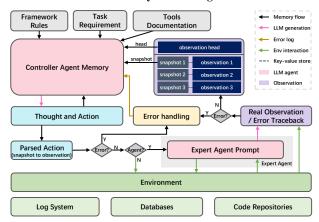

Figure 1: Overview of the action cycles from RCAgent. The cycle involves generating verbal thoughts, taking actions, and receiving observation from the environment, all of which are recorded in the prompt alongside the initial memory to boost reasoning. Besides, RCAgent includes the key-value store for observation retrieval, allowing the agent to operate on lengthy text data. After parsing the action, RCAgent executes it directly or invokes an expert agent, depending on its type.

handling unseen anomalies without laboriously annotated training data, several critical challenges exist.

Privacy. A general LLM method for RCA and other cloud AIOps tasks should be internally hosted for security concerns. Specifically, transmitting production-level confidential data to external API induces privacy risks. This means stronger models like ChatGPT cannot be used except for those close collaboration enterprises. While trading off the model's ability for security, we need techniques to mitigate the gap to some extent.

Context Length. A fundamental problem for agent usage in realistic cloud environments is context length because various kinds of data, such as logs, code, and database query results, tend to be enormous. Even if the LLMs can extrapolate to larger context length[35], processing unnecessarily excessive tokens is highly inefficient.

Action Validity. Open-ended action generation for LLMs is of great challenge because less sufficiently aligned LLMs have a larger possibility of generating invalid actions[26]. These errors severely damage the performance of autonomous agents on comprehensive tasks. The model restriction from privacy concerns and noisy cloud data this problem even more arduous.

3 METHODOLOGY

To systematically and reliably prompt the LLM as a tool-augmented autonomous agent for cloud RCA, we propose **RCAgent**, an enhanced reasoning and acting framework. An overview of our methodology at the decision-loop level is shown in Figure 1. For disambiguousity, the LLM agent with the prompt of thought-action-observation loop is named the *controller agent* responsible for coordinating actions, and RCAgent additionally employs the LLM as tools called the *expert agents* for domain-specific functionalities.

In concordance with the typical implementation of ReAct tool agent prompt framework [38], the controller agent is injected with three basic prompts: (i) framework rules that describe the thought-action-observation loop, (ii) task requirements that contain instructions for the RCA tasks with basic cloud knowledge, and (iii) tools documentation that describes the description of all invokable tools. Because of its flexibility and readability, JSON is chosen as the data interchange format for all generations in the action step from LLM. We also define a tool named 'finalize' as an exit point that allows the model to freely decide when to report findings in a parsable format. Note that RCAgent discards the few-shot examples compared to the original ReAct because of limited context length.

Starting from the tool agent version of ReAct, we propose several enhancements to address the challenges of using tool-augmented LLM agents in cloud RCA. To deal with the context length challenge described in § 2, we first invent an observation management method for compressing token usage introduced in § 3.1. Then, because the LLM itself does not have access to and enough domain-specific knowledge about the cloud system, we build tools including LLM-augmented ones for the RCAgent, whose designs are described in § 3.2. Addressing the action validity problem, RCAgent has stabilizing methods presented in § 3.3. Lastly, to improve the performance of RCAgent since we are using less capable locally hosted LLM, we utilize the aggregation method in RCAgent described in § 3.4.

3.1 Observation Snapshot Key

One of the basic challenges of building autonomous agents in a comprehensive cloud environment is context length [10]. The most inflating part of the agent prompt is the observation content in the action trajectories, containing a large amount of logs, table entries, etc. To overcome the information loss from truncating and summarizing observations, we propose OBservation Snapshot Key (**OBSK**), a new method to address the context length problem in realistic cloud tasks. As shown in Figure 1, OBSK only shows the head of observation to the controller agent, leaving a hash ID (snapshot key) for further usage. A key-value store is built for mapping the snapshot key to real observation. Thus, when a snapshot key is found in a parsed action, RCAgent queries through the key-value store and returns the corresponding observation. This ensures necessary information with controlled length is provided for the controller agent as supportive information for decision-making.

3.2 Tool Preparation

We employ data querying functions as *information-gathering tools* and LLM-based expert agents as *analytical tools*, similar to the data collection and analysis process done by human SREs.

3.2.1 Information-gathering Tools. Information-gathering tools are designed in an easy-to-use way, hiding all unrelated details in accessing data in cloud systems. For example, instead of giving SQL interface and Log query API to LLM, these tools only accept simple parameters like the ID of entities. This semantically minimalist tool setting will reduce the threshold for LLMs to take valid actions, preventing useless exploration in large data warehouses. Also, to avoid the agent's ineffective analysis, we deduplicate similar information and exclude messages beneath the WARNING level.
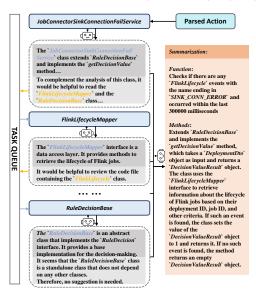

Figure 2: Code analysis tool in RCAgent.

