InsightsSumm - Summarization of ITOps Incidents through In-Context Prompt Engineering

Suranjana Samanta
IBM Research
India
suransam@in.ibm.com

Oishik Chatterjee
IBM Research
India
oishik.chatterjee@ibm.com

Neil Boyette
IBM Software
USA
neil.boyette@us.ibm.com

Guangya Liu
IBM Software
USA
gyliu@ibm.com

Prateeti Mohapatra
IBM Research
India
pramoh01@in.ibm.com

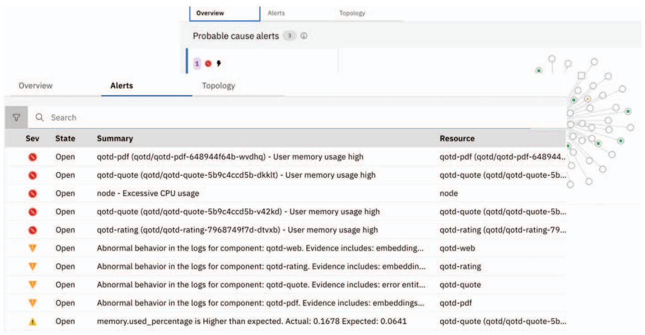
***Abstract*—AI has been extensively used to help Site Reliability Engineers (SREs) to resolve faults in cloud services and applications. It helps to accelerate resolution time by navigating through the vast amount of heterogeneous data (logs, metrics, alerts, etc) related to a fault. A good ITOps system should help SREs by giving precise and meaningful insights for a quick understanding of the data at hand. In this paper, we design a framework to summarize the context or insight present in the heterogeneous data related to a fault. The proposed framework constructs queries/prompts, specific to the ITOps domain, which helps us to generate more insightful abstractive summaries using state-of-the-art text generator models. Initial study on simulated faults shows promising results, which can be expanded to accommodate other datatype, providing summaries for real-world cases.**

***Index Terms*—Abstractive summary, in-context learning, prompt engineering, metric evaluation, incident insights.**

I. Introduction

The increase in IT deployments and cloud adoption owing to rapid digitization has led to many challenges. One of the main challenges for an operations team is how to derive insights from heterogeneous data coming from disparate sources (such as metrics, logs, alerts/anomalies, and topology) and application lifecycle stages to help remediate a failure. Artificial Intelligence for IT Operations (ITOps) aims to transform IT operational services by analyzing and bringing together data around a fault (*context*) to assist human decisions for faster fault resolution. Insights can include all alerts, topological information, detected anomalous behaviors, and probable cause identification including pieces of evidences, related to the fault. AI is able to do this faster while drawing deeper insights and at a far greater scale than manually possible, especially when considering the shrinking error budgets (aka allowed time an application can be impacted) and exploding quantities of data.

Most ITOps products use an incidents insights dashboard to collate and view these key insights for a given incident. Figure 1 presents one such example snapshot of a part of an insights dashboard where the overview tab presents the top probable causes and related historical tickets, the alerts tab includes all alerts associated with the fault and the topology tab shows the relationships of impacted resources - all together constitute a *context* associated with a fault. However, with such rich and exhaustive context present on these dashboards, it is still challenging and time-consuming to comprehend the key insights as it may require a lot of cognitive and perceptual efforts. This paper introduces the task of *InsightsSumm for ITOps*. Namely, given heterogeneous insights/context data around a fault (e.g., anomalies and alerts with their details, topology information, evidences), the objective is to generate an abstractive summary through prompt engineering. These summaries could serve as input to perform various other downstream tasks such as query generation, question answering, ticket generation, and prompt design for code generation.



