Triangle: Empowering Incident Triage with Multi-LLM-Agents

ZHAOYANG YU, Tsinghua University, China
MINGHUA MA[*], Microsoft, USA
XIAOYU FENG, Nankai University, China
RUOMENG DING, Georgia Tech, USA
CHAOYUN ZHANG, Microsoft, China
ZE LI, Microsoft, USA
MURALI CHINTALAPATI, Microsoft, USA
XUCHAO ZHANG, Microsoft, USA
RUJIA WANG, Microsoft, USA
CHETAN BANSAL, Microsoft, USA
SARAVAN RAJMOHAN, Microsoft, USA
QINGWEI LIN, Microsoft, China
SHENGLIN ZHANG, Nankai University, China
CHANGHUA PEI, Chinese Academy of Sciences, China
DAN PEI, Tsinghua University, China

As cloud service systems grow in scale and complexity, incidents that indicate unplanned interruptions and outages become unavoidable. Rapid and accurate triage of these incidents to the appropriate responsible teams is crucial to maintain service reliability and prevent significant financial losses. However, existing incident triage methods relying on manual operations and predefined rules often struggle with efficiency and accuracy due to the heterogeneity of incident data and the dynamic nature of domain knowledge across multiple teams. To solve these issues, we propose Triangle, an end-to-end incident triage system based on a Multi-LLM-Agent framework. Triangle leverages a semantic distillation mechanism to tackle the issue of semantic heterogeneity in incident data, enhancing the accuracy of incident triage. Additionally, we introduce multi-role agents and a negotiation mechanism to emulate human engineers' workflows, effectively handling decentralized and dynamic domain knowledge from multiple teams. Furthermore, our system incorporates an automated troubleshooting information collection and mitigation mechanism, reducing the reliance on human labor and enabling fully automated end-to-end incident triage. Extensive experiments conducted on real-world cloud production environment demonstrate that Triangle significantly improves the accuracy of incident triage more than 20% and reduces the time to engage about 3 time units per incident compared to

[*]Minghua Ma is the corresponding author. Email: minghuama@microsoft.com

Authors' addresses: Zhaoyang Yu, Tsinghua University, Beijing, China; Minghua Ma, Microsoft, Redmond, USA; Xiaoyu Feng, Nankai University, Tianjin, China; Ruomeng Ding, Georgia Tech, Atlanta, USA; Chaoyun Zhang, Microsoft, Beijing, China; Ze Li, Microsoft, Redmond, USA; Murali Chintalapati, Microsoft, Redmond, USA; Xuchao Zhang, Microsoft, Redmond, USA; Rujia Wang, Microsoft, Redmond, USA; Chetan Bansal, Microsoft, Redmond, USA; Saravan Rajmohan, Microsoft, Redmond, USA; Qingwei Lin, Microsoft, Beijing, China; Shenglin Zhang, Nankai University, Tianjin, China; Changhua Pei, Chinese Academy of Sciences, Beijing, China; Dan Pei, Tsinghua University, Beijing, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2025 Association for Computing Machinery.
XXXX-XXXX/2025/2-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

state-of-the-art methods. Triangle has been successfully deployed in a system with tens of millions of users at a leading global technology company.

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

Additional Key Words and Phrases: Key Words

ACM Reference Format:
Zhaoyang Yu, Minghua Ma, Xiaoyu Feng, Ruomeng Ding, Ze Li, Murali Chintalapati, Xuchao Zhang, Rujia Wang, Chetan Bansal, Saravan Rajmohan, Qingwei Lin, Shenglin Zhang, Changhua Pei, Dan Pei. 2025. Triangle: Empowering Incident Triage with Multi-LLM-Agents. 1, 1 (February 2025), 21 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As cloud service systems continue to scale and become more complex, incidents within these systems are inevitable [1, 5, 6, 27–29]. These incidents often indicate unplanned interruptions and even system outages. Thus, when an incident occurs, it is crucial for the responsible team to address it promptly to prevent further failures and avoid significant financial losses [29]. However, in today's large-scale cloud service systems, even a single system may involve many teams with different functions. Therefore, incidents should be assigned to the appropriate responsible team, a process known as **Incident Triage** [5, 19]. If an incident is assigned to an unsuitable team, the incident usually can not be solved properly and should be reassigned based on the feedback from that team until the correct team is identified. A bad incident triage can significantly extend the time to engage (TTE), increasing the system's risk exposure. Thus, rapid and accurate incident triage is critical for reducing recovery time and ensuring service quality.

Traditional incident triage processes typically rely on manual operations combined with pre-defined rules, where engineers utilize various tools to further investigate issues associated with the incident. This often involves ad hoc meetings across multiple relevant teams, consuming a significant amount of human resources and time, and making rapid fault resolution difficult.

To automate the incident triage process, we can draw a direct parallel to the bug triage problem. Recent research in academia has extensively explored bug triage [8, 12, 14, 18, 21, 24], typically involving a unified model pre-trained on historical datasets to assign bugs to various teams through a one-time classification. This approach, however, has limitations and does not meet the performance requirements for incident triage. The core reason for this issue is that, unlike bug triage, the original incident information is often either automatically generated by system components following certain rules or manually submitted by users, resulting in a lack of rich information [5]. Therefore, directly classifying incidents based on the original data makes it difficult to achieve the accuracy needed for practical applications.

As shown in Figure 1, a customer reported incident highlights a sign-in issue encountered by a service client on Mac devices. Initially, the incident is manually triaged to Team A, who attempt to resolve the issue through various steps, but are ultimately unable to fix it. Consequently, the incident is transferred to Team B based on their domain knowledge. Team B follows their troubleshooting guide (TSG), conducts further examinations, but still does not resolve the problem, leading them to triage the incident to Team C based on the TSG. Team C then uses their specialized tools to collect domain-specific logs, identify the root cause, and resolve the incident.

This process, which involves three teams and many engineers, can be both time-consuming and complex. It is not feasible to directly identify Team C as the right team without the thorough examinations performed by Teams A and B. To address this, our core idea is to replace the teams

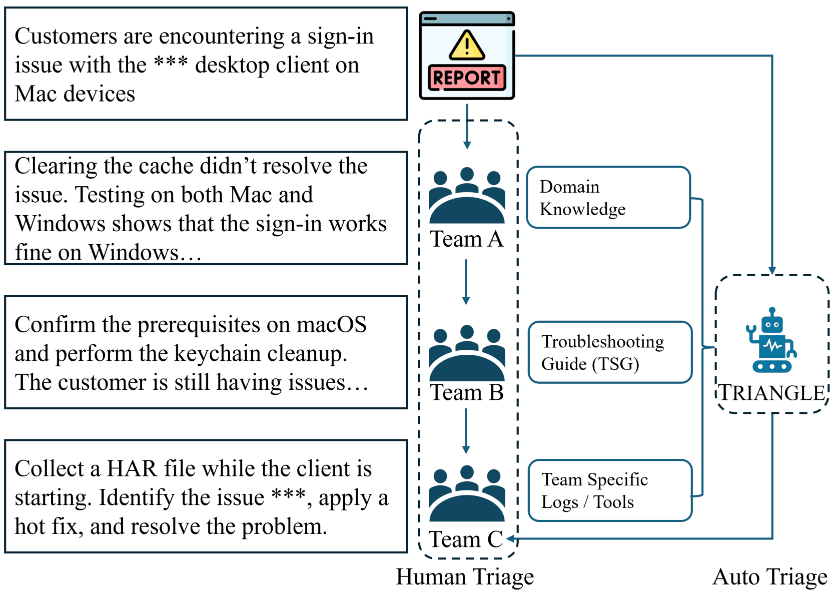

Customers are encountering a sign-in issue with the *** desktop client on Mac devices
Clearing the cache didn't resolve the issue. Testing on both Mac and Windows shows that the sign-in works fine on Windows…
Confirm the prerequisites on macOS and perform the keychain cleanup. The customer is still having issues…
Collect a HAR file while the client is starting. Identify the issue ***, apply a hot fix, and resolve the problem.
REPORT
Team A
Team B
Team C
Domain Knowledge
Troubleshooting Guide (TSG)
Team Specific Logs / Tools
TRIANGLE
Human Triage
Auto Triage

Fig. 1. An illustration of current incident triage system

with an LLM-driven agent. This agent would assimilate domain knowledge and utilize existing tools to automate the triage process, streamlining and expediting incident resolution.

