LLM-Augmented Ticket Aggregation for Low-cost Mobile OS Defect Resolution

Yongqian Sun
Nankai University
Tianjin, China

Bowen Hao
Nankai University
Tianjin, China

Xiaotian Wang
Nankai University
Tianjin, China

Chenyu Zhao
Nankai University
Tianjin, China

Yongxin Zhao
Nankai University
Tianjin, China

Binpeng Shi
Nankai University
Tianjin, China

Shenglin Zhang
Nankai University
Tianjin, China

Qiao Ge
Huawei Inc.
Wuhan, China

Wenhu Li
Huawei Inc.
Wuhan, China

Hua Wei
Huawei Inc.
Wuhan, China

Dan Pei
Tsinghua University
Beijing, China

Abstract

Mobile OS often exhibit defects due to their complexity and frequent updates, adversely affecting user experience. To resolve these defects, OS providers enlist beta users to test the OS. During testing, any defects encountered are recorded in tickets and reported back to the providers. However, the sheer volume of these tickets significantly challenges the efficiency of the ticket handling system, especially during the triage phase, where engineers assign tickets to appropriate development teams for resolution. Enhancing triage efficiency is possible by aggregating duplicate tickets related to the same defect, allowing simultaneous assignment of multiple tickets. Nonetheless, current ticket aggregation methods demand substantial labeled data for training, imposing a labor cost that hinders their practical implementation in production environments. To reduce this labor cost, we propose *TixFusion*, an LLM-augmented ticket aggregation framework. *TixFusion* employs unsupervised clustering to aggregate tickets, minimizing the need for labeled data, and integrates an LLM to extract discriminative information from tickets, improving aggregation accuracy. Extensive experiments using a dataset collected from the production environment of a top-tier global mobile OS provider *H* demonstrate that *TixFusion* outperforms existing methods while maintaining a low labor cost. Additionally, *TixFusion* has been deployed in *H* for over three months, processing more than 200,000 tickets, and has increased the processing speed of triage engineers by 3.78 times.

CCS Concepts

• **Software and its engineering → Software defect analysis**.

Keywords

Ticket Aggregation, Mobile Operating Systems, Defect

FSE '25, Trondheim, Norway

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

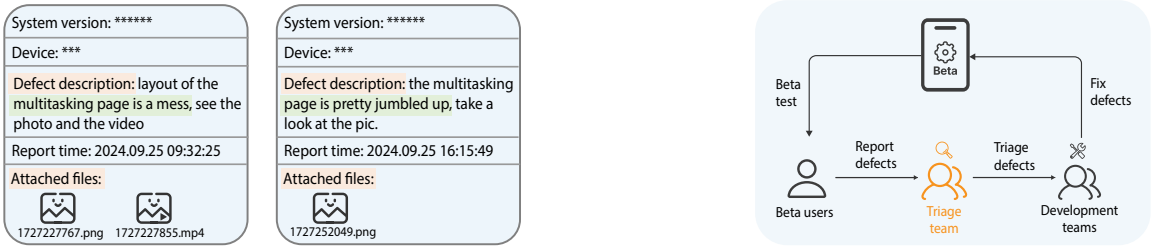
ACM Reference Format:
Yongqian Sun, Bowen Hao, Xiaotian Wang, Chenyu Zhao, Yongxin Zhao, Binpeng Shi, Shenglin Zhang, Qiao Ge, Wenhu Li, Hua Wei, and Dan Pei. 2018. LLM-Augmented Ticket Aggregation for Low-cost Mobile OS Defect Resolution. In *Companion Proceedings of the 33rd ACM Symposium on the Foundations of Software Engineering (FSE '25), June 23–27, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

With the advancement of mobile Internet technology, the global user base of smart mobile devices has exceeded 4.3 billion [17]. Mobile OS, as the foundation of these devices, plays a vital role in modern life. However, the complexity and continuous evolution of the mobile OS often introduce defects, ranging from minor errors to critical malfunctions [37], adversely affecting user experience and potentially causing significant business consequences. For example, a defect named "MediaProvider" was identified within Google's Android OS, which permitted attackers to execute arbitrary code on vulnerable devices by manipulating specially crafted media files. This defect affected approximately 1.5 billion Android users, severely harming Google's reputation [45].

To identify and fix such defects, mobile OS providers usually implement strict testing procedures before releasing updates. The beta test is a crucial final stage [26], during which various beta users interact with the updated OS to report defects. As demonstrated in Figure 1a, each defect is recorded in a dedicated ticket. A vast volume of tickets is generated every day (e.g., over 3,000 in a top-tier global mobile OS provider). The ticket handling system is responsible for triaging, and fixing these tickets to ensure a high-quality user experience [3], as shown in Figure 1b.

During the triage stage, collected tickets must be assigned to the appropriate development teams by triage engineers. This process is time-consuming, requiring triage engineers to carefully review the text and images uploaded by beta users before making assignments. On average, an experienced triage engineer can process only about 40 tickets per day. Furthermore, the number of experienced triage engineers is limited, as they require specialized knowledge of the mobile OS and its associated development teams. As a result, the reliance on manual triage and the scarcity of skilled triage engineers



(a) Example of duplicate tickets **(b) Ticket handling system**

Figure 1: Illustration of duplicate tickets and overview of the ticket handling system.

often lead to delays, reducing the overall efficiency of the ticket handling system.

Within the extensive volume of tickets, many duplicate tickets refer to the same defect. These duplicate tickets waste valuable time for the triage engineers because they must assign these tickets individually. To allow the triage engineers to assign duplicate tickets collectively and improve efficiency, we can aggregate these tickets [27]. Over the years, a series of ticket aggregation methods have been proposed [5, 8, 27, 32]. However, these methods rely heavily on a large number of labeled tickets to achieve satisfactory performance. They require the labeling of connections between tickets, a process more labor-intensive than the triage itself, rendering these methods impractical for large-scale production environments (as detailed in Section 5.4).

Therefore, we try to propose a low-cost ticket aggregation method that relies on the fewest labeled tickets. Intuitively, we can apply an unsupervised clustering approach to aggregate duplicate tickets based on their semantic information. However, it faces the following two challenges (see Section 3 for more details).

Challenge 1: Utilization of discriminative information. Due to the lack of domain-specific training, unsupervised clustering cannot effectively identify and utilize discriminative information in tickets to distinguish between defects, impeding aggregation performance.

Challenge 2: Lack of explainability. Neither clustering approaches nor existing ticket aggregation methods can provide aggregation results with explanatory content. Triage engineers still need to invest substantial time and effort in further examining the tickets before they are assigned to the appropriate development teams.

This paper presents *TixFusion*, an LLM-augmented ticket aggregation framework. To address Challenge 1, we propose an in-context learning-based method to extract discriminative information from tickets. To address Challenge 2, we generate explanations for each group of aggregated tickets. Additionally, recognizing the direct visual insights from image data, we integrate image data into our framework to enhance explainability.

The contributions of this paper are summarized as follows:

- We propose an LLM-augmented ticket aggregation framework that achieves accurate aggregation while significantly reducing the labeling overhead.
- We introduce an in-context learning-based method for discriminative information extraction, enhancing extraction effectiveness by incorporating domain knowledge and segmenting the process into multiple rounds. The discriminative information is then used to improve aggregation accuracy.
- To the best of our knowledge, we are the first to integrate image data in ticket aggregation, offering a simple yet effective strategy for utilizing image data. By combining image and text data, we generate effective and readable explanations for aggregation results, thereby enhancing the explainability.
- To substantiate the effectiveness of *TixFusion*, we conduct a comprehensive evaluation using a dataset collected from a top-tier global mobile OS provider H. The F_1-score of *TixFusion* demonstrates a significant improvement of 0.44 compared to baseline methods. Furthermore, *TixFusion* has been deployed in H for over three months, processing more than 200,000 tickets, and improving the processing speed of triage engineers by 3.78 times.