Sev	State	Summary	Resource
🔴	Open	qotd-pdf (qotd/qotd-pdf-648944f64b-wvdhq) - User memory usage high	qotd-pdf (qotd/qotd-pdf-648944..
🔴	Open	qotd-quote (qotd/qotd-quote-5b9c4ccd5b-dkklt) - User memory usage high	qotd-quote (qotd/qotd-quote-5b...
🔴	Open	node - Excessive CPU usage	node
🔴	Open	qotd-quote (qotd/qotd-quote-5b9c4ccd5b-v42kd) - User memory usage high	qotd-quote (qotd/qotd-quote-5b...
🔴	Open	qotd-rating (qotd/qotd-rating-7968749f7d-dtvxb) - User memory usage high	qotd-rating (qotd/qotd-rating-79...
▼	Open	Abnormal behavior in the logs for component: qotd-web. Evidence includes: embedding...	qotd-web
▼	Open	Abnormal behavior in the logs for component: qotd-rating. Evidence includes: embeddin...	qotd-rating
▼	Open	Abnormal behavior in the logs for component: qotd-quote. Evidence includes: error entit...	qotd-quote
▼	Open	Abnormal behavior in the logs for component: qotd-pdf. Evidence includes: embeddings...	qotd-pdf
⚠	Open	memory.used_percentage is Higher than expected. Actual: 0.1678 Expected: 0.0641	qotd-quote (qotd/qotd-quote-5b...

Fig. 1. ITOps Incidents Dashboard, showing context for a simulated fault.

Recent methods of summarization assume the input to be a paragraph, where the sentences are interlinked and grammatically correct. For our summarization task, we consider the list of alerts and anomalies (from the context) as input, where a particular resource is linked to a symptom due to a fault. We generate the summary using state-of-the-art large language models (LLM) like Flan T5 XXL [1] and ChatGPT [2] using few-shot in-context instruction learning or prompt engineering [3], [4]. This work is in the direction of generating a summary using heterogeneous data, like alerts, topology, logs, and other important derived cues like Golden Signal [5] which are related to a fault in a software system. The main contributions of this paper are:

1) Generating prompts for few-shot learning from incident insights data.
2) Proposing ITOps domain-specific evaluation metrics for evaluating generated summary by focusing on resources and symptoms mentioned in the input.

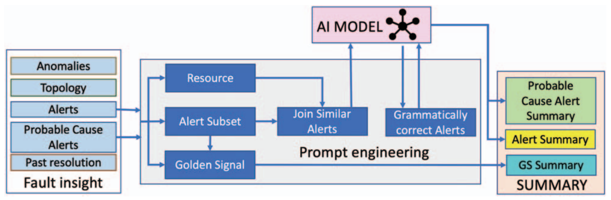


Fig. 2. Flowchart of the proposed InsightsSumm for generating fault summary through prompt engineering.

II. Proposed Method

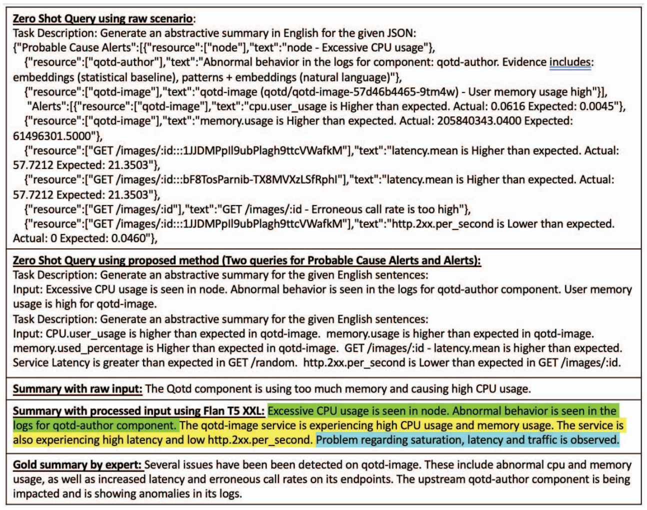
In this section, we describe our proposed method of *InsightsSumm*, for generating summaries using custom-made queries/prompts/input (Figure 2). We use the *context* that includes the list of alerts/anomalies along with affected resources for generating summaries. A subset of these alerts has been already marked as Probable Cause Alerts by the existing ITOps system. We divide the list of alerts into two sets - The Probable Cause Alert list, and the Alerts list containing the rest of the alerts. We follow the rest of the steps described below to generate intermediate summaries from both lists.

