Leveraging LLMs for Structured Information Extraction and Analysis from Cloud Incident Reports (Work In Progress Paper)

Xiaoyu Chu
x.chu@vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Shashikant Ilager
s.s.ilager@uva.nl
Universiteit van Amsterdam
The Netherlands

Yizhen Zang
y.zang-3@student.tudelft.nl
Delft University of Technology
The Netherlands

Sacheendra Talluri
s.talluri@vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Alexandru Iosup
a.iosup@vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Abstract

Incident management is essential to maintain the reliability and availability of cloud computing services. Cloud vendors typically disclose incident reports to the public, summarizing the failures and recovery process to help minimize their impact. However, such reports are often lengthy and unstructured, making them difficult to understand, analyze, and use for long-term dependability improvements. The emergence of LLMs offers new opportunities to address this challenge, but how to achieve this is currently understudied. In this paper, we explore the use of cutting-edge LLMs to extract key information from unstructured cloud incident reports. First, we collect more than 3,000 incident reports from 3 leading cloud service providers (AWS, AZURE, and GCP), and manually annotate these collected samples. Then, we design and compare 6 prompt strategies to extract and classify different types of information. We consider 6 LLM models, including 3 lightweight and 3 state-of-the-art (SotA), and evaluate model accuracy, latency, and token cost across datasets, models, prompts, and extracted fields. Our study has uncovered the following key findings: (1) LLMs achieve high metadata extraction accuracy, 75%–95% depending on the dataset. (2) Few-shot prompting generally improves accuracy for meta-data fields except for classification, and has better (lower) latency due to shorter output-tokens but requires 1.5–2× more input-tokens. (3) Lightweight models (e.g., Gemini 2.0, GPT 3.5) offer favorable trade-offs in accuracy, cost, and latency; SotA models yield higher accuracy at significantly greater cost and latency. Our study provides tools, methodologies, and insights for leveraging LLMs to accurately and efficiently extract incident-report information. The FAIR data and code are publicly available at https://github.com/atlarge-research/llm-cloud-incident-extraction.

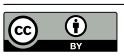
CCS Concepts

• **Computer systems organization → Reliability**.

Keywords

Incident Report, Cloud Computing, Information Extraction, Large Language Models (LLMs), Root Cause Analysis, Artificial Intelligence for IT Operations (AIOps)

ACM Reference Format:
Xiaoyu Chu, Shashikant Ilager, Yizhen Zang, Sacheendra Talluri, and Alexandru Iosup. 2026. Leveraging LLMs for Structured Information Extraction and Analysis from Cloud Incident Reports (Work In Progress Paper). In *Companion of the 17th ACM/SPEC International Conference on Performance Engineering (ICPE Companion '26), May 04–08, 2026, Florence, Italy.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3777911.3801103



This work is licensed under a Creative Commons Attribution 4.0 International License.
ICPE Companion '26, Florence, Italy
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2326-1/2026/05
https://doi.org/10.1145/3777911.3801103

1 Introduction

Incident management is an important and complex process in cloud computing operations [1, 18, 21]. A cloud incident could occur for multiple reasons, including hardware or software failures, deployment and runtime errors, network disruptions, or security breaches [12, 22]. Such incidents can disrupt user services, affecting the reliability of the system. To minimize the impact, once an incident occurs, cloud operators provide public incident reports [4, 10, 23] to inform users about the incident management process and root causes. However, these reports are very complex, lengthy, and too unstructured to understand or use for long-term analysis and mitigation. As a result, while individual reports are disclosed for each incident, there is a lack of a public dataset and long-term analysis of these incidents. Currently, LLMs have become powerful tools and offer new opportunities to address such problem. LLMs have already been effectively used in performance engineering, such as log parsing [3, 16], root cause analysis [1, 7, 11, 21, 28]. Therefore, we propose to leverage advanced LLMs to extract and analyze key information from public cloud incident reports.

We design a workflow to adapt LLMs for report data extraction. Our work addresses three main challenges in this workflow process: First, **there is no publicly available labeled dataset of cloud incident reports, nor open source tools for automatically extracting and analyzing their key information.** Previous studies on cloud incident and outage analysis have relied on manual or rule-based data extraction [12, 25], which is highly time-consuming and labor-intensive due to the length and complexity of the reports. For example, in our collected datasets from Azure and GCP, the average report length exceeds 500 words (see Table 1). Second, **there is a lack of methodology and systematic evaluation for assessing**

Table 1: Summary of cloud incident reports.

ID	Name	Period	# Rows	# Labeled	Avg. Words
1	AWS	2016 - 2022	774	150 (19%)	151
2	AZURE	2019 - 2024	127	95 (75%)	575
3	GCP	2016 - 2021	2,186	215 (10%)	533
TOTAL	3 sources	2016 - 2024	3,087	460 (15%)	–

Table 2: Summary of extracted fields. Legend: ● indicates fully extracted from the report, ● indicates inferred by LLMs. EM = Exact Match, TK = Token-Level F1, BS = BERTScore.

