Assess and Summarize: Improve Outage Understanding with Large Language Models

Pengxiang Jin*
Shenglin Zhang*
Nankai University
China

Minghua Ma
Microsoft
China

Haozhe Li
Peking University
China

Yu Kang
Liqun Li
Microsoft
China

Yudong Liu
Bo Qiao
Microsoft
China

Chaoyun Zhang
Pu Zhao
Microsoft
China

Shilin He
Microsoft
China

Federica Sarro
University College
London
UK

Yingnong Dang
Saravan
Rajmohan
Microsoft
USA

Qingwei Lin
Dongmei Zhang
Microsoft
China

ABSTRACT

Cloud systems have become increasingly popular in recent years due to their flexibility and scalability. Each time cloud computing applications and services hosted on the cloud are affected by a cloud outage, users can experience slow response times, connection issues or total service disruption, resulting in a significant negative business impact. Outages are usually comprised of several concurring events/source causes, and therefore understanding the context of outages is a very challenging yet crucial first step toward mitigating and resolving outages. In current practice, on-call engineers have to manually assess and summarize outages when they happen, which is time-consuming and labor-intensive. In this paper, we first present a large-scale empirical study investigating the way on-call engineers currently deal with cloud outages at Microsoft, and then present and empirically validate a novel approach (dubbed Oasis) to help the engineers in this task. Oasis is able to automatically assess the impact scope of outages as well as to produce human-readable summarization. Specifically, Oasis first assesses the impact scope of an outage by aggregating relevant incidents via multiple techniques. Then, it generates a human-readable summary by leveraging fine-tuned large language models like GPT-3.x. The impact assessment component of Oasis was introduced in Microsoft over three years ago, and it is now widely adopted, while the outage summarization component has been recently introduced, and in this article we present the results of an empirical evaluation we carried out on 18 real-world cloud systems as well as a human-based evaluation with outage owners. The results obtained show that Oasis can effectively and efficiently summarize outages, and lead Microsoft to deploy its first prototype which is currently under experimental adoption by some of the incident teams.

CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**; • **Software and its engineering** → **Maintaining software**.

KEYWORDS

Outage Understanding, Large Language Model, Cloud Systems

ACM Reference Format:
Pengxiang Jin, Shenglin Zhang, Minghua Ma, Haozhe Li, Yu Kang, Liqun Li, Yudong Liu, Bo Qiao, Chaoyun Zhang, Pu Zhao, Shilin He, Federica Sarro, Yingnong Dang, Saravan Rajmohan, Qingwei Lin, Dongmei Zhang. 2024. Assess and Summarize: Improve Outage Understanding with Large Language Models. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With the trend of large IT enterprises such as Microsoft, Amazon, and Google deploying services to the cloud platforms, cloud systems have had a booming development in recent years [4, 8, 22, 25, 31]. Tremendous efforts have been devoted to improving the reliability of cloud systems, however, unplanned incidents or performance degradation are still inevitable due to the complex and dynamic nature of cloud systems. Often these incidents escalate to a so called *outage*, which impacts multiple services and customers.

Once an outage occurs to a cloud system, it is crucial to understand its *impact scope* as soon as possible in order to promptly notify customers [36, 39, 41], mitigate issues [4, 35], and ultimately resolve the outage [20, 26, 38], aiming at reducing as much as possible the loss associate with it. Nevertheless, a cloud system is quite complex and involves many services such as across-region infrastructures, virtual machines, networking, and database systems, thus making this task very challenging. To support engineers in monitoring the reliability of the cloud system, each cloud system service has multiple monitors that create an incident each time something wrong occurs. For example, Figure 1 shows the timeline of an incident caused by a flawed configuration change in the Storage service. The failed storage affected several SQL databases, and the failure was further propagated to web application instances. Finally, the outage

*Equal contribution.
Work is done during the internship of Pengxiang Jin and Haozhe Li at Microsoft. Qingwei Lin is the corresponding author.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

is declared and associated with the multiple incidents occurred in the storage, SQL, and web application services. Doing this job manually is not trivial, and in some cases not even feasible. Being able to efficiently aggregate all and only those incidents which are relevant to a given outage, would empower the engineers to promptly investigate the impact scope, as it greatly reduces the number of incidents that need to be investigated.

Previous studies [8, 11, 17, 34] have devoted a lot of efforts to dealing with incidents aggregation or linking the relevant incidents to the outage. However, based on our real-world experience in Microsoft, we observe that on-call engineers (OCEs) still need to manually check the detailed information of relevant incidents and write a *summary* of outages (a real-world example in Section 3.2), which is helpful to further handle the outage in terms of notification, mitigation, diagnosis, and resolution. To the best of our knowledge, extensive studies on outage understanding are lacking. Therefore, in this paper, we first empirically investigate the negative effects of outages in worldwide popular cloud systems in Microsoft and how engineers currently deal with them. To this end, we exploit data collected from the usage of 18 real-world cloud systems (many of which are worldwide popular systems) over the past three years. We found that most outages have a substantial negative impact on customers, and the median summarization time is one time unit[1]. Therefore, in practice, engineers have to spend significant efforts to understand outages. Besides, the content of outage summaries often contains detailed when, where, who, what, and why. This information is complex and cannot simply adopt as a template because it must be readable by engineers from various component teams. Thus, it is necessary to automatically summarize outages for understanding quickly.

These results motivated us to explore automated ways to improve engineers' understanding of outages. To this end, we propose Oasis, which has two components: *impact scope assessment* and *summary generation*. As for impact scope assessment, we adopt three techniques *i.e.*, rule-based, historical lookup, and deep learning based to aggregate relevant incidents to the outage. To embed domain knowledge of cloud systems, engineers implement some linking rules from incidents to outages. To automatically learn the correlations among components of cloud systems, we propose a historical lookup algorithm to form a component graph based on the historical incident linkage and match new incidents in the graph. To capture the rapid evolution of cloud systems, the deep learning based linking approach is used. The impact scope of an outage is composed of the relevant incidents aggregated by these three techniques. We have deployed the impact assessment component of Oasis in Microsoft, which is running for over three years and has achieved significant results in impact scope assessment.

After we obtain relevant incidents of the outage, we adopt the most popular pre-trained large language models GPT-3.x (both GPT-3.0 and GPT-3.5), to automatically generate outage summaries. This task presents two main challenges: 1) identifying which information on relevant incidents is helpful to outage summarization; 2) identifying how to effectively generate domain-specific outage summaries with complex cloud-related information. For the first challenge, our empirical study provides some guidelines on summarizing outages, which reveals the importance of incident severity and description. To tackle the second challenge, we fine-tune the pre-trained large language model, which can generate human-readable sentences and embed with knowledge from cloud systems.

[1]Due to company policy, we hide the actual time and normalize it as a time unit.

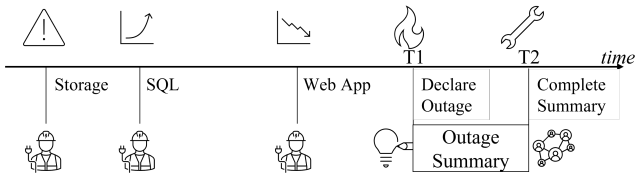


Figure 1: The timeline of handling an outage, where multiple incidents should be summarized when declare the outage.