2 Background

2.1 Beta Test and Ticket

The inherent complexity and frequent updates of mobile OS inevitably introduce numerous defects. A **defect** is any flaw that causes the OS to behave incorrectly or deviate from its intended functionality, encompassing issues from minor glitches to major malfunctions [37]. Such defects can degrade user experience and result in adverse business impacts. To address these defects, mobile OS providers usually conduct rigorous tests before releasing updates. The beta test is the last phase of this testing process [26]. During this phase, providers distribute the updated OS to a selected group of beta users to identify as many defects as possible.

Upon encountering defects in the updated OS, beta users report them to the provider via tickets, as shown in Figure 1a. Each ticket serves as a detailed record, capturing all necessary information for the triage, diagnosis, and resolution of the defect. A ticket includes the following fields: system version, device name, report time, defect description (user's detailed description of the encountered defect), and attached files (screenshots or videos reflecting the defect).

2.2 Ticket Triage

Mobile OS providers commonly employ a robust ticket handling system to efficiently manage tickets reported by beta users. We illustrate the workflow of a ticket handling system of H in Figure 1b. The system first collects tickets from beta users. Next, the triage team assigns these tickets to appropriate development teams by

examining the detailed information in the tickets [4, 43, 49]. Upon receiving assigned tickets, the development teams analyze the corresponding defects and fix them. Through this systematic process, the providers aim to ensure a seamless experience for mobile OS users by identifying and rectifying defects in the OS. The triage phase is pivotal in this workflow, as it ensures timely and accurate ticket assignment, optimizing the development teams' productivity, and enhancing the overall efficiency of the ticket handling system.

3 Motivation

The ticket handling system of H generates over 3,000 tickets per day, presenting a significant challenge to the efficiency of triage engineers. Among the vast number of these tickets, many duplicate tickets point to the same defect. Aggregation of these duplicate tickets can improve the efficiency of triage engineers by allowing collective processing.

Existing methods for ticket aggregation [5, 8, 27, 32] are generally effective but depend on an extensive amount of labeled tickets. They require the labeling of connections between tickets, a task more labor-intensive than the triage due to the presence of thousands of distinct defects. Our evaluation in Section 5.4 reveals that preparing the dataset for large-scale deployment of existing methods would require over 1,000 person-days. Furthermore, the dynamic nature of the mobile OS introduces new defects, necessitating frequent updates to the labeled data and compounding the labor cost. These factors render existing methods impractical for our production environment.

Therefore, we try to propose a low-cost ticket aggregation method that relies on the fewest labeled tickets. Intuitively, we can apply an unsupervised clustering approach to aggregate duplicate tickets based on their semantic information. However, it confronts the following two challenges: utilization of discriminative information and lack of explainability.

3.1 Challenge 1: Utilization of Discriminative Information

During manual ticket processing, triage engineers concentrate on defect-related information, including affected OS components, functions, and defect behavior. This discriminative information is crucial for distinguishing between defects. We categorize it into three segments: component (the OS component affected by the defect), function (the function of the affected component), and behavior (the observed behavior of the defect), as shown in Figure 3a.

Since beta users typically do not possess specialized knowledge of the mobile OS, discriminative information is often not directly present in the raw tickets and requires extraction by triage engineers based on their expertise. This absence leads to poor aggregation performance of unsupervised clustering, as it lacks domain-specific training to effectively extract discriminative information from tickets.

To address this, we propose a two-step strategy: first, extract discriminative information from tickets, and then use this information as input for unsupervised clustering. This strategy allows for direct leveraging of discriminative information to differentiate between defects, thereby potentially improving aggregation performance.

Given that one of the primary data in tickets is the detailed user description of defects, and considering the powerful natural language processing capabilities of Large Language Models (LLMs), we employ an LLM to perform the extraction [58]. Furthermore, LLMs can achieve satisfactory performance on downstream tasks without requiring a large amount of labeled data, which aligns with our goal of reducing the labor cost of labeling [56].

3.1.1 Empirical Study. We conduct an empirical study to assess the effectiveness of discriminative information extraction on unsupervised clustering.

The DBSCAN clustering algorithm is tested with three embedding models from a widely recognized text embedding benchmark MTEB [36] on a test set comprising 1,479 tickets collected from H. We use the widely accepted Rand Index (RI) [38] as the evaluation metric, which computes precision, recall, and F_1-score by comparing the aggregation results with the ground truth.

Table 1: Comparison of aggregation performance.

Data	Embedding model	Precision	Recall	F_1-score
Raw	acge [22]	0.163	0.332	0.219
	conan_v1 [29]	0.141	0.416	0.211
	xiaobu_v2 [42]	0.134	0.437	0.205
Discriminative	acge	0.657	0.835	0.735
	conan_v1	0.635	0.826	0.718
	xiaobu_v2	0.644	0.823	0.723

The results are presented in Table 1. The top three rows represent the aggregation performance of unsupervised clustering using raw tickets, while the bottom three rows show its performance when discriminative information is used. The results reveal that aggregation using raw tickets exhibits poor performance, supporting the finding that unsupervised clustering algorithms cannot effectively utilize the discriminative information in tickets. In contrast, when discriminative information extracted by the LLM is employed, the average F_1-score improves by 0.514, demonstrating the significant enhancement in aggregation performance achieved by this extraction strategy.

3.2 Challenge 2: Lack of Explainability

Clustering approaches and existing ticket aggregation methods [5, 8, 27, 32] only deliver aggregation results without offering any explanatory content. This lack of explainability restricts their effectiveness in facilitating the triage phase, as the triage engineers must inspect multiple tickets within each group to assign them to the appropriate development teams. The review process is time-consuming, requiring engineers to examine 20% to 30% of the tickets in each group to confirm assignments, with each ticket review taking about one minute.

To address this challenge, our framework generates explanations for each group of aggregated tickets. These explanations offer clear overviews of common characteristics within each group, enabling triage engineers to comprehend the results without examining individual tickets. Tickets often contain screenshots and videos reflecting the defects, this image data can significantly enhance the

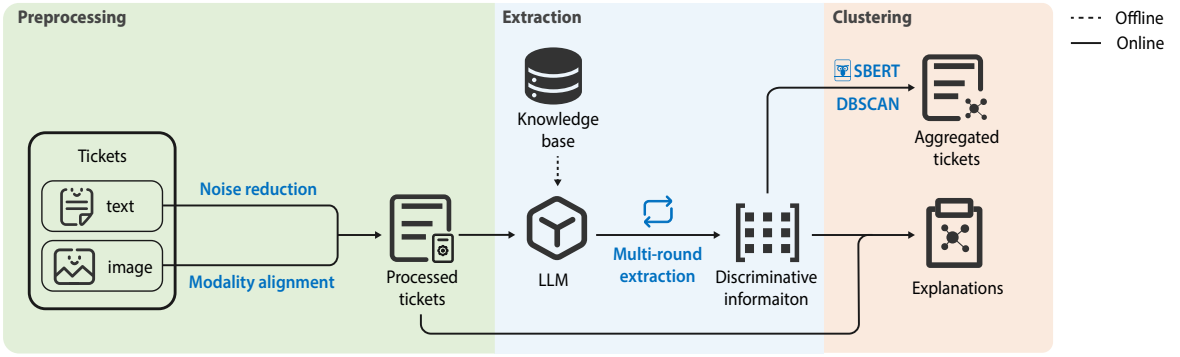


Figure 2: The framework of *TixFusion*.

explainability by providing direct visual insights of the OS's state. Therefore, we integrate image data into our framework to enhance explainability.