ID	Extracted Fields	Type	Datasets			Eval. Metric		
			AWS	AZURE	GCP	EM	TK	BS
1	Service Name	entity	●	●	●	✓		
2	Location	entity	●	●	-	✓		
3	Start Time	entity	●	●	●	✓		
4	End Time	entity	●	●	●	✓		
5	Timezone	entity	●	●	●	✓		
6	Service Categ.	class	●	●	●	✓		
7	Root Cause Categ.	class	-	●	-	✓		
8	User Symp. Categ.	multi-class	●	●	●		✓	
9	User Symptom	text	●	●	●			✓
10	Root Cause	text	-	●	-			✓

the performance and cost of applying LLMs to incident report data extraction. Although LLMs are widely used for information extraction [24, 30], their design and application for incident report extraction remain unclear. In particular, it is not well understood how to structure data, how to design prompts, and how to adapt LLMs for cloud incident analysis. Third, **there are limited studies to perform long-term statistical characterization on incident reports.** Previous studies have conducted incident report analysis on cloud services [12, 25], but these analyses are typically based on labeled datasets and cover only limited fields, making them difficult to generalize to long-term datasets or diverse incident reports.

Addressing these challenges, our key contributions are:

(1) **Dataset and toolbox construction (Section 2):** This work is the first to provide open-sourced datasets and toolbox for systematically exploring and evaluating LLMs for structured information extraction and analysis of cloud incident reports. We collect over 3,000 cloud incident reports with annotated subsets for report-data evaluation. We also develop a toolbox for leveraging LLMs for report data extraction, evaluation, and analysis.

(2) **LLMs adaption and evaluation (Section 3 and 4):** This work demonstrates the usefulness of SotA LLMs such as GPT-4o for incident information extraction and analysis. We systematically evaluate 6 LLMs models of different size with diverse prompt strategies, we compare their accuracy and cost with different evaluation metrics. Our results achieve 75%–95% accuracy on metadata extraction, based on the dataset. We also provide recommendations for selecting prompts and LLMs based on their latency and cost.

(3) **Open science:** We follow the FAIR principles of open science and release the datasets[1] and software[2] to enable reproducibility and further research.

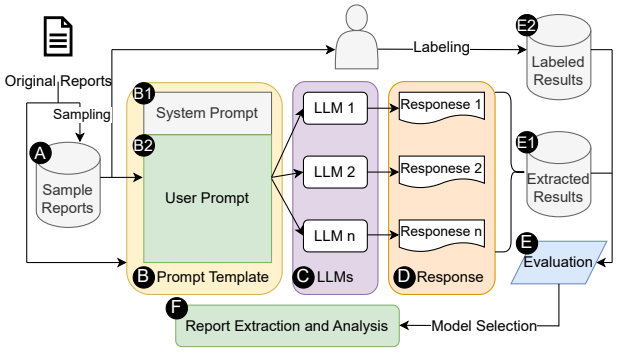


Figure 1: Overview of the workflow (structured method) for LLM-based data extraction from cloud incident reports.

2 Data Collection and Preparation

2.1 Dataset Collection

Incident reports from cloud service providers are frequently published online via their respective web platforms [4, 10, 23]. However, these reports vary in structure, and the extent of publicly available information differs across providers. To compile a diverse set of incident report data, we sourced reports from 3 major cloud service providers: AWS, AZURE, and GCP, covering the period from 2016 to 2024. In total, we collected approximately 3,000 incident reports. Each report consists of important information such as the name of the service, the location, the user symptom, and the root cause.

To *collect* the data, we scraped and archived the incident report web pages from the 3 cloud providers. We then *processed* the data using the methodology described in Section 2.3. Additionally, we *manually labeled* approximately 15% of the reports to serve as ground truth for evaluation. Although the proportion of labeled data varies by provider, the resulting sample size yields statistically meaningful insights; further details are provided in Section 2.3. Table 1 presents a summary of this study's datasets.

2.2 Data Processing

In this stage, we process the raw dataset to prepare it for prompt construction. Table 2 presents the target fields identified for extraction. These fields are categorized into entities, classes, and free-text segments, as each type requires distinct evaluation strategies, which are discussed in detail in Section 3.4. Accordingly, our data processing includes the following steps: (1) *Raw datasets*: We extract information from each externally scraped file in HTML format and save it as a *Parquet* file for each operator. (2) *Cleaned datasets*: Because the structure of each vendor's report is different, we processed them separately to filter out invalid metrics (e.g., `None` values and `duplicates`). After this, we obtained cleaned datasets with an identical structure, which can be easily combined for unified analysis. (3) *Sample datasets*: Because of scale, labeling the entire dataset is impractical, so we applied *K-means* clustering to select representative sample datasets for data annotation. (4) *Labeled datasets*: We manually annotated the fields listed in Table 2 from the sample datasets. Following a strict data annotation process (see Section 2.3), we obtained the labeled datasets.

[1] Zenodo: https://zenodo.org/records/14010282
[2] GitHub: https://github.com/atlarge-research/llm-cloud-incident-extraction

Table 3: Model configurations. Model Type: L–lightweight model; S–state of the art. Models marked with (*) do not provide time fingerprints, but the dates when these models were run are shown in Section 4. Price: T=Token.