3.2.2 Analytical Tools. Analytical tools are proposed to extend the domain knowledge and abilities of the controller agent augmented by LLMs with their reasoning ability. We name this kind of analytical tool the expert agent, which is shown in Figure 1. We provide two expert agents for RCAgent as complementary knowledge tools, called the *Code analysis tool* and the *Log analysis tool*. Both generate analyses and aggregations prompted by the zero-shot Chain-of-Thought (CoT) [19] and answer extraction instructions.

Code analysis tool. The code analysis tool works in a recursive manner, which is shown in Figure 2. Given a class name, the code analysis tool searches the corresponding file in the code repository. After the LLM reading and analyzing the code file, it is prompted to suggest any other classes that would be helpful to analyze as supportive information. These suggestions from each code-reading round will be stored in a task queue, managing all pending tasks. With this exhaustive search, the code analysis tool stops parsing when no more code files of interest are recommended, or when all remaining recommended files are external dependencies. Then, we utilize an LLM to summarize all the code files, whose result is presented to the controller agent as the observation.

Log analysis tool. The log analysis tool operates in an in-context RAG paradigm with some adaptions to lengthy log data. The complete mechanism is shown in Algorithm 1. We split the log L into lines S and built edges between lines with the cosine similarity of embedding exponentially decayed by document distance as weights W. This yields a weighted undirected dense graph (S, W) regarding lines as vertices, describing the relevance of lines that are weakened when other lines are inserted in the middle. Then the graph is clustered with Louvain community detection [4], and the overlaps between clusters are removed by greedily switching the minimum amount of clustering labels. This clustering functions as semantic partitioning, and the result log chunks P are then fed into the log agent one chunk per round to perform Retrieval-Augmented Generation (RAG) [22]. Moreover, we instruct the expert agent to output evidence supporting its analysis by directly copying log content, preventing hallucinations of analyzing examples rather

Algorithm 1 Pseudo code for log expert agent.

Require: Log L, Max prompt length N
Ensure: Interpretations $\tilde{R}$, Evidences $\tilde{E}$
1: $S \leftarrow$ split L using delimiters (e.g., newline)
2: $\mathbf{v}_s \leftarrow$ EMBEDDINGMODEL(s) for each s in S
3: $W = \{w_{ij}\}$ empty weight matrix
4: **for** pairs (s_i, s_j) in $S \times S$ where $j - i \in (0, 200]$ **do**
5:     $d_{ij} \leftarrow$ position distance between s_i and s_j in L
6:     $w_{ij} \leftarrow CosSim(\mathbf{v}_{s_i}, \mathbf{v}_{s_j}) \times exp(-d_{ij})$
7: **end for**
8: $C \leftarrow$ LOUVAINCLUSTERING(S, W)
9: $C' \leftarrow$ GREEDYOVERLAPREMOVAL(C)
10: $P \leftarrow$ partitions from L indicted by components C'
11: $R', E' \leftarrow$ empty initialized filtered results
12: **for** each partition p in P **do**
13:     $E, A \leftarrow$ retrieved sorted examples and answers
14:     $ICP \leftarrow E[0:N], A[0:N]$ ▷ In-context prompt
15:     $R, E \leftarrow$ LLMANALYSIS(ICP, p)
16:     **for** each (r, e) in R, E **do**
17:         **if** LEVENSHTEIN(e, p) < L(p) - L(e) × 0.9 **then**
18:             $R', E' \leftarrow R' \cup r, E' \cup e$ ▷ Filter hallucinations
19:         **end if**
20:     **end for**
21: **end for**
22: $\tilde{R}, \tilde{E} \leftarrow$ LLMSUMMARY(R', E')

than the partitioned chunk. If the evidence listed by LLMs cannot be fuzzy-matched to the chunk p, the analysis result is discarded. Thus, we ensure reliable RAG on lengthy non-natural language.

3.3 Stabilization

To overcome the degradation of action validity induced by noisy data and less capable local LLMs, we introduce two stabilizations.

3.3.1 JSON Repairing.
One of the vital problems in real-world applications of tool-augmented LLM autonomous agents is structured inference for parsable data. To our knowledge, there is no pain-free method to guarantee a specific data format (e.g., JSON) for interactions between LLM agents and the environment. Even though there are some structure helper toolkits, such as Outlines [53] and TypeChat [15], they either cannot generate free-form JSON with extensive escape characters while not impair generation quality, or solely rely on LLMs' capability of token error correction. To solve this issue, we employ an intuitive and effective method to generate structured interchange data named **JsonRegen**.