4 Methodology

In this section, we introduce *TixFusion*, an LLM-augmented ticket aggregation framework. *TixFusion* takes text and image data from tickets (example in Figure 1a) as input and outputs aggregated tickets with the same defect along with explanations.

The overall framework of *TixFusion* is illustrated in Figure 2, which comprises two stages. In the offline stage, *TixFusion* collects a small number of labeled tickets for in-context learning (ICL) samples [10], constructs a knowledge base, and fine-tunes the LLM. In the online stage, *TixFusion* first preprocesses the tickets for noise reduction and multimodal alignment. Then, *TixFusion* extracts discriminative information from the processed tickets using LLM. After that, to aggregate duplicate tickets, a clustering module is built to vectorize this discriminative information through a pre-trained embedding model and perform aggregation through unsupervised clustering. Finally, *TixFusion* combines the extracted discriminative information and original tickets to generate explanations for each group of tickets with the same defect and presents the results to the triage engineers for assignment.

4.1 Preprocessing Module

Tickets comprise both text and image data. In this stage, *TixFusion* removes noise present in tickets and aligns the modalities.

Noise reduction. The text data in tickets refers to **defect descriptions**, which are beta users' detailed descriptions of the encountered defects. Since defect descriptions are manually written, they sometimes contain irrelevant information such as users' complaints, which act as noise and obstruct subsequent processing. Large Language Models (LLMs) have demonstrated powerful capabilities in summarizing and refining text [58]. Therefore, *TixFusion* inputs the raw text of defect descriptions into an LLM (an original GLM-4-9B) and requests a refined version. To help the LLM identify and filter out noise, *TixFusion* incorporates five fixed examples into the prompt.

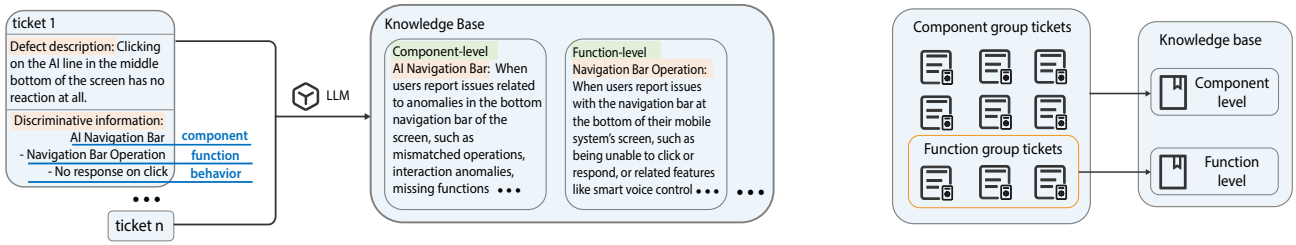
Modality alignment. The image data comprises screenshots and videos submitted by beta users to depict defects. To align the image data with the defect descriptions in text form, *TixFusion* converts this image data into text. Specifically, we individually process each image using a multimodal large language model (mLLM, such as GLM-4v[16]), generating detailed **image descriptions**. For videos, we extract frames at a constant interval of 180 frames and feed these frames into the mLLM for descriptions.

4.2 Extraction Module

To facilitate the subsequent unsupervised clustering, *TixFusion* utilizes an LLM to extract discriminative information from the preprocessed **defect descriptions** and **image descriptions**. However, tickets contain certain domain knowledge, and the performance of direct extraction through LLM is not satisfactory (see Section 5.3.1 for details). In light of this, in addition to conventional techniques such as in-context learning prompting, chain-of-thought prompting[51], and fine-tuning, *TixFusion* introduces two enhancements: knowledge base construction and multi-round extraction.

4.2.1 Knowledge Base Construction. A knowledge base that encompasses triage engineers' understanding of defects can provide the LLM with additional context and enhance the quality of discriminative information extraction. Such knowledge bases are usually constructed through domain-specific documents [15]. However, there are no such documents in our scenario, and creating them from scratch requires a substantial amount of manual labor. Therefore, we request experienced triage engineers to label a small number (e.g., 10,000 in our scenario) of tickets and extract their knowledge from these labels. The labels represent discriminative information in the defect descriptions, formatted as "component (the OS component affected by the defect) - function (the function of the affected component) - behavior (the observed behavior of the defect)", as illustrated in the left part of Figure 3a. This information is critical to triage engineers during manual ticket processing, as outlined in Section 3.1. The knowledge reflects typical behaviors of components and functions in defect descriptions.

We derive two types of knowledge from the labeled tickets, corresponding to the OS's components and the functions of those components. Initially, we group the labeled tickets by component. Within each group, we batch the defect descriptions and corresponding labels, input them into an LLM (an original GLM-4-9B) along with five fixed knowledge examples, and request summarizations



(a) An example of discriminative information and knowledge **(b) Construction workflow**

Figure 3: Illustration of knowledge base construction.

about how components are described in defect descriptions. Next, within each component group, we further group tickets by function and request similar summarizations. These summarizations collectively form a knowledge base, which provides additional context to the LLM to enhance the quality of discriminative information extraction.

4.2.2 Multi-Round Extraction. Inspired by inference scaling laws [1, 41, 52], which suggest that increased computational resources during inference can improve the model's performance, *TixFusion* adopts a multi-round extraction approach for **defect descriptions** to enhance discriminative information extraction.

Initially, we prompt an LLM (a fine-tuned GLM-4-9B for extraction) to identify the OS component impacted by the defect. Subsequently, we direct the LLM to determine the specific function of the identified component. Throughout the initial two rounds, we integrate relevant knowledge about components and functions from the knowledge base to assist identification. Finally, we ask the LLM to summarize the anomalous behavior of the defect based on the previously identified component and function. These three rounds of interaction yield a structured "component - function - behavior" format for discriminative information, as illustrated in the left part of Figure 3a. During each extraction round, *TixFusion* calculates the similarity between the current defect description and the training set and incorporates the top-ranked (five in this paper) examples into the prompts. Motivated by [12, 15, 47], we employ the pre-trained Sentence-BERT [39] embedding model for similarity calculation. Note that the calculation of similarity is not the contribution of *TixFusion*. To more clearly illustrate the multi-round extraction process, we use the following equations. Here, C represents the identified component, F represents the identified function, B represents the summarized anomalous behavior, K represents the knowledge matched through the components and functions of the examples.

$$\text{LLM}(\text{query}, \text{examples}, K_{\text{comp}}) \rightarrow C_{\text{identified}} \quad (1)$$

$$\text{LLM}(\text{query}, \text{examples}, C_{\text{identified}}, K_{\text{func}}) \rightarrow F_{\text{identified}} \quad (2)$$

$$\text{LLM}(\text{query}, \text{examples}, C_{\text{identified}}, F_{\text{identified}}) \rightarrow \text{B}_{\text{anomalous}} \quad (3)$$

Although multi-round extraction with the LLM requires more computational time, we have adopted a daily-batch processing workflow for deployment (see Section 6.1), where tickets received during the day are processed at night. Therefore, the time consumption is not a significant concern.