To investigate the effectiveness of Oasis, we conduct extensive experiments using real-world outages from Microsoft. The results show that Oasis is able to effectively and efficiently generate outage summaries and titles for cloud systems, and significantly outperform all the compared approaches [24, 32]. More specifically, Oasis achieves scores of 0.665 (BLEU-4), 0.742 (ROUGE-L), and 0.734 (METEOR) with its summarization which outperforms state-of-the-art approaches by at least 32.3%. Furthermore, to investigate the usefulness and readability of our generated summaries, we conduct a preliminary human evaluation involving 54 outage owners. Based on the rankings of summaries produced by models and the original OCEs, we find that Oasis can achieve human-level summaries much more quickly (251.2 times faster than the median of manual summarization). Based on the above results, the Oasis outage assessment component has already been in usage for over three years at Microsoft, while the more recent summarization component has been now prototyped and used by some of the incident teams at Microsoft in a phase preceding the final rolling in production.

To sum up, our work has the following contributions:

- We are the first to identify outage understanding, a practical scenario for large-scale cloud services. We have conducted an empirical study of 18 cloud systems to investigate this scenario.
- We propose Oasis, the first automated approach to tackle the problem of outage understanding based on impact scope assessment and large language models (LLMs). We are the first to propose LLM-based summary generation of outages.
- Our impact scope assessment of Oasis has been deployed in Microsoft for over three years and achieved significant impact. We conduct an extensive study and human evaluation to demonstrate the efficacy and potential usage of Oasis.

2 BACKGROUND

Cloud systems. Cloud systems have become increasingly popular in recent years, as they offer a range of benefits such as scalability, accessibility, and cost-efficiency. To ensure the reliability of these systems, engineers use various monitoring tools and techniques, *e.g.*, Azure Monitor, to track and analyze the performance and health of different levels and components of the cloud system [10, 14]. If the monitors detect anomalies, incidents will be reported.

Incidents. Incidents are unplanned interruptions to cloud service. Incident Management is the process of logging those interruptions,

and resolving those in a timely manner [5, 7, 10, 16, 19, 21, 28]. An incident is reported with many fields, for example, the time and source of the incident creation, and a text field describing the problem. The text description can be generated by the monitor based on pre-defined templates or filled in manually by the engineers. Moreover, engineers assign a severity level to each incident, ranging from 0 to 4, where a severity of 0 means highest priority and large customer impact, and a severity of 4 means lowest priority.

Outages. Outages are severe incidents that require collaboration across many services or result in customer impact [9, 34]. Different products and teams may define outages differently depending on service level agreements (SLAs), customer expectations, or other criteria. When an outage happens, it tends to affect various aspects of the cloud system, causing many incidents to be reported. OCEs need to go through these incidents to fully understand the outage.

IcM system. To facilitate mitigating and resolving outages, Microsoft has developed an Incident Management system (IcM) for cloud systems. After a monitor reports an incident, an associated incident is created on the IcM. Then engineers can discuss the incident, check the information, and update the status of the incidents on the IcM page, *etc.* An incident may escalate and is declared as an outage if it impacts multiple services or customers as shown in Figure 1. During these processes, records of incidents and logs of the actions are persistently stored in the IcM database.

3 OUTAGE UNDERSTANDING: A CASE STUDY

To better understand the impact of outages and the need for automatic support in outage scope and summary production, we conduct a case study on real-world outages and their summaries. To this end, we collect outages from 18 systems over three years in the IcM database of Microsoft, which serves millions of users worldwide. To ensure that the outages have undergone careful examination and their summaries are ready, we keep over 6000 outages whose state is 'MITIGATED' or 'RESOLVED' during collection. We are not able to make all the details public due to the company's policy.

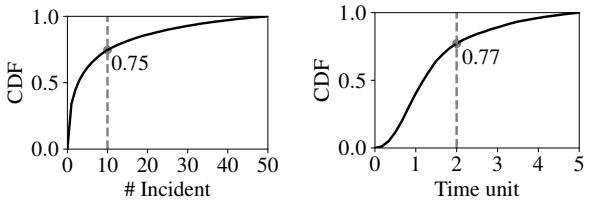
In this study, we address the following research questions:

- **RQ1**: What is the impact of outages?
- **RQ2**: What are the information included in outage summaries?
- **RQ3**: What is the time cost of manually summarizing outages?

3.1 RQ1: Impact of Outages

Impact on customers. When OCEs deal with outages, it is important to decide the impact on customers, especially the number of customers affected. For each outage, OCEs determine whether it impacts a large number of users and record this determination. We statistically analyze the outages that OCEs considered as impacting a large number of users and found that such outages accounted for as much as 86.4% of all outages. Outages usually have a significant impact on cloud systems, resulting in a degraded user experience for a large number of customers. Therefore, it is crucial to quickly and effectively respond to outages.

Another aspect of the customer impact is whether an outage has resulted in persistent impacts. *Persistent* impact means the service is consistently inaccessible, while *intermittent* impact means the service is accessible at certain times but not consistently. The number of outages resulting in persistent impacts is 1.81 times more than the number of outages that have intermittent impacts. The impact of an outage on a cloud system is frequently severe.



(a) Incidents (b) TTS

Figure 2: CDF of (a) Number of relevant incidents to outages. (b) Time to Summary (TTS).

Relevant incidents. Several incidents in cloud systems are continuously reported and escalate to one outage, as they share a common root cause. The distribution of incidents associated with outages is illustrated in Figure 2(a), with 25% of outages having more than 10 associated incidents. The average number of relevant incidents to an outage is 9.36. Based on this data, the outages bring about many incidents, consuming the efforts and time of the OCEs.

3.2 RQ2: Outage Summary Information

To help understand what information needs to be summarized for an outage, we demonstrate a real-world outage summary written by OCEs and its relevant incidents.

Incident 1 Title: Email service errors of batch events.
Description: The Email Service was experiencing connectivity issues to their replica database in the West US Region. Due to this issue, System-Cloud customers globally were not receiving any type of System-Cloud notifications.
Severity: 2
Start time: 14:28
Service: SQL

Incident 2 Title: No Success Signal in the last 60 minutes.
Description: Calls to the API-Sub failed with a 5xx HTTP error. Approximately α_1 customers could not upgrade their subscriptions on URL-Cloud-Portal.
Severity: 2
Start time: 15:30
Service: Commercial

Incident 3 Title: Email Orchestrator Health in Cluster_1
Description: Email notifications sent to customers could be delayed.
Severity: 3
Start time: 14:33
Service: Business Intelligence

Incident 4 Title: Api request failed with multiple -1 responses. Target: URL-Cloud-Email
Description: Calls to the API-Marketplace service failed which prevented the service from sending emails to the customers and affected α_2 customers.
Severity: 3
Start time: 17:06
Service: Marketplace

Incident 5 Title: Email Service calls are failing for Monitor-Email-Exceptions evaluated on MonitorRule$_1$ unhealthy
Description: Customers could not view Customer renewal and subscription alerts were delayed. In addition, users were unable to get authentication codes to verify login and new account sign-ups.
Severity: 3
Start time: 14:44
Service: Notification

The title, times, and summary of the outage are listed below:

Outage Title: Outage for Email Service - Triage
Impact start time: 14:20
Outage declared time: 14:28
OCEs engage time: 14:29
Outage Summary: The Email Service experienced connectivity issues to their replica database in the West US Region. This affected customer email delivery for approximately α_3 internal company services. Due to this issue, System-Cloud customers were not receiving notifications including purchase, renewal, and monitor alert notifications. The Portal team reported that approximately α_1 customers were unable to upgrade their subscriptions on URL-Cloud-Portal.