Before LLM inference, all sensitive characters that may correspond to control symbols in JSON are replaced with insensitive ones for a clean prompt. When trivial cleaning fails to make the JSON-like string parsable, a regeneration process is performed. To enforce the LLM to understand JSON structure, we instruct it to convert the content to YAML. The LLM is then prompted to regenerate a JSON with the same structure and content. The regeneration proceeds for several rounds or until a valid JSON is parsed.

3.3.2 Error Handling.
The previous work [38] demonstrates that LLMs in tool invocation tend to propagate errors, limiting exploratory actions. These issues are even more pronounced in less
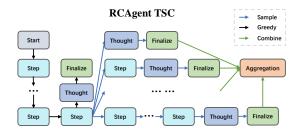

Figure 3: Trajectory-level Self-Consistency. Every Step in RCAgent means a sequential procedure of thought, action, and observation.

capable LLMs. Inspired by [42], we use pre-defined criteria to mark problematic actions or states as erroneous. As shown in Figure 1, we provide error messages and suggestions to the controller agent, including these circumstances: (i) duplicate invocation of stateless tools with the same set of arguments, (ii) trivial input to expert agents, and (iii) early finalizing without thorough investigation. These error messages can reduce the frequency of meaningless actions taken by the control agent by alerting it.

3.4 Self-Consistency Aggregation

Self-Consistency (SC) [50] has proved its efficacy in various close-ended NLP tasks while aggregating sampled open-ended multi-step generation like RCA with LLM agent is underexplored. To our knowledge, utilizing SC on ReAct style trajectories is also not well-defined. Thus, we propose applying the SC paradigm to free-form generation on the topic of LLM autonomous agents.

3.4.1 Self-Consistency for Text Data.
To apply SC to text data, we utilize two methods in our experiments:

Vote with embedding. . We directly generalize the idea of un-weighted SC (majority vote), which performs best across all tasks [50]. The voting can be rewritten as $\arg\max_i(\text{Sim}(\mathbf{a}_i, \frac{1}{K}\sum_j^K(\mathbf{a}_j)))$, where K is sample count, and $\mathbf{a}_i$ is a one-hot vector representing sampled result i with each position as a candidate choice or numerical result. We simply replace $\mathbf{a}$ as semantic embeddings for text output. This intuitively means the text result closest to the majority is chosen.

Aggregate with LLMs. . Considering the possible diversity of generated content, we prompt LLM to aggregate the candidates and output in similar form and length.

3.4.2 Self-Consistency for Tool Using Trajectories.
SC has been comprehensively tested on CoT reasoning paths, and can naturally be utilized on ReAct style trajectories. However, directly sampling multiple cycles of thought-action-observation can be expensive. This is even more costly while some actions, such as activating expert agents, require heavy consumption. Moreover, random sampling without any history actions or few-shot examples leads to flooding errors, e.g. consecutive calling non-existent tools.

Therefore, we propose a mid-way sampling method named Trajectory-level Self-Consistency (**TSC**) as shown in Figure 3. Specifically, only when the controller agent is stepping into finalization does the sampling start. This sampling strategy shares most preliminary steps between trajectory samples and reduces unnecessary consumption. Besides, the more stable action history from greedy decoding provides exemplification without additional context-length consumption from few-shot examples, suppressing

the validity drop from sampling. This method strikes a balance between full-process SC on agent trajectories and one-step CoT SC.

4 EXPERIMENT

We develop and evaluate RCAgent on the Real-time Compute Platform in Alibaba Cloud, an enterprise-level and high-performance system capable of real-time stream data computation based on Apache Flink. This system achieves a throughput of 100 million data records per second during peak hours.

4.1 Model Configuration

Our implementation is based on **Vicuna-13B-V1.5-16K** [62] with vLLM [20] backend on a single NVIDIA A100 SXM4 GPU (80 GB). We use the greedy decoding strategy by default for better reproducibility and stability. During self-consistency where random sampling is required, we use the default configuration of Vicuna.

The embedding model we use is **GTE-LARGE** [24], for its slightly better results on MTEB [30] than text-embedding-ada-002, providing an internally deployable substitute.

For Self-Consistency results, we use 10 output samples by default. We employ a step-wise Self-Consistency denoted as **SC**, which only accepts samples that finalize synchronously after the greedy decoding trajectory, allowing no additional action steps.

4.2 Dataset Preparation

4.2.1 Anomaly Selection. For root cause analysis, we collect a dataset of 15, 616 anomalous jobs of one-month cloud system history, either unrecoverable fail, or fail to start in 6 minutes. We filter the data and obtain ~5, 000 non-trivial anomalous jobs with substantial log content. We use the Flink Advisor knowledge base, which is a large rule set distilled from experienced SREs' domain knowledge, to create analysis results for these jobs. Due to the imbalance of anomalies, which means a large proportion of anomalies have the same root cause, we reduce the successfully analyzed jobs to an offline dataset of 161 jobs. The reduction is done with the class-balance constraint that no more than two jobs have identical root causes. The required annotation of these jobs contains four items: the root cause, solution, evidence, and responsibility determination. We first use LLM to summarize the analysis from Flink Advisor and output the above four items. Then the SRE team proofreads and annotates the target output.