4.2.3 Image Data Utilization. **Image descriptions** are detailed depictions of device screens where users encounter defects, offering direct visual insights of the OS's state. These descriptions are derived from user-uploaded screenshots and videos through the preprocessing module.

We extract discriminative information from image descriptions following the experience of triage engineers. During manual ticket processing, triage engineers require only information from image data that indicates the specific page or application in which the user encountered the defect. This coarse-grained information derived from image data is sufficient to facilitate the assignment of tickets. Therefore, we input the image descriptions into an LLM and prompt it to identify the pages being described (e.g., wifi connection page) and take these as discriminative information from image data.

4.3 Clustering Module

The extracted discriminative information from **defect descriptions** and **image descriptions** will be vectorized first by a pre-trained Sentence-BERT model acge [22]. Then, the vectors are combined through a weighted sum (e.g., 0.9 for defect descriptions and 0.1 for image descriptions) to create a single vector representation for each ticket. After that, *TixFusion* employs the DBSCAN [11] algorithm to perform clustering. As one iteration of clustering with a fixed epsilon (eps) value cannot effectively cluster tickets due to potential variations in the vector distances between different groups of duplicate tickets, we execute the clustering twice. Initially, we run DBSCAN with a large eps (0.2) for coarse-grained clustering. For each group of tickets obtained from the initial clustering, we calculate the local density of the group and use it as the epsilon value for the second round of clustering.

After obtaining the aggregated tickets from the clustering, we enhance explainability by providing explanations for each group. These explanations facilitate a comprehensive understanding of each group, allowing triage engineers to assign entire groups at once without inspecting individual tickets, thus streamlining the triage process. We create these explanations based on the insights of triage engineers.

When triage engineers identify duplicate tickets, they first pinpoint discriminative information within the tickets. Subsequently, they locate similar tickets in the dataset that point to the same defect based on this information. Through this process, triage engineers gather groups of duplicate tickets and gain a basic understanding of each group.

We use an LLM to simulate this manual process and turn this basic understanding into concrete explanations. Specifically, we input discriminative information, extracted from defect descriptions and image descriptions based on triage engineers' domain knowledge, into the LLM to generate explanations. The discriminative information from defect descriptions provides essential information for distinguishing between defects, while that from image descriptions offers direct visual insights into the state of the mobile OS. Additionally, we also input the original defect descriptions to provide necessary details for generating explanations.

5 Evaluation

In this section, we conduct a comprehensive evaluation of *TixFusion* to answer the subsequent research questions (RQs):

- **RQ1:** How does *TixFusion* perform overall in aggregating duplicate tickets?
- **RQ2:** Does each component of *TixFusion* contribute significantly to *TixFusion*'s performance?
- **RQ3:** How does the labor cost of the labeling data for *TixFusion* compare to that of baseline methods?
- **RQ4:** How do we select the embedding model and the backbone LLM for *TixFusion*?

5.1 Experiment Setup

5.1.1 Dataset. We collect a total of 11,431 tickets from the production environment of H. Among these, 9,952 tickets are used for training, and the remaining 1,479 tickets are utilized to evaluate aggregation performance.

5.1.2 Implementation Details. We conduct all the experiments with two NVIDIA A30 GPUs, PyTorch 2.0.0, and CUDA toolkit 11.4. We use GLM-4-9B and GLM-4V-9B [16] as the backbone models. The fine-tuning is conducted with LLaMAFactory [59] and LoRA [19].

5.1.3 Metrics. Following existing methods [27, 32], we adopt the widely accepted Rand Index (RI) [38] to evaluate the performance of ticket aggregation. The RI calculates Precision, Recall, and F_1-score by measuring the similarity between the aggregation result and the ground truth. Given a set of tickets, let C denote the aggregation result, and G denote the ground truth. RI defines the following metrics:

- **True Positive (TP):** The number of pairs of tickets correctly identified as correlated in both C and G.
- **True Negative (TN):** The number of pairs of tickets correctly identified as not correlated in both C and G.
- **False Positive (FP):** The number of pairs of tickets incorrectly identified as correlated in C, but actually not correlated in G.
- **False Negative (FN):** The number of pairs of tickets incorrectly identified as not correlated in C, but actually correlated in G.

Using these four basic metrics, RI derives: $Precison = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, $F_1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

5.1.4 Baselines.

- **DBSCAN:** We use a pre-trained Sentence-BERT model acge [22] to vectorize tickets and subsequently apply DBSCAN clustering to aggregate duplicate tickets.
- **LiDAR [8]:** LiDAR is a deep learning-based approach to identify linked incidents in cloud systems. It integrates textual and structural information to predict connections among incidents. For this evaluation, only the textual information of tickets is used to identify links.
- **iPACK [32]:** iPACK is an incident-aware method used to aggregate duplicate customer support tickets in cloud systems. It operates in three steps: alert parsing, incident profiling, and ticket-event correlation. For this study, we focus on the ticket-event correlation step, which employs an attentive interaction network to identify and aggregate duplicate tickets.
- **COLA [27]:** COLA combines correlation mining and LLM reasoning to aggregate alerts in cloud systems. COLA leverages statistical evidence from frequent alerts and enhances LLM performance through domain-specific documents. For our evaluation, correlation mining is implemented using text similarity, while the LLM reasoning module is left unchanged.

To construct the training set for baseline methods, we calculate the semantic similarity between the labeled discriminative information of tickets. A threshold is then determined based on insights from triage engineers to establish connections among tickets. To ensure comparability between the labor costs of labeling training data for baseline methods and *TixFusion*, the size of the baseline training dataset is fixed at 2,000 tickets. This decision aligns with the labor cost comparison ratio discussed in Section 5.4.

5.2 RQ1: The Overall Performance

The comparative analysis, summarized in Table 2, highlights the superior performance of *TixFusion*, which achieves an F_1-score of 0.735, significantly outperforming the baseline approaches. The learning-based approaches, including LiDAR, iPACK, and COLA, exhibit suboptimal aggregation performance, likely due to the constrained size of their training datasets (see Section 5.1.4). This limitation restricts their ability to capture the connections between tickets effectively.

The unsupervised aggregation approach, leveraging a pre-trained embedding model combined with DBSCAN clustering, also demonstrates relatively poor performance. This outcome aligns with our finding that unsupervised aggregation approaches, lacking explicit training, struggle to identify and exploit discriminative information within tickets to differentiate between distinct defects.

TixFusion addresses these challenges by employing an LLM to extract discriminative information from tickets before applying unsupervised aggregation, resulting in a marked performance improvement.

Table 2: Performance comparison of different methods.

Method	Precision	Recall	F_1-score
DBSCAN	0.107	0.599	0.181
LiDAR	0.135	0.379	0.199
iPACK	0.124	0.348	0.182
COLA	0.203	0.657	0.310
TixFusion	**0.657**	**0.835**	**0.735**

5.3 RQ2: Ablation Study

To assess the contributions of individual components in *TixFusion*, we conduct ablation studies focusing on the extraction module, clustering module, and the utilization of image data.

Table 3: Effectiveness of each component in *TixFusion*.

Method	Precision	Recall	F_1-score
TixFusion w/o EM	0.163	0.332	0.219
TixFusion w/o ICL-e	0.586	0.714	0.643
TixFusion w/o 2R-DBSCAN	0.656	0.812	0.726
TixFusion w/o Image	0.654	0.830	0.732
TixFusion	**0.657**	**0.835**	**0.735**

5.3.1 Extraction Module. The extraction module employs an LLM to extract discriminative information from tickets, enhancing aggregation performance (detailed in Section 4.2). To further refine extraction, two additional techniques are integrated: knowledge base construction and multi-round extraction. These techniques are collectively considered as an In-Context-Learning (ICL)-based enhancement, as the constructed knowledge is embedded into multiple rounds of interaction with the LLM.