Model Alias	Model Type / Name	Price [$/$10^6$T] Input	Output
GPT 4o	S / gpt-4o*	2.50	10.00
GPT 3.5	L / gpt-3.5-turbo*	0.50	1.50
Claude Sonnet 4	S / claude-sonnet-4-20250514	3.00	15.00
Claude 3.5	L / claude-3-5-haiku-20241022	0.80	4.00
Gemini 2.5	S / gemini-2.5-pro*	1.25	10.00
Gemini 2.0	L / gemini-2.0-flash*	0.10	0.40

Table 4: Six prompt strategies with different components. Acronyms: ZS=Zero-Shot; FS=Few-shot.

Label	Components
Full-ZS	Task + CoT + Category + Format
Full-FS	Task + CoT + Category + Examples + Format
Basic-ZS	Task + Format
Basic-FS	Task + Examples + Format
CoT-ZS	Task + CoT + Format
Categ-ZS	Task + Category + Format

2.3 Data Annotation

Data annotation is a crucial step, as ground truth data is necessary to compare against the LLMs' extracted results and to assess their accuracy. Since no annotated cloud incident datasets are publicly available for evaluation, we created our own annotated datasets. To ensure annotation accuracy and consistency, we selected 3 researchers with a background in computer systems as annotators. The annotation process is as follows: First, Annotators 1 and 2 were asked to independently label the sample datasets following the provided instructions. After completing the labeling, they compared their results to check for alignment. In cases of disagreement, they had discussions to reach a consensus. If no agreement was possible, Annotator 3 intervened to make the final decision.

3 Setup of LLM Experiments

3.1 Use of LLMs for Report Extraction

Figure 1 provides a methodological overview of this study. Following data collection and annotation, the process moves to using LLMs for report extraction on sample datasets (A). First, we design a prompt template (B) that includes general context about cloud incidents and step-by-step instructions for data extraction, covering both zero-shot and few-shot settings. Figure 2 depicts an example full template, which contains five components. The template varies slightly depending on the cloud operator, as their reports differ in structure and available information.

For each report, we insert its content into the template as part of the user prompt (B2). A system prompt (B1) is set for LLMs to assume the role of a system operator to help perform data extraction from cloud incident reports. Combined with system prompt, the complete prompt is then sent to a set of candidate LLMs (C), and the model responses (D) are collected in JSON format. We evaluate (E) the accuracy of the extracted results (E1) by comparing them with the labeled datasets (E2), which serve as the ground

Task Description (Task)

Analyze the incident report step by step to extract structured information.

Chain-of-Thought Instruction (CoT)

Follow the reasoning steps:
1. Identify the service name and service location.
2. From `{service_category_lst}`, select one most relevant service category.
3. Extract the relevant sentence(s) that describe user symptoms. Then, from `{user_symp_lst}`, select one or more categories that best match the extracted symptoms.
4. Identify the start time, end time, and timezone. Format times as "HH:MM:SS" (24-hour).

Categorization Instruction (Category)

The definition for user symptom category are:
`{user_symp_instruction}`.

Answering Format (Format)

Finally, return the extracted information in a JSON object with the keys:
service_name, *location*, *service_category*, *start_time*, *end_time*, *timezone*, *user_symptom*, *user_symptom_category*.

Few-shot Examples (Examples)

Here are a few examples of report content (labeled **Q**) and extracted information (labeled **A**).
Q: title: **Amazon CloudWatch** (**Ireland**)
status: [RESOLVED] Delayed CloudWatch Metrics
description: (...) 2:56 PM PST Between **10:26 AM** and **02:40 PM** PST, **we experienced increased delays for CloudWatch log event processing for metric filter extraction and log subscriptions in the EU-WEST-1 Region**.
(...)
A: "service_name": "**Amazon CloudWatch**", "location": "**Ireland**", "service_category": "**management**", "start_time": "**10:26:00**", "end_time": "**14:40:00**", "timezone": "**PST**", "user_symptom_category": "**DELAY**", "user_symptom": "**we experienced increased delays for CloudWatch log event processing for metric filter extraction and log subscriptions in the EU-WEST-1 Region.**" *(Continued...)*

Figure 2: Full-FS user prompt template for extracting data from AWS reports, with five components: *Task*, *CoT*, *Category*, *Format*, and *Examples*. Colors: blue=extracted, red=classified.

Table 5: Extraction accuracy ("Acc.") of different prompt strategies for GPT-3.5 on the AWS dataset. For accuracy metrics EM, TK, BS, see Table 2. For each field, boldface mygreen text indicates best-performers, and boldface myred text indicates worst-performers.