We can see from the above example that each relevant incident describes various aspects of the outage, and the information about an outage fall into many different categories. For example, West US Region is a physical location, and {System-Cloud, Email Service, API-Marketplace} are software components at different layers that are affected, and {α_1, α_2, α_3} are specific numbers describing the number of impacted customers or services, and *5xx HTTP error* is a software bug that affects the service functionality. The information of an outage usually involves 5W (when, where, who, what, why):

When. *When does the outage start impact, get declared, and engaged?* Engineers pay attention to several time points and periods of an outage. For example, the time when the outage starts to make an impact, when the outage is declared, and when the OCEs start to engage are important signals for assessing the availability and reliability of the system. Additionally, when assessing the impact of an outage, it is also useful to know the time window period when a certain function is unavailable.

Where. *Where does the outage come from?* The physical location of an outage can lie in various levels of the cloud infrastructure. The physical location can have an impact on the time required to resolve it and the potential for cascading failures. Additionally, the physical location of an outage can be a key factor in determining the impact on customers, as local or nearby customers may be affected more severely. The physical location of the *cloud* infrastructure at Microsoft is structured in a hierarchical manner [18] with *regions* and *availability zones* at the top level, which is directly accessible to customers. Each region can consist of up to three *availability zones*, each containing one or more *datacenters*. These *datacenters* are further divided into *clusters*. Despite the fact that other cloud systems may exhibit different location hierarchies, it is as important to know the location of outages.

Who. *Which services are suffering from the outage?* Services of cloud systems can be divided into different layers: (1) application layer: this layer contains the actual code and functionality of the cloud system, where frontend and backend services are located, (2) platform layer: this layer provides the operating system, middleware, and runtime environment for the cloud system, which may include virtualization software, container orchestration software, or serverless computing framework, (3) data layer: this layer handles the storage and management of data used by the cloud system, which may include databases, data lakes, and data warehouses, (4) infrastructure layer: this layer provides the underlying physical and virtual resources that are used to run the cloud system, which includes host servers, storage, and networking. Each layer has its fine-grained components. Assessing which parts of the cloud system are affected by the outage is helpful to handle the outages.

What. *What happens to the cloud system in the outage?* Previous research has shown [14] some common symptoms of outages, including: (1) *code bugs*, such as buggy or incompatible code that generates error results, (2) *dependency failures*, such as an unhealthy dependent service that impacts the functioning of downstream services, (3) *infrastructure issues*, such as high CPU utilization of a server that prevents the service from functioning normally, (4) *deployment errors*, such as an engineer deploying an incorrect certificate. There are also other less frequent symptoms, such as *configuration bugs*, *database/network* issues, *authentication failures*, *etc.*

Why. *Why did the outage happen?* Previous research has investigated the four most common root causes of outages [23]: (1) insufficient or erroneous mechanism of *fault handling* (*e.g.*, error component, unresponsive component, and silent corruption), (2) *data format* incompatibility between different software components, (3) *timing*(*e.g.*, concurrent) bugs, and (4) misconfigured or outdated *constant values*. However, the bugs in production cloud systems are highly diversified. The underlying causes of misbehavior require thorough manual investigation by OCEs.

Focus of outage summarization. We can see from the example that the summary of an outage is not simply a list of information. OCEs when writing outage summaries are more favorable to high-severity incidents. Moreover, textual description is an important reference in the outage summarization process. For example, the dashed sentences are taken directly from the textual descriptions of two high-severity incidents, *i.e.*, Incident 1 and 2.
Finding: High-severity incidents and their textual descriptions are important in outage summarization.

3.3 RQ3: Time to Summary

After an outage starts to make an impact on customers, OCEs need to quickly respond to the outage. One key step is to summarize the context of the outage. Therefore, we investigate the time to manually write outage summaries. Specifically, we retrieved the impact start time (T1 in Figure 1) of the outage and its summary completed time (T2). The time needed to summarize outages is calculated by T2 − T1.

Figure 2(b) shows the CDF of the time needed to summarize outages. From this figure, there are nearly 23% of outages cannot be summarized within two time units after the outage starts. The median time needed to summarize outages is one time unit. Therefore, outage summarization is time-consuming and labor-intensive.

3.4 Summary

According to our empirical study, the impact summary of an outage may include domain specific terminology of physical or logical locations, service name, code change name, *etc.* Besides, what and why of outages are even more difficult to summarize simply using templates. OCEs must have a thorough grasp of the relevant incidents in order to effectively summarize an outage, and the text descriptions of these incidents provide crucial information for this comprehension. Nowadays, pretrained large language models (*e.g.*, GPT-x) show their ability in many tasks, such as Q&A and summarization in ChatGPT. Therefore, we aim to employ large language models to help outage summarization. In this work, we aim to generate outage summaries with the following goals:

Usefulness. Usefulness measures whether the outage summary contains relevant and valuable information.

Readability. Readability measures whether the outage summary read fluently, especially considering the context information of the outage and the affected system.

Reducing TTS (time to summary). It is desirable to summarize outage in a short time because it helps improve the overall outage handling process, improve communnication, shorten the lifecycle of outage, and in turn, improve customers' satisfaction.

4 OUR PROPOSAL: OASIS

4.1 Overview

In this paper, we aim to automatically generate summaries for outages of cloud systems. However, outage summarization faces two challenges. The first challenge is determining which information on relevant incidents is helpful to outage summarization. Since the cloud system is complex and rapidly growing, it is not trivial to extract domain-specific terminologies of incidents. The second challenge is how to effectively generate human-readable outage summaries with complex cloud-related information.

To solve these challenges, we propose Oasis. The overview of Oasis is shown in Figure 3, which consists of the following two components. In the first component, *i.e.*, impact scope assessment, Oasis identifies relevant incidents via three types of linking to comprehensively assess the impact of the outage. In the second component, *i.e.*, summary generation, Oasis first performs domain-specific text processing to denoise and prioritize important information from the obtained relevant incidents, thus addressing the first challenge. Then Oasis employs a fine-tuned large language model, *i.e.*, GPT-3.x, to understand the incidents and generate a compact summary for the outage, thus addressing the second challenge.

4.2 Impact Scope Assessment

Assessing the impact scope of an outage is about comprehensively understanding different aspects of an outage such as the when, where, who, what, why, *etc.* As shown in Section 3, the impact of these aspects of the outage is collectively described by many relevant incidents. However, there is no simple and direct way to identify the set of relevant incidents, since incidents with the same underlying root cause can have different properties and spread across services. Meanwhile, if an OCE determines two incidents are highly relevant to each other, she can formally *link* the two incidents together, which is a feature provided by the IcM system. Linking an incident with relevant incidents reduce the effort of OCEs in many ways, for example, reducing the number of incidents that require manual examination, auto-resolving less severe incident if a more severe linked incident is resolved, *etc.*

Inspired by the process, Oasis assesses the impact scope of an outage by linking its relevant incidents. To completely link the relevant incidents of an outage, Oasis incorporates domain knowledge and historical linking patterns. Specifically, Oasis performs three types of incident linking: linking by rule, linking by historical lookup, and linking by prediction model.