To develop an incident triage system that meets real-world requirements for efficiency and accuracy, we conducted an in-depth investigation into the incident management practices of a leading global technology company's cloud services. Based on our practical experience, we have identified three key challenges in achieving efficient and accurate incident triage:

- **Incident Semantic Heterogeneity:** In incident data, semantic expressions closely related to incident triage often exhibit heterogeneity. Semantic information relevant to incident triage is usually contained within several key phrases scattered throughout the text data of an incident. Since these key phrases are not generated from predefined templates, different incidents can have heterogeneous expressions. This situation leads to incidents with similar language patterns belonging to different teams, while those with vastly different patterns may belong to the same team.
- **Decentralized and Dynamic Domain Knowledge:** Triage of an incident is often difficult to determine based solely on the domain knowledge of a single team. Effective incident triage typically requires combining knowledge from multiple teams. Additionally, a team's responsibilities are constantly evolving, making the domain knowledge relevant to incident triage dynamic and ever-changing.
- **High Human Labor:** Due to the heavy reliance on domain knowledge for incident triage, injecting this knowledge often incurs significant human labor costs. This not only makes end-to-end automation of incident triage challenging but also significantly increases the time to engage.

To address these issues, we designed the Triangle system, an end-to-end incident triage system based on a Multi-LLM-Agent framework. We developed a semantic distillation mechanism, leveraging the strong semantic understanding capabilities of LLMs to effectively comprehend the semantics of incident data and teams' domain knowledge for triage, thereby addressing incident semantic heterogeneity and improving the accuracy of incident triage. Additionally, we emulated the workflow of human engineers in solving incident triage problems by innovatively designing multi-role agents and proposing an effective negotiation mechanism. This allows for the distributed and dynamic handling of multi-team domain knowledge, effectively mitigating the impact of decentralized and dynamic domain knowledge on triage performance. By utilizing the robust semantic understanding of LLMs, we designed an automated Team Information Enrichment mechanism throughout the entire incident triage process, enabling end-to-end triage even in scenarios requiring reassignment, without additional human labor costs.

We conducted extensive experiments with Triangle using incident triage data collected from a real-world production environment serving tens of millions of users. The results demonstrate that Triangle achieves efficient and accurate end-to-end incident triage. Compared to state-of-the-art methods, our model improves the average accuracy of incident triage by more than 20% and reduces the time to engage by about 3 time units [1] per incident. Our model has been successfully deployed in a system with tens of millions of users at a leading global technology company.

Our contributions are summarized as follows:

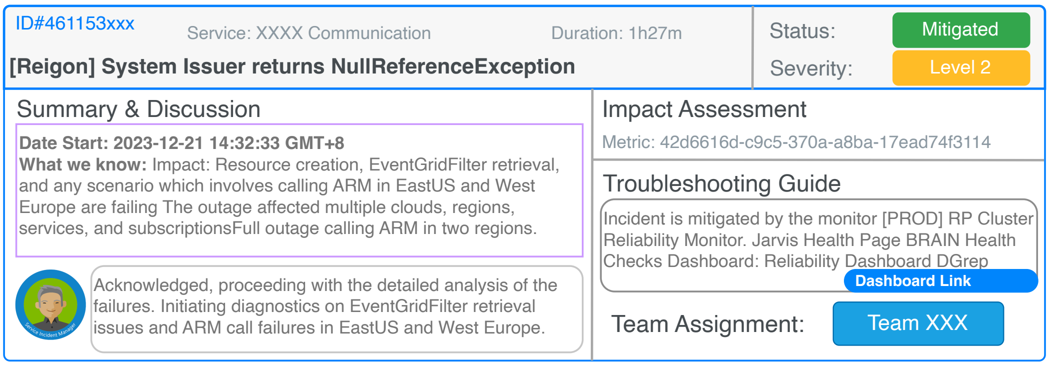
- To the best of our knowledge, Triangle is the first end-to-end incident triage system designed using a Multi-LLM-Agent framework to automate the incident triage process in large-scale cloud service environments. This approach improves both efficiency and accuracy, addressing the specific challenges inherent in incident triage scenarios.
- We propose a novel semantic distillation mechanism that leverages the powerful semantic understanding capabilities of LLMs to address the issue of incident semantic heterogeneity. This mechanism enhances the system's ability to accurately interpret incident data and domain knowledge, significantly improving triage accuracy.
- We design a multi-role agent framework with an effective negotiation mechanism that emulates the workflow of human engineers in incident triage tasks. This framework dynamically manages multi-team domain knowledge, effectively mitigating the challenges posed by decentralized and dynamic domain knowledge.
- We develop an automated Team Information Enrichment mechanism integrated throughout the entire triage process, which allows Triangle to perform end-to-end triage without incurring additional human labor costs, even in cases where reassignment is needed.
- We conduct extensive experiments with real-world incident triage data from a production environment, demonstrating that Triangle significantly outperforms state-of-the-art methods in terms of triage accuracy, reducing reassignments, and lowering time to engagement. The system has been successfully deployed in a large-scale environment serving tens of millions of users at a leading global technology company.

The rest of this paper is organized as follows. In Section 2, we introduce the background of the incident triage task and the multi-LLM-agent. In Section 3 we provide a detailed description of the design of our incident triage system, Triangle. Then, we conducted extensive experiments that demonstrate the outstanding performance of Triangle in Section 4. Moreover, We discuss our future work and the threats to validity. Then we review related work. Finally, we conclude our work in Section 7.

[1] A time unit is a metric used internally by the company to evaluate time to triage. There is a linear correspondence between a time unit and real-world time. However, due to privacy reasons, we cannot disclose the relationship between the time unit and actual physical time.

2 BACKGROUND

2.1 Incident Data


ID#461153xxx Service: XXXX Communication Duration: 1h27m

[Reigon] System Issuer returns NullReferenceException

Status: Mitigated

Severity: Level 2

Summary & Discussion

Date Start: 2023-12-21 14:32:33 GMT+8
What we know: Impact: Resource creation, EventGridFilter retrieval, and any scenario which involves calling ARM in EastUS and West Europe are failing The outage affected multiple clouds, regions, services, and subscriptionsFull outage calling ARM in two regions.

Acknowledged, proceeding with the detailed analysis of the failures. Initiating diagnostics on EventGridFilter retrieval issues and ARM call failures in EastUS and West Europe.

Impact Assessment

Metric: 42d6616d-c9c5-370a-a8ba-17ead74f3114

Troubleshooting Guide

Incident is mitigated by the monitor [PROD] RP Cluster Reliability Monitor. Jarvis Health Page BRAIN Health Checks Dashboard: Reliability Dashboard DGrep

Team Assignment: Team XXX

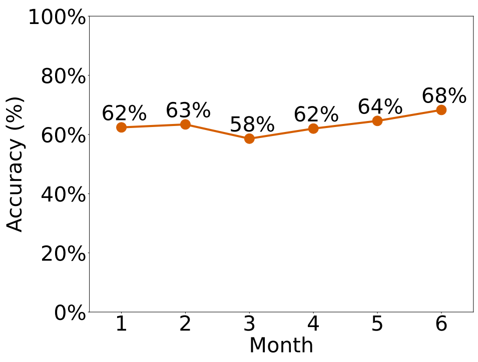
Fig. 2. An example of incident data after triage and manual processing

Incident data serves as a critical indicator of service quality in large-scale cloud service systems. An incident is often triggered by anomalies, faults, or unplanned interruptions within a system, which could lead to significant service degradation or outages. The rapid identification and resolution of incidents are essential to maintaining high service quality and minimizing financial losses. However, modern cloud environments are highly complex, often involving numerous interconnected components and teams with distinct functions. This complexity amplifies the challenges associated with understanding and processing incident data effectively.

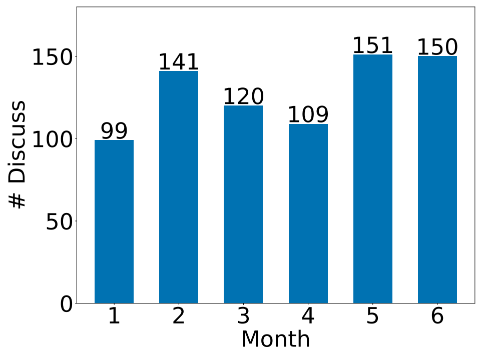
Incident data typically comprises system-generated logs, telemetry, alerts, and user reports, which can vary widely in their format, granularity, and relevance. Incident data are usually semi-structured data and natural language is the most important data type in incidents, as shown in Figure 2. The same incident pattern might correspond to different teams depending on subtle contextual factors, while different patterns might lead to the same team. This variability complicates the classification and assignment of incidents to the correct teams, impacting the effectiveness of incident triage systems.