Experiments are conducted under two configurations: one without the extraction module, using raw ticket descriptions directly (denoted as w/o EM), and another with the extraction module but without the ICL-based enhancement (denoted as w/o ICL-e).

As listed in Table 3, the results confirm the effectiveness of the extraction module in identifying discriminative information, significantly improving aggregation performance. Moreover, the ICL-based enhancement further boosts the quality of extraction, contributing to superior results.

5.3.2 Clustering Module. The clustering module converts the extracted discriminative information into vectors and applies unsupervised clustering to aggregate duplicate tickets. Due to variations in the distances between groups of duplicate tickets after Sentence-BERT embedding, a fixed epsilon value in DBSCAN clustering may not yield optimal results for all groups. To address this, we propose a two-round DBSCAN clustering approach. The first round employs a larger epsilon value for coarse-grained clustering, and the resulting clusters are refined in the second round using locally calculated density-based epsilon values.

This method is compared against a single-round DBSCAN approach using a smaller, fixed epsilon value (denoted as w/o 2R-DBSCAN). The results in Table 3 demonstrate that the two-round clustering approach effectively enhances aggregation performance by dynamically adjusting epsilon values to better suit varying cluster densities.

5.3.3 Image Data Utilization. User-provided image data, including screenshots and videos illustrating reported defects, is incorporated into *TixFusion* through a multimodal LLM. We evaluate the impact of image data on both aggregation performance and explainability.

Aggregation performance. When evaluating aggregation results without utilizing image data (denoted as w/o Image), we observe only minimal improvements in the F_1-score when image data is included. This outcome is anticipated, as the granularity of the discriminative information extracted from images is relatively coarse, often lacking the specificity required for effective aggregation.

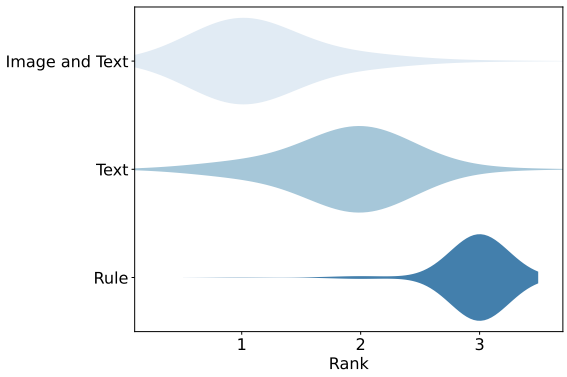


Figure 4: The ranking given by triage engineers.

Explainability. Beyond aggregation, *TixFusion* generates explanations for each group of tickets, utilizing discriminative information derived from both text (defect descriptions) and image data (screenshots and videos). This aids triage engineers in comprehending the aggregation results.

To validate the role of image data in enhancing explainability, we generate three types of explanations for each group of aggregated tickets: (1) using both text and image data, (2) using only text data, and (3) rule-based explanations. Triage engineers rank the explanations on a scale from 1 to 3, where a ranking of 1 denotes the most effective and readable explanation, while a ranking of 3 indicates the least effective and readable. Effectiveness is evaluated based on the explanation's ability to comprehensively encapsulate the discriminative information of the entire group, whereas readability is assessed in terms of clarity, simplicity, logical structure, and grammatical accuracy.

As shown in Figure 4, explanations integrating both text and image data are rated as the most effective and readable in over 86% of groups. This underscores the significant contribution of image data to the explainability of the aggregation results, providing valuable context for triage engineers.

5.4 RQ3: Labor Cost of Labeling

Existing ticket aggregation approaches rely on extensive labeled datasets to identify connections between tickets, incurring substantial labor costs. To address this limitation, *TixFusion* adopts unsupervised clustering to aggregate tickets and integrates an LLM to enhance performance (detailed in Section 3). Specifically, *TixFusion* utilizes an LLM to extract discriminative information from tickets, which are then fed into the unsupervised clustering as input. While the discriminative information extraction process still requires labeled data, such as examples of in-context learning or fine-tuning, *TixFusion* shifts the focus of labeling from identifying connections between tickets to labeling the discriminative information of individual tickets. This shift significantly reduces labeling labor costs.

To evaluate *TixFusion*'s effectiveness in reducing labeling labor costs, we conducted a study involving two groups of triage engineers tasked with labeling 1,200 tickets using two distinct methods. The total time consumption (measured in person-days) is recorded for both methods:

- **Labeling of connections:** This method, corresponding to existing ticket aggregation approaches, involves two steps: (1) ticket understanding, where engineers comprehend the defect based on text and image data within the tickets, and (2) duplicate ticket search, where engineers identify other tickets in the dataset that reference the same defect.
- **Labeling of discriminative information:** This method aligns with the labeling required by *TixFusion* and also comprises two steps: (1) ticket understanding and (2) discriminative information output, where engineers articulate the discriminative information corresponding to the defect based on their understanding of the ticket.

The total time consumption and the breakdown of each step for both labeling methods are summarized in Table 4. Labeling discriminative information for 1,200 tickets using *TixFusion*'s method required only 2.0 person-days, reflecting a 79% reduction in labor compared to labeling connections. This significant reduction primarily results from eliminating the most time-intensive step in the connection labeling process: duplicate ticket search, where engineers must scan the dataset to identify related tickets.

Table 4: Comparison of labeling methods.

Labeling	Step 1 Prop.	Step 2 Prop.	Time Consumption
Connections	0.15	0.85	9.6 person-days
Discriminative	0.5	0.5	2.0 person-days

Moreover, the labor-saving potential of *TixFusion*'s labeling method is expected to increase as the dataset size grows. The time required for duplicate ticket search scales with the size of the dataset, whereas the discriminative information labeling process remains relatively constant regardless of dataset size.

To provide additional context, existing aggregation methods in the baselines have leveraged labeled datasets containing hundreds of thousands of tickets for large-scale deployments in cloud systems. Given the complexity of mobile OS, which shares similarities with cloud systems—such as intricate ticket categories and the generation of thousands of tickets daily [21]—it is reasonable to anticipate that similar labeling requirements would apply for large-scale deployments in our production environment. For *TixFusion*, we estimate that approximately 100,000 labeled tickets will be required to ensure robust performance in large-scale deployment. Based on this scale, we project that *TixFusion* will require 167 person-days to label the dataset, representing an 89% reduction compared to the 1,480 person-days required by baseline methods.

5.5 RQ4: Selection of Embedding Models and LLMs

This section presents a comparative analysis of the embedding model and LLM selections utilized by *TixFusion*.

For the embedding model, we employ acge [22], which serves two purposes: (1) identifying examples for ICL and (2) vectorizing tickets for the clustering module. To ensure optimal performance, we select acge based on the results of the MTEB benchmark [36], a widely used text embedding evaluation framework. We compare acge against two other embedding models from the same benchmark. As shown in Table 5, acge outperforms the alternatives, confirming its suitability for *TixFusion*.

For the LLM, we utilize GLM-4-9B [16] to extract discriminative information from tickets. Its performance is compared against two similarly sized state-of-the-art models: Qwen-2.5-7B [44] and InternLM-2.5-7B [2]. The comparison results, also presented in Table 5, demonstrate that GLM-4-9B achieves the best performance among the three models, further validating its selection for *TixFusion*.