Linking by rules. Automated incident linking is a capability in IcM that correlates and de-duplicates incidents to reduce alert storms and noise. Engineers can set up specific rules to create links between incidents upon various triggers, which represent the domain knowledge of the engineers. For example, an engineer can set up a rule to have incidents that are triggered by the same KPI anomaly be linked. These are structural incident links that can be directly queried from the IcM database. During the operation of cloud systems and the corresponding IcM system, a large number of historical incidents and rule-based links are persistently recorded. These data are a natural source of labeling for learning, which facilitates the following two types of linking.

Linking by historical lookup. We propose a heuristic lookup mechanism to utilize the historical links between incidents. The mechanism consists of an offline phase to memorize the historical linking pattern, and an online phase to apply the patterns to current incidents. In each phase, we use the field of component that is reported along with the incident. Components are fine-grained parts of cloud systems that are defined by engineers. In the offline phase, we build a component linking graph by summing up links between incidents, *i.e.*, if incident A and incident B are linked, then we link their component in the component graph. In the online phase, we check whether there are active incidents (incidents within a short time range) on the linked components.

Linking by prediction model. Another way to automatically discover the relationship between incidents is by employing deep learning techniques [8, 17]. It has the advantage of being highly automated and can be applied to a large number of incidents. Also, it can be applied to scenarios where new incident detection criteria are created and the engineers have not set up rules and historical links fail to apply because of the lack of historical data. We train a

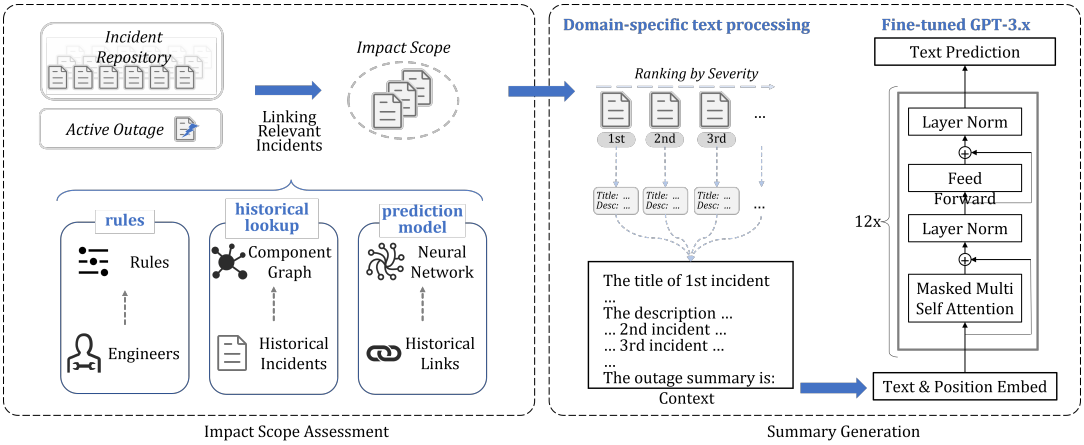


Figure 3: Overview of Oasis.

neural network to predict the link between incidents. The neural network takes the titles and descriptions of two incidents as input and outputs the similarities between the titles. If the similarity of two incidents is larger than a threshold, we determine the two incidents are linked. The neural network has been trained on pairs of incidents to learn the relationships between incident linking and incidents' titles and descriptions. In the training set, incident pairs that were actually linked by engineers are labeled positive and pairs that were not linked are labeled negative.

Put them together. Oasis periodically assesses relevant incidents to the outage by querying the information of incidents within a time window to the outage. We take advantage of three linking approaches: the rule-based linking has the highest confidence and interoperability; the historical lookup may find hidden dependencies; the prediction model can adapt to the rapid evolution of cloud systems. These three linking techniques complement each other and enhance our linking capabilities. To provide a complete overview of the impacted components, Oasis uses the aggregation of incidents linked by these three approaches. This serves as the foundation for creating the outage summary.

4.3 Summary Generation

After gathering relevant incidents of an outage, Oasis generates a summary of the outage based on the incidents' information. To overcome the challenge of noisy information, we fine-tune an LLM, *i.e.*, GPT-x, to summarize the relevant incidents.

GPT-x. Generative Pre-trained Transformer x (GPT-x) [3] is a large pretrained language model that can tackle a wide range of natural language processing (NLP) tasks. One typical usage scenario of GPT-x is text completion, where the model is given a block of text as *context* and generates text as the *completion* of the context. It has been explored to recommend the root causes of cloud incidents [2].

The GPT-x model is based on Transformers [33], which takes advantage of the attention mechanism to assign weights to different parts of the text. Thus it is suitable to summarize noisy information. There are different sizes (number of parameters) of GPT-x model. In our work, we implement Oasis with two parameter sizes: *GPT-3-Curie* and *GPT-3.5-DaVinci* (see Section 5.4 for more details).

Domain-specific text processing. In this step, we process the structural incident information into a paragraph of text so that the GPT-3 model can take it as input, *i.e.*, context. Inspired by the findings from Section 3, we propose to process incidents in a way that the high-severity incidents and textual descriptions are emphasized. First, we sort the relevant incidents by their severity so that the incidents with higher severity precede the ones with lower severity. Then we transform the incidents into a piece of text in the following way: for the incident with sorted order i, the text is [*The title of* i^{th} *incident is .... The description of* i^{th} *incident is .... The service of* i^{th} *incidents is .....*] Finally, we append an instruction to the end of the text to hint the GPT-3 model to generate a summary: [*The outage summary is:*].

Fine-tuning. The GPT-3 model was trained on a general corpus that allows the model the learn various knowledge like linguistics, common knowledge, factual knowledge, basic logical inference ability, *etc.* To achieve better summary generation, we use our IcM-specific data to fine-tune the GPT-3 model so that it learns the domain knowledge of the applied cloud system and incidents. Moreover, the training samples presented to the model teach it to emphasize the aspects that are of interest to OCEs, thereby improving its ability to summarize information from noisy sources. The data we use to fine-tune the model is in the same form of summary generation, *i.e.*, for each outage, we provide the relevant incidents as context and the outage summary written by engineers as the desired completion.

4.4 System Implementation

We have deployed Oasis as an aid to the IcM system of Microsoft. We will introduce the integration of Oasis with the IcM workflow and the underlying implementation details.

The implementation of Oasis in production consists of four parts, as shown in Figure 4. The Oasis backend periodically queries the local database to get active incidents within the time window, as well as rule-based and historical-lookup links of all current outages

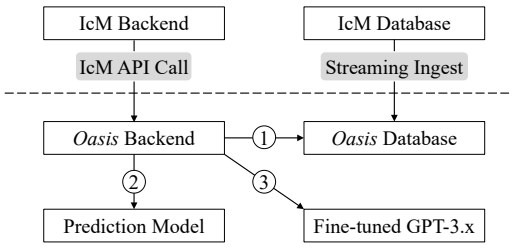


Figure 4: Oasis in production. The upper part is the IcM system. The lower part is the architecture of Oasis.

①. On receiving the API call initiated by the IcM Backend querying a specific outage, the Oasis backend applies the prediction model to determine what other active incidents should be linked to the outage ②. After that, it performs domain-specific text processing for the outage's relevant incidents and feeds the processed text to the fine-tuned GPT-3.x model ③. Finally, the backend returns the summary generated by GPT-3.x to the IcM Backend.

The local database of Oasis is ingested from the IcM database in a streaming manner. Compared to batch ingestion, streaming ingestion is more smooth in resource utilization. The prediction model has already been trained in ML-dedicated servers and exported as a binary file to minimize the operation effort of Oasis in production.