Field / Prompt Strategy	Acc.	Full-ZS	Full-FS	Basic-ZS	Basic-FS	CoT-ZS	Categ.-ZS
Service Name	EM	86.00	**100.00**	**52.00**	**100.00**	83.33	60.67
Location	EM	48.00	**96.67**	38.00	83.33	57.33	**44.67**
Start Time	EM	86.00	**91.33**	**70.00**	89.33	83.33	72.00
End Time	EM	83.33	**86.00**	**64.00**	**86.00**	78.67	71.33
Timezone	EM	**98.67**	**98.67**	**98.67**	**98.67**	**98.67**	**98.67**
Service Categ.	EM	77.33	71.33	64.67	76.0	**80.00**	**61.33**
User Symptom Categ.	TK	88.50	**88.94**	9.76	50.19	**9.33**	90.00
User Symptom	BS	84.33	92.90	84.79	**93.79**	83.09	**81.63**
Overall	**Average**	71.08	**79.23**	**49.74**	73.60	61.44	62.44

truth. We also analyze and assess the latency and cost of candidate LLMs. Finally, we select the best-performing LLMs for each operator and conduct incident report extraction and analysis (Ⓕ).

3.2 Environmental Setup and LLM Models

Table 3 presents the models selected in our experiments for comparison. We selected models from popular LLM providers (OpenAI, Claude, and Gemini). To estimate cost per model, Table 3 also lists the official-website pricing of input and output tokens.

3.3 Prompt Engineering

Each LLM extracts information from input data guided by a prompt, which significantly affects the accuracy and cost of extraction results. To design accurate and effective prompts, we use the *Chain of Thought (CoT)* and *In-Context Learning (ICL)* prompting techniques, which have increasingly become popular in LLM downstream applications [15]. CoT prompts guide LLMs to reason step by step, yielding more accurate answers [14, 29]. In contast, ICL provides a few demonstration examples directly within the input prompt, enabling LLMs to perform diverse downstream NLP tasks [6, 19].

Prompt components: To investigate how different prompt components influence report data extraction, we define five components inspired by prior work [11]: Task description (Task), Chain-of-Thought instruction (CoT), Categorization instruction (Category), In-context examples (Examples), and Answering format (Format). The *Task* description instructs the model to assume the role of a system operator and extract information from cloud incident reports. (*CoT*) This instruction provides step-by-step guidance to facilitate reasoning about the information. For in-context *Examples*, two samples from the dataset are provided to illustrate the expected answers. Finally, the answering *Format* specifies the JSON structure in which the model should return its output.

Six prompt strategies: By combining the above introduced prompt components in various ways, we construct six prompting strategies, summarized in Table 4. Among them, Task and Format serve as the foundational elements present in all prompts, while the remaining components are selectively incorporated to evaluate their individual and combined effects.

An exemplary prompt strategy: Figure 2 shows our full prompt with few-shot examples (Full-FS) for extracting report information. The full prompt incorporates all five components examined in this study: Task, CoT, Category, Examples, and Format. Whereas, the other prompt templates are comparatively less component-rich.

3.4 Evaluation Metrics

Accuracy: LLMs extract information for all the fields listed in Table 2, with accuracy different per field. We first evaluate the extraction accuracy of *entity* and *class* fields using *Exact Match (EM)*, which measures whether the extracted output matches the labeled ground truth exactly [20]. For the multi-class field *user symptom category*, we use the evaluation measure for classification tasks, which is *Token-level F1 (TK)*. It computes the harmonic mean of precision and recall based on overlapping tokens [17, 24]. For the textual fields *user symptoms* and *root causes*, which are sentence-level and require semantic accuracy, therefore, we use *BERTScore (BS)* [1, 32] to measure semantic similarity using pre-trained BERT models.

4 Performance Analysis of LLMs

4.1 Comparison of Prompt Strategies

We present an experimental comparison of the six LLM-prompting strategies introduced in Section 3.3. Each strategy is evaluated on the AWS dataset (Table 1) using GPT-3.5 as the underlying model. Table 5 presents the accuracy achieved by each prompting strategy, with per-strategy averages computed using the arithmetic mean reported in the bottom-row.

Finding 1: *As expected, the most component-rich strategy, Full-FS, achieves the highest overall accuracy (79.23%), and is the best-(single-best-) performer among the six strategies for 6 of the 8 fields. In-context Examples are the most effective single-prompt component: Adding it to Basic-ZS, results in Basic-FS's accuracy of 73.60%; accuracy increases by nearly 24% over Basic-ZS and becomes comparable with the more component-rich Full-ZS. Similarly, adding Chain-of-Thought (CoT-ZS) and categorization instructions (Categ-ZS) to Basic-ZS leads to 11.7% and 12.7% better accuracy, respectively.*

4.2 Extraction Accuracy

Table 6 reports the general accuracy of the exact match between different models and prompt methods.

Table 6: Exact match accuracy on selected (meta-data) fields [%]. For each row, mygreen = best (highest), myred = worst (lowest).

