Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 91-103
DOI: https://doi.org/10.51903/jtie.v5i2.538

Cost-Aware LLM-Style Routing for AIOps Log Analysis: Log Parsing, Anomaly Detection, Fault Diagnosis, and Incident Summarization on LogEval Task Files

Chenyu Li[1], Ge Liu[*2], Zoe Zhao[3]
Email: fretin13@gmail.com
[1]*Applied Analytics, Columbia University, NY, USA*
[2]*Computer Science, USC, CA, USA*
[3]*Computer Science, UCSD, CA, USA*
*Corresponding Author

Abstract
This study investigates a local and cost-aware routing framework for AIOps log analysis using the LogEval benchmark. The evaluation covers four tasks: log parsing, anomaly detection, fault diagnosis, and incident summarization. Instead of relying on external large language model APIs, the experiment implements deterministic local policies that simulate zero-shot and few-shot LLM-style inference under controlled token-cost and latency assumptions. Six approaches were compared: regex normalization, TF-IDF with machine learning, a local character-based classifier, zero-shot policy, few-shot retrieval policy, and a routing cascade. At a risk threshold of 0.20, the router directed only 12.9% of queries to the few-shot retrieval policy while achieving parsing accuracy of 0.991, anomaly F1-score of 1.000, diagnosis accuracy of 1.000, ROUGE-L of 0.743, and BLEU-1 of 0.814. The routing strategy reduced simulated token cost by 80.1% compared with always using few-shot retrieval. Additional unseen-template evaluation revealed limited generalization for closed-label classifiers and retrieval methods when encountering unseen patterns. The findings indicate that routing can effectively reduce AIOps inference costs, while further validation with real LLMs and stronger generalization testing are required before production deployment.

Keywords: *AIOps; Anomaly detection; Cost-aware inference; Incident summarization; Log analysis.*

I. INTRODUCTION

Logs are among the most stable observability sources in software, cloud, high-performance computing, and networked systems. Operators use logs to detect abnormal states, reconstruct failure sequences, and communicate incident context after alerts. Classical studies of console logs showed that operational messages contain both routine state transitions and rare failure evidence (Oliner & Stearley, 2007; Xu et al., 2009). Modern AIOps systems build on the same signal but must process much larger streams and more heterogeneous message formats, which makes manual triage impractical.

Large language models offer a new design point for log analysis because they can follow task instructions, read irregular text, and use examples in context (Brown et al., 2020). However, high-quality LLM inference also introduces latency, governance, and token-cost concerns. A routing architecture is therefore attractive: routine or high-confidence records can be handled locally, while ambiguous or high-risk records are escalated to a more expensive endpoint. This idea is consistent with cost-aware model cascading, where cheaper models answer easier inputs and expensive models are reserved for cases where their additional context is likely to matter (Chen et al., 2023).

Received on Juni 2025; Revised on July 2025; Accepted on August 2025; Published on August 2025

The research question is direct: can a local model plus a risk-based router retain most of the quality of a few-shot LLM-style endpoint while reducing cost across multiple AIOps log tasks? This paper evaluates that question on the LogEval task files, which organize log analysis into parsing, anomaly detection, fault diagnosis, and summarization. The results are reported as a local experiment on the task data rather than as a leaderboard submission or as evidence from external API-based LLM calls.

The study makes three contributions. First, it replaces the earlier simulation framing with direct experiments on the LogEval task JSON files. Second, it compares six methods under fixed splits and reports task quality together with route rate, simulated latency, and simulated token cost. Third, it adds residual-error analysis and an unseen-template parsing split so that strong within-distribution results are not mistaken for robustness to novel log templates.

The motivation is not only accuracy. In an operations center, a model that flags an anomaly but cannot explain the fault still leaves responders with manual triage work. A model that explains faults but is called for every routine log can become expensive at scale. A useful AIOps design therefore needs to balance template recovery, anomaly recall, fault labeling, summary usefulness, latency, and cost in the same evaluation.

II. LITERATURE REVIEW

Automated log analysis has long been organized around parsing, anomaly detection, and diagnosis. Log parsing converts unstructured messages into event templates that downstream analytics can count and join. Drain introduced an online fixed-depth tree for efficient template extraction (He et al., 2017), and later benchmark work showed why consistent parser evaluation is essential across datasets and tools (Zhu et al., 2019).

Log anomaly detection moved from statistical and workflow features to sequence models and learned representations. DeepLog modeled log keys as sequences for anomaly detection and diagnosis (Du et al., 2017). LogAnomaly added sequential and quantitative signals for unstructured logs (Meng et al., 2019), while LogRobust studied unstable templates across changing software versions (Zhang et al., 2019). Surveys emphasize that deep learning can improve pattern recognition, but evaluation remains sensitive to labels, preprocessing, and class imbalance (He et al., 2021; Landauer et al., 2023).