5 OASIS EVALUATION: EMPIRICAL STUDY DESIGN

To assess the effectiveness of OASIS, we investigate the following:

- **RQ4**: Is Oasis effective at summarizing outages?
  Generating the summary of outages is the main task of Oasis. We are interested in the ability of Oasis to generate a reasonable outage summary with automatic impact scope assessment.
- **RQ5**: Is Oasis effective at proposing outage titles?
  The title of an outage is a short piece of text stating the problem that is happening. Proposing outage titles also demonstrates the ability of Oasis to understand and summarize the outage.
- **RQ6**: Does Oasis get better at summarizing outages if the outage title is given?
  We explore whether Oasis can produce more effective summaries of outages if it has access to the title written by OCEs, which is typically written before the summary in practical settings.
- **RQ7**: What is the time efficiency of Oasis?
  Since Oasis needs to work in the production environment, it is important for Oasis to summarize outages efficiently.

5.1 Study data

In the study, we applied Oasis to the same 18 cloud systems and the same time range (3 years) in Section 3.1 to evaluate the effectiveness of Oasis. In particular, we split the data in chronological order using a 7:1:2 ratio for the training (fine-tuning), validation, and test sets, respectively. Each data point, representing an outage, is presented as a *context-completion* pair. The *context* consists of the processed text from relevant incidents linked by impact scope assessment. The *completion*, on the other hand, is provided as the summary of the outage written by OCEs.

5.2 Compared approaches

To better answer the RQ 4 to 7, we compare the performance of Oasis with some baseline approaches. We formulate the task as a text generation problem, therefore we compare with 3 methods that have been proven capable of generating summarization. In answering each research question, we provide the same context (information of relevant incidents) to the baselines and to the GPT-3 model of Oasis.

- Joint incident summary (**Rule-based**): A straightforward rule-based method that concatenates all the information of incidents. This method imitates the behavior that OCEs read through relevant incidents when handling outages.
- Information retrieval (**IR**): NNGen [24] leverages bag-of-words embedding and nearest neighbor to retrieve summaries from similar history outages.
- **GPT-2**: Generative Pre-training Transformer 2 (GPT-2) is a language model that is trained to generate coherent text. We use GPT-2 with 117M parameters.

5.3 Metrics

Following the existing work [1, 2, 15], we use the BLEU-4, ROUGE-L, and METEOR to evaluate Oasis and its baselines in terms of readability. The BLEU-4 compares the matching of n-grams between generated text and the ground truth. The ROUGE-L is widely used in Machine Translation evaluation, which measures the overlap of the longest sequence between hypothesis and reference. The METEOR calculates the harmonic mean of unigram precision and recall with consideration of stemming and synonym matching.

Specifically, we get five candidate generated texts from each model, except for the joint incident summary which can only give one piece of generation. To better evaluate the quality of generation models, we calculate the Top1 metrics using the first generated text, and the Top5 metrics using the best of five generated text.

We also measure the running time of each approach. Specifically, we record the overall time needed to train/fine-tune the model, and the average time spent on generating a summary for an outage.

To further evaluate usefulness and readability, we conduct a human evaluation in Section 7.

5.4 Experiment environment

Generation model. We implement Oasis with two GPT-3 variants, *i.e.*, Curie and DaVinci:
Curie (GPT-3) is a fast GPT-3 model with 6.7 billion parameters, which was pre-trained on a natural language corpus.
DaVinci (GPT-3.5) is a large GPT-3 model with 175 billion parameters, which was pre-trained on both text and code.
We fine-tune these generation models using the training and validation set from Section 5.1.

Experiment environment. We implement all training with one NVIDIA GeForce A100 GPU, PyTorch 1.11, and CUDA toolkit 11.3.1.

Implementation of baselines. Baselines are implemented using Python 3.8 and scikit-learn 1.0.2. The number of GPT-2's training epochs is 20. The temperature of GPT-2 is 0.7, which is recommended by a previous study [2].

Table 1: Effectiveness of models at summarizing outages

Model	BLEU-4		ROUGE-L		METEOR	
	Top1	Top5	Top1	Top5	Top1	Top5
IR	0.042	0.051	0.144	0.180	0.115	0.146
Rule-based	0.277	NA	0.508	NA	0.629	NA
GPT-2	0.455	0.51	0.561	0.592	0.536	0.574
Curie	0.654	0.701	0.73	0.777	0.721	0.767
DaVinci	**0.664**	**0.706**	**0.742**	**0.782**	**0.734**	**0.776**

Table 2: Effectiveness of models at proposing outage titles

Model	BLEU-4		ROUGE-L		METEOR	
	Top1	Top5	Top1	Top5	Top1	Top5
IR	0.170	0.211	0.398	0.427	0.342	0.369
Rule-based	0.069	NA	0.211	NA	0.316	NA
GPT-2	0.624	0.673	0.672	0.694	0.639	0.688
Curie	0.826	0.88	0.88	0.9	0.84	0.894
DaVinci	**0.857**	**0.893**	**0.883**	**0.913**	**0.869**	**0.907**

6 OASIS EVALUATION: EMPIRICAL STUDY RESULTS

6.1 RQ4: Performance of Summary Generation

Table 1 lists the effectiveness of baselines and Oasis in summarizing outages. Oasis with DaVinci, the largest GPT-3 model, achieves the best metrics with both Top1 and Top5 summary generation. The advantage of DaVinci over Curie comes from the larger parameter size and the extra code corpus used in pretraining since some incidents contain API names or investigating code. However, the performance gain of DaVinci over Curie, the fastest GPT-3 model, is modest in both Top1 and Top5 generations.

We observe IR method is especially not suitable for outage summary generation. The major reason that the scores of IR are poor is that the rapid evolution of cloud systems has resulted in significant changes in the architecture of the systems over time, so similar outages are not likely to appear repeatedly, therefore historically useful summaries fail to depict the new outages. Although Rule-based summaries have a higher METEOR score than GPT-2, their BLEU-4 score is far lower than that of GPT-2. This is because the METEOR score takes into account the precision and recall of the unigram rather than subsequences, resulting in a more lenient evaluation of summaries. Rule-based summaries are often too long as the original incidents, which is not helpful for engineers to understand the context of outages.

6.2 RQ5: Performance of Title Generation

The title of an outage is a compact description of the outage. The example of an outage title and summary is shown in Section 3.2. The performance of baselines and Oasis at summarizing outages in the form of titles are listed in Table 2. By comparing Table 2 with Table 1, we achieve a higher generation score (0.826-0.857 BLEU-4) at generating titles for outages than generating the whole summary (0.654-0.664 BLEU-4). Similarly, GPT2, which is also a large Transformer based language model, scores higher in summarizing outages in the form of a title than the whole summary. The ROUGE-L and METEOR score exhibit the same trend for Oasis and GPT-2. The reason for this performance improvement lies in the nature of outage title and summary. The title of outage has a stronger pattern than that of outage summary. Firstly, a large portion of outage titles starts with "Outage for". This pattern is easy for LLM to learn, so the titles generated by LLM tend to have more overlapping words, and consequently, higher scores. Secondly, the words used in titles are usually either in a dictionary (*e.g.*, the "Triage" in the example title), or have been mentioned in incidents (*e.g.*, the "Email Service" in the example title).