2.2 Incident Triage

Incident triage is the process of quickly and accurately assigning incidents to the appropriate teams to ensure timely mitigation. In large-scale cloud service environments, a poorly managed triage can significantly increase Time to Engage (TTE), which is detrimental to both service quality and customer trust. Traditional approaches to incident triage rely heavily on predefined rules, human expertise, and manual operations. Engineers often need to investigate incident details using various tools, collaborate across multiple departments, and adjust triage decisions based on evolving insights and feedback. Even for human engineers, incident triage remains a very challenging task. The triage process involves extensive discussions across multiple teams, which results in a very long TTE in real-world industrial settings, sometimes even stretching to several weeks. We show a real-word caes in Figure 3. In this example, we calculated the average accuracy of manual incident triage in a system over six months and the average number of discussion meetings caused by incidents each month. From this, it is evident that an automated end-to-end incident triage system is an urgent need in the industry.



(a) Manual triage accuracy over time



(b) The number of discussions for incidents

Fig. 3. Empirical study of manual incident triage performance: (a) illustrates the accuracy of manual triage over a six-month period. (b) shows the average number of discussion meetings per month required for incident resolution.

Incident triage shares similarities with bug triage, both being multi-class, single-label classification problems where different teams represent different classes. However, unlike bug triage, incident data is often generated automatically by system components or manually by users, lacking the richness needed for one-time classification models to perform effectively. This lack of context-rich information necessitates more sophisticated methods that can dynamically incorporate evolving domain knowledge and adapt to changing system states. For current incident triage systems, the initial assignment is often based on static rules or simple heuristics. In practice, incidents frequently need multiple reassignments before they reach the right team, increasing Transfer Hop Counts, TTE and TTM. This reliance on manual routing and human experience further exacerbates inefficiencies and inconsistencies, underscoring the need for more automated and intelligent triage systems.

2.3 Multi-LLM-Agent

Large Language Models (LLMs) have emerged as a transformative technology in natural language processing (NLP), enabling a wide range of applications, from text generation to complex problem-solving. However, as these models grow in size and complexity, it becomes evident that a single model, no matter how advanced, may not be sufficient to tackle all tasks optimally. This realization has led to the development of multi-LLM-agent systems, where multiple LLMs are deployed in a coordinated manner to enhance performance, robustness, and adaptability.

A multi-LLM-agent system involves the orchestration of several LLMs, each potentially specialized in different tasks or aspects of a broader problem. By leveraging the strengths of various models, these systems aim to overcome individual limitations, such as biases, contextual misunderstandings, or performance degradation on specific tasks. The interaction between these models can be designed to mirror human teamwork, where different agents contribute their expertise to achieve a common goal.

The concept of multi-LLM-agent systems also addresses scalability and resource efficiency. In a rapidly evolving technological landscape, where new models and updates are frequently released, a multi-agent approach allows for more flexible integration of cutting-edge innovations. This modularity ensures that the system remains adaptive and can be optimized continuously without the need to overhaul the entire architecture.

In recent years, several architectures and frameworks have been proposed to implement multi-LLM-agent systems, each with its unique approach to coordination, communication, and decision-making. These systems have demonstrated potential in various domains, including automated reasoning, dialogue systems, and incident triage, where the synergy between different LLMs can lead to more accurate and reliable outcomes.

3 APPROACH

3.1 Overall Workflow

The overall workflow of Triangle is illustrated in Figure 4. The core process of Triangle can be divided into three phases: **Semantic Distillation, Team Candidate Selection, and Incident Assignment Loop**.

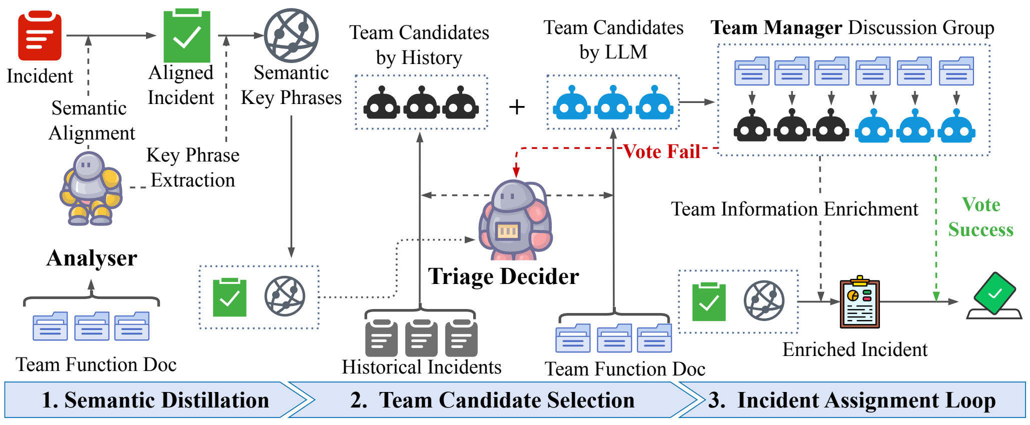


Fig. 4. The overall workflow of Triangle.

When a new incident arises in the system, it enters the Semantic Distillation phase. In this phase, we make Analyser Agent to extract semantic information from the incident that is closely related to triage. The detailed process is described in Section 3.2. The primary goal in this phase is to effectively extract key phrases semantically relevant to triage in order to reduce the impact of incident semantic heterogeneity.

In the Team Candidate Selection phase, the Triage Agent receives the semantically distilled incident information and raw incident. At this point, the Triage Agent attempts to select suitable team candidates for the current incident according to the historical incident data and team function documents. This process leverages the language understanding capabilities of LLMs to find the most semantically similar team. The Triage Agent combines two kinds of team candidates to build a discussion group.

In the Incident Assignment Loop, the core task is to find correct team for the incident by negotiation among Team Manager Agents. A intuitive approach is to vote. At this phase, the team candidates selected by the Triage Decider form a discussion group. Each Team Manager in the group has the ability to access external monitoring databases and provides enriched discussion information from their team's perspective for the current incident. Subsequently, multiple Team Managers vote to decide the incident assignment, as detailed in Section 3.3. If a suitable team is successfully selected through voting, this result is considered the final outcome of the triage. If the voting does not result in a consensus, the discussion results and the enriched incident information will be sent back to the Triage Decider for another round of team candidate selection. To prevent infinite loops, we have set the maximum number of reassignment cycles to 5, based on practical experience. If the reassignment cycles exceed this maximum, the TRIANGLE will directly assign the incident according to the last voting result and involve engineers in the process.

3.1.1 Multi-Role Agent Corporation. In the TRIANGLE system, there are three different agent roles with distinct functionalities: Analyser, Triage Decider, and Team Manager, as illustrated in Figure Figure 4. These three roles work collaboratively to efficiently perform the incident triage task, with each playing a unique part in the overall process.

Analyser Agent. The primary task of the Analyser is to preprocess incoming incidents and perform semantic distillation (see more details in Section 3.2). Specifically, when a new incident is introduced to the system, the Analyser first extracts basic information and preprocesses it by cleaning data, parsing text, identifying key entities according to the team function documents. Then, the Analyser conducts semantic distillation to extract the most important key phrases semantically from the incident data. This agent aims to reduce unnecessary noise while providing more accurate and concise input data for subsequent decision-making steps.

Triage Decider Agent. The core task of the Triage Decider is essentially singular: to identify suitable team candidates for the current incident. It is central to triage decision-making. Intuitively, if an incident is very similar to a historically occurred incident, it is likely that this incident belongs to the same team as the historical one. Similarly, if an incident closely matches the functional description of a certain team, it is highly probable that this incident is closely related to that team.

Algorithm 1 shows the pseudocode of the Triage Decider process. Following these two principles, during the initial assignment, the team triage first calculates the similarity with historical incidents. We use TF-IDF to vectorize all incident data and use cosine similarity to determine the top K most similar incidents from history and take their handling teams as candidates (Step 1). Considering time and computational cost at this stage, we do not use LLM. However, team candidates obtained solely through TF-IDF similarity calculations are not accurate. Therefore, the Triage Decider also uses LLM to directly match the current incident with the team's functional documents and retrieves N team candidates. Due to context limitations, we pre-compress and summarize the team's functional documents with LLM to meet the context length requirements of LLM (Step 2). Subsequently, the Triage Decider combine K candidates from historical incidents and N candidates from team function document. The Triage Decider utilizes LLM's language understanding capabilities to rank the current candidates based on semantic matching to the current incident (Step 3). At this stage, we can reduce the compression rate of the team's functional documents, since we only need to use $K + N$ teams. Finally, according to the rank of team candidates, Triage Decider select top M ($M < K + N$) teams to build a discussion group. Due to the limitation of LLM's context window length, in TRIANGLE, we set $K = N = M = 5$.