Transformer and LLM research expanded the scope of log analysis beyond classification. Attention-based models support scalable sequence representation (Vaswani et al., 2017), BERT demonstrated strong text classification after task adaptation (Devlin et al., 2019), and few-shot prompting showed that large models can infer tasks from examples (Brown et al., 2020). For

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 91-103
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

AIOps, these capabilities motivate natural-language fault interpretation and incident summarization, but the same capabilities raise questions about cost and latency.

LogEval is directly relevant because it evaluates LLM behavior across log parsing, anomaly detection, fault diagnosis, and log summarization (Cui et al., 2024). The benchmark framing is useful for routing research because a router should be assessed across multiple operational tasks, not only on one binary anomaly label. The present study follows that task scope while using deterministic local endpoint policies so that the routing behavior is repeatable and auditable.

Cost-aware LLM routing addresses the practical problem that not every input requires the same model. FrugalGPT formalized cascaded model selection and showed that cost can be reduced when inexpensive models handle easy cases and expensive models are reserved for harder cases (Chen et al., 2023). The same principle applies to logs: low-risk routine records should not consume the same budget as ambiguous, severe, or incident-critical records.

III. RESEARCH METHOD

The experiments use the English LogEval task files. The few-shot versions are used as the label source because they contain the complete diagnosis label set, while the zero-shot versions are used to estimate prompt length for the cost simulation. The task files share the same input-output schema but are not a single unified multi-task table. Therefore, each task is evaluated on its own held-out split rather than on the same 4,000 records.

Table 1. LogEval task files and experimental splits.

Task	Records	Label space	Train	Held-out test	Split rule
Log parsing	4,000	269 templates	3,034	966	stratified over non-singleton templates; singleton templates kept in training
Log anomaly detection	4,000	2 labels	3,000	1,000	stratified by anomaly label
Fault diagnosis	4,000	8 fault labels	3,000	1,000	stratified by fault label
Incident summarization	200	114 unique references	150	50	random split because many references are singleton summaries

Table 1 summarizes the task files and splits. Parsing, anomaly detection, and diagnosis each contain 4,000 records. Incident summarization contains 200 log sequences, each composed of multiple log messages. The anomaly file is highly imbalanced, with 143 abnormal records and 3,857 normal records. The diagnosis file contains eight fault labels, shown in Table 2 and visualized in Figure 2. The task files do not expose a source-family column, so this study does not report per-source-family routing accuracy. Figure 2 plots the same diagnosis distribution and

highlights the imbalance between common virtual-machine, management-port, and mirror labels and the rarer virtual-machine-port label.

Table 2. Fault-label distribution in the diagnosis task file.

Fault label	Records
Mirror does not specify output	650
Management port down	650
Virtual Machine Startup Failure	650
Processor CPU Caterr	569
Memory Throttled Uncorrectable Error Correcting Code	569
Hard Disk Drive Control Error Computer System Bus Short Circuit Programmable Gate Array Device Unknown	568
Operational port down	272
Virtual Machine Port Down	72

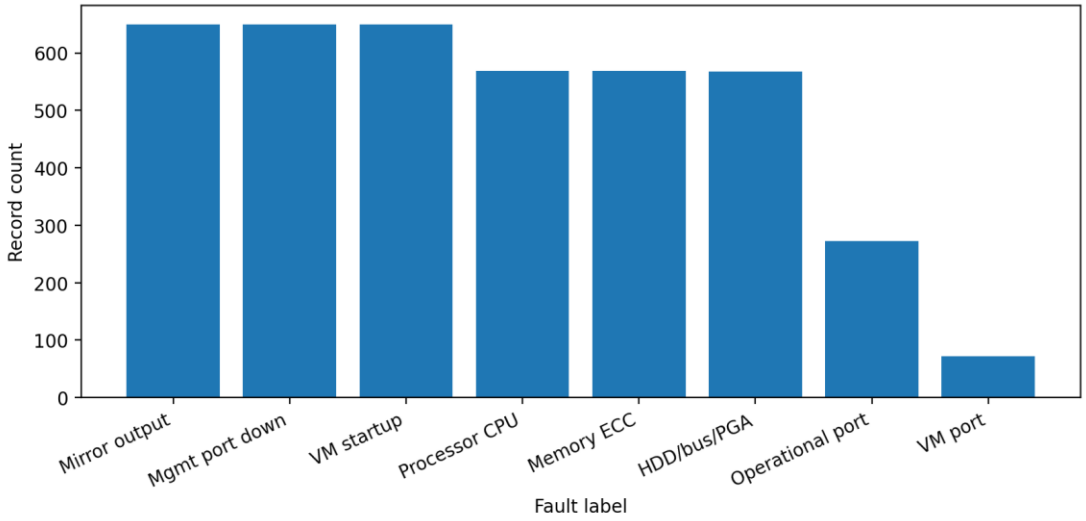

Figure 2. Fault-label distribution in the LogEval diagnosis task file.