Golden Signals are key performance indicators that are used to monitor and analyze the health of a system. The four Golden Signals are: Error, Saturation, Latency, and Traffic [6]. We adopt the method in [5] for getting a subset of important alerts. Sentences in the alert description/text that do not contain any golden signal or resource name are not used further. For e.g., we drop the second sentence in the alert text: "Abnormal behavior in the logs for the component: qotd-author. Evidence includes: embeddings (statistical baseline), patterns+embeddings (natural language)", as it does not contain any golden signal or resource name.

A context may contain similar alerts from different endpoints or resources. We combine all the resource names together to form a singular alert. For e.g., three similar alerts present in a context, such as:
{*"resource":["GET /random"], "text": "GET /random - Increase in latency (95th percentile)"*},
{*"resource":["GET /"], "text": "GET / - Increase in latency (95th percentile)"*},
{*"resource":["GET /ratings/:id"], "text": "GET /ratings/:id - Increase in latency (95th percentile)"*}
can be combined to form a single alert: {*"resource":["GET /random", "GET /", "GET /ratings/:id"], "text": "Increase in latency (95th percentile)"*}. Next, we use an LLM to convert the ill-formed sentences in the alert text into grammatically correct sentences. We use BLOOMZ[1] with the greedy method as it generates better sentences. We build the query as a triplet: $\{Instruction, Input, Output\}$, using three scenarios for few-shot learning. Finally, we use Flan T5 XXL [1] and ChatGPT to generate the summaries using the engineered prompts.

Prompt Engineering [3], [4] enables LLMs to produce output as desired, without changing any weights. The instruction used for generating a sentence with correct grammar is "Task Description: Generate grammatically correct sentence in English:". We have used a similar triplet format for generating summarization with the instruction "Task Description: Generate an abstractive summary for the given English sentences:". We mention the golden signals detected from the context in a sentence through the slot-filling technique [7]. The final summary is a concatenation of the intermediate summaries generated from the Probable Cause Alert list, Alert list and detected Golden signals.

[1]https://huggingface.co/bigscience/bloomz



Zero Shot Query using raw scenario:
Task Description: Generate an abstractive summary in English for the given JSON:

```
{"Probable Cause Alerts":[{"resource":["node"],"text":"node - Excessive CPU usage"},
   {"resource":["qotd-author"],"text":"Abnormal behavior in the logs for component: qotd-author. Evidence includes:
embeddings (statistical baseline), patterns + embeddings (natural language)"},
   {"resource":["qotd-image"],"text":"qotd-image (qotd/qotd-image-57d46b4465-9tm4w) - User memory usage high"}],
    "Alerts":[{"resource":["qotd-image"],"text":"cpu.user_usage is Higher than expected. Actual: 0.0616 Expected: 0.0045"},
   {"resource":["qotd-image"],"text":"memory.usage is Higher than expected. Actual: 205840343.0400 Expected:
61496301.5000"},
   {"resource":["GET /images/:id:::1JJDMPpll9ubPlagh9ttcVWafkM"],"text":"latency.mean is Higher than expected. Actual:
57.7212 Expected: 21.3503"},
   {"resource":["GET /images/:id:::bF8TosParnib-TX8MVXzLSfRphl"],"text":"latency.mean is Higher than expected. Actual:
57.7212 Expected: 21.3503"},
   {"resource":["GET /images/:id"],"text":"GET /images/:id - Erroneous call rate is too high"},
   {"resource":["GET /images/:id:::1JJDMPpll9ubPlagh9ttcVWafkM"],"text":"http.2xx.per_second is Lower than expected.
Actual: 0 Expected: 0.0460"},
```