During the reassignment process, the Triage Decider performs a similar function; however, it does not use TF-IDF to select team candidates from historical incidents. Instead, it removes the incorrect team from the previous round and combines the enriched discussion information obtained in the last hop to select team candidates from the team's functional documents.

Team Manager Agent. In the system, each team has a corresponding Team Manager that can connect to the team's own monitoring database. The role of the Team Manager is primarily to determine whether the current incident falls within its area of responsibility and to provide reasons for either accepting or rejecting it. Team Manager can actively retrieves external monitoring information using tools and supplements additional information for the current incident from the team's perspective by Team Information Enrichment mechanism (see Section 3.3 for more details). Team Managers exchange their analysis results and reasons to ensure that the final

Algorithm 1 Triage Decider for Incident Team Assignment

```
Require: Incident $I$, Historical Incidents $\mathcal{H}$, Team Documents $\mathcal{D}$
Ensure: Candidate Teams $\mathcal{T}^*$
 1: // Step 1: Compute similarity with historical incidents
 2: $\mathbf{V} \leftarrow \text{TFIDF}(\mathcal{H})$    ▷ Vectorize $\mathcal{H}$ using TF-IDF
 3: $S(I, \mathcal{H}) \leftarrow \text{cosine}(\text{TFIDF}(I), \mathbf{V})$    ▷ Compute similarity
 4: $\mathcal{T}_1 \leftarrow \{\text{teams of top-}K(S(I, \mathcal{H}))\}$    ▷ Select top $K$ team candidates
 5: // Step 2: Candidate teams refinement using LLM
 6: $\mathcal{D}' \leftarrow \text{LLMcompress}(\mathcal{D})$    ▷ Compress team documents, high compress rate
 7: $\mathcal{T}_2 \leftarrow \{\text{top-}N(\text{LLMmatch}(I, \mathcal{D}'))\}$    ▷ Retrieve top $N$ team candidates
 8: // Step 3: Final ranking of candidates
 9: $\mathcal{D}'' \leftarrow \text{LLMcompress}(\mathcal{D})$    ▷ Compress team documents, low compress rate
10: $\mathcal{T}^* \leftarrow \text{LLMrank}(I, \mathcal{T}_1 + \mathcal{T}_2, \mathcal{D}'', \text{key phrases})$    ▷ Rank candidates
11: return $\mathcal{T}^*$
```

assignment decision is the most reasonable one. This negotiation is conducted in a structured and transparent manner to optimize incident handling efficiency and resource utilization. When a team's responsibilities change, engineers only need to update the team's functional documentation dynamically, without needing to retrain or fine-tune TRIANGLE. This design not only eliminates the need for manual intervention but also effectively addresses the problem of decentralized and dynamic domain knowledge.

This multi-agent design allows the TRIANGLE system to better simulate and execute the actual operational process of incident triage, improving efficiency and accuracy through automation and intelligence. Moreover, the independence and complementarity of each agent role provide flexibility and scalability for the system.

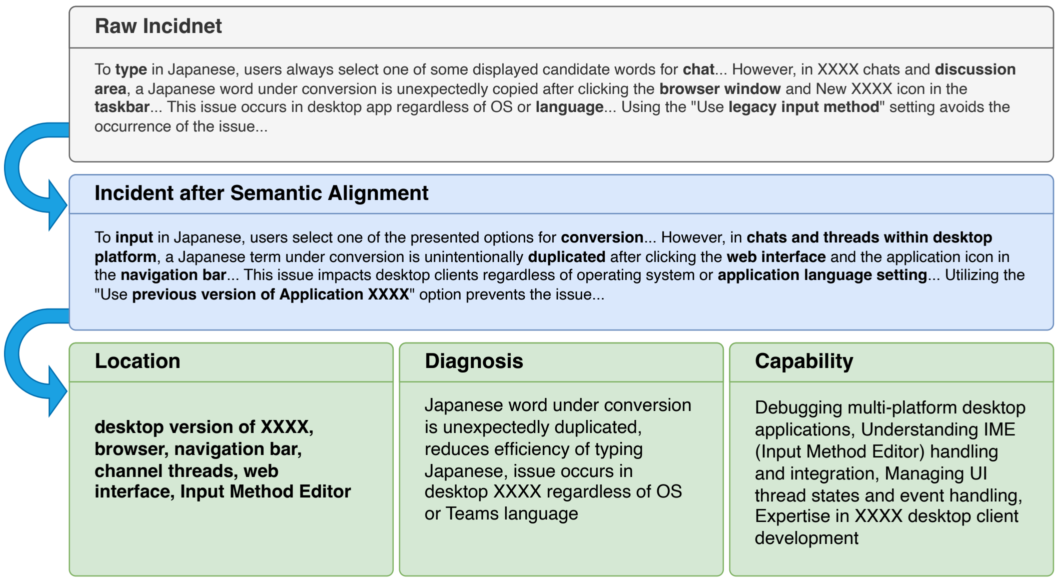
3.1.2 *Design of Negotiation Mechanism.*

To fully leverage the collaborative potential among multiple teams, we designed a voting-based multi-agent negotiation mechanism. In this process, the Team Manager is the primary participant. When the Triage Decider selects suitable team candidates, a discussion group is formed among these team candidates. First, each Team Manager invokes their own Team Information Enrichment capability to extract additional information from their respective monitoring data and supplement the incident data. The additional enriched discussion information provided by all teams is then aggregated and appended to the current incident. Subsequently, this enriched incident is sent to each Team Manager, allowing each to individually decide which team in the current discussion group is the best match for the incident. If a voting result surpasses more than half of the teams in the discussion group, the incident is assigned to that team. Otherwise, the negotiation fails, and the enriched incident is sent back to the Triage Decider to reselect team candidates.

The design of the voting-based negotiation mechanism is inspired by real-world practices of incident triage in the industry. In traditional manual triage, a temporary discussion group is typically formed by bringing different team candidates into a discussion to collaboratively determine a responsible team based on their diverse domain knowledge. TRIANGLE uses a Multi-Agent system to automate this process with LLM, effectively utilizing the domain knowledge of different teams and enhancing the accuracy of the triage. Experiments demonstrate that the introduction of the voting-based negotiation mechanism significantly improves the accuracy of both initial assignment and reassignment.

3.2 Semantic Distillation

Whether the incident is automatically generated by system or reported manually, the content that determines the outcome of incident triage constitutes only a small portion of the entire incident. Therefore, effectively extracting triage-relevant information is crucial for the effectiveness of incident triage. In traditional manual triage processes, the extraction of triage-related information is typically done in the minds of skilled engineers [27]. In deep learning based incident triage methods, the extraction of triage information is reflected in the forward propagation of the network, where triage-related information is continuously extracted as feature vectors in the hidden layers of the model until it is finally provided to the classifier [5]. The information extraction capability of such specially trained deep models heavily depends on the training data and is generally built on statistical significance, making it difficult to deeply understand the semantic information within an incident. Therefore, we propose **Semantic Distillation**, which utilizes statistical methods combined with the semantic understanding capabilities of LLMs to fully extract information related to triage from incidents for subsequent triage. This mechanism is designed as a core task of the Analyser agent. The Semantic Distillation process is mainly divided into two parts: semantic alignment and key phrase extraction. There is an example of the process on a real-world incident in Figure 5. In this example, through semantic alignment, we aligned this data with team functional documents without changing the semantic expression. Following this, we extracted three types of keywords using a Key Phrase Extraction mechanism. These three types of keywords are the core information related to incident triage. In the subsequent process, these key phrases will serve as the most critical information for triage, with the LLM focusing primarily on the key phrases, while the original incident data will serve as secondary reference material for the LLM.


Raw Incidnet

To **type** in Japanese, users always select one of some displayed candidate words for **chat**... However, in XXXX chats and **discussion area**, a Japanese word under conversion is unexpectedly copied after clicking the **browser window** and New XXXX icon in the **taskbar**... This issue occurs in desktop app regardless of OS or **language**... Using the "Use **legacy input method**" setting avoids the occurrence of the issue...

Incident after Semantic Alignment

To **input** in Japanese, users select one of the presented options for **conversion**... However, in **chats and threads within desktop platform**, a Japanese term under conversion is unintentionally **duplicated** after clicking the **web interface** and the application icon in the **navigation bar**... This issue impacts desktop clients regardless of operating system or **application language setting**... Utilizing the "Use **previous version of Application XXXX**" option prevents the issue...