Table 5: Comparison of embedding models and LLMs.

Method	Precision	Recall	F_1-score
ICL-xiaobu_v2 [42]	0.644	0.823	0.723
ICL-conan_v1 [29]	0.635	0.826	0.718
ICL-acge [22]	**0.657**	**0.835**	**0.735**
Clustering-xiaobu_v2 [42]	0.581	0.729	0.647
Clustering-conan_v1 [29]	0.554	0.676	0.609
Clustering-acge [22]	**0.657**	**0.835**	**0.735**
LLM-Qwen-2.5-7B [44]	0.569	0.669	0.615
LLM-InternLM-2.5-7B [2]	0.574	0.626	0.599
LLM-GLM-4-9B [16]	**0.657**	**0.835**	**0.735**

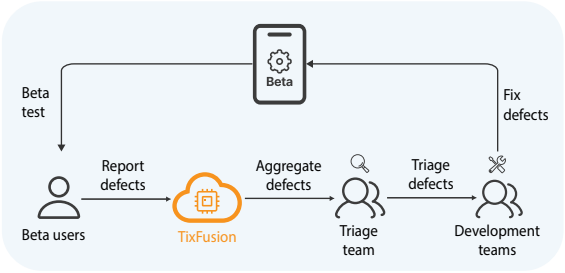


Figure 5: The deployment of *TixFusion* on the ticket handling system.

6 Discussion

6.1 Deployment

We have integrated the proposed method, *TixFusion*, into the ticket handling system of *H* for trial implementation. Over a stable operation period exceeding three months, *TixFusion* has processed more than 200,000 tickets and has improved the efficiency of triage engineers by 3.78 times. The deployment architecture of *TixFusion* is illustrated in Figure 5.

The system employs a daily batch-processing workflow to process tickets. All tickets submitted by beta users are forwarded by the

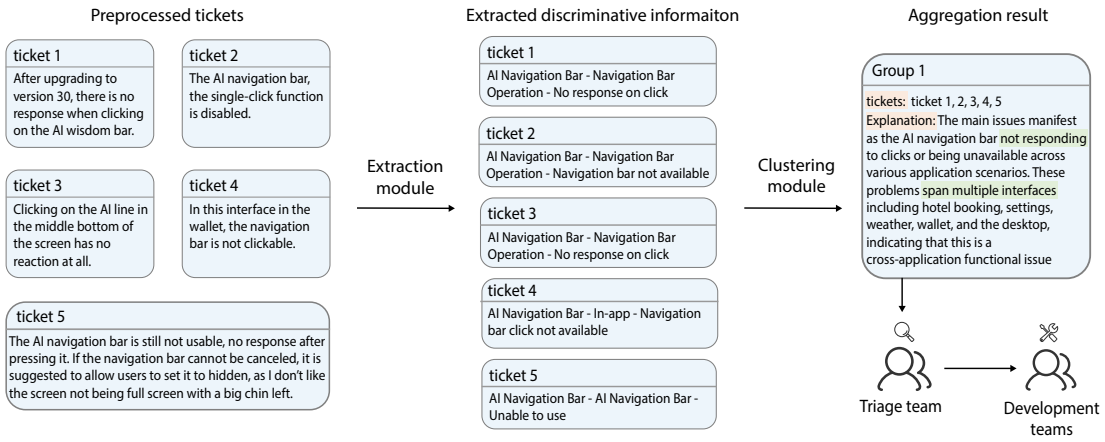


Figure 6: The workflow of an example.

ticket handling system to a GPU server hosting *TixFusion*. Within the server, *TixFusion* executes a series of operations, including preprocessing, discriminative information extraction, and clustering. Once processing is completed, *TixFusion* returns the aggregation results to the ticket handling system. These results consist of grouped tickets, each accompanied by detailed explanations for the respective groups. The ticket handling system then presents these aggregated groups and their explanations to triage engineers, facilitating efficient ticket assignment.

6.2 Case Study

To comprehensively demonstrate the workflow of *TixFusion* and its pivotal role in the triage process, we present a detailed case study. This case examines a group of duplicate tickets identified by *TixFusion*. To uphold strict user privacy standards, all sensitive information, including screenshots, recordings, and related data, has been excluded from the description. The group comprises five tickets, all reporting the same system defect: the unresponsiveness or malfunction of the "AI navigation bar".

As shown in Figure 6, the tickets are first processed through the preprocessing and discriminative information extraction stages. During this phase, discriminative information is extracted from the tickets to serve as input for subsequent analysis. It is then passed to the clustering module, where they are vectorized using Sentence-BERT and aggregated using the DBSCAN algorithm. The clustering module identifies these five tickets as belonging to the same group and generates a concise explanation based on both the extracted discriminative information and the original ticket content. This explanation provides a synthesized overview of the grouped tickets, allowing triage engineers to quickly understand the aggregation results. Consequently, the engineers can assign the entire group collectively, eliminating the need to review each ticket individually.

Further analysis is provided in Figure 7, which presents the cosine similarity matrices comparing the original defect descriptions and the extracted discriminative information. The relatively low cosine similarities among the original descriptions indicate

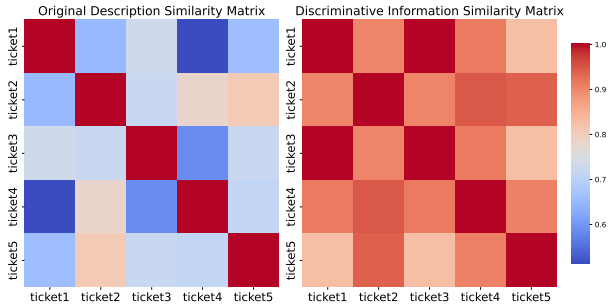


Figure 7: Similarity matrices of the example.

challenges in grouping the tickets directly based on raw input, potentially leading to reduced aggregation performance. However, the higher cosine similarities among the extracted discriminative information highlight the effectiveness of the extraction step in enhancing aggregation accuracy.

6.3 Lessons Learned

6.3.1 Explainability of the Results. Initially, we only presented aggregated tickets to the triage engineers. However, they feedbacked that this output was of limited assistance, as they still needed to examine multiple tickets within each group to decide appropriate development teams for assignment. Therefore, we utilized an LLM to generate detailed explanations for each group of aggregated tickets, thereby aiding triage engineers in comprehending the aggregation results.

6.3.2 Presentation of the Results. Given that the purpose of our framework is to facilitate triage engineers, it is essential to prioritize their needs when presenting the results. A clumsy presentation format could potentially compromise the practical effectiveness of the framework. Consequently, we iteratively refined the presentation format of the aggregation results based on the feedback from triage engineers and ultimately integrated it into their existing triage platform.

6.4 Generality of Results

Adaptability. This research is based on data collected from a top-tier global mobile OS provider, which ensures the representativeness of our conclusions and methodologies. Furthermore, the methodological framework, which utilizes LLMs to extract discriminative information from data and combines them with traditional machine learning techniques, demonstrates a high degree of versatility. While initially designed for the scenarios addressed in this study, the framework can be effectively extended to other domains that involve natural language-rich data. Examples include the analysis of failure reports in cloud systems and the processing of support tickets in customer service operations. This versatility highlights the potential for our method to serve as a reference for advancing research and applications in related fields.