Another observation is that title generation is the only task where IR outperforms the Rule-based method. Since the Rule-based method performs simple concatenation, the generated title is long and contains unnecessary words, thus resulting in lower scores, while IR method retrieves titles from historical outages which conforms better with the pattern of outage titles in general. Oasis achieves significantly high scores, with considerable improvement over baselines (at least 30.0% of BLEU-4, 30.9% of ROUGE-L, 31.4% of METEOR), indicating that applying Oasis to production outages title generation is very promising.

6.3 RQ6: Performance of Summary Generation Given Title

We evaluate the performance of the outage summary generation when the title of the outage is given. Remember that we provide the information of relevant incidents to these methods as context. In this experiment, we include the title written by OCEs as part of the context. More specifically, we set the information of incidents, followed by the title of the outage.

The results of this experiment are listed in Table 3. Surprisingly, the performance of Oasis, GPT-2, and Rule-based degrade slightly, except IR. The relative order of methods in Table 3 keeps the same as Table 1, for their tasks are very similar. The performance degradation is because the title of the outage is not a grammatically-complete sentence.

6.4 RQ7: Efficiency Comparison

Table 4 lists the fine-tuning (if any) and summary generation times for each model. The fine-tuning time reflects the total time spent on all outages from the training set, while the summary generation time is the average time taken to generate a summary for one outage. Despite the differences in parameter size, all models have a very small generation time. The fine-tuning time for LLMs (GPT-2, Curie, DaVinci) increases as the parameter size increases, although not linearly. Please note that Oasis only needs to be fine-tuned once on historical outages and incidents and can then be used for outage summary generation. In other words, the time to generate an outage summary of Oasis is only 13.3 or 39.6 ($\times 10^{-5}$) time units, which is at least 251.2 times faster than the median of manual summarization. In conclusion, the fine-tuning time for Oasis is reasonable, and its short generation time of summary demonstrates its practicality in cloud systems.

Table 3: Effectiveness of models at generating outage summaries given outage titles

Model	BLEU-4		ROUGE-L		METEOR	
	Top1	Top5	Top1	Top5	Top1	Top5
IR	0.037	0.055	0.156	0.189	0.109	0.142
Rule-based	0.247	NA	0.505	NA	0.614	NA
GPT-2	0.428	0.504	0.548	0.59	0.515	0.569
Curie	0.65	0.697	0.729	0.776	0.719	0.764
DaVinci	**0.652**	**0.699**	**0.734**	**0.779**	**0.724**	**0.77**

Table 4: Average time cost of models

Time	Rule	IR	GPT-2	Curie	DaVinci
Fine-tuning ($10^{-1}\times$time unit)	NA	NA	3.0	3.4	8.7
Generation ($10^{-5}\times$ time unit)	2.8	13.9	11.1	13.3	39.6

7 OASIS PRELIMINARY HUMAN EVALUATION

7.1 Methodology

Summarizing an outage is a challenging task that requires a deep understanding of both the service and the specific outage, as well as a comprehensive knowledge of the relevant context and domain. To ensure the accuracy of the generated summaries, we ask the owners of these outages to evaluate the generated summaries from RQ4. The process is done through Email and eventually, a total of 54 outage owners respond to our email and provide their evaluations.

For each outage, we present the generated summaries from all the methods, with the output of Oasis with Curie and DaVinci treated as independent summaries. This results in five summaries for each outage. We present the outage and the summaries in the following order: (1) first we give the IcM link to the outage so that the outage owner can better recall the details of the outage, (2) next we present the original outage summary written by OCEs and tell the outage owners this is the human-written summarization, (3) then we present the five outage summaries generated by models (Oasis-Curie, Oasis-DaVinci, GPT-2, Rule-based, IR), and we shuffle the order of these summaries to minimize the effect of default ordering. To ensure the objectivity of the evaluation and avoid the subjectivity and bias of scoring for each summary, we ask the outage owners to rank the summaries from 1 to 5, where 1 for the most useful and readable and 5 for the least useful and readable. *Useful* means that the summary contains useful and relevant information on the outage. *Readable* means the ease with which the summary can be understood, which may be characterized by clear and simple language, logical organization, grammatical correctness, *etc.* Besides ranking, we also ask outage owners to share their opinions and comments on model-generated summaries.

7.2 Results

Figure 5 shows the ranking of outage owners. In general, the results of human evaluation are in accordance with automatic metrics presented in Table 1. The outage owners report positive feedback

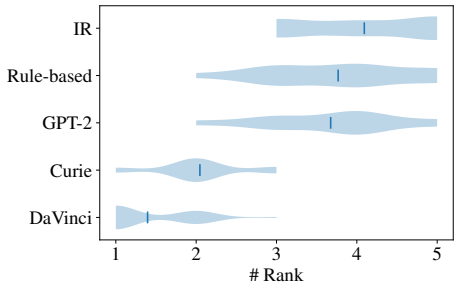


Figure 5: The ranking given by outage owners. Rank #1 means the most preferred summary.

regarding the readability and usefulness of Oasis. Notably, 32 out of 54 OCEs rank the summaries produced by Oasis-DaVinci as their top preference. To investigate whether the rankings of outage owners are consistent with each other, we conduct Friedman Test [12] at the significant level of 0.05. The null hypothesis is that there is no significant difference between the rankings of outage owners. The calculated p-value on our rank by outage owner is larger than the level of significance, which means that the outage owners basically conform to each other in evaluating the summaries.

More encouragingly, the majority of outage owners have a favorable attitude toward the practice of generating outage summaries using Oasis: ***"I absolutely believe in the ability of AI to assist with incident management and outage summaries."***

8 DISCUSSION

Case Study. As described in Section 4.4, Oasis serves as a supportive tool in the production IcM, *i.e.*, OCEs have the option to use or not use Oasis to generate a reference summary when handling outages. Since it is difficult to determine the extent to which Oasis contributes to the generated summary for outages where it is utilized, we randomly selected an outage from recent outages that were handled without the use of Oasis.

In this outage, a misconfiguration of the load balancer led to an overwhelming number of requests being directed to a single service endpoint. As a result, this endpoint was unable to function properly, causing the unified account API to fail. This failure cascaded to the downstream account APIs of the Cloud, Productivity, and Partner systems. As a result, the signup, ordering, and billing services of the Cloud and Productivity systems were affected. Below is the actual outage summary, written by an experienced OCE:

Outage Summary by OCE: The $\text{API}_{account}$ failed with HTTP 5xx errors (over α_1 fall failures) because of bad gateway errors to the endpoint_1. Due to this issue, commercial customers could not sign-up for System-Cloud or System-Productivity via endpoint_2 or endpoint_3, and perform management related actions on endpoint_4. Additionally, customers could not complete purchases within these ecosystems. Partner is also impacted.

Outage Summary by Oasis: The $\text{API}_{account}$ failed with HTTP 5xx errors (over α_1 fall failures) because of bad gateway errors to the endpoint_1. Due to this issue, commercial customers could not sign-up for System-Cloud or System-Productivity via endpoint_2 or endpoint_3, and perform management related actions on endpoint_4. Additionally, users were not able to access their billing accounts and invoices on System-Cloud portal. Approximately α_2 unique users were impacted.

