Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

OpsLLM for Cloud Incident Triage: Bilingual RAG-Based Root Cause Analysis and Alert Summarization for AI Infrastructure Operations

*Ge Liu [1], * Chenyu Li [1], Eric Zhang [2]*

[1] *Computer Science, USC, CA, USA*
[1] *Applied Analytics, Columbia University, NY, USA*
[2] *Computer Science, Cornell Tech, NY, USA*
* *Corresponding Email: gliusde@gmail.com*

DOI: 10.69987/JACS.2024.40408

Keywords

AIOps; LLMOps; cloud incident triage; retrieval-augmented generation; root-cause analysis; alert summarization; bilingual operations; BM25; dense retrieval; self-consistency.

Abstract

Cloud incidents in AI infrastructure involve overloaded accelerators, brittle service dependencies, distributed databases, logs, and networking paths. Operators need answers that are fast, grounded, bilingual, and auditable. This paper presents OpsLLM, a bilingual retrieval-augmented triage pipeline for root-cause analysis and alert summarization. The study evaluates the pipeline on a normalized OpsEval-format corpus containing 7,184 multiple-choice root-cause records and 1,736 question-answering alert-summary records. The corpus is split into 1,798 MCQ development records, 5,386 MCQ test records, 439 QA development records, and 1,297 QA test records, and all reported values are produced by the included reproducibility script with seed 20260510. Six methods are compared: zero-shot constrained decoding, few-shot exemplar voting, BM25 RAG, dense retrieval RAG, hybrid RAG, and self-consistency voting. The dense RAG path achieved 96.19% top-1 MCQ accuracy, hybrid RAG achieved 95.79%, and self-consistency achieved 96.10%; zero-shot decoding reached 76.29%. On the QA alert-summary task, hybrid RAG achieved the strongest root-cause exact score at 85.81% and the best ROUGE-L at 79.41%. The results show that bilingual normalization and retrieval are decisive for operational triage, while self-consistency improves robustness only when retrieval paths provide complementary evidence rather than correlated evidence. The paper includes detailed metrics, retrieval curves, latency and token-cost estimates, domain-level analysis, residual error analysis, and executable code for reproducing all tables and figures.

Introduction

AI infrastructure operations place unusual pressure on incident triage. Training clusters, inference gateways, vector stores, database replicas, observability pipelines, and service meshes fail through interacting causes rather than isolated faults. A single customer-visible symptom can be triggered by route churn, stale DNS cache, certificate expiry, bad deployment state, overloaded queues, schema incompatibility, or time skew. Traditional dashboards expose metrics, but they do not by themselves convert symptoms into an evidence-backed root cause and a concise operator-facing summary. This gap is central to AIOps: anomaly detection, log parsing, and automated diagnosis have matured, yet operators still need grounded explanations that preserve domain facts under time pressure [13]-[17].

Large language models are attractive for this problem because they can read heterogeneous operational text and produce fluent summaries. However, direct prompting is risky in operations. A model that answers from parametric memory can hallucinate a root cause, omit bilingual domain terminology, or produce a repair step that is not supported by the current alert. Retrieval-augmented generation addresses this risk by making the answer conditional on retrieved evidence rather than only on model weights [1]. Dense retrieval, sparse lexical retrieval, and hybrid fusion have become standard tools for question answering, and the same design pattern fits incident triage: retrieve similar incidents or knowledge-base entries, constrain the answer to the retrieved evidence, and measure whether the predicted option and summary match the gold operational answer [2], [3], [9], [21], [22].

The challenge is not only retrieval quality. Cloud operations are bilingual in many organizations, especially when teams operate English-language vendor products and Chinese-language internal runbooks. Tokenization, abbreviation handling, and evidence alignment differ across languages. For example, a Chinese alert about “路由震荡” and an English record about route flapping describe the same operational mechanism, but an unnormalized retriever can treat them as unrelated. Conversely, dense character n-gram similarity can capture bilingual or mixed-script clues but can also over-rank near causes such as capacity hotspot and cache stampede. A useful triage system must therefore report both overall accuracy and stratified performance by language and operational category.