We guarantee that our annotations do not show uninformative patterns like "The root cause of this anomaly is …" that cause some of the semantic scores untrustworthy discovered in [2, 17].

4.2.2 Data Sources. The available data sources, on which we build information-gathering tools for the LLM agent, include:

- Log data at three levels: platform, runtime, and infrastructure, stored in SLS (Simple Log Service) of Alibaba Cloud.
- Database containing the history of advisor services
- Repositories containing the code of advisor services

For log and database entries, only data before the detection time of the anomaly can be retrieved, preventing the analysis of future information and adhering to real-world usage.

The retrieval log database for the expert agent is a history subset of Flink Advisor with no overlap with the content used for labeling.

Table 1: Results of root cause prediction on Flink jobs.

Model	Semantic Metrics				LLM Metric
	METEOR	EmbScore	BLEURT	BARTScore	G-Correctness
ReAct	6.44	89.64	25.17	-6.20	3.06
RCAgent	15.15	91.47	31.57	-5.74	5.22
w/o LLM experts	9.60	90.33	27.77	-6.02	3.97
w/o JsonRegen	13.89	90.74	27.72	-5.84	4.19
w/o OBSK	12.37	90.97	29.67	-5.77	4.53
w/o Obs Head	12.27	91.30	30.47	-5.87	4.98
w/ Summary Head	14.64	91.34	32.11	-5.82	5.16
w/ SC (LLM)	$15.94_{\pm 0.44}$	$91.59_{\pm 0.06}$	$33.74_{\pm 0.44}$	$-5.48_{\pm 0.04}$	$5.38_{\pm 0.01}$
w/ TSC (LLM)	$\mathbf{16.49_{\pm 0.09}}$	$\mathbf{91.67_{\pm 0.03}}$	$\mathbf{34.43_{\pm 0.59}}$	$\mathbf{-5.40_{\pm 0.01}}$	$\mathbf{5.47_{\pm 0.06}}$

Table 2: Results of solution generation on Flink jobs.

Model	Semantic Metrics			LLM Metric
	METEOR	BLEURT	BARTScore	G-Helpfulness
ReAct	6.42	26.97	-4.90	3.41
RCAgent	12.94	34.68	-4.17	5.48
w/o LLM experts	8.46	30.46	-4.63	4.13
w/o JsonRegen	11.41	31.25	-4.40	4.58
w/o OBSK	10.34	32.13	-4.37	4.68
w/ SC (LLM)	$15.27_{\pm 0.19}$	$37.94_{\pm 0.03}$	$-4.00_{\pm 0.00}$	$5.55_{\pm 0.03}$
w/ TSC (LLM)	$\mathbf{16.45_{\pm 0.06}}$	$\mathbf{39.18_{\pm 0.13}}$	$\mathbf{-3.94_{\pm 0.08}}$	$\mathbf{5.69_{\pm 0.02}}$

4.3 Evaluation Metrics

Besides semantic metric scores including **METEOR** [3], **NUBIA** [18] (6-dim), **BLEURT** [41], and **BARTScore** [57] (F-Score, CNNDM), we use additional embedding Score (**EmbScore**), the cosine similarity from the default embedding model in our experiment.

We also follow the common practice of using stronger models to estimate model prediction [12, 38, 62]. We use greedy decoding **gpt-4-0613**, a frozen version of GPT4[33] for better reproducibility. We prompt the model to judge the accuracy and helpfulness of root cause and solution predictions, marked as **G-Correctness** and **G-Helpfulness**, respectively, and give a score within $0 \sim 10$.

5 RESULT

5.1 Effectiveness

We present the effectiveness of RCAgent on the offline dataset in Table 1, 2, and 3. RCAgent outperforms the original ReAct in all aspects of comprehensive RCA encompassing root cause, solution, and evidence prediction. The performance superiority is evident and consistent across all metrics, including +8.71 and +6.52 METEOR against ReAct in the root cause and solution prediction subtasks.

Employing TSC aggregation using LLM summarization, the overall performance of RCAgent gains further enhancements, especially on solution prediction, witnessing gains of +3.51 METEOR, +4.50 BLEURT, and +2.28% G-Helpfulness. This boost can be explained by the broader diversity of solution sampling.

Table 3: Semantic scores of evidence of methods.

Model	Semantic Metrics		
	METEOR	EmbScore	BARTScore
ReAct	11.82	90.03	-5.74
RCAgent	28.10	92.14	-4.62
w/o LLM experts	13.10	90.63	-5.63
w/o OBSK	17.79	91.12	-5.13
w/ Summary Head	18.09	91.67	-5.14
w/ SC (LLM)	$30.15_{\pm 0.83}$	$92.60_{\pm 0.06}$	$-4.41_{\pm 0.05}$
w/ TSC (LLM)	$\mathbf{30.84_{\pm 0.43}}$	$\mathbf{92.78_{\pm 0.02}}$	$\mathbf{-4.29_{\pm 0.02}}$

Table 4: Trajectory statistics of different settings.