The evaluated tasks and metrics are listed in Table 3. Parsing and diagnosis use accuracy and macro-F1 because they are multi-class tasks. Anomaly detection uses accuracy, precision, recall, and F1 with abnormal as the positive label. Summary evaluation uses token-based ROUGE-L and BLEU-1. These automatic summary metrics measure lexical overlap only; they do not replace a human assessment of whether a summary is useful during an incident.

Table 3. Task schema and evaluation metrics.

Task	Prediction target	Primary metrics	Output type
Log parsing	Normalized template string	Accuracy and macro-F1	Template text with <*> wildcards
Log anomaly detection	normal / abnormal	Accuracy, precision, recall, F1	Binary decision
Fault diagnosis	8-way fault label	Accuracy and macro-F1	Fault class
Incident summarization	Reference summary text	ROUGE-L and BLEU-1	One concise summary

Table 4 describes the six compared systems. The names zero-shot and few-shot refer to endpoint style rather than to real LLM calls. The zero-shot endpoint is a deterministic rule policy. The few-

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 91-103
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

shot endpoint is a deterministic retrieval policy that selects nearest examples from the training split. This design isolates the routing problem and keeps the experiment repeatable, while future work can replace these endpoints with API-based or open-weight LLMs.

Table 4. Model and routing configurations.

Method	Implementation	Data used	Operational role
Regex baseline	Hand-written normalization and keyword rules	No training data	Transparent low-cost reference
TF-IDF + ML classifier	Word n-gram TF-IDF with Naive Bayes or nearest-neighbor retrieval for summaries	Training split	Low-cost supervised baseline
Local character classifier	Character n-gram classifier; balanced linear model for anomaly and character n-gram classifier/retriever for other tasks	Training split	Default local stage for the router
Zero-shot local policy	Task-description rules without examples	No training examples	Deterministic zero-shot endpoint proxy
Few-shot retrieval policy	Nearest-example retrieval using character n-gram similarity	Training split as example pool	Deterministic few-shot endpoint proxy
Router cascade	Local character classifier first; escalate when risk score >= 0.20	Training split plus risk rule	Cost-aware cascade

Figure 1 shows the routing architecture. Each task query first receives local features and a risk score based on local confidence, input complexity, and severity-related lexical cues. Diagnosis records use confidence-based routing without a severity boost because the fault labels in this task are already strongly indicated by label-specific terms. Queries below the threshold keep the local output; queries above it are sent to the few-shot retrieval endpoint.

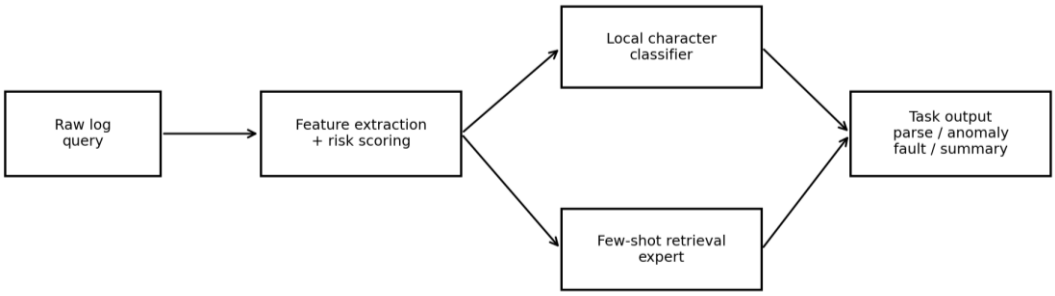

Figure 1. Cost-aware router architecture used in the experiment.

The cost analysis is a token-cost simulation, not a claim about a specific commercial provider. Zero-shot and few-shot prompt lengths are estimated from the corresponding LogEval instructions and inputs. Local methods receive small fixed compute charges per 1,000 queries. Endpoint latencies are deterministic service-latency assumptions: 350 ms for zero-shot and 620 ms for few-shot. The router always pays the local cost and pays the few-shot cost only for routed queries.

To address generalization beyond within-template records, the study also includes an unseen-template parsing split. In that split, 21 templates are entirely held out from the training pool, producing 1,070 parsing test records. This split is stricter than the main record-level split because closed-label classifiers cannot predict labels that never appear during training.

IV. FINDINGS DISCUSSION

The main results are shown in Tables 5 through 10 and summarized visually in Figures 3 through 7. At threshold 0.20, the router routes 12.9% of held-out task queries to the few-shot retrieval policy. It improves parsing over the local character classifier, preserves perfect anomaly and diagnosis scores on the held-out record splits, and obtains the strongest automatic summary ROUGE-L among the evaluated systems. The same result should be interpreted as a controlled task-file result rather than as a production robustness claim, because the anomaly and diagnosis labels contain strong lexical cues.