AWS	GPT 3.5		GPT 4o		Claude 3.5		Claude 4		Gemini 2.0		Gemini 2.5	
	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few
Service Name	84.67	**100.00**	**100.00**	**100.00**	**76.67**	**100.00**	88.00	**100.00**	79.33	**100.00**	98.67	**100.00**
Location	**48.00**	96.67	83.33	**98.67**	48.67	96.67	55.33	97.33	76.67	96.67	78.00	96.00
Start Time	88.00	91.33	95.33	**96.00**	**84.67**	94.67	95.33	95.33	88.67	94.67	95.33	95.33
End Time	83.33	86.00	87.33	**88.00**	**66.00**	86.00	85.33	86.67	83.33	**88.67**	86.67	**88.00**
Timezone	98.67	98.67	98.00	98.67	**97.33**	98.00	98.67	98.67	**97.33**	98.67	98.67	98.67
Service Categ.	73.33	73.33	90.00	**68.00**	85.33	87.33	88.00	89.33	84.00	86.00	**90.67**	**90.67**
Average	79.33	91.00	92.33	91.56	**76.44**	93.78	85.11	94.56	84.89	94.11	91.34	**94.78**

AZURE	GPT 3.5		GPT 4o		Claude 3.5		Claude 4		Gemini 2.0		Gemini 2.5	
	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few
Service Name	56.84	63.16	55.79	61.05	66.32	**67.37**	61.05	55.79	56.84	65.26	57.89	**52.63**
Location	63.16	67.37	65.26	64.21	64.21	69.47	66.32	**70.53**	**52.63**	67.37	60.00	65.26
Start Time	97.89	**100.00**	98.95	94.74	**67.37**	96.84	98.95	**100.00**	68.42	**100.00**	98.95	98.95
End Time	92.63	**96.84**	94.74	93.68	65.26	93.68	93.68	**96.84**	**64.21**	93.68	95.79	**96.84**
Timezone	97.89	97.89	97.89	95.79	97.89	97.89	97.89	**98.95**	**86.32**	**98.95**	**98.95**	**98.95**
Service Categ.	64.21	**58.95**	**67.37**	**67.37**	63.16	63.16	64.21	66.32	60.00	65.26	62.11	62.11
Root Cause Categ.	**61.05**	63.16	67.37	64.21	65.26	71.58	63.16	66.32	**61.05**	**73.68**	67.37	70.53
Average	76.24	78.20	78.20	77.29	69.92	80.00	77.89	79.25	**64.21**	**80.60**	77.29	77.90

GCP	GPT 3.5		GPT 4o		Claude 3.5		Claude 4		Gemini 2.0		Gemini 2.5	
	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few
Service Name	83.72	**89.77**	85.12	86.05	73.02	88.37	83.72	87.91	**59.53**	**89.77**	79.07	89.30
Start Time	**25.58**	37.21	47.44	55.35	33.49	44.65	**64.65**	49.77	45.12	46.98	57.67	47.44
End Time	**32.09**	44.19	78.60	84.19	66.98	80.47	85.12	86.98	73.95	77.67	85.12	**88.37**
Timezone	74.42	77.67	77.67	87.91	73.49	74.42	**89.77**	82.79	**54.88**	77.67	86.98	88.84
Service Categ.	61.86	53.02	61.86	**40.93**	64.19	56.74	62.33	61.86	62.33	**67.91**	62.33	64.19
Average	**55.53**	60.37	70.14	70.89	62.23	68.93	**77.12**	73.86	59.16	72.00	74.23	75.63

Finding 2: *Few-shot prompting generally improves accuracy for most fields, with a maximum average improvement of 17.34%. However, it does not improve performance for category extraction. In some cases, models with examples even perform worse.*

Generally, few-shot prompting improves accuracy in most cases. In our case, we observed improvements in 76.67% (23/30) of model–field combinations for GCP, 75.00% (27/36) for AWS, and 64.29% (27/42) for AZURE. However, it is less effective for service category and root category extraction, with some cases even showing notable decreases. For example, GPT-4o shows a 22.00% drop in service category accuracy on AWS and a 20.93% drop on GCP. A possible explanation is limited patterns in these classification tasks, which may lead to overfitting when few-shot examples are used with advanced reasoning models such as GPT-4o.

Finding 3: *Lightweight models sometimes perform better than advanced models, especially with few-shot examples.*

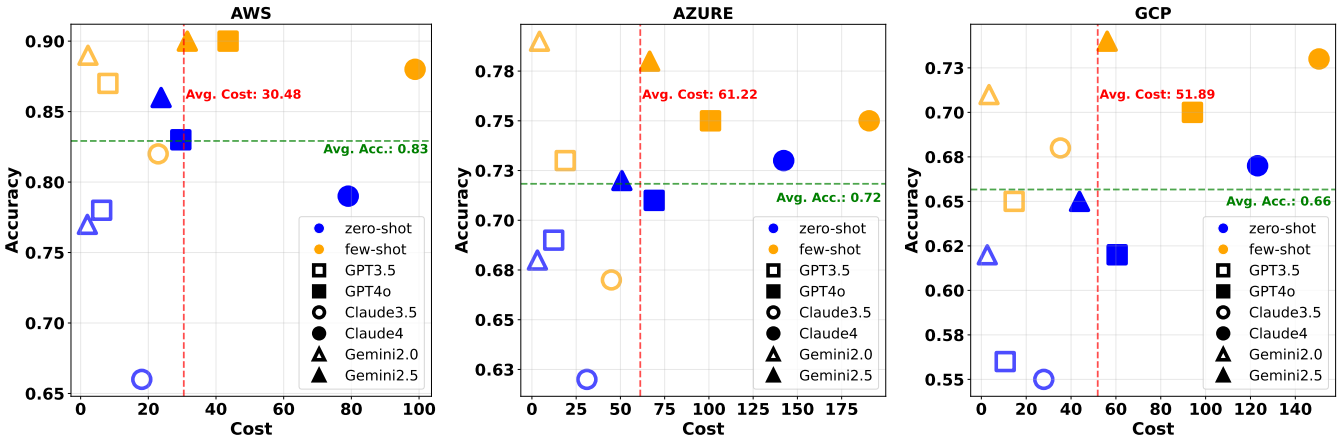
Advanced models do not necessarily ensure better performance. For example, the highest average accuracy for AZURE is 80.60% with Gemini 2.0 few-shot, compared to 77.90% for Gemini 2.5 few-shot. In terms of average accuracy, for Azure, lightweight models with few-shot learning could outperform advanced models without few-shot learning. In contrast, for GCP, advanced models achieve higher accuracy than general models regardless of few-shot prompting. This suggests that the optimal choice of model and prompting strategy varies across datasets and is not sufficient to select a single model or a fixed combination.