Model	Pass Rate	Trajectory Length	Invalid Rate
ReAct	86.33	7.48	22.82
RCAgent	**99.38**	**6.78**	**7.93**
w/o LLM experts	92.55	6.93	16.24
w/o JsonRegen	85.71	7.91	18.75
w/o OBSK	96.89	7.21	18.34
w/ Sampling	70.19	10.66	44.80
w/ SQL tools	65.84	10.55	70.94

5.2 Ablation Study

To gauge the contribution of each component of RCAgent, we conduct an ablation study by removing enhancements introduced by RCAgent, including LLM expert agents, JsonRegen, and OBSK. The ablative result is shown in Table 1,2, and 3.

5.2.1 w/o LLM Expert Agents. We see a drastic drop in all metrics, such as +8.71 to +3.16 METEOR on root cause prediction, leaving only marginal improvement over ReAct. This shows the power of building analytical tools for the LLM, relieving the burden of the controller agent directly analyzing complex data.

5.2.2 w/o JsonRegen. The controller and expert agents generate more malformed output, and RCAgent loses a large proportion of its performance, primarily due to erroneous decisions.

5.2.3 w/o OBSK. The controller agent cannot use snapshots anymore and has to operate on truncated data. The absence of snapshots impacts the overall metrics, including -1.90 BLEURT and -0.69 G-Correctness on root cause prediction, though not as dramatically as excluding the LLM experts. This indicates that the controller can still put analysis on the log with the expert, while a large part of the environmental observation is lost.

Additionally, we test altering the observation head shown to the controller agent. When the observation head is removed and only a snapshot is shown (w/o Obs Head), the performance slightly downgrades for -1.10 BLEURT, implying the major benefit of the snapshot mechanism. When we prompt the LLM to summarize the original observation as the alternative head, minimal performance degradation is observed including -0.06 G-Correctness. The difference except for BLEURT shows that the truncated observation helps more than the summarized content for the controller agent.

5.3 Stability

We study the action trajectories of different settings in Table 4. With all enhancements, the RCAgent achieves a 99.38% Pass Rate within 15 steps and a 7.93% Invalid Rate, meaning nearly perfect stability and a significant edge over ReAct. With such a minuscule chance of generating problematic actions, RCAgent consistently delivers more accurate and helpful RCA results with shorter trajectories.

When the LLM expert or OBSK is removed, the controller agent maintains a Pass Rate exceeding 90% while both absences lead to an error-prone exploration, diverting some of its actions toward redundant tool invocations. The removal of JsonRegen significantly damages the stability due to surging invalid data interchange.

Moreover, when the default decoding strategy for the controller agent is changed to nucleus sampling (w/ Sampling), the stability collapses to 70.19% Pass Rate and 44.80% Invalid Rate with tons of erroneous actions and hallucinations. Such results highlight the
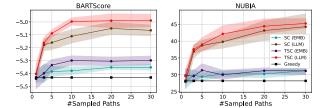

Figure 4: Performance of Self-Consistency at different scales and methods. The solid line is the mean score, and the shade represents the standard deviation. The score is calculated on the concatenated solution and root cause.

vitality of optimal decoding during initial steps and lead to the design of our mid-way TSC rather than a pure full-process SC.

We also test using the SQL and SLS query execution tools as the information-gathering tool set. A thorough description of all relevant databases is present in the prompt. The substitution drastically downgrades the Pass Rate by 33.54%, and the gap mainly results from a 70.95% Invalid Rate. This clearly shows the necessity of providing semantically minimalist tools for locally hosted LLMs.

5.4 Self-Consistency

We study combinations of SC methods and sample counts each for 10 different runs. The results are detailed in Figure 4. We evaluate concatenated predicted root causes and solutions for better numerical readability.

The statistics show that every SC method consistently augments the performance of RCAgent in terms of BARTScore and NUBIA. This enhancement seems to plateau when the number of samples reaches 20. Among different methods, TSC brings superiority due to its diverse action sampling. In all metrics, LLM aggregation outperforms embedding voting, and this gap broadens with an increasing number of samples, illustrating LLM aggregation's ability to offer more comprehensive results as the candidate pool grows.

6 DEPLOYMENT

RCAgent has been integrated as a feedback mechanism into the internal operations management platform in our company, aiding in the detection of potential platform-side errors and bugs. Specifically, RCAgent analyzes all Out-of-Domain (OoD) jobs that existing automatic SRE tools cannot properly handle. Then, jobs labeled as platform responsibility by RCAgent, along with the RCA results, are handed to the SRE team of Apache Flink to aid human diagnosis. This procedure improves the efficiency of spotting anomalies that require the DevOps team's attention and the workflow of the after-sale service team.