Table 5. Log parsing results on the held-out parsing split.

Method	Parse accuracy	Parse macro-F1
Regex baseline	0.862	0.400
TF-IDF + ML classifier	0.977	0.821
Local character classifier	0.983	0.859
Zero-shot local policy	0.862	0.400
Few-shot retrieval policy	0.991	0.916
Router cascade	0.991	0.912

Table 5 shows that the few-shot retrieval policy and the router both reach parsing accuracy of 0.991. The router makes nine template mismatches on 966 held-out parsing records, compared with sixteen mismatches for the local character classifier. The remaining gap is mostly a template-boundary issue rather than an anomaly or fault-label issue.

Table 6. Anomaly detection results on the held-out anomaly split.

Method	Accuracy	Precision	Recall	F1
Regex baseline	0.911	0.243	0.694	0.360
TF-IDF + ML classifier	0.998	0.947	1.000	0.973
Local character classifier	1.000	1.000	1.000	1.000
Zero-shot local policy	0.911	0.243	0.694	0.360
Few-shot retrieval policy	1.000	1.000	1.000	1.000
Router cascade	1.000	1.000	1.000	1.000

Table 6 shows that the local character classifier, the few-shot retrieval policy, and the router all achieve anomaly F1 = 1.000 on the held-out split. This reflects the strong lexical separability of this LogEval anomaly file and should not be generalized to noisier incident streams without additional validation.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 91-103
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

Table 7 shows that the TF-IDF classifier, the local character classifier, and the router reach fault accuracy = 1.000 under the held-out record split. The result is useful for comparing routing cost, but it is also a limitation: the diagnosis task contains fault labels that are often explicitly signaled by words such as CPU, memory, mirror, management port, and interface state.

Table 7. Fault diagnosis results on the held-out diagnosis split.

Method	Fault accuracy	Fault macro-F1
Regex baseline	0.368	0.325
TF-IDF + ML classifier	1.000	1.000
Local character classifier	1.000	1.000
Zero-shot local policy	0.368	0.325
Few-shot retrieval policy	0.999	0.999
Router cascade	1.000	1.000

Table 8 shows that summary evaluation is the weakest part of the experiment. The router obtains ROUGE-L = 0.743 and BLEU-1 = 0.814, slightly above the local character classifier on ROUGE-L but not on BLEU-1. Because the summary task contains only 50 held-out sequences and many singleton references, these automatic scores should be treated as overlap indicators rather than as evidence of incident-message usefulness.

Table 8. Incident summarization results on the held-out summary split.

Method	ROUGE-L	BLEU-1
Regex baseline	0.316	0.290
TF-IDF + ML classifier	0.727	0.820
Local character classifier	0.736	0.822
Zero-shot local policy	0.316	0.290
Few-shot retrieval policy	0.743	0.814
Router cascade	0.743	0.814

Table 9 shows the operating-point advantage of routing. Sending every query to the few-shot endpoint costs $0.1641 per 1,000 task queries in the simulation, while the router costs $0.0326. The corresponding cost reduction is 80.1%, and average latency falls from 620.0 ms/query to 80.5 ms/query. Figure 3 places the four main quality metrics on one radar chart. It shows that the router is close to the strongest endpoint on parsing, anomaly detection, and diagnosis, while summary quality remains more modest across all methods.

Table 9. Latency, token-cost, and routing results.

Method	Latency ms/query	Cost USD/1k queries	Route rate
Regex baseline	0.020	0.0000	0.000
TF-IDF + ML classifier	0.180	0.0020	0.000
Local character classifier	0.350	0.0070	0.000
Zero-shot local policy	350.000	0.0382	1.000
Few-shot retrieval policy	620.000	0.1641	1.000
Router cascade	80.522	0.0326	0.129

Table 10 and Figure 4 show the cost-quality frontier. Lower thresholds route more records and improve parsing and summary ROUGE-L, while higher thresholds reduce cost further and increasingly resemble the local character classifier. The selected threshold of 0.20 is used because

it gives the best mean quality across the four task metrics while still routing only 12.9% of held-out task queries.

Table 10. Router threshold sweep.

Threshold	Route rate	Parsing acc.	Anomaly F1	Fault acc.	ROUGE-L	Cost USD/1k	Saving vs few-shot
0.20	0.129	0.991	1.000	1.000	0.743	0.0326	80.1%
0.35	0.024	0.991	1.000	1.000	0.733	0.0160	90.3%
0.50	0.011	0.989	1.000	1.000	0.733	0.0124	92.4%
0.55	0.009	0.984	1.000	1.000	0.733	0.0120	92.7%
0.60	0.007	0.984	1.000	1.000	0.738	0.0110	93.3%
0.65	0.004	0.983	1.000	1.000	0.738	0.0095	94.2%
0.70	0.000	0.983	1.000	1.000	0.736	0.0072	95.6%
0.75	0.000	0.983	1.000	1.000	0.736	0.0070	95.7%
0.80	0.000	0.983	1.000	1.000	0.736	0.0070	95.7%