To study the ability of Oasis in summarizing online outages, we triggered Oasis manually, limiting it to only knowing the information at the time of the outage. Oasis managed to find six relevant incidents, and the generated summary is presented above. (We indicate sentences that diverge from the OCE outage summary by underlining them with wavy lines) In the above summaries, endpoints 1-4 are URLs that serve the API calls. We notice that Oasis failed to identify System-Partner as impacted because the impact of System-Partner can only be determined by knowing the prefix of endpoint_4 refers to System-Partner. This knowledge is difficult to learn even after the LLM has been fine-tuned using incident and outage corpus. Despite this, we can see from the above example that Oasis is capable of generating reference summaries for outages. In this study, we adopt fine-tuning of the GPT-x models to generate outage summaries, which perform much better than prompt tuning. Because the outage summary is very domain-specific and fine-tuned models may capture the domain knowledge.

Threats to Validity. Threats to *internal validity* mainly lie in our implementation of Oasis and compared approaches. To reduce this threat, we implemented these approaches based on well-established frameworks, which have been described in Section 5.4. Additionally, two authors carefully examined the code and configurations. Threats to *external validity* mainly lie in the subjects used in our study. Our study and evaluation are conducted on 18 major cloud systems of Microsoft. Since the incidents and outages we used are only from Microsoft, modifications may be necessary when applying to other incident management systems. However, the cloud systems we used in our experiments include a variety of types, such as infrastructure, productivity, communication, game, search engine, *etc.* Moreover, these cloud systems serve millions of customers, thus having a certain degree of representativeness. In the future, we plan to extend our evaluations to include more cloud systems. Threats to the *construct validity* mainly lie in the evaluation metrics we adopted. Automated evaluation metrics (BLEU-4, ROUGE-L, and METEOR) may not fully reflect the readability and usefulness of the outage summary. To address this limitation, we will consider using additional metrics in the future to better measure these factors. Moreover, we reached out to the owners of outages to conduct a human evaluation, and the evaluation results are basically aligned with automated metrics.

9 RELATED WORK

Incident storm / outage handling. Handling outages (incident storms) in cloud systems has been widely studied in previous work [8, 11, 17, 27, 34]. A series of works perform incident linking to provide engineers with more relevant information. LiDAR [8] calculates both textual similarity and component similarity to determine whether two incidents should be linked. LinkCM [17] argues that linking customer incidents (reported by customers) with system incidents (reported by monitors) can lead to more efficient incident triage. The above works utilize neural networks to learn from historical linking patterns. GRLIA [11] employs graph embedding, with additional concerns about node closeness with KPI trend similarities. Another series of works focuses on alert reduction or prioritization [6, 40]. OAS [6] combines semantic and behavioral features of alerts to decide the groups of alerts and then correlate alerts within a time window. Zhao *et al.* [40] first calculate the textual and topological similarity of alerts to reduce the number of alerts. They then use DBSCAN to group similar alerts and selected the centroid alert of each cluster as the representative incident to show to engineers. Our impact scope assessment is similar to these approaches, and we also include domain-specific knowledge via rule-based incident linking.

Large Language Models (LLM) for Software Engineering. In recent years, the rise of LLM has brought new opportunities to the field of software engineering [2, 13, 29, 30, 37]. Mastropaolo *et al.* [30] studied the ability of fine-tuned T5 in the following tasks: automatic bug fixing, generation of assert statements in test methods, code summarization, and injection of code mutants. LANCE [29] uses fine-tuned T5 to automatically generate logging statements for Java methods. VulRepair [13] also fine-tune T5 on vulnerability repairs datasets to automatically propose vulnerability fixes. The above works fine-tune LLM on task-specific datasets. Zhang *et al.* [37] propose to use prompting for LLM to improve code version control. They further integrate k-shot learning to resolve code merge conflicts. GPT-3.x models are used to recommend root causes and mitigation steps to facilitate cloud incident management [2]. Different from previous studies, Oasis is the first work to leverage the capabilities of LLM in to summarize outages for cloud systems.

10 CONCLUSION

In this paper, we identify the problem of outage understanding in real-world cloud systems. Through our empirical study on 18 industrial cloud systems, we show that understanding outage is time-consuming and involves complex contexts. To improve the process of outage understanding, we present Oasis, the first framework to automatically assess impacts and summarize outages. Oasis incorporates an assessment of outage impact scope and a fine-tuned large language model, *i.e.*, GPT-3.x. Our experiments on 18 cloud systems within Microsoft demonstrate that Oasis outperforms baseline approaches. We also received feedback from outage owners, which further validates the effectiveness of Oasis.

ACKNOWLEDGMENTS

We thank our colleagues in the production teams for the valuable feedback and labeling. Pengxiang Jin, and Shenglin Zhang are supported by the National Natural Science Foundation of China (Grant No. 62272249, 62072264), the Natural Science Foundation of Tianjin (Grant No. 21JCQNJC00180). Federica Sarro is supported by the European Research Council (Grant no. 741278).

REFERENCES

[1] Toufique Ahmed and Premkumar Devanbu. 2022. Multilingual training for software engineering. In *Proceedings of the 44th International Conference on Software Engineering*. 1443–1455.

[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 *ICSE 2023*.

[3] 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. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[4] 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 *IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 111–120.

[5] Junjie Chen, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. Continuous incident triage for large-scale online service systems. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 364–375.

[6] Jia Chen, Peng Wang, and Wei Wang. 2022. Online Summarizing Alerts through Semantic and Behavior Information. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. 1646–1657. https://doi.org/10.1145/3510003.3510055

[7] Junjie Chen, Shu Zhang, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Yu Kang, Feng Gao, Zhangwei Xu, Yingnong Dang, et al. 2020. How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 373–384.

[8] Yujun Chen, Xian Yang, Hang Dong, Xiaoting He, Hongyu Zhang, Qingwei Lin, Junjie Chen, Pu Zhao, Yu Kang, Feng Gao, et al. 2020. Identifying linked incidents in large-scale online service systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 304–314.

[9] Yujun Chen, Xian Yang, Qingwei Lin, Hongyu Zhang, Feng Gao, Zhangwei Xu, Yingnong Dang, Dongmei Zhang, Hang Dong, Yong Xu, et al. 2019. Outage prediction and diagnosis for cloud service systems. In *The world wide web conference*. 2659–2665.

[10] Zhuangbin Chen, Yu Kang, Liqun Li, Xu Zhang, Hongyu Zhang, Hui Xu, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, et al. 2020. Towards intelligent incident management: why we need it and how we make it. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1487–1497.

[11] Zhuangbin Chen, Jinyang Liu, Yuxin Su, Hongyu Zhang, Xuemin Wen, Xiao Ling, Yongqiang Yang, and Michael R Lyu. 2021. Graph-based incident aggregation for large-scale online service systems. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 430–442.

[12] Rob Eisinga, Tom Heskes, Ben Pelzer, and Manfred Te Grotenhuis. 2017. Exact p-values for pairwise comparison of Friedman rank sums, with application to comparing classifiers. *BMC bioinformatics* 18, 1 (2017), 1–18.

[13] Michael Fu, Chakkrit Tantithamthavorn, Trung Le, Van Nguyen, and Dinh Phung. 2022. VulRepair: a T5-based automated software vulnerability repair. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 935–947.

[14] 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 *Proceedings of the 13th Symposium on Cloud Computing*. 126–141.

[15] David Gros, Hariharan Sezhiyan, Prem Devanbu, and Zhou Yu. 2020. Code to comment" translation" data, metrics, baselining & evaluation. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 746–757.