Zero Shot Query using proposed method (Two queries for Probable Cause Alerts and Alerts):
Task Description: Generate an abstractive summary for the given English sentences:
Input: Excessive CPU usage is seen in node. Abnormal behavior is seen in the logs for qotd-author component. User memory usage is high for qotd-image.
Task Description: Generate an abstractive summary for the given English sentences:
Input: CPU.user_usage is higher than expected in qotd-image. memory.usage is higher than expected in qotd-image. memory.used_percentage is Higher than expected in qotd-image. GET /images/:id - latency.mean is higher than expected. Service Latency is greater than expected in GET /random. http.2xx.per_second is Lower than expected in GET /images/:id.

Summary with raw input: The Qotd component is using too much memory and causing high CPU usage.

Summary with processed input using Flan T5 XXL: Excessive CPU usage is seen in node. Abnormal behavior is seen in the logs for qotd-author component. The qotd-image service is experiencing high CPU usage and memory usage. The service is also experiencing high latency and low http.2xx.per_second. Problem regarding saturation, latency and traffic is observed.

Gold summary by expert: Several issues have been been detected on qotd-image. These include abnormal cpu and memory usage, as well as increased latency and erroneous call rates on its endpoints. The upstream qotd-author component is being impacted and is showing anomalies in its logs.

Fig. 3. Example showing (i) query using raw alert lists (ii) query using proposed prompt engineering (iii) generated summary using a query as shown in the first row (iv) generated summary using the constructed query as shown in the second row (proposed). Input for generated intermediate summaries in color codes: green - Probable Cause Alerts, yellow - Alerts, cyan - golden signal; and (v) Gold summary written by Site Reliability Engineer (SRE).

Figure 3 illustrates how the proposed method processes the list of alerts into a query. We also show the generated summary, with and without prompt engineering, when three other examples were used as few-shot learning in the query. The comparison clearly shows the effectiveness of the proposed method of query building for a better summarization task. For the generated summary using the proposed method, the color coding is as follows: green - intermediate summary from Probable Cause Alerts, yellow - intermediate summary from Alerts, cyan - intermediate summary from Golden Signals.

III. Proposed Metrics and Initial Experiments

We present the initial results of InsightsSumm via the constructed prompts. We use simulated faults from the "Quote Of The Day" (QOTD) application [8] for experimental study. First, we manually construct the summaries of a few fault scenarios, which we use for in-context summary generation through prompt engineering. We use the three (most diverse) out of ten scenarios along with their handwritten summaries as examples in the prompt. We use the Flan T5 XXL from Hugging Face [1] for summary generation, with seed value 124 and a temperature value of 0.3.

Evaluation Metrics - It is important to evaluate the generated summaries using proper metrics. It is essential that major problems along with their corresponding resources are referenced in the generated summary. This motivates us to use four new proposed ITOps-specific metrics along with RougeL [9] and BertScore [10] for evaluation, which are defined as:

- **Probable Cause Resource Precision (PCRP)**: Fraction of resources reported in the summary that are present in the Probable Cause Alerts list. PCRP = 1 indicates that the summary is not talking about resources that have no issues; i.e. no "ghost" resource is present in the summary.
- **Probable Cause Resource Recall (PCRR)**: Fraction of resources present in the Probable Cause Alerts list that are reported in the summary. A high PCRR indicates that the summary has a comprehensive overview of the Probable Cause Alerts and has not missed important details that could help in further downstream tasks such as query/ticket generation etc.
- **Alert Resource Precision (ARP)**: Fraction of resources reported in the summary that are present in the Alerts list.
- **Alert Resource Recall (ARR)**: Fraction of resources present in Alerts that are reported in the summary.