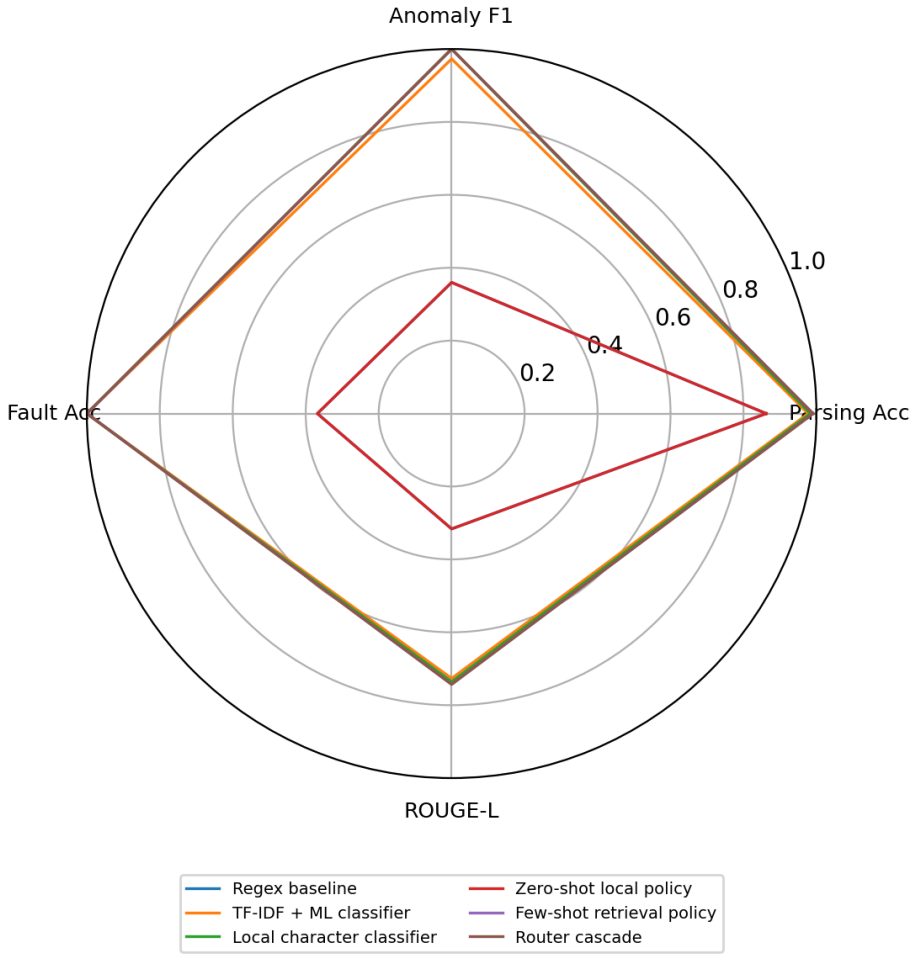

Figure 3. Multi-task performance radar across methods.

Figure 5 visualizes the latency values from Table 9. The router sits between local-only inference and all-few-shot inference because it pays the endpoint latency only for the routed subset. Table 11 and Figure 6 show the residual error profile. The router makes nine parsing template errors, no anomaly errors, no fault-label errors, and eleven summary outputs with ROUGE-L below 0.60.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 91-103
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

This pattern indicates that exact template recovery and summary wording remain more fragile than binary anomaly and fault-label classification in these task files.

Table 11. Router residual error distribution.

Error type	Count
parse template mismatch	9
anomaly false positive/negative	0
fault label mismatch	0
summary ROUGE-L below 0.60	11