6.1 Performance on OoD Jobs

We evaluate the performance of RCAgent on OoD jobs in the first two weeks of the system's deployment. These OoD job samples are semantically clustered, all of which are beyond the capability of current rules and are labeled by experienced SREs for system evaluation. The SRE team is instructed to assign a 0-5 **H-Helpfulness** score with a set of Likert Scales, spanning from misleading to exceptionally helpful diagnosis.

Table 5: Evaluations on the online OoD anomalies. Bold denotes the best results.

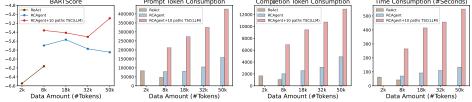| Model	Root Cause					Responsibility	Human
	METEOR	NUBIA	BLEURT	BARTScore	G-Correctness	Precision	H-Helpfulness
XGBoost	-	-	-	-	-	77.65	-
Finetune T5	4.18	8.20	19.32	-6.61	2.62	77.85	-
LLM Summary	7.88	14.57	25.40	-6.00	3.58	77.21	-
ReAct	5.21	10.38	20.33	-6.25	2.24	73.53	$1.36_{\pm0.03}$
RCAgent	13.77	19.48	31.52	-5.59	3.82	80.74	$2.47_{\pm0.17}$
w/ TSC (LLM)	$\mathbf{15.72_{\pm0.61}}$	$\mathbf{26.79_{\pm2.54}}$	$\mathbf{35.72_{\pm0.58}}$	$\mathbf{-5.29_{\pm0.03}}$	$\mathbf{4.36_{\pm0.01}}$	$\mathbf{82.06_{\pm0.42}}$	$\mathbf{2.92_{\pm0.21}}$

Figure 5: Performance and resource consumption at different data scales.

The result is shown in 5. Consistent with all LLM and semantic metrics, human evaluators rate our method with an H-Helpfulness of 2.92, indicating moderate support for RCA. A Tukey's HSD test shows that our method with ($t = 5.84, p = 0.001$) and without ($t = 4.08, p = 0.001$) TSC substantially outperforms ReAct. Furthermore, equipped with TSC (LLM), RCAgent demonstrates a precision of 82.06% in determining the responsibility.

We also examine non-agent RCA solutions for comparison. We use all possible types of relevant data of a job, truncated if exceeding the model length constraint, to train XGBoost using document embeddings, fine-tune T5, and feed to LLM with a summarization instruction. Despite all the instabilities an autonomous agent solution would incur, which might explain ReAct being worse than traditional methods, our design outperforms non-agent approaches, underscoring the capability of LLM's autonomous decision-making.

6.2 Computational Scalability

We show the statistics of LLM agent performance and resource consumption against varying amounts of data included for different jobs during our deployment in Figure 5. The data amount is calculated by tokenizing all data either presented to the controller agent or queried by the OBSK mechanism. While utilizing more data and consuming more resources, RCAgent and its TSC enhancement clearly show better performance compared to ReAct. Moreover, our methods' resource consumption scales nearly linearly as the data amount grows (Pearson Correlation p-value < 0.05 for any type of consumption), without substantial performance degradation ($p = 1.0$ based on Kruskal-Wallis H-test). This breakdown of performance and consumption shows RCAgent's effectiveness in handling a large amount of data.

7 RELATED WORK

7.1 LLM as Autonomous Agents

As LLMs demonstrate impressive capabilities [32, 33, 52], some literatures [36, 44, 54] leverage these models to construct LLM-based agents. A popular paradigm is autonomous agents [36, 43, 48, 51, 56], in which LLM agents explore self-directedly without human intervention and step-by-step instructions. They perform tasks

with an action trajectory, adapting their intentions and outputs according to the environment [25]. While autonomous agents have been implemented and tested on a variety of toy tasks [26, 63], our RCAgent is the first work to introduce autonomous LLM agents to realistic cloud RCA tasks.

7.2 LLM Augmented by Tools

Recent studies [37, 38, 40, 47] have showcased the proficiency of LLMs to invoke tools and make decisions across a wide range of tasks. The tools, in the form of simple functions or external APIs, extend LLM's knowledge and capability evidently [6, 23, 39]. While stronger LLM can easily grasp tools and accomplish tasks, others can be taught by generated and filtered trajectories [38, 40]. In this paper, we aim to augment agents with a comprehensive toolset, extending the tool-using paradigm to the real-world cloud RCA.

7.3 Cloud RCA with LLMs

RCA in large cloud services is a prominent subject of study within software engineering communities [7, 13]. A large part of RCA is coupled with NLP due to subtasks like log analysis [14, 21, 28]. As LLMs advance, they are leveraged for cloud RCA tasks with fine-tuning [2, 17] or in-context learning [10, 16]. However, these models are not aware of the workflow of cloud RCA, leaving them simply analytical tools. We thus investigate tool-augmented LLM as agents for the ever-changing environment of cloud RCA.