Finding 4: *Few-shot-CoT exhibits lower latency, even though it requires 1.5–2× more input tokens, likely due to the shorter output tokens.*

Although few-shot-CoT prompts require 1.5–2× more input tokens, they may exhibit lower latency because the structured output guided by the few-shot examples reduces the number of output tokens, as output tokens are the primary contributor to latency [2]. For example, on GCP, Claude 4 few-shot completes in 10.00 s, reducing latency by 1.08 s, compared to zero-shot (11.08 s).

Finding 5: *The most expensive models cost 50–60x more than the least costly models.*

Average costs vary significantly across models, with the most expensive models costing 50–60× more than the least costly ones across datasets. For example, on Azure, Claude 4 few-shot costs 190.54 (10^{-4} \$), which is 61.5× and 44.6× higher than Gemini 2.0 zero-shot and few-shot, and costs only 3.10 (10^{-4} \$) and 4.27 (10^{-4} \$),



(a) AWS. (b) AZURE. (c) GCP.

Figure 3: Model selection trade-off between accuracy and cost.

respectively. However, the Gemini 2.0 few-shot shows higher accuracy and much lower latencies. This highlights that lightweight models, such as Gemini 2.0, can achieve competitive or even superior accuracy while incurring lower cost and latency, emphasizing the importance of considering performance–cost tradeoffs when selecting LLMs for practical applications.

4.3 Implications for Model Selection

Finding 6: *Lightweight models, including Gemini 2.0 and GPT 3.5, offer a strong balance of accuracy, cost, and latency across datasets. Advanced models, such as Gemini 2.5 and GPT-4o, can achieve higher accuracy but at substantially higher cost and latency, while Claude need further optimization (e.g. fine-tuning) for effective use.*

Figure 3 illustrates the relationship between accuracy and cost across datasets, models, and prompts. Models positioned toward the upper-left corner indicate higher accuracy at lower cost. The green line represents average accuracy, while the red line represents average cost. Overall, when using LLMs for incident report extraction, we recommend employing few-shot prompts and starting with lightweight models such as Gemini 2.0 and GPT 3.5. Advanced models, such as Gemini 2.5 and GPT-4o, may be considered if higher accuracy is required. Claude series models require further optimization for better application in this task.

5 Threats to Validity

Our work shows how LLMs can be used to extract information from cloud incident reports accurately and effectively. However, our approach has certain threats and limitations:

(1) The *accuracy* of the models is constrained by the number and content of the selected few-shot examples. Further research is needed to optimize the design of few-shot prompts for improved accuracy;

(2) The *quality* of the data annotation is limited by the subclassification of specific fields, such as service category, user symptom category, etc; Also, This "closed-world" classification approach may fail to capture complex and evolving incident types that do not fit neatly into the pre-set buckets, potentially losing critical nuances.

(3) The *depth* of current incident analysis is constrained by the availability of public incident reports. While many numerical results are presented and visually summarized, deeper corelation analysis or causal insights are limited due to the lack of detailed information. For instance, AWS and GCP do not disclose detailed root cause information in their reports, which limits the depth of our analysis.

6 Related Work

Incident Management and Analysis: Cloud incident management and analysis are essential to understand and improve the reliability of cloud computing services, as they provide valuable insights that support root cause detection and mitigation [1, 18, 21]. Previous work has been carried out on empirical characterization of outage and incident reports in cloud services [12, 25, 26] , LLM services [9, 31], and frameworks for automation of such analysis [5]. However, these studies either rely on manual or rule-based approaches for information extraction, or focus only on incident metadata fields, without deeper analysis such as user impact and root cause classification. The emergence of AI provides solutions for such problems. There has been research exploring ways to automate the extraction of structured information from incident records using machine learning, such as SoftNER [24], Bayesian networks [8], and CUA [25]. However, there is a lack of studies that systematically explore the application of LLMs.