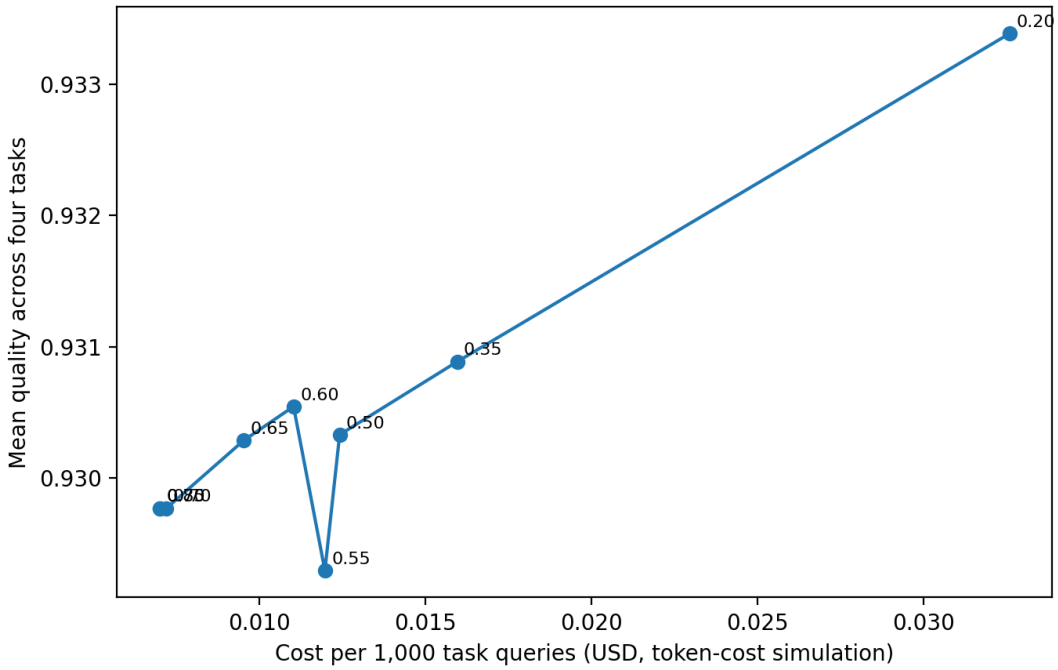

Figure 4. Cost-quality frontier for router thresholds.

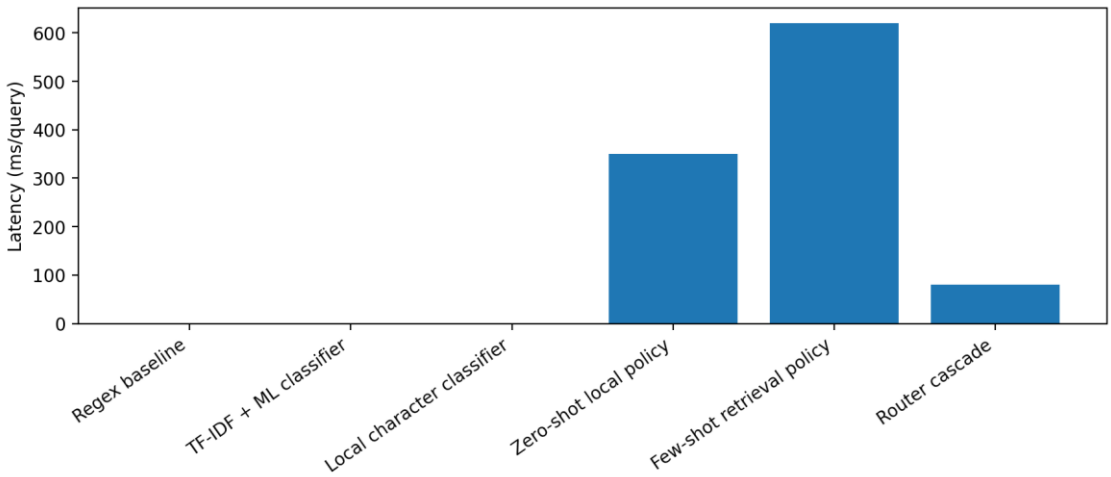

Figure 5. Latency comparison by method.

Table 12 gives representative router predictions. The examples show that routed and non-routed records can both be correct; routing is not a substitute for accuracy by itself, but a cost-control decision based on risk. Figure 7 shows the router confusion matrix for fault diagnosis. The diagonal structure is expected because the held-out diagnosis records remain within the same lexical label distribution as the training split.

Table 12. Sample router predictions.

Task	ID	Routed	Gold output	Router output
parsing	238	yes	exception syndrome register: <*>	exception syndrome register: <*>
parsing	2960	no	data_thread() got not answer from any [<*>] datasource	data_thread() got not answer from any [<*>] datasource
parsing	819	no	generating core.<*>	generating core.<*>
parsing	1593	no	iar <*> dear <*>	iar <*> dear <*>
parsing	1140	yes	<*> floating point alignment exceptions	<*> floating point alignment exceptions
parsing	1117	yes	<*> floating point alignment exceptions	<*> floating point alignment exceptions
parsing	1516	yes	ciod: Error loading <*>: invalid or missing program image, No such file or directory	ciod: Error loading <*>: invalid or missing program image, No such file or directory
anomaly	18	no	normal	normal
anomaly	1223	yes	normal	normal
anomaly	1459	yes	normal	normal
anomaly	3856	no	normal	normal
anomaly	1660	yes	normal	normal