8 CONCLUSION

In this work, we introduce RCAgent, a tool-augmented LLM autonomous agent tailored for cloud root cause analysis. RCAgent ensures secure industrial usage of LLM agents in cloud systems by utilizing internally deployed models instead of powerful external ones like ChatGPT. Our methodology encompasses a spectrum of enhancements including unique Self-Consistency for action trajectories, a comprehensive prompting framework, expert agents, and stabilization methods. Furthermore, RCAgent's efficacy is demonstrated by its practical application in the Real-time Compute Platform for Apache Flink of Alibaba. In general, this work pioneers the real-world application of LLM agents in the cloud RCA field.

REFERENCES

[1] Pooja Aggarwal, Ajay Gupta, Prateeti Mohapatra, Seema Nagar, Atri Mandal, Qing Wang, and Amit Paradkar. 2020. Localization of operational faults in cloud applications by mining causal dependencies in logs using golden signals. In *International Conference on Service-Oriented Computing*. 137–149.

[2] Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas Zimmermann, Xuchao Zhang, and Saravan Rajmohan. 2023. Recommending Root-Cause and Mitigation Steps for Cloud Incidents Using Large Language Models. In *International Conference on Software Engineering*. 1737–1749.

[3] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. 65–72.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*. 1877–1901.

[6] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712* (2023).

[7] Haicheng Chen, Wensheng Dou, Yanyan Jiang, and Feng Qin. 2019. Understanding exception-related bugs in large-scale cloud systems. In *International Conference on Automated Software Engineering*. 339–351.

[8] Junjie Chen, Xiaoting He, Qingwei Lin, Yong Xu, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. An empirical investigation of incident triage for online service systems. In *International Conference on Software Engineering: Software Engineering in Practice*. 111–120.

[9] Pengfei Chen, Yong Qi, and Di Hou. 2016. CauseInfer: Automated end-to-end performance diagnosis with hierarchical causality graph in cloud environment. *IEEE Transactions on Services Computing* 12, 2 (2016), 214–230.

[10] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, et al. 2023. Empowering Practical Root Cause Analysis by Large Language Models for Cloud Incidents. *arXiv preprint arXiv:2305.15778* (2023).

[11] Qian Cheng, Doyen Sahoo, Amrita Saha, Wenzhuo Yang, Chenghao Liu, Gerald Woo, Manpreet Singh, Silvio Saverese, and Steven CH Hoi. 2023. AI for IT Operations (AIOps) on Cloud Platforms: Reviews, Opportunities and Challenges. *arXiv preprint arXiv:2304.04661* (2023).

[12] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. 2023. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010* (2023).

[13] Supriyo Ghosh, Manish Shetty, Chetan Bansal, and Suman Nath. 2022. How to fight production incidents? an empirical study on a large-scale cloud service. In *Symposium on Cloud Computing*. 126–141.

[14] Haixuan Guo, Shuhan Yuan, and Xintao Wu. 2021. LogBERT: Log anomaly detection via bert. In *International Joint Conference on Neural Networks*. 1–8.

[15] Anders Hejlsberg, Steve Lucco, Daniel Rosenwasser, Pierce Boggan, Umesh Madan, Mike Hopcroft, , and Gayathri Chandrasekaran. 2023. Introducing Type-Chat. https://microsoft.github.io/TypeChat/blog/introducing-typechat/

[16] Yuxuan Jiang, Chaoyun Zhang, Shilin He, Zhihao Yang, Minghua Ma, Si Qin, Yu Kang, Yingnong Dang, Saravan Rajmohan, Qingwei Lin, et al. 2023. Xpert: Empowering Incident Management with Query Recommendations via Large Language Models. *arXiv preprint arXiv:2312.11988* (2023).

[17] Pengxiang Jin, Shenglin Zhang, Minghua Ma, Haozhe Li, Yu Kang, Liqun Li, Yudong Liu, Bo Qiao, Chaoyun Zhang, Pu Zhao, et al. 2023. Assess and Summarize: Improve Outage Understanding with Large Language Models. *arXiv preprint arXiv:2305.18084* (2023).

[18] Hassan Kane, Muhammed Yusuf Kocyigit, Ali Abdalla, Pelkins Ajanoh, and Mohamed Coulibali. 2020. NUBIA: NeUral based interchangeability assessor for text generation. In *ACL Workshop on Evaluating NLG Evaluation*. 28–37.

[19] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*. 22199–22213.

[20] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Symposium on Operating Systems Principles*. 611–626.

[21] Van-Hoang Le and Hongyu Zhang. 2022. Log-based anomaly detection with deep learning: How far are we?. In *International Conference on Software Engineering*. 1356–1367.

[22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*. 9459–9474.

[23] Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-Bank: A benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244* (2023).