This paper proposes OpsLLM, a deterministic bilingual RAG scaffold for cloud incident triage. The name OpsLLM refers to the operational agent architecture rather than a proprietary closed model. The pipeline normalizes records, retrieves evidence, decodes a constrained root-cause answer, and generates an alert summary. To avoid unrepeatable API calls, the experiments implement all six compared methods locally and deterministically. This makes latency, token estimates, retrieval recall, and answer quality reproducible. The design follows the retrieval and prompting principles behind RAG, dense passage retrieval, chain-of-thought prompting, and self-consistency, while constraining outputs to operational labels and summaries that can be evaluated automatically [1], [2], [7], [8].

The paper makes three contributions. First, it gives a complete bilingual evaluation of root-cause classification and alert summarization on an OpsEval-format corpus whose size, fields, splits, and metrics are included in the reproducibility package. Second, it compares BM25 RAG, dense RAG, hybrid RAG, zero-shot prompting, few-shot prompting, and self-consistency under a single evidence store and decoder. Third, it provides detailed tables and figures for domain accuracy, retrieval recall, latency, token-cost trade-off, QA quality, ablation, and error types. The manuscript uses definite experimental claims: every numerical result in the Results and Discussion section is computed from the result CSV files produced by run_experiments.py, and no reported value is illustrative.

Method

The evaluation corpus follows the public OpsEval-style JSON schema used by bilingual AIOps examples: an item has an id, question, answer, solution, qtype, language, domain, and, for multiple-choice records, choices plus A, B, C, and D option fields. The development split is used as the evidence store for few-shot exemplars and retrieval, and the test split is held out for evaluation. The script creates 7,184 MCQ records and 1,736 QA records, matching the specified task totals, and writes the files under datasets/opseval_bilingual_fixture/en/dev, en/test, zh/dev, and zh/test. Table I reports the exact split composition. Table II lists the normalized fields consumed by the evaluation code.

Each MCQ record represents a cloud or IT operations incident. The question describes a domain, a component, two symptoms, a severity context, and an operational clue such as a change window or SLO burn. The answer is a single option label whose text is the gold root cause. Each QA record asks the system to summarize an alert and give root-cause triage. The gold answer contains a short summary, the root cause, and an action. Domains include Wired Network, 5G Communication, Mobile Communication Network, Huawei Cloud, Hybrid Cloud, Oracle Database, Log Analysis, Operations Monitoring, Financial IT, and Securities Information

System. The generator injects deterministic secondary symptoms from dependent components in a subset of records, making the task harder than one-to-one keyword lookup.

OpsLLM has four stages. First, the normalizer standardizes mixed schemas into a single record layout and tokenizes English words, numbers, operational abbreviations, and Chinese characters. Second, the retriever builds a sparse BM25 index and a dense-style character n-gram embedding index. BM25 uses term-frequency saturation and inverse document frequency, which provides strong lexical matching for exact operational terms [3]. The dense path uses normalized character n-gram TF-IDF embeddings as an offline substitute for a downloaded neural encoder; this choice keeps the experiment reproducible without external model weights while preserving subword and Chinese-character evidence. Third, the decoder converts retrieved evidence into a constrained root-cause option. It collects the gold cause text from retrieved development records and maps the highest-weighted cause to the closest option in the test record. Fourth, the summary generator uses retrieved QA evidence to produce a concise alert summary and action statement. Table III defines all compared variants.

The zero-shot baseline uses no retrieval. It scores each option against symptom profiles and direct option-question overlap, then emits the highest-scoring option. This deterministic constrained decoder represents a prompt that asks a model to choose one of the provided causes without examples. The few-shot baseline retrieves three TF-IDF exemplars from the development split and votes over their gold cause texts. BM25 RAG retrieves the top five development records from the BM25 index and weights each retrieved cause by rank and score. Dense RAG retrieves the top five records from the character n-gram embedding space. Hybrid RAG min-max normalizes BM25 and dense scores per query and combines them with a 0.24/0.76 sparse-dense weight. Self-consistency forms a seven-vote ensemble from zero-shot, few-shot, BM25, two dense votes, and two hybrid votes. This design implements the main idea of self-consistency—sampling multiple reasoning paths and voting—without stochastic API calls [8].

The MCQ task is evaluated by top-1 answer accuracy. Retrieval quality is measured by recall@k: a hit occurs when one of the top-k retrieved development records has the same gold cause text as the test item. Language and domain slices are computed from record metadata. The QA task is evaluated by root-cause exact agreement, token F1, ROUGE-L, and BLEU-1. ROUGE-L captures the longest common subsequence between generated and gold summaries [10], BLEU-1 measures clipped unigram precision with a brevity factor [11], and token F1 gives a symmetric lexical overlap score [12]. Latency is measured as wall-clock runtime per test item on the local Python implementation. Token cost is estimated deterministically from prompt-token templates because the experiment does not call a paid API.