[16] Jiazhen Gu, Chuan Luo, Si Qin, Bo Qiao, Qingwei Lin, Hongyu Zhang, Ze Li, Yingnong Dang, Shaowei Cai, Wei Wu, et al. 2020. Efficient incident identification from multi-dimensional issue reports via meta-heuristic search. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 292–303.

[17] Jiazhen Gu, Jiaqi Wen, Zijian Wang, Pu Zhao, Chuan Luo, Yu Kang, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, et al. 2020. Efficient customer incident triage via linking with system incidents. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1296–1307.

[18] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E. Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, and Thomas Moscibroda. 2020. Protean: VM Allocation Service at Scale. In *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020*. USENIX Association, 845–861. https://www.usenix.org/conference/osdi20/presentation/hadary

[19] Jiajun Jiang, Weihai Lu, Junjie Chen, Qingwei Lin, Pu Zhao, Yu Kang, Hongyu Zhang, Yingfei Xiong, Feng Gao, Zhangwei Xu, et al. 2020. How to mitigate the incident? an effective troubleshooting guide recommendation technique for online service systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1410–1420.

[20] Liqun Li, Xu Zhang, Shilin He, Yu Kang, Hongyu Zhang, Minghua Ma, Yingnong Dang, Zhangwei Xu, Saravan Rajmohan, Qingwei Lin, et al. 2023. CONAN: Diagnosing Batch Failures for Cloud Systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 138–149.

[21] Liqun Li, Xu Zhang, Xin Zhao, Hongyu Zhang, Yu Kang, Pu Zhao, Bo Qiao, Shilin He, Pochian Lee, Jeffrey Sun, et al. 2021. Fighting the Fog of War: Automated Incident Detection for Cloud Systems. In *USENIX Annual Technical Conference*. 131–146.

[22] Qingwei Lin, Tianci Li, Pu Zhao, Yudong Liu, Minghua Ma, Lingling Zheng, Murali Chintalapati, Bo Liu, Paul Wang, Hongyu Zhang, et al. 2023. EDITS: An Easy-to-difficult Training Strategy for Cloud Failure Prediction. In *Companion Proceedings of the ACM Web Conference 2023*. 371–375.

[23] Haopeng Liu, Shan Lu, Madan Musuvathi, and Suman Nath. 2019. What bugs cause production cloud incidents?. In *Proceedings of the Workshop on Hot Topics in Operating Systems*. 155–162.

[24] Zhongxin Liu, Xin Xia, Ahmed E Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. 2018. Neural-machine-translation-based commit message generation: how far are we?. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 373–384.

[25] Minghua Ma, Yudong Liu, Yuang Tong, Haozhe Li, Pu Zhao, Yong Xu, Hongyu Zhang, Shilin He, Lu Wang, Yingnong Dang, Saravanakumar Rajmohan, and Qingwei Lin. 2022. An Empirical Investigation of Missing Data Handling in Cloud Node Failure Prediction. In *Proceedings of the European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. 1453 – 1464.

[26] Minghua Ma, Zheng Yin, Shenglin Zhang, Sheng Wang, Christopher Zheng, Xinhao Jiang, Hanwen Hu, Cheng Luo, Yilin Li, Nengjun Qiu, et al. 2020. Diagnosing root causes of intermittent slow queries in cloud databases. *Proceedings of the VLDB Endowment* 13, 8 (2020), 1176–1189.

[27] Minghua Ma, Shenglin Zhang, Junjie Chen, Jim Xu, Haozhe Li, Yongliang Lin, Xiaohui Nie, Bo Zhou, Yong Wang, and Dan Pei. 2021. Jump-Starting Multivariate Time Series Anomaly Detection for Online Service Systems. In *Proceedings of the Annul Technical Conference (ATC)*. USENIX, 413–426.

[28] Minghua Ma, Shenglin Zhang, Dan Pei, Xin Huang, and Hongwei Dai. 2018. Robust and rapid adaption for concept drift in software system anomaly detection. In *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 13–24.

[29] Antonio Mastropaolo, Luca Pascarella, and Gabriele Bavota. 2022. Using Deep Learning to Generate Complete Log Statements. In *Proceedings of the 44th International Conference on Software Engineering (ICSE '22)*. 2279–2290.

[30] Antonio Mastropaolo, Simone Scalabrino, Nathan Cooper, David Nader Palacio, Denys Poshyvanyk, Rocco Oliveto, and Gabriele Bavota. 2021. Studying the usage of text-to-text transfer transformer to support code-related tasks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 336–347.

[31] Xiaoting Qin, Minghua Ma, Yuheng Zhao, Jue Zhang, Chao Du, Yudong Liu, Anjaly Parayil, Chetan Bansal, Saravan Rajmohan, Íñigo Goiri, et al. 2023. How Different are the Cloud Workloads? Characterizing Large-Scale Private and Public Cloud Workloads. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 522–530.

[32] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[34] Yaohui Wang, Guozheng Li, Zijian Wang, Yu Kang, Yangfan Zhou, Hongyu Zhang, Feng Gao, Jeffrey Sun, Li Yang, Pochian Lee, et al. 2021. Fast outage analysis of large-scale production clouds with service correlation mining. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 885–896.

[35] Xiaohan Yan, Ken Hsieh, Yasitha Liyanage, Minghua Ma, Murali Chintalapati, Qingwei Lin, Yingnong Dang, and Dongmei Zhang. 2023. Aegis: Attribution of Control Plane Change Impact across Layers and Components for Cloud Systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 222–233.

[36] Zhengran Zeng, Yuqun Zhang, Yong Xu, Minghua Ma, Bo Qiao, Wentao Zou, Qingjun Chen, Meng Zhang, Xu Zhang, Hongyu Zhang, et al. 2023. TraceArk: Towards Actionable Performance Anomaly Alerting for Online Service Systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 258–269.

[37] Jialu Zhang, Todd Mytkowicz, Mike Kaufman, Ruzica Piskac, and Shuvendu K Lahiri. 2022. Using pre-trained language models to resolve textual and semantic

merge conflicts (experience paper). In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 77–88.

[38] Shenglin Zhang, Pengxiang Jin, Zihan Lin, Yongqian Sun, Bicheng Zhang, Sibo Xia, Zhengdan Li, Zhenyu Zhong, Minghua Ma, Wa Jin, et al. 2023. Robust Failure Diagnosis of Microservice System through Multimodal Data. *arXiv preprint arXiv:2302.10512* (2023).

[39] Chenyu Zhao, Minghua Ma, Zhenyu Zhong, Shenglin Zhang, Zhiyuan Tan, Xiao Xiong, LuLu Yu, Jiayi Feng, Yongqian Sun, Yuzhi Zhang, et al. 2023. Robust Multimodal Failure Detection for Microservice Systems. In *Proceedings of the SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM.

[40] Nengwen Zhao, Junjie Chen, Xiao Peng, Honglin Wang, Xinya Wu, Yuanzong Zhang, Zikai Chen, Xiangzhong Zheng, Xiaohui Nie, Gang Wang, et al. 2020. Understanding and handling alert storm for online service systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 162–171.

[41] Zhenyu Zhong, Qiliang Fan, Jiacheng Zhang, Minghua Ma, Shenglin Zhang, Yongqian Sun, Qingwei Lin, Yuzhi Zhang, and Dan Pei. 2023. A Survey of Time Series Anomaly Detection Methods in the AIOps Domain. *arXiv preprint arXiv:2308.00393* (2023).