Results - We generate summaries using few-shot in-context learning settings, with (proposed) and without (raw) the proposed method for prompt construction, using both ChatGPT and Flan T5 XXL [1]. Results in Table I show that the query constructed using our proposed framework gives much better summaries than the raw query for both ChatGPT and Flan T5 XXL models.

TABLE I
PERFORMANCE OF GENERATED SUMMARIES WITH AND WITHOUT USING OUR PROPOSED METHOD FOR PROMPT CONSTRUCTION, USING CHATGPT AND FLAN T5 XXL. FOR ROUGEL WE REPORT THE ROUGEL SCORE AND FOR BERTSCORE WE REPORT THE F1 SCORE.

Metric	ChatGPT		Flan T5 XXL	
	raw	proposed	raw	proposed
RougeL	0.159	0.196	0.021	**0.299**
BertScore	0.813	0.83	0.216	**0.848**
PCRP	0.094	**0.691**	0	0.367
PCRR	0.094	**0.918**	0	0.277
ARP	0.115	0.253	0.285	**0.717**
ARR	0.12	**0.468**	0.05	0.364

Discussions - Table I shows that our proposed prompt engineering technique outperforms raw input significantly for both ChatGPT and Flan T5 models. This is because there is a lot of irrelevant information present in the raw input which makes it difficult for the models to focus on the relevant issues. In the prompt engineered input, relevant, concise and grammatically correct details are passed and hence the LLMs are able to generate better summaries using prompt engineered queries. Also, it is observed that ChatGPT generates longer and more elaborate summaries, with ghost resource names that are not present in the input. Hence, recall is higher for ChatGPT summary as compared to precision for the resources. For Flan T5, the summaries are shorter and contain fewer resources from the input, which causes a better value of precision than recall. It is interesting to notice that Flan T5 adapts to the style of the gold summaries (passed as few-shot examples in prompt) more than ChatGPT, causing higher BertScore and RougeL. By comparing the metrics for both raw and prompt engineered inputs for ChatGPT, we can see that RougeL and BertScore are quite close whereas there is a huge difference in the precision and recall metrics for the resources. Though ChatGPT produces aesthetically better summaries, it captures the issues present in the input less effectively. Hence, proposed resource specific precision and recall metrics are required for evaluating generated summaries for ITOps.

IV. CONCLUSION AND FUTURE WORK

The proposed method of query construction for summarization using in-context learning uses the heterogeneous data associated with a fault. The preliminary results show promising results, on which we can build an improved summarization module that can ingest topology information, past resolved similar tickets, graphs, tables, etc in the future. We have plans to generate better summaries by continual pretraining of the language models. We also plan to have more examples with gold summaries for creating soft prompts through example selection for few-shot learning. Overall, the proposed method of InsightsSumm is the first step towards building a robust summarization method of fault insights for enterprise use.

REFERENCES

[1] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei, "Scaling instruction-finetuned language models," 2022.

[2] "Chatgpt version 3.5," https://openai.com/blog/chatgpt, 2023.

[3] E. Saravia, "Prompt Engineering Guide," *https://github.com/dair-ai/Prompt-Engineering-Guide*, 2022.

[4] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.

[5] S. Nagar, S. Samanta, P. Mohapatra, and D. Kar, "Building golden signal based signatures for log anomaly detection," in *IEEE 15th International Conference on Cloud Computing, CLOUD 2022*. IEEE, 2022.

[6] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, *Site Reliability Engineering: How Google Runs Production Systems*, 2016. [Online]. Available: http://landing.google.com/sre/book.html

[7] H. Weld, X. Huang, S. Long, J. Poon, and S. C. Han, "A survey of joint intent detection and slot filling models in natural language understanding," *ACM Comput. Surv.*, vol. 55, no. 8, 2022.

[8] "Quote of the day - version 4.1.0," https://gitlab.com/quote-of-the-day/quote-of-the-day, 2023.

[9] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Association for Computational Linguistics, 2004.

[10] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations*, 2020.