Location

desktop version of XXXX, browser, navigation bar, channel threads, web interface, Input Method Editor

Diagnosis

Japanese word under conversion is unexpectedly duplicated, reduces efficiency of typing Japanese, issue occurs in desktop XXXX regardless of OS or Teams language

Capability

Debugging multi-platform desktop applications, Understanding IME (Input Method Editor) handling and integration, Managing UI thread states and event handling, Expertise in XXXX desktop client development

Fig. 5. An Example of the Semantic Alignment and Key Phrase Extraction in Semantic Distillation Mechanism.

Semantic Alignment. In Semantic Distillation, the first task is semantic alignment. The goal of semantic alignment is to align the information in incident data with the team function documents semantically. The team function documents are texts, maintained by engineers of each team, that

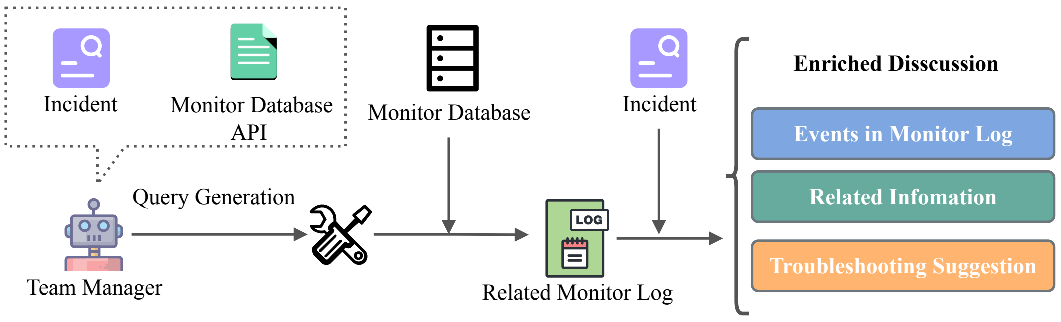


Fig. 6. Team Information Enrichment Mechanism.

describe their primary functions in natural language. When human engineers try to read and understand incident data, this process naturally occurs in the human brain, automatically building connections with the functions of different teams. Thus, our design mimics the operations of a human engineer. The Analyser agent traverses all team function documents and generates a vocabulary. Then, the Analyser provides this vocabulary and the current raw incident data to the LLM, prompting the model to attempt replacing the text in the current incident with words from the vocabulary without changing the semantic expression. This process is achieved through prompt engineering, without the need to train or fine-tune the model. The purpose of this is to make the incident semantically aligned with the team's function documents without changing the semantic information.

Key Phrase Extraction. After completing the alignment, we want the Analyser to extract key phrases from the incident data related to triage on a semantic level to achieve data distillation based on semantic information from the incident data. Due to the context length limitations of LLMs, we cannot provide all team function information to the LLM. Therefore, we combine statistical methods at this stage. The Analyser calculates the TF-IDF [3] value of each word in the incident data relative to team function documents. When a word has a high TF-IDF value, it usually indicates that the word frequently appears in the incident data but rarely in the team function documents. This likely means that the word is highly relevant to the current incident data but only related to a few teams' function descriptions. Therefore, this word has a higher probability of being relevant to incident triage. The Analyser uses the TF-IDF values as weights for each word, providing the weight information and the semantically aligned incident data to the LLM. Through prompt engineering, the LLM is prompted to summarize the incident text like an incident triage expert, extracting three types of key phrases: those related to the failure location, the symptoms of the failure, and the capabilities needed to resolve the incident. These three types of key phrases are the core factors that human engineers consider when performing incident triage.

After completing Semantic Distillation, the Analyser appends the extracted semantic key phrases to the original incident data and provides it to the subsequent Triage Decider. Experimental results show that Semantic Distillation effectively extracts triage-related key phrases from raw incident data, and these key phrases are highly aligned with the team's function descriptions in terms of semantic expression, significantly improving the accuracy of incident triage.

3.3 Team Information Enrichment

To address the issue of insufficient information in raw incident data, we propose a Team Information Enrichment mechanism. In raw incident data, the available information may not be sufficient to support accurate assignment. Consequently, during subsequent reassignments, engineers often take various steps—such as reviewing monitoring logs or consulting system documentation to provide additional information valuable for triage. We refer to this type of information as “enriched discussion”. The process of obtaining enriched discussions usually requires significant expert knowledge and human effort, leading to a notable increase in Time to Triage. The core task of the Team Information Enrichment mechanism is to automate this process using a LLM and integrate it into a multi-agent workflow to enable end-to-end incident triage in Triangle.

The Team Information Enrichment mechanism is unique to the Team Manager agent. The core idea behind our design is to enable the Team Manager to mimic the process of a human engineer in obtaining enriched discussions. Specifically, the Team Manager automatically queries the monitoring database for Monitor Logs directly related to the current incident and summarizes enriched discussions by analyzing the events in both the Monitor Logs and the incident itself. To facilitate the Team Manager’s access to the monitoring database, we provide each Team Manager with an interface for querying the monitoring database associated with its team. The Team Manager extracts the time range and component names from the incident and automatically generates and executes database queries.

It is worth noting that for manually reported incidents, the component names are often not automatically recorded by the system, resulting in the absence of certain key fields in the incident data. In such cases, the Team Manager infers the relevant data based on the database query interface documentation and the descriptions provided in the incident report. Therefore, even when some fields are missing in the incident data, the Team Manager can still generate appropriate queries to retrieve information closer to the system’s underlying details from the monitoring database.

When we obtain the Monitor Log related to the current incident, we do not use it directly as enriched discussion information. This is because the original Monitor Log is not only very large in volume but also contains a significant amount of redundant information that is irrelevant to incident triage. Directly adding it to the incident would consume a large number of tokens and cause the information related to triage to be overwhelmed by the redundant data. Therefore, after obtaining the Monitor Log, we use an LLM to infer the relationship between the events described in the current incident and the actual Monitor Log. The Monitor Log is then summarized into three parts: what events might have occurred in the Monitor Log, what relevant information the Monitor Log has with the current incident description, and what troubleshooting suggestions the Monitor Log provides for the incident. These three parts of information are then added to the incident data as the final enriched discussion, along with an indication of which team provided the information.

Through the Team Information Enrichment mechanism, we leverage the language understanding and reasoning capabilities of LLM to effectively extract Monitor Log information from the external environment and summarize it into enriched discussion. This enriched discussion augments the original incident information, enabling Triangle to perform automatic reassignment more effectively. Experiments in a real cloud service environment have demonstrated that, with the support of enriched discussion information, Triangle can achieve end-to-end incident reassignment and significantly improve the accuracy of incident triage.

4 INDUSTRY EVALUATION

In this study, to fully evaluate the performance of Triangle in incident triage within a real-world production environment, we aim to address the following research questions:

- RQ1: What is the accuracy of Triangle in the continuous incident triage process within large-scale cloud service systems?

- RQ2: What is the contribution of each key component to the overall performance of Triangle?
- RQ3: How efficient is Triangle in terms of time savings during incident triage in real-world industry scenarios?

4.1 Dataset

To evaluate the performance of Triangle in real-world scenarios, we collected 15 months of real incident data from large-scale cloud service systems serving tens of millions of users at a leading global technology company. These cloud services involve hundreds of engineering teams. To ensure a quantitative and objective experiment, we concentrated solely on incidents that had been resolved, as their confirmed assignments facilitate an accurate assessment of the incident triage process. Specifically, we split the data into a 12-month period for historical incident data and a subsequent 3-month period for evaluating the performance of Triangle. For our experimental analysis, we concentrated solely on incidents that had been resolved, as their confirmed assignments facilitate an accurate assessment of the incident triage process. Specifically, we split the data into a 12-month period for building the knowledge base and a subsequent 3-month period for testing the performance of Triangle.

4.2 Experiment Setup

4.2.1 Metrics. Accuracy and Time to Engage are the two most crucial evaluation metrics in incident triage. Below, we will provide a detailed introduction to these two metrics.

Accuracy: Accuracy is a widely used metric in classification tasks and is a core indicator for evaluating the end-to-end performance of incident triage. However, in incident triage, due to the involvement of reassignment, we further refine the concept of accuracy. We introduce Hop Accuracy. Its calculation is the same as traditional accuracy, but with a restriction on the number of hops for reassignment. Hop N Accuracy ($N \geq 1$) represents the accuracy when the number of assignments does not exceed N by the time the model completes the final assignment. This places a higher requirement on the model's capabilities.

Time to Engage (TTE): TTE refers to the time elapsed from when an incident is reported to when it is assigned to the correct team. TTE is a key factor in measuring the efficiency of incident triage. In practical scenarios, the model's runtime accounts for a minimal portion of the entire triage process. This is because, during triage, engineers from different teams may conduct further analysis of the incident, and there may also be meetings between teams. The time spent by human engineers in these activities constitutes the majority of the triage process.