AI for Performance Engineering: AI can provide powerful insights into system performance engineering [13]. Currently, there are two main application areas that stand out: LLMs for log parsing [3, 16], and LLMs for root cause analysis (RCA) [1, 7, 21, 27, 28], where LLMs have shown promising performance. For example, RCACopilot [7] integrates language models into diagnostic workflows, and TAMO [27] combines logs, traces, and metrics in a tool-assisted LLM agent to overcome the context and modality limits of RCA. However, these works rely mainly on telemetry and internal monitoring data rather than public incident reports. Ours is the first study to construct an annotated dataset of incident reports, systematically compare LLMs and prompts for structured information extraction, and perform longitudinal analyses of multiple incident characteristics.

7 Conclusion and Future Work

Accurate and efficient data extraction and analysis from cloud incident reports are important for improving the dependability of cloud computing services. To address this challenge, we propose a methodology that demonstrates how to leverage LLMs for structured data extraction.

In this work, we collect 3,000 incident reports from three cloud operators, and annotate 460 of them for evaluation. We propose five prompt components and design six strategies. Using both lightweight and advanced LLMs, we then develop data extraction pipeline to extract ten types of information from textual incident reports. After that, we evaluate and compare the accuracy, latency, and cost of six LLMs, providing insights into prompt and model selection adapting to different requirements in practical report extraction. Overall, we summarize 6 key findings, and provide open-source artifacts as valuable resources for system researchers, cloud engineers, and service users to better understand and improve cloud incident management.

Our future work includes: (1) Optimized evaluation of prompts. Further research is required to optimize prompt design, and to evaluate different components through controlled experiments. (2) Proactive incident prediction. The extracted information can be further used to predict incident duration and root causes, which helps better demonstrating downstream utility in incident mitigation. (3) Advanced LLM techniques. More advanced LLM techniques, such as fine-tuning and Retrieval Augmented Generation (RAG), can be applied to further improve extraction accuracy in complex fields thorough historical patterns.

Acknowledgments

This work is partially supported by EU MSCA CloudStars (101086248) and Horizon Graph Massivizer (101093202), and by the NL National Growth Fund 6G flagship project Future Network Services. We acknowledge ChatGPT use for grammar and clarity only; this content is original and was written entirely by the authors.

References

[1] 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 *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*. IEEE, 1737–1749. https://doi.org/10.1109/ICSE48619.2023.00149

[2] OpenAI API. 2025. Latency Optimization. https://platform.openai.com/docs/guides/latency-optimization, Accessed: 2025-08-18.

[3] Merve Astekin, Max Hort, and Leon Moonen. 2024. A Comparative Study on Large Language Models for Log Parsing. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (Barcelona, Spain) *(ESEM '24)*. Association for Computing Machinery, New York, NY, USA, 234–244. https://doi.org/10.1145/3674805.3686684

[4] Microsoft Azure. 2025. Azure Status History. https://azure.status.microsoft/en-us/status/history/, Accessed: 2025-09-09.

[5] Sándor Battaglini-Fischer, Nishanthi Srinivasan, Bálint László Szarvas, Xiaoyu Chu, and Alexandru Iosup. 2025. FAILS: A Framework for Automated Collection and Analysis of LLM Service Incidents. In *Companion of the 16th ACM/SPEC International Conference on Performance Engineering* (Toronto ON, Canada) *(ICPE '25)*. Association for Computing Machinery, New York, NY, USA, 187–194. https://doi.org/10.1145/3680256.3721320

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[7] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, Jun Zeng, Supriyo Ghosh, Xuchao Zhang, Chaoyun Zhang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Tianyin Xu. 2024. Automatic Root Cause Analysis via Large Language Models for Cloud Incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems, EuroSys 2024, Athens, Greece, April 22-25, 2024*. ACM, 674–688. https://doi.org/10.1145/3627703.3629553

[8] Yujun Chen, Xian Yang, Qingwei Lin, Hongyu Zhang, Feng Gao, Zhangwei Xu, Yingnong Dang, Dongmei Zhang, Hang Dong, Yong Xu, Hao Li, and Yu Kang. 2019. Outage Prediction and Diagnosis for Cloud Service Systems. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 2659–2665. https://doi.org/10.1145/3308558.3313501

[9] Xiaoyu Chu, Sacheendra Talluri, Qingxian Lu, and Alexandru Iosup. 2025. An Empirical Characterization of Outages and Incidents in Public Services for Large Language Models *(ICPE '25)*. Association for Computing Machinery, New York, NY, USA, 69–80. https://doi.org/10.1145/3676151.3719372

[10] Google Cloud. 2025. Google Cloud Service Health. https://status.cloud.google.com/summary, Accessed: 2025-09-09.