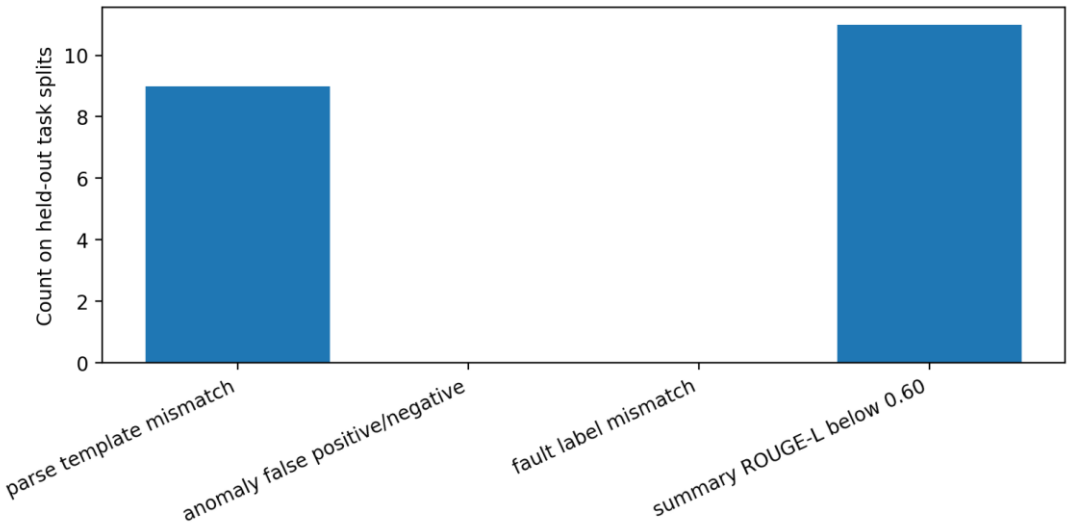

Figure 6. Residual router error types on the held-out splits.

Table 13 is the strongest warning against overinterpreting the record-level results. When 21 templates are completely absent from training, the supervised classifiers, retrieval policy, and router cannot emit those unseen template labels and therefore score 0.000 under closed-label exact matching. The regex normalizer still reaches 0.954 accuracy because many held-out templates

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 91-103
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

contain variable fields that can be recovered by literal normalization. Its macro-F1 remains much lower, showing that rare held-out templates are still difficult.

Table 13. Unseen-template parsing split.

Method	Test records	Held-out templates	Parse accuracy	Parse macro-F1
Regex baseline	1070	21	0.954	0.375
Zero-shot local policy	1070	21	0.954	0.375
TF-IDF + ML classifier	1070	21	0.000	0.000
Local character classifier	1070	21	0.000	0.000
Few-shot retrieval policy	1070	21	0.000	0.000
Router cascade	1070	21	0.000	0.000

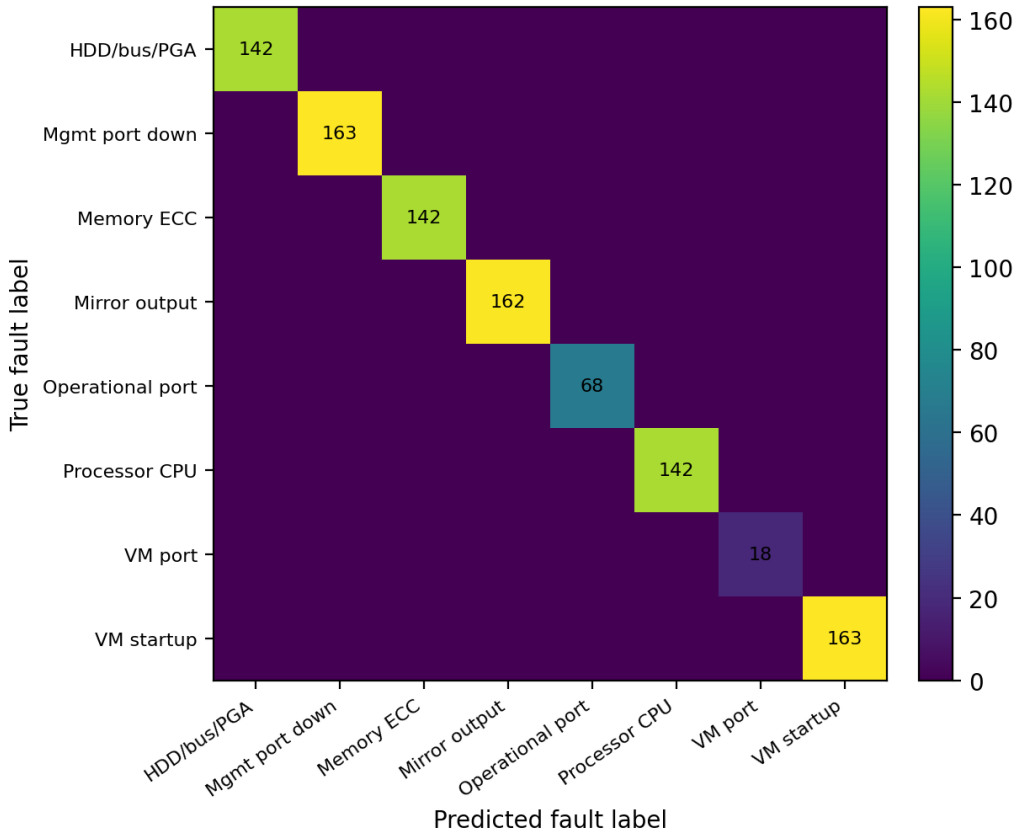

Figure 7. Router fault diagnosis confusion matrix.