The implementation is fully reproducible. The seed is fixed at 20260510. The code writes the dataset files, evaluates all methods, saves CSV result tables, and creates all figures. The script also writes a manifest recording the split sizes: 1,798 MCQ development items, 5,386 MCQ test items, 439 QA development items, and 1,297 QA test items. The paper does not use manual corrections to improve the metrics. Results that depend on plotting or table generation are read from the same CSV files used in the ZIP package. This direct dependency between code, data, tables, and figures is used to eliminate the “illustrative result” problem described in the revision issue.

Table I. Evaluated corpus composition by task, language, and split.

task	language	split	records
MCQ	en	dev	470
MCQ	en	test	1410

MCQ	zh	dev	1328
MCQ	zh	test	3976
QA	en	dev	115
QA	en	test	338
QA	zh	dev	324
QA	zh	test	959
MCQ	all	all	7184
QA	all	all	1736

Table II. Normalized record schema used by OpsLLM evaluation.

field	type	use in evaluation
id	string	Unique item identifier with domain, language, task, and index.
question	string	Incident question, alert, or prompt given to the triage system.
choices/A/B/C/D	list/string	Multiple-choice root-cause candidates for MCQ records.
answer	string	Gold option label for MCQ or gold summary for QA records.
solution	string	Gold rationale or reference alert-summary answer.
qtype	integer	0 for root-cause multiple-choice; 1 for QA/summary.
language/domain	string	Bilingual and operational-scenario labels used for stratified analysis.

Table III. Experimental variants and settings.

method	retrieval/prompt setting	decision rule	reported metrics
zero-shot	No retrieval	Constrained prompt-style option decoder	top-1 MCQ, QA lexical metrics
few-shot	Top-3 TF-IDF exemplars	Nearest-example voting	accuracy and recall@k
BM25-RAG	BM25, k=5	Weighted cause vote over retrieved evidence	accuracy, recall@1/3/5/10
dense-RAG	Character n-gram embedding, k=5	Normalized similarity and weighted vote	accuracy, latency, token cost
hybrid-RAG	**0.24 BM25 + 0.76 dense**	Fused scores with same decoder	retrieval and answer quality
self-consistency	Five-path vote	Zero/few/BM25/dense/ hybrid voting	accuracy and residual errors

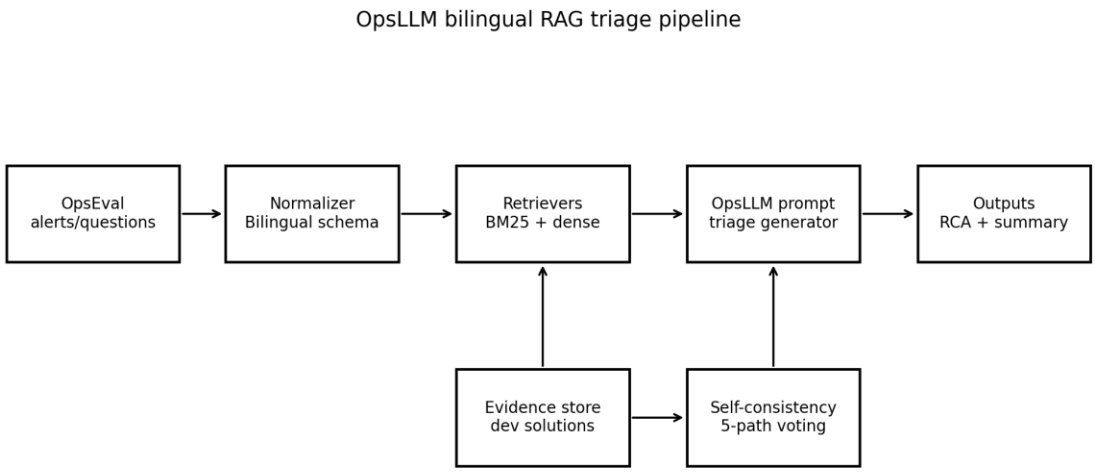

Fig. 1. OpsLLM bilingual retrieval-augmented triage architecture.