4.2.2 Baselines. To evaluate the performance of Triangle, we introduce several baseline methods.

- ContentBased [19]: Uses locality-sensitive hashing to find suitable teams, helping mitigate cold start issues by identifying patterns in new or sparse data.
- IvertedIndex [26]: Builds a inverted index table re-ranked by IDF scores to rank teams.
- LGBM [13]: Employs a one-vs-all LightGBM model to handle sparse and unstructured data.
- MART [7]: Utilizes a multiple additive regression tree (MART) model, trained with a one-vs-all FastTree [2] classifier to assign incidents.
- DeepCT [5]: The state-of-the-art incident triage method based on deep learning.

The first four methods are traditional machine and statistical learning methods, that are widely used in the industry. DeepCT [5] is a state-of-the-art incident triage method based on deep learning. DeepCT utilizes Convolutional Neural Networks (CNNs) to encode domain-specific discussions. It then leverages Gated Recurrent Units (GRUs) to capture temporal relations and applies attention mechanisms to reduce the impact of noise. This method relies heavily on the availability of extensive discussions from engineers.

4.3 RQ1: Overall Performance

To verify the effectiveness of Triangle in real-world scenario, we compare the end-to-end incident triage performance of Triangle with other baseline methods. Based on the maximum hop count of manual triage in historical incident data, we evaluated the accuracy for hop counts ranging from 1 to 5. It is worth noting that triage models based on traditional machine learning methods in DeepTriage [19](ContentBased, InvertedIndex, LGBM and MART) are unable to perform continuous triage. In contrast, we use the same discussion text as DeepCT to achieve continuous triage results. As a result, their hop accuracy does not vary across different hops. Additionally, because DeepCT requires manually provided enriched discussions, we sequentially provided DeepTriage with the manually added enriched discussions from the incident data in chronological order. In contrast, our method did not use manually provided enriched discussions, but instead utilized the Team Manager for automatic generation. The experimental results are shown in Table 1.

Table 1. End-to-end performance of Triangle and baseline methods. The maximum of hop count is 5 according to the historical manual triage results. Triangle and DeepCT have the capability of continous triage.

Method	Hop Accuracy [%]				
	Hop ≤ 1	Hop ≤ 2	Hop ≤ 3	Hop ≤ 4	Hop ≤ 5
ContentBased	9.43	17.3	21.9	25.4	29.6
InvertedIndex	14.4	24.8	34.4	42.8	49.4
LGBM	3.12	3.65	4.66	5.11	5.96
MART	4.23	6.17	7.28	10.22	13.56
DeepCT	43.4	54.6	60.4	64.4	67.6
Triangle	**54.7**	**70.4**	**80.5**	**86.0**	**91.7**

According to the experimental results shown in Table 1, our proposed model, Triangle, shows outstanding performance compared to traditional machine learning methods and the state-of-the-art method, DeepCT in end-to-end incident triage. Besides, Triangle shows a significant improvement in accuracy as the hop count increases. This highlights its capability to handle complex scenarios involving multiple reassignment hops effectively.

For hop counts up to 1, Triangle achieves an accuracy of 54.7%, surpassing all other methods, including DeepCT, which stands at 43.4%. As the hop count increases to 5, Triangle maintains its superior performance, reaching an accuracy of 91.7%, a substantial improvement over DeepCT's 67.6%. This demonstrates Triangle's robustness and effectiveness in continuous triage without the need for manually enriched discussions. The automatic generation of enriched discussions by the Team Manager in Triangle plays a crucial role in achieving this enhanced performance, making it a highly effective solution for real-world incident triage scenarios.

Further in-depth analysis reveals that as the Hop Count increases, the performance improvement of DeepCT is less than that of Triangle. We believe this is due to the forgetting phenomenon caused by the GRU model in DeepCT when the sequence length increases. In contrast, Triangle benefits from the powerful memory and comprehension capabilities of the Transformer model in LLM for long sequences. Therefore, its performance is not affected by the increased sequence length when the Hop Count increases.

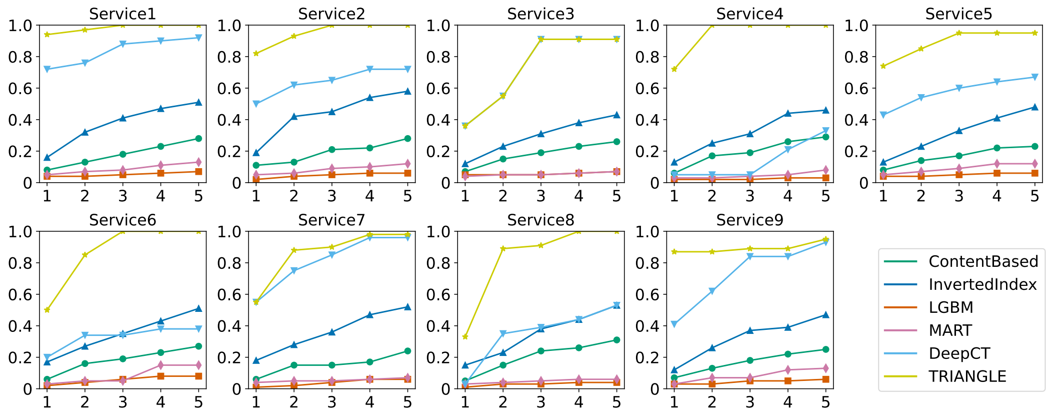


Fig. 7. Effectiveness comparison among Triangle, ContentBased, InvertedIndex, LGBM, MART and DeepCT, for each studied cloud services (the x-axis represents the number of triage hops and the y-axis presents the accuracy of incident triage).

To assess the robustness and generalization of Triangle, we randomly selected nine different services from the system and evaluated the performance of various incident triage models across these services. The results are illustrated in Figure 7.

The experimental findings reveal that Triangle consistently achieves superior Hop Accuracy across the majority of services, with notable improvements over baseline methods observed at the 2nd or 3rd hop. This enhancement is attributed to Triangle's multi-agent negotiation mechanism, which effectively aggregates information from multiple teams. This process introduces substantial external information to incidents that initially lack sufficient details, thereby significantly improving triage performance.

Our experiments demonstrate that Triangle excels in end-to-end incident triage performance in real-world scenarios.

4.4 RQ2: Ablation Study

To evaluate the contribution of each key components in our approach, we conducted an ablation study following the experimental setup of Section 4.3. We removed the semantic distillation (w/o ST), Multi-Agent negoTiation mechanism (w/o MAT), and Team Information Enrichment mechanism (w/o TIE) respectively. Since multi-agent negotiation is the core operation of incident triage, to ensure the normal operation of Triangle after removing the multi-agent negotiation mechanism, we allowed the Triage Decider to directly assign based on the ranking of team candidates. Table 2 shows our experimental results.

From the experimental results presented in Table 2, it is evident that each of the key components in our proposed approach contributes significantly to the overall performance. The results show that removing any of the components leads to a decrease in Hop Accuracy across all Hop count (Hop ≤ 1 to Hop ≤ 5). Specifically, without the Semantic Distillation (w/o ST), the performance drops notably, achieving only 49.1% Hop ≤ 1 accuracy, which is a 5.6% decrease compared to the full model. This drop in performance indicates that semantic distillation is crucial for accurate hop prediction, allowing the system to make more informed decisions based on enriched semantic information.

Table 2. Ablation study results of different components of TRIANGLE on Hop Accuracy.

Method	Hop Accuracy [%]				
	Hop ≤ 1	Hop ≤ 2	Hop ≤ 3	Hop ≤ 4	Hop ≤ 5
w/o ST	49.1	62.2	76.8	81.1	86.1
w/o MAT	42.8	54.6	63.4	67.5	70.4
w/o TIE	49.6	60.1	61.7	63.8	65.8
TRIANGLE	**54.7**	**70.4**	**80.5**	**86.0**	**91.7**

The most substantial performance degradation is observed when the multi-agent negotiation mechanism is removed(w/o MAT). The accuracy drops to 42.8% for Hop ≤ 1, which is almost a 12% reduction compared to the full model. The Hop ≤ 5 accuracy also sees a significant decline to 70.4%. This demonstrates that the negotiation mechanism is vital for optimizing the triage decision-making process through collaborative decision-making among agents, rather than relying on a naive ranking approach.

The absence of the Team Information Enrichment mechanism (w/o TIE) also results in a significant reduction in performance. The model's Hop ≤ 2 and Hop ≤ 5 accuracies decrease by 10.3% and 25.9%, respectively, compared to TRIANGLE. These results confirm that enriched discussion is a key factor for effective incident triage, as it provides crucial context that enhances the decision-making capability of the multi-agent system.