[11] Drishti Goel, Fiza Husain, Aditya Singh, Supriyo Ghosh, Anjaly Parayil, Chetan Bansal, Xuchao Zhang, and Saravan Rajmohan. 2024. X-Lifecycle Learning for Cloud Incident Management using LLMs. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, FSE 2024, Porto de Galinhas, Brazil, July 15-19, 2024*, Marcelo d'Amorim (Ed.). ACM, 417–428. https://doi.org/10.1145/3663529.3663861

[12] Haryadi S. Gunawi, Mingzhe Hao, Riza O. Suminto, Agung Laksono, Anang D. Satria, Jeffry Adityatama, and Kurnia J. Eliazar. 2016. Why Does the Cloud Stop Computing? Lessons from Hundreds of Service Outages. In *Proceedings of the Seventh ACM Symposium on Cloud Computing, Santa Clara, CA, USA, October 5-7, 2016*. ACM, 1–16. https://doi.org/10.1145/2987550.2987583

[13] Lizy K. John. 2025. AI for Performance Engineering and Performance Engineering for AI. In *Proceedings of the 16th ACM/SPEC International Conference on Performance Engineering* (Toronto ON, Canada) *(ICPE '25)*. Association for Computing Machinery, New York, NY, USA, 1–2. https://doi.org/10.1145/3676151.3720528

[14] 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 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. http://papers.nips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html

[15] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 9, Article 195 (Jan. 2023), 35 pages. https://doi.org/10.1145/3560815

[16] Yilun Liu, Shimin Tao, Weibin Meng, Jingyu Wang, Wenbing Ma, Yuhang Chen, Yanqing Zhao, Hao Yang, and Yanfei Jiang. 2024. Interpretable Online Log Analysis Using Large Language Models with Prompt Strategies. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension* (Lisbon, Portugal) *(ICPC '24)*. Association for Computing Machinery, New York, NY, USA, 35–46. https://doi.org/10.1145/3643916.3644408

[17] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, USA.

[18] Victor Ion Munteanu, Andrew Edmonds, Thomas Michael Bohnert, and Teodor-Florin Fortis. 2014. Cloud Incident Management, Challenges, Research Directions, and Architectural Approach. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. 786–791. https://doi.org/10.1109/UCC.2014.128

[19] Alec Radford and et al. 2019. Language Models are Unsupervised Multitask Learners. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[20] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 2383–2392. https://doi.org/10.18653/v1/D16-1264

[21] Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro Henrique B. Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. 2024. Exploring LLM-Based Agents for Root Cause Analysis. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering* (Porto de Galinhas, Brazil) *(FSE 2024)*. Association for Computing Machinery, New York, NY, USA, 208–219. https://doi.org/10.1145/3663529.3663841

[22] Amrita Saha and Steven C. H. Hoi. 2022. Mining root cause knowledge from cloud service incident investigations for AIOps. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice* (Pittsburgh, Pennsylvania) *(ICSE-SEIP '22)*. Association for Computing Machinery, New York, NY, USA, 197–206. https://doi.org/10.1145/3510457.3513030

[23] Amazon Web Services. 2025. AWS Service Health. https://health.aws.amazon.com/health/status, Accessed: 2025-09-09.

[24] Manish Shetty, Chetan Bansal, Sumit Kumar, Nikitha Rao, Nachiappan Nagappan, and Thomas Zimmermann. 2021. Neural Knowledge Extraction From Cloud Service Incidents. In *43rd IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2021, Madrid, Spain, May 25-28, 2021*. IEEE, 218–227. https://doi.org/10.1109/ICSE-SEIP52600.2021.00031

[25] Sacheendra Talluri, Dante Niewenhuis, Xiaoyu Chu, Jakob Kyselica, Mehmet Cetin, Alexander Balgavy, and Alexandru Iosup. 2025. Cloud Uptime Archive: Open-Access Availability Data of Web, Cloud, and Gaming Services. *CoRR* abs/2504.09476 (2025). https://doi.org/10.48550/ARXIV.2504.09476 arXiv:2504.09476

[26] Sacheendra Talluri, Leon Overweel, Laurens Versluis, Animesh Trivedi, and Alexandru Iosup. 2021. Empirical Characterization of User Reports about Cloud Failures. In *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. 158–163. https://doi.org/10.1109/ACSOS52086.2021.00039

[27] Qi Wang, Xiao Zhang, Mingyi Li, Yuan Yuan, Mengbai Xiao, Fuzhen Zhuang, and Dongxiao Yu. 2025. TAMO:Fine-Grained Root Cause Analysis via Tool-Assisted LLM Agent with Multi-Modality Observation Data. *CoRR* abs/2504.20462 (2025). https://doi.org/10.48550/ARXIV.2504.20462 arXiv:2504.20462

[28] Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Jihong Wang, Fengbin Yin, Lunting Fan, Lingfei Wu, and Qingsong Wen. 2024. RCAgent: Cloud Root Cause Analysis by Autonomous Agents with Tool-Augmented Large Language Models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024*, Edoardo Serra and Francesca Spezzano (Eds.). ACM, 4966–4974. https://doi.org/10.1145/3627673.3680016

[29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[30] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2023. Large Language Models for Generative Information Extraction: A Survey. *CoRR* abs/2312.17617 (2023). https://doi.org/10.48550/ARXIV.2312.17617 arXiv:2312.17617

[31] Haoran Yan, Yinfang Chen, Minghua Ma, Ming Wen, Shan Lu, Shenglin Zhang, Tianyin Xu, Rujia Wang, Chetan Bansal, Saravan Rajmohan, Chaoyun Zhang, and Dongmei Zhang. 2025. An Empirical Study of Production Incidents in Generative AI Cloud Services. *CoRR* abs/2504.08865 (2025). https://doi.org/10.48550/ARXIV.2504.08865 arXiv:2504.08865

[32] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=SkeHuCVFDr