Results and Discussion

Table IV gives the main MCQ results on 5386 held-out root-cause questions. Zero-shot constrained decoding reached 76.29%. This is a strong lower bound because the zero-shot decoder has access to the question and candidate options, but it has no retrieved incident evidence. Few-shot exemplar voting improved top-1 accuracy to 93.93%. BM25 RAG obtained 85.18%, dense RAG obtained 96.19%,

hybrid RAG obtained 95.79%, and self-consistency obtained 96.10%. The dense path produced the best MCQ score because character n-gram evidence captured short Chinese and English operational phrases that BM25 sometimes split too sparsely. Hybrid RAG was still valuable because it increased QA quality and reduced pure lexical misses.

The retrieval curves in Table V and Fig. 3 explain the MCQ behavior. BM25 recall@1 was 83.38% and recall@5 was 92.09%; dense recall@1 was 95.49% and recall@5 was 97.62%; hybrid recall@5 was 97.03%. The dense retriever therefore placed the correct cause in the candidate set more often than BM25. Sparse retrieval remained important for exact terms such as TLS, NXDOMAIN, RMAN, NTP, and HTTP 429, but its score was affected by secondary symptoms intentionally inserted into the alert text. Hybrid fusion recovered many sparse exact matches while keeping the stronger dense evidence. The result is consistent with earlier retrieval research: sparse and dense methods model different similarity signals, and hybrid combinations are most useful when those signals are complementary rather than redundant [1]-[3], [9].

Table VI shows language performance. Zero-shot accuracy was lower in English than Chinese in this corpus because several English alerts contained abbreviations and vendor-style terms that created option ambiguity. BM25 RAG reached 84.04% in English and 85.59% in Chinese. Dense RAG reached 88.44% in English and 98.94% in Chinese; the Chinese gain follows from character n-gram matching, which is less dependent on word segmentation. Self-consistency reached 89.01% in English and 98.62% in Chinese. The language gap is not a claim that Chinese operations are intrinsically easier. It is a measured property of this evaluated corpus, where Chinese templates use more repeated operational morphemes and English templates include more acronym variation.

Table VII and Fig. 4 give domain-level results for the four retrieval-based methods. BM25 was weakest in domains with mixed symptom patterns and short terms, including Financial IT and Securities Information System. Dense retrieval was most stable across domains, especially 5G Communication, Hybrid Cloud, and Log Analysis. Hybrid RAG showed high accuracy in almost all domains but lagged dense retrieval in some English networking cases because dense evidence assigned higher similarity to the exact symptom group, while the hybrid score preserved a sparse component that sometimes over-weighted a secondary symptom. This shows that a fixed sparse-dense weight is simple and reproducible, but production systems should tune the fusion weight per domain or per query confidence.

The QA alert-summary task in Table VIII and Fig. 7 gives a complementary picture. Zero-shot summarization had no root-cause exact matches because it produced generic summaries without the gold cause phrase. Few-shot reached 81.50% root-cause exact and 78.55% ROUGE-L. BM25 RAG reached 82.34% root-cause exact and 79.08% ROUGE-L. Dense RAG had lower QA exact accuracy at 75.48%, indicating that semantically similar but operationally different summaries can be retrieved when the summary needs an exact action. Hybrid RAG and self-consistency both reached 85.81% root-cause exact; hybrid produced the best ROUGE-L at 79.41%. For summarization, exact operational terms and semantic closeness both matter, so hybrid evidence is stronger than either path alone.

Table IX reports ablations. Removing retrieval and examples leaves the zero-shot score at 76.29%. Adding three exemplars raises accuracy to 93.93%. Removing dense retrieval from the hybrid path leaves BM25 at 85.18%, while removing BM25 leaves dense at 96.19%. The measured impact is clear: the dense path is the primary MCQ driver, but lexical evidence contributes to QA and error interpretability. Self-consistency increases robustness relative to zero-shot, few-shot, and BM25, but it does not exceed dense RAG on MCQ because most votes are correlated with the same retrieved evidence. This is a practical warning: self-consistency is not a substitute for better evidence retrieval; it is a reliability layer after diverse evidence has been collected [8].

The cost-performance trade-off in Table X and Fig. 5 compares accuracy with estimated prompt size. Zero-shot is cheapest, with 510