Notably, Team Information Enrichment has the greatest impact on the performance of TRIANGLE. This is because the key reason for the inaccuracy in incident triage is the insufficient amount of information in raw incidents. The role of Team Information Enrichment is to automatically obtain external relevant information through agents, so the introduction of external information has a decisive effect on the performance of incident triage.

In contrast, our proposed method, TRIANGLE, consistently outperforms all ablated versions across all metrics, achieving the highest Hop Accuracy at every threshold. This indicates that the combined use of semantic distillation, multi-agent negotiation, and team information enrichment provides a synergistic effect that leads to superior triage performance.

4.5 RQ3: Efficiency of TRIANGLE

Time to Engage (TTE) is an important metric for evaluating the efficiency of incident triage models. We followed the experimental setup from Section 4.3 to assess the impact of different incident triage models and various variants of TRIANGLE on TTE. We use Time Unit as the unif of TTE. As defined, the Time Unit is an internal metric used by the company to evaluate the efficiency of triage processes. Although the exact relationship between a Time Unit and real-world time cannot be disclosed, it is a linearly correlated measure, allowing for a fair comparison across different models. We used the TTE of manual triage as a baseline and tested how much different methods could reduce TTE. The results are shown in Figure 8.

As shown in Figure 8, the average TTE reductions achieved by different models and variants of TRIANGLE demonstrate the superior performance of our proposed method. Notably, TRIANGLE outperforms all other baseline models with an average TTE reduction of 24.58 time units, significantly improving the efficiency of incident triage compared to manual triage.

In the baseline models, ContentBased and InvertedIndex achieve moderate improvements in TTE, reducing it by 7.07 and 16.76 time units, respectively. However, these traditional methods are limited in their ability to capture complex semantic relationships in the incident data, leading to

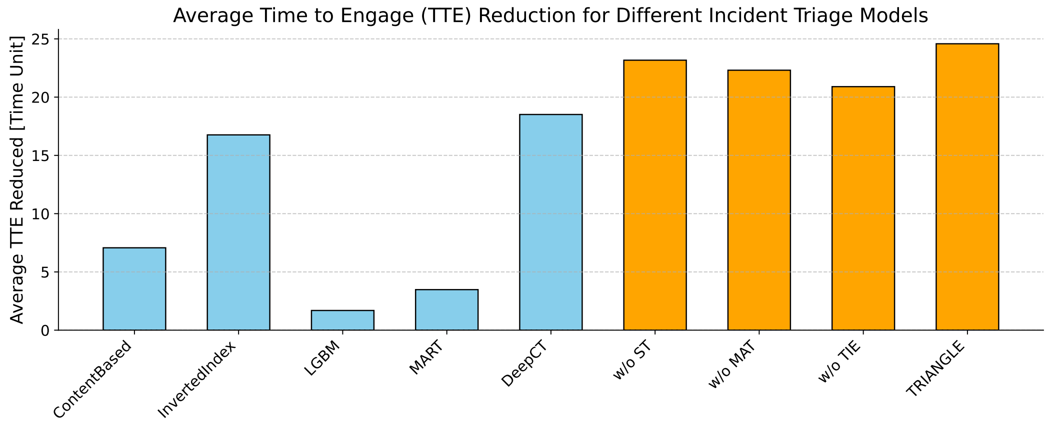


Fig. 8. Average Time to Engage (TTE) Reduction for Different Incident Triage Models and Variants of Triangle. The TTE of manual triage is used as a baseline.

less substantial TTE reductions compared to more advanced methods. Machine learning-based models such as LGBM and MART show limited improvements, reducing TTE by only 1.69 and 3.48 time units, respectively. This indicates that while they are able to capture some patterns in the data, their effectiveness in optimizing the incident triage process is constrained, possibly due to their reliance on predefined features and shallow architectures.

Among the more advanced methods, DeepCT delivers a notable reduction of 18.51 time units, underscoring the strength of deep learning approaches in handling unstructured incident data. However, DeepCT still falls short of the performance of Triangle, suggesting that the Multi-LLM-Agent framework and additional optimizations introduced in our approach, contribute significantly to enhancing triage efficiency.

5 DISCUSSION

Lessons learned. To enhance the accuracy of incident triage, we have identified several key lessons related to improving troubleshooting guides and team documentation [22]. Firstly, it is crucial to ensure that documentation is clear and detailed, with step-by-step instructions and information on responsible team, to facilitate easy understanding and reduce resolution times. Regular updates and reviews are essential to keep the information current and relevant, while incorporating feedback from users helps identify and address any gaps or ambiguities. Standardizing the format and terminology of documents ensures consistency and usability. Additionally, leveraging advanced technology, such as knowledge management systems, can further streamline document updates and accessibility. By implementing these practices, we aim to simplify the process and enhance understanding for LLM agents, ultimately leading to more accurate and efficient incident triage.

Auto-mitigation. Effectively managing and triaging an incident is just the beginning of our process. It's not merely about directing the issue to the appropriate team; it's crucial to actively work towards minimizing the time to mitigation [15, 17]. To address this challenge, we will further implement Triangle to enhance its capabilities to incorporate an automated mitigation workflow. This innovation will streamline the incident response process, allowing us to resolve incidents more swiftly and efficiently. Looking ahead, we envision that these advancements will pave the way towards achieving a fully autonomous cloud operation, where the system can proactively handle and resolve issues with minimal human intervention. This progressive step aims to not only improve our current operational efficiency but also set the stage for a more self-sufficient and resilient cloud infrastructure in the future.

Threat to Validity. *Internal* validity threats mainly stem from the implementation of our method Triangle and the comparison methods. To mitigate this threat, two authors thoroughly review the code. Specifically, we implement these methods based on a mature industry framework.

External validity threats primarily concern the subjects used. In our study, we employed data from several large-scale cloud service systems. Although these data are derived from real industry applications, the subjects may not fully represent online service systems in other companies. In future work, we will apply Triangle to a broader range of cloud service systems.

Construct validity threats primarily lie in the choice of parameters and metrics used. To mitigate the threat from parameters, we employ grid search to optimize the parameters in both Triangle and the comparison methods. To address the threat from metrics, we utilize the most commonly used accuracy and time cost metrics in our study. In future work, we plan to incorporate additional metrics, such as false positive rate and recall, to more comprehensively evaluate the effectiveness and efficiency of Triangle.

6 RELATED WORK

Bug triage. Research on bug triage for traditional software is extensive , focusing mainly on two approaches: learning-based and information-retrieval-based methods. Learning-based approaches treat bug triage as a supervised classification problem, using techniques such as ensemble learning [12], and deep learning with Convolutional Neural Networks (CNNs) [14, 21] to classify bugs. Information-retrieval methods focus on leveraging expertise and historical data, with approaches like Latent Dirichlet Allocation (LDA) [18] to match developers to bugs, topic-modeling [24] to map bug report terms to topics, and historical bug-fix analysis [8] to link developers, code components, and bugs. However, incident triage presents a more complex challenge in industry practice because of the intricate nature of cloud systems.

Incident triage. Recent advancements in incident triage have utilized deep learning to enhance accuracy and efficiency [4]. DeepTriage [19] uses various machine learning models to automate triage, improving accuracy by learning from historical data. The most similar work is DeepCT [5], which performs continuous incident triage using Convolutional Neural Networks (CNNs) to encode domain-specific text and Gated Recurrent Units (GRUs) to extract temporal relationships, complemented by attention mechanisms to reduce noise. Its effectiveness depends on extensive human discussions, which cannot be fully automated. In contrast, our multi-LLM-agent based solution can collect troubleshooting information and manage negotiation processes like a human.

LLM for cloud systems. In recent years, the integration of Large Language Models (LLMs) into cloud systems has gained significant traction, reflecting a broader trend toward enhancing automation and efficiency in cloud operations. Research and practical implementations have demonstrated how LLMs can be leveraged for various tasks, including incident detection [16, 27], assessment [11, 30], and diagnosis [1, 6, 9, 10]. For example, RCAgent [23] enhances LLM-generated root cause reports with a Self-Consistency mechanism and domain-specific knowledge integration. ReAct [20] applies LLMs to root cause analysis in cloud management, showing high performance and accuracy with real-world data. DB-GPT [25] merges LLMs with traditional databases to improve natural language query responses, featuring a retrieval-augmented generation system and adaptive learning. To the best of our knowledge, no existing multi-LLM-agent solutions have been proposed specifically for incident triage. Triangle is the first end-to-end multi-LLM-agent based incident triage approach. Nonetheless, it is quite natural to leverage LLMs to emulate human capabilities in performing triage tasks.