Robustness. Our method exhibits significant robustness, even under varying data availability conditions. While image data is incorporated into the ticket aggregation process to enhance accuracy and explainability, the method retains its effectiveness in scenarios where image data is unavailable. This resilience ensures that the method maintains consistent performance and reliability, regardless of the completeness of the data. Such robustness underscores the suitability of *TixFusion* for deployment in heterogeneous environments, where data completeness may vary significantly.

Flexibility. *TixFusion*'s modular design endows it with exceptional flexibility, granting mobile OS providers substantial autonomy in adapting the framework to their specific needs. While preserving the overall effectiveness of the framework, mobile OS providers can customize various components, such as substituting clustering algorithms or selecting LLMs tailored to their application contexts and operational scales. This high degree of adaptability is particularly critical in dynamic business environments, enabling the framework to accommodate diverse requirements and challenges. By allowing organizations to optimize the solution based on their unique demands, our method maximizes operational efficiency and effectiveness, further solidifying its value in real-world applications.

7 Related Work

7.1 Ticket Aggregation

Ticket aggregation methods aim to identify duplicate tickets that refer to the same bug, cloud incident, or system defect, allowing engineers to focus on resolving the issue and avoid wasting time inspecting redundant tickets. LinkCM [18] aggregates customer report tickets by matching them with system incidents to improve customer service efficiency. LiDAR [8] aggregates related incidents by combining semantic and service dependency representations to facilitate incident mitigation. Warden [28] groups related alerts to detect potential failures in cloud systems, enhancing incident management efficiency and reducing downtime. GRILA [9] aggregates incidents by learning from the cascade graph of cloud failures to narrow the scope of incidents and improve incident management efficiency. OAS [5] aggregates alerts by learning their semantics and behavioral representations, summarizing grouped alerts to aid maintenance engineers in understanding system failures. iPACK [32] aggregates customer support tickets using co-occurrence patterns to improve customer ticket management efficiency. COLA [27] aggregates alerts by combining correlation mining and LLM to promote cloud system fault resolution.

Despite the excellent results achieved by the aforementioned aggregation methods, they typically require extensive labeled data for training, which incurs significant labor costs. To address this, we adopt unsupervised clustering to aggregate duplicate tickets and utilize an LLM to enhance its aggregation performance by extracting discriminative information from tickets.

7.2 LLM for Software Engineering

The rapid advancement of Large Language Models (LLMs) has led to their widespread application across various domains of software engineering, including code generation [6, 13, 14, 33, 35, 46, 54], log analysis [20, 24, 30, 31, 34, 53, 55, 60] , and cloud system maintenance [7, 23, 25, 40, 48, 50, 57].

In the domain of code generation, significant progress has been made by integrating LLMs with specific tasks. For instance, DCGen [46] translates webpage designs into corresponding UI code, streamlining the development process. Similarly, GPTDroid [35] utilizes LLMs to generate GUI test scripts by passing page information to the model, thereby automating the testing phase. For log analysis, UniLog [53] proposes an automatic logging framework based on in-context learning (ICL). LILAC [24] conducts log parsing with improved ICL techniques.

In the realm of cloud system maintenance, LLMs have been utilized to improve the management and resolution of incidents. OASIS [25] generates human-readable summaries of system outages, aiding maintenance engineers in quickly understanding the context and severity of issues. RCAgent [48] conducts root cause analysis on cloud incidents through a tool-augmented agent framework, enabling more effective resolution of cloud incidents. Different from the aforementioned methods, which use LLMs as output generators, we leverage LLM as an auxiliary tool alongside the traditional clustering algorithm to aggregate tickets.

8 Conclusion

In this paper, we introduce *TixFusion*, an LLM-augmented ticket aggregation framework, specifically designed to streamline the triage process for tickets generated during the beta testing of mobile OS. By leveraging LLMs to extract discriminative information and enhance aggregation performance, *TixFusion* effectively achieves high-quality aggregation with minimal reliance on labor-intensive data labeling. We propose an in-context learning-based extraction method to improve the quality of discriminative information extraction. Furthermore, by integrating a multimodal LLM, *TixFusion* incorporates image data into the ticket aggregation process for the first time, enabling a more comprehensive analysis of tickets. To assist triage engineers, *TixFusion* generates concise and interpretable explanations for each group of aggregated tickets, facilitating efficient decision-making. Extensive experiments conducted on a ticket dataset from the production environment of a top-tier global mobile OS provider *H* demonstrate that *TixFusion* outperforms all state-of-the-art methods. Additionally, *TixFusion* has been deployed in *H* for over three months, during which it has achieved a 3.78-fold increase in the processing efficiency of triage engineers.

References

[1] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. arXiv:2407.21787 [cs.LG] https://arxiv.org/abs/2407.21787

[2] Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. 2024. InternLM2 Technical Report. arXiv:2403.17297 [cs.CL]

[3] 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)*. 111–120. doi:10.1109/ICSE-SEIP.2019.00020

[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] 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*. 1646–1657.

[6] Yinghao Chen, Zehao Hu, Chen Zhi, Junxiao Han, Shuiguang Deng, and Jianwei Yin. 2024. ChatUniTest: A Framework for LLM-Based Test Generation. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering* (Porto de Galinhas, Brazil) *(FSE 2024)*. Association for Computing Machinery, New York, NY, USA, 572–576. doi:10.1145/3663529.3663801

[7] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, et al. 2024. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*. 674–688.

[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] 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.

[10] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234* (2022).

[11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, Vol. 96. 226–231.

[12] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6491–6501.

[13] Shuzheng Gao, Cuiyun Gao, Wenchao Gu, and Michael Lyu. 2024. Search-Based LLMs for Code Optimization. *arXiv preprint arXiv:2408.12159* (2024).

[14] Shuzheng Gao, Xin-Cheng Wen, Cuiyun Gao, Wenxuan Wang, Hongyu Zhang, and Michael R Lyu. 2023. What makes good in-context demonstrations for code intelligence tasks with llms?. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 761–773.

[15] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL] https://arxiv.org/abs/2312.10997

[16] Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools. arXiv:2406.12793

[17] GSMA. 2023. The State of Mobile Internet Connectivity 2023. https://www.gsma.com/r/wp-content/uploads/2023/10/The-State-of-Mobile-Internet-Connectivity-Report-2023.pdf. Accessed: 2024-11-22.

[18] 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.

[19] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL] https://arxiv.org/abs/2106.09685

[20] Junjie Huang, Zhihan Jiang, Zhuangbin Chen, and Michael R Lyu. 2024. ULog: Unsupervised Log Parsing with Large Language Models through Log Contrastive Units. *arXiv preprint arXiv:2406.07174* (2024).

[21] 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.

[22] INTSIG. 2024. *acge_text_embedding*. https://huggingface.co/aspire/acge_text_embedding Accessed: 2024-12-28.

[23] Yuxuan Jiang, Chaoyun Zhang, Shilin He, Zhihao Yang, Minghua Ma, Si Qin, Yu Kang, Yingnong Dang, Saravan Rajmohan, Qingwei Lin, et al. 2024. Xpert: Empowering incident management with query recommendations via large language models. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[24] Zhihan Jiang, Jinyang Liu, Zhuangbin Chen, Yichen Li, Junjie Huang, Yintong Huo, Pinjia He, Jiazhen Gu, and Michael R Lyu. 2024. Lilac: Log parsing using llms with adaptive parsing cache. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 137–160.

[25] Pengxiang Jin, Shenglin Zhang, Minghua Ma, Haozhe Li, Yu Kang, Liqun Li, Yudong Liu, Bo Qiao, Chaoyun Zhang, Pu Zhao, et al. 2023. Assess and summarize: Improve outage understanding with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1657–1668.