[24] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards General Text Embeddings with Multi-stage Contrastive Learning. *arXiv preprint arXiv:2308.03281* (2023).

[25] Ruibo Liu, Ruixin Yang, Chenyan Jia, Ge Zhang, Denny Zhou, Andrew M Dai, Diyi Yang, and Soroush Vosoughi. 2023. Training Socially Aligned Language Models in Simulated Human Society. *arXiv preprint arXiv:2305.16960* (2023).

[26] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688* (2023).

[27] Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, et al. 2023. BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents. *arXiv preprint arXiv:2308.05960* (2023).

[28] Steven Locke, Heng Li, Tse-Hsun Peter Chen, Weiyi Shang, and Wei Liu. 2021. LogAssist: Assisting log analysis through log summarization. *IEEE Transactions on Software Engineering* 48, 9 (2021), 3227–3241.

[29] Minghua Ma, Yudong Liu, Yuang Tong, Haozhe Li, Pu Zhao, Yong Xu, Hongyu Zhang, Shilin He, Lu Wang, Yingnong Dang, et al. 2022. An empirical investigation of missing data handling in cloud node failure prediction. In *ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1453–1464.

[30] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. In *European Chapter of the Association for Computational Linguistics*. 2006–2029.

[31] Hiep Nguyen, Zhiming Shen, Yongmin Tan, and Xiaohui Gu. 2013. Fchain: Toward black-box online fault localization for cloud systems. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, 21–30.

[32] OpenAI. 2022. OpenAI: Introducing ChatGPT. https://openai.com/blog/chatgpt

[33] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774

[34] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*. 27730–27744.

[35] Arka Pal, Deep Karkhanis, Manley Roberts, Samuel Dooley, Arvind Sundararajan, and Siddartha Naidu. 2023. Giraffe: Adventures in Expanding Context Lengths in LLMs. *arXiv preprint arXiv:2308.10882* (2023).

[36] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442* (2023).

[37] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. 2023. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354* (2023).

[38] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789* (2023).

[39] Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Xingyu Zeng, and Rui Zhao. 2023. TPTU: Task planning and tool usage of large language model-based AI agents. *arXiv preprint arXiv:2308.03427* (2023).

[40] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761* (2023).

[41] Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Association for Computational Linguistics*. 7881–7892.

[42] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*.

[43] Significant-Gravitas. 2023. AutoGPT: the heart of the open-source agent ecosystem. https://github.com/Significant-Gravitas/Auto-GPT. GitHub repository.

[44] Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. 2023. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427* (2023).

[45] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971

[46] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. LLaMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[47] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. 2023. Chatgpt for robotics: Design principles and model abilities. *Technical Report MSR-TR-2023-8, Microsoft* 2 (2023), 20.

[48] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432* (2023).

[49] Lingzhi Wang, Nengwen Zhao, Junjie Chen, Pinnong Li, Wenchi Zhang, and Kaixin Sui. 2020. Root-cause metric location for microservice systems via log anomaly detection. In *International Conference on Web Services*. 142–150.

[50] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *International Conference on Learning Representations*. 1–24.

[51] Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, et al. 2023. Interactive natural language processing. *arXiv preprint arXiv:2305.13246* (2023).

[52] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research* (2022), 1–30.

[53] Brandon T Willard and Rémi Louf. 2023. Efficient Guided Generation for Large Language Models. *arXiv e-prints* (2023), arXiv–2307.

[54] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864* (2023).

[55] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*. 1–33.

[56] yoheinakajima. 2023. BabyAgi. https://github.com/yoheinakajima/babyagi. GitHub repository.

[57] Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*, Vol. 34. 27263–27277.

[58] Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. 2022. TFAD: A decomposition time series anomaly detection architecture with time-frequency analysis. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2497–2507.

[59] Chaoli Zhang, Zhiqiang Zhou, Yingying Zhang, Linxiao Yang, Kai He, Qingsong Wen, and Liang Sun. 2022. NetRCA: an effective network fault cause localization algorithm. In *IEEE International Conference on Acoustics, Speech and Signal Processing*. 9316–9320.

[60] Dylan Zhang, Xuchao Zhang, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. 2023. PACE: Prompting and Augmentation for Calibrated Confidence Estimation with GPT-4 in Cloud Incident Root Cause Analysis. *arXiv preprint arXiv:2309.05833* (2023).

[61] Yingying Zhang, Zhengxiong Guan, Huajie Qian, Leili Xu, Hengbo Liu, Qingsong Wen, Liang Sun, Junwei Jiang, Lunting Fan, and Min Ke. 2021. CloudRCA: A root cause analysis framework for cloud computing platforms. In *ACM International Conference on Information & Knowledge Management*. 4373–4382.

[62] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685* (2023).

[63] Xuanhe Zhou, Guoliang Li, and Zhiyuan Liu. 2023. Llm as dba. *arXiv preprint arXiv:2308.05481* (2023).