7 CONCLUSION

Effective and accurate incident triage is crucial for maintaining service quality and reducing time to engagement and mitigate in large-scale cloud service systems. In this paper, we present TRIANGLE, an end-to-end incident triage system designed using a Multi-LLM-Agent framework. We introduce a novel semantic distillation mechanism that leverages the powerful semantic understanding capabilities of LLMs to tackle the issue of incident semantic heterogeneity, significantly enhancing triage accuracy. Additionally, we develop a multi-role agent framework equipped with an effective negotiation mechanism, allowing the system to dynamically manage multi-team domain knowledge and simulate the workflow of human engineers. Moreover, TRIANGLE includes an automated team information enrichment mechanism, enabling end-to-end triage without incurring additional human labor costs, even in scenarios requiring incident reassignment. Extensive experiments conducted with real-world incident triage data from a large-scale production environment demonstrate that TRIANGLE outperforms state-of-the-art methods, improving average triage accuracy more than 20% and reducing time to engagement. The deployment of TRIANGLE in a production system serving tens of millions of users at a leading global technology company has shown its effectiveness and reliability in real-world environments. We believe that our approach can provide valuable insights and serve as a foundation for future research and development in automated incident triage systems for large-scale cloud services.

8 DATA AVAILABILITY

The data of our industry experiment is highly confidential that contains private information about internal systems and services, and therefore, we cannot disclose our data due to company policy.

ACKNOWLEDGMENTS

Acknowledgment

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 *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1737–1749.

[2] Zeeshan Ahmed, Saeed Amizadeh, Mikhail Bilenko, Rogan Carr, Wei-Sheng Chin, Yael Dekel, Xavier Dupré, Vadim Eksarevskiy, Senja Filipi, Tom Finley, Abhishek Goswami, Monte Hoover, Scott Inglis, Matteo Interlandi, Najeeb Kazmi, Gleb Krivosheev, Pete Luferenko, Ivan Matantsev, Sergiy Matusevych, Shahab Moradi, Gani Nazirov, Justin Ormont, Gal Oshri, Artidoro Pagnoni, Jignesh Parmar, Prabhat Roy, Mohammad Zeeshan Siddiqui, Markus Weimer, Shauheen Zahirazami, and Yiwen Zhu. 2019. Machine Learning at Microsoft with ML.NET. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 2448–2458. https://doi.org/10.1145/3292500.3330667

[3] Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management* 39, 1 (2003), 45–65.

[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 *2019 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 *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*. IEEE, 364–375. https://doi.org/10.1109/ASE.2019.00042

[6] 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

[7] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[8] Hao Hu, Hongyu Zhang, Jifeng Xuan, and Weigang Sun. 2014. Effective bug triage based on historical bug-fix information. In *2014 IEEE 25th international symposium on software reliability engineering*. IEEE, 122–132.

[9] Junjie Huang, Jinyang Liu, Zhuangbin Chen, Zhihan Jiang, Yichen Li, Jiazhen Gu, Cong Feng, Zengyin Yang, Yongqiang Yang, and Michael R Lyu. 2024. FaultProfIT: Hierarchical Fault Profiling of Incident Tickets in Large-scale Cloud Systems. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*. 392–404.

[10] Yuxuan Jiang, Chaoyun Zhang, Shilin He, Zhihao Yang, Minghua Ma, Si Qin, Yu Kang, Yingnong Dang, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2024. Xpert: Empowering Incident Management with Query Recommendations via Large Language Models. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024*. ACM, 92:1–92:13. https://doi.org/10.1145/3597503.3639081

[11] 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, and Dongmei Zhang. 2023. Assess and Summarize: Improve Outage Understanding with Large Language Models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023*. ACM, 1657–1668. https://doi.org/10.1145/3611643.3613891

[12] Leif Jonsson, Markus Borg, David Broman, Kristian Sandahl, Sigrid Eldh, and Per Runeson. 2016. Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts. *Empirical Software Engineering* 21 (2016), 1533–1578.

[13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 3146–3154. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[14] Sun-Ro Lee, Min-Jae Heo, Chan-Gun Lee, Milhan Kim, and Gaeul Jeong. 2017. Applying deep learning based automatic bug triager to industrial projects. In *Proceedings of the 2017 11th Joint Meeting on foundations of software engineering*. 926–931.

[15] Sebastien Levy, Randolph Yao, Youjiang Wu, Yingnong Dang, Peng Huang, Zheng Mu, Pu Zhao, Tarun Ramani, Naga Govindaraju, Xukun Li, et al. 2020. Predictive and Adaptive Failure Mitigation to Avert Production Cloud {VM} Interruptions. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 1155–1170.

[16] Jun Liu, Chaoyun Zhang, Jiaxu Qian, Minghua Ma, Si Qin, Chetan Bansal, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2024. Large Language Models can Deliver Accurate and Interpretable Time Series Anomaly Detection. *arXiv preprint arXiv:2405.15370* (2024).

[17] Chang Lou, Cong Chen, Peng Huang, Yingnong Dang, Si Qin, Xinsheng Yang, Xukun Li, Qingwei Lin, and Murali Chintalapati. 2022. {RESIN}: A Holistic Service for Dealing with Memory Leaks in Production Cloud Infrastructure. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. 109–125.

[18] Hoda Naguib, Nitesh Narayan, Bernd Brügge, and Dina Helal. 2013. Bug report assignee recommendation using activity profiles. In *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 22–30.

[19] Phuong Pham, Vivek Jain, Lukas Dauterman, Justin Ormont, and Navendu Jain. 2020. DeepTriage: Automated Transfer Assistance for Incidents in Cloud Services. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3281–3289. https://doi.org/10.1145/3394486.3403380

[20] Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro 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*. 208–219.

[21] CN dos Santos and MADC Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING*, Vol. 25. 69.

[22] Manish Shetty, Chetan Bansal, Sai Pramod Upadhyayula, Arjun Radhakrishna, and Anurag Gupta. 2022. Autotsg: learning and synthesis for incident troubleshooting. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1477–1488.

[23] Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Lunting Fan, Lingfei Wu, and Qingsong Wen. 2023. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. *arXiv preprint arXiv:2310.16340* (2023).

[24] Xin Xia, David Lo, Ying Ding, Jafar M Al-Kofahi, Tien N Nguyen, and Xinyu Wang. 2016. Improving automated bug triaging with specialized topic model. *IEEE Transactions on Software Engineering* 43, 3 (2016), 272–297.

[25] Siqiao Xue, Caigao Jiang, Wenhui Shi, Fangyin Cheng, Keting Chen, Hongjun Yang, Zhiping Zhang, Jianshan He, Hongyang Zhang, Ganglin Wei, et al. 2023. Db-gpt: Empowering database interactions with private large language models. *arXiv preprint arXiv:2312.17449* (2023).

[26] Hao Yan, Shuai Ding, and Torsten Suel. 2009. Inverted index compression and query processing with optimized document ordering. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl (Eds.). ACM, 401–410. https://doi.org/10.1145/1526709.1526764

[27] Zhaoyang Yu, Minghua Ma, Chaoyun Zhang, Si Qin, Yu Kang, Chetan Bansal, Saravan Rajmohan, Yingnong Dang, Changhua Pei, Dan Pei, Qingwei Lin, and Dongmei Zhang. 2024. MonitorAssistant: Simplifying Cloud Service Monitoring via Large Language Models. 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, 38–49. https://doi.org/10.1145/3663529.3663826

[28] Zhaoyang Yu, Changhua Pei, Xin Wang, Minghua Ma, Chetan Bansal, Saravan Rajmohan, Qingwei Lin, Dongmei Zhang, Xidao Wen, Jianhui Li, Gaogang Xie, and Dan Pei. 2024. Pre-trained KPI Anomaly Detection Model Through Disentangled Transformer. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, Ricardo Baeza-Yates and Francesco Bonchi (Eds.). ACM, 6190–6201. https://doi.org/10.1145/3637528.3671522

[29] Zhaoyang Yu, Changhua Pei, Shenglin Zhang, Xidao Wen, Jianhui Li, Gaogang Xie, and Dan Pei. 2023. AutoKAD: Empowering KPI Anomaly Detection with Label-Free Deployment. In *34th IEEE International Symposium on Software Reliability Engineering, ISSRE 2023, Florence, Italy, October 9-12, 2023*. IEEE, 13–23. https://doi.org/10.1109/ISSRE59848.2023.00063

[30] Xin Zhou, Bowen Xu, Kisub Kim, DongGyun Han, Hung Huu Nguyen, Thanh Le-Cong, Junda He, Bach Le, and David Lo. 2024. Leveraging Large Language Model for Automatic Patch Correctness Assessment. *IEEE Transactions on Software Engineering* (2024).