DISCUSSION

The results support cost-aware routing as an operating pattern rather than as a universal accuracy booster. The router improves parsing over the local character classifier and matches or preserves strong anomaly and diagnosis performance, but its main advantage is the cost-quality operating point: most task queries remain local, while selected higher-risk queries are escalated to the few-shot retrieval endpoint. The deterministic endpoint design is also a limitation. The zero-shot and few-shot policies are not real LLM calls, and their behavior is narrower than an API-based or open-weight LLM. The study should therefore be read as a routing simulation for LLM-style inference. It shows how to measure route rate, token cost, latency, and task quality together, but it does not prove that the same numbers will hold under a production LLM.

The perfect anomaly and diagnosis metrics require cautious interpretation. They are achieved on held-out records from the same task distributions, where the labels are often signaled by explicit words. Such values are useful for checking cost savings under stable conditions, but they should not be presented as evidence that the system is robust to all real operational logs. The summary results are intentionally interpreted more conservatively. ROUGE-L and BLEU-1 reward lexical overlap with one reference summary, and the summary file contains only 200 sequences. A stronger summary evaluation should include more diverse references, human incident-response judgments, or model-assisted assessments that focus on usefulness, missing context, and hallucinated detail.

The unseen-template split clarifies generalization. Closed-label classifiers and retrieval policies work well when test records share templates with the training pool, but they cannot predict template strings that are absent from training. A production parser should therefore combine learned classifiers with template-mining or normalization methods that can emit new templates

V. CONCLUSION AND RECOMMENDATION

This study evaluates a cost-aware LLM-style router for AIOps log analysis on the LogEval task files. At threshold 0.20, the router routes 12.9% of held-out task queries and reduces simulated token cost by 80.1% relative to all-few-shot inference. It achieves parsing accuracy of 0.991, anomaly F1 of 1.000, fault diagnosis accuracy of 1.000, ROUGE-L of 0.743, and BLEU-1 of 0.814.

The practical recommendation is to deploy routing as a guarded cascade rather than as a single model replacement. Local classifiers should handle routine or high-confidence records, while the expert endpoint should be reserved for low-confidence, complex, or incident-critical inputs. The system should log the route decision, confidence score, selected examples, endpoint output, and final task output so that escalations can be audited.

The research recommendation is to validate the same router with actual API-based or open-weight LLMs, real token accounting, stronger summary assessment, and splits that hold out templates, software versions, or source families when those metadata are available. The present results show a useful cost-quality pattern, but the deterministic endpoints and task-file splits leave a clear path for stronger validation.

REFERENCES

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. In Advances in Neural Information Processing Systems, 33, 1877-1901.

Chen, L., Zaharia, M., & Zou, J. (2023). FrugalGPT: How to use large language models while reducing cost and improving performance. arXiv.

Cui, T., Ma, S., Chen, Z., Xiao, T., Tao, S., Liu, Y., Zhang, S., Lin, D., Liu, C., Cai, Y., Meng, W., Sun, Y., & Pei, D. (2024). LogEval: A comprehensive benchmark suite for large language models in log analysis. arXiv:2407.01896.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT (pp. 4171-4186).

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (pp. 1285-1298).

He, P., Zhu, J., Zheng, Z., & Lyu, M. R. (2017). Drain: An online log parsing approach with fixed depth tree. In 2017 IEEE International Conference on Web Services (pp. 33-40).

He, S., He, P., Chen, Z., Yang, T., Su, Y., & Lyu, M. R. (2021). A survey on automated log analysis for reliability engineering. ACM Computing Surveys, 54(6), 1-37.

Landauer, M., Skopik, F., Wurzenberger, M., & Rauber, A. (2023). Deep learning for anomaly detection in log data: A survey. ACM Computing Surveys, 56(2), 1-37.

Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., Chen, Y., Zhang, R., Tao, S., Sun, P., & Zhou, R. (2019). LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In Proceedings of the IEEE/ACM International Conference on Automated Software Engineering.

Oliner, A., & Stearley, J. (2007). What supercomputers say: A study of five system logs. In 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (pp. 575-584).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems, 30.

Xu, W., Huang, L., Fox, A., Patterson, D., & Jordan, M. I. (2009). Detecting large-scale system problems by mining console logs. In Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (pp. 117-132).

Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., Chen, J., He, X., Yao, R., Lou, J.-G., & Chintalapati, M. (2019). Robust log-based anomaly detection on unstable log data. In Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 807-817).

Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2019). Tools and benchmarks for automated log parsing. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (pp. 121-130).