[26] Mateja Kocbek and Marjan Hericko. 2013. Beta Testing of a Mobile Application: A Case Study.. In *SQAMIA*. Citeseer, 29–34.

[27] Jinxi Kuang, Jinyang Liu, Junjie Huang, Renyi Zhong, Jiazhen Gu, Lan Yu, Rui Tan, Zengyin Yang, and Michael R Lyu. 2024. Knowledge-aware Alert Aggregation in Large-scale Cloud Systems: a Hybrid Approach. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*. 369–380.

[28] 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 *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 131–146.

[29] Shiyu Li, Yang Tang, Shizhe Chen, and Xi Chen. 2024. Conan-embedding: General Text Embedding with More and Better Negative Samples. arXiv:2408.15710 [cs.CL] https://arxiv.org/abs/2408.15710

[30] Yichen Li, Yintong Huo, Zhihan Jiang, Renyi Zhong, Pinjia He, Yuxin Su, Lionel C Briand, and Michael R Lyu. 2024. Exploring the Effectiveness of LLMs in Automated Logging Statement Generation: An Empirical Study. *IEEE Transactions on Software Engineering* (2024).

[31] Yichen Li, Yintong Huo, Renyi Zhong, Zhihan Jiang, Jinyang Liu, Junjie Huang, Jiazhen Gu, Pinjia He, and Michael R Lyu. 2024. Go static: Contextualized logging statement generation. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 609–630.

[32] Jinyang Liu, Shilin He, Zhuangbin Chen, Liqun Li, Yu Kang, Xu Zhang, Pinjia He, Hongyu Zhang, Qingwei Lin, Zhangwei Xu, et al. 2023. Incident-aware duplicate ticket aggregation for cloud systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2299–2311.

[33] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2024. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems* 36 (2024).

[34] Yilun Liu, Shimin Tao, Weibin Meng, Feiyu Yao, Xiaofeng Zhao, and Hao Yang. 2024. LogPrompt: Prompt Engineering Towards Zero-Shot and Interpretable Log Analysis. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings* (Lisbon, Portugal) *(ICSE-Companion '24)*. Association for Computing Machinery, New York, NY, USA, 364–365. doi:10.1145/3639478.3643108

[35] Zhe Liu, Chunyang Chen, Junjie Wang, Mengzhuo Chen, Boyu Wu, Xing Che, Dandan Wang, and Qing Wang. 2024. Make llm a testing expert: Bringing human-like interaction to mobile gui testing via functionality-aware decisions. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[36] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. MTEB: Massive Text Embedding Benchmark. *arXiv preprint arXiv:2210.07316* (2022). doi:10.48550/ARXIV.2210.07316

[37] Chanathip Pornprasit and Chakkrit Kla Tantithamthavorn. 2023. DeepLineDP: Towards a Deep Learning Approach for Line-Level Defect Prediction. *IEEE Transactions on Software Engineering* 49, 1 (2023), 84–98. doi:10.1109/TSE.2022.3144348

[38] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66, 336 (1971), 846–850.

[39] N Reimers. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084* (2019).

[40] Manish Shetty, Yinfang Chen, Gagan Somashekar, Minghua Ma, Yogesh Simmhan, Xuchao Zhang, Jonathan Mace, Dax Vandevoorde, Pedro Las-Casas, Shachee Mishra Gupta, et al. 2024. Building AI Agents for Autonomous Clouds: Challenges and Design Principles. *arXiv preprint arXiv:2407.12165* (2024).

[41] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. arXiv:2408.03314 [cs.LG] https://arxiv.org/abs/2408.03314

[42] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. 2020. Circle loss: A unified perspective of pair similarity optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 6398–6407.

[43] Yongqian Sun, Binpeng Shi, Mingyu Mao, Minghua Ma, Sibo Xia, Shenglin Zhang, and Dan Pei. 2024. ART: A Unified Unsupervised Framework for Incident Management in Microservice Systems. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 1183–1194.

[44] Qwen Team. 2024. Qwen2.5: A Party of Foundation Models. https://qwenlm.github.io/blog/qwen2.5/

[45] thenextweb. 2025. *Android vulnerability lets hackers hijack your phone with malicious videos.* https://thenextweb.com/news/google-android-vulnerability-malicious-video Accessed: 2025-01-14.

[46] Yuxuan Wan, Chaozheng Wang, Yi Dong, Wenxuan Wang, Shuqing Li, Yintong Huo, and Michael R Lyu. 2024. Automatically Generating UI Code from Screenshot: A Divide-and-Conquer-Based Approach. *arXiv preprint arXiv:2406.16386* (2024).

[47] Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 19206–19214.

[48] Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Jihong Wang, Fengbin Yin, Lunting Fan, Lingfei Wu, and Qingsong Wen. 2024. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4966–4974.

[49] Zexin Wang, Minghua Ma, Ze Li, Chetan Bansal, Saravan Rajmohan, Qingwei Lin Qingwei lin, and Dongmei Zhang. 2024. Large Language Models Can Provide Accurate and Interpretable Incident Triage. In *2024 International Symposium on Software Reliability Engineering*. https://www.microsoft.com/en-us/research/publication/large-language-models-can-provide-accurate-and-interpretable-incident-triage/

[50] Zexin Wang, Minghua Ma, Ze Li, Chetan Bansal, Saravan Rajmohan, Qingwei Lin Qingwei lin, and Dongmei Zhang. 2024. Large Language Models Can Provide Accurate and Interpretable Incident Triage. In *2024 International Symposium on Software Reliability Engineering*. https://www.microsoft.com/en-us/research/publication/large-language-models-can-provide-accurate-and-interpretable-incident-triage/

[51] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[52] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference for Problem-Solving with Language Models. arXiv:2408.00724 [cs.AI] https://arxiv.org/abs/2408.00724

[53] Junjielong Xu, Ziang Cui, Yuan Zhao, Xu Zhang, Shilin He, Pinjia He, Liqun Li, Yu Kang, Qingwei Lin, Yingnong Dang, et al. 2024. UniLog: Automatic Logging via LLM and In-Context Learning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 1–12.

[54] Junjielong Xu, Ying Fu, Shin Hwei Tan, and Pinjia He. 2024. Aligning LLMs for FL-free Program Repair. *arXiv preprint arXiv:2404.08877* (2024).

[55] Junjielong Xu, Ruichun Yang, Yintong Huo, Chengyu Zhang, and Pinjia He. 2024. DivLog: Log Parsing with Prompt Enhanced In-Context Learning. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–12.

[56] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. 2024. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data* 18, 6 (2024), 1–32.

[57] Xuchao Zhang, Supriyo Ghosh, Chetan Bansal, Rujia Wang, Minghua Ma, Yu Kang, and Saravan Rajmohan. 2024. Automated root causing of cloud incidents using in-context learning with GPT-4. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 266–277.

[58] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2024. A Survey of Large Language Models. arXiv:2303.18223 [cs.CL] https://arxiv.org/abs/2303.18223

[59] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Association for Computational Linguistics, Bangkok, Thailand. http://arxiv.org/abs/2403.13372

[60] Aoxiao Zhong, Dengyao Mo, Guiyang Liu, Jinbu Liu, Qingda Lu, Qi Zhou, Jiesheng Wu, Quanzheng Li, and Qingsong Wen. 2024. LogParser-LLM: Advancing Efficient Log Parsing with Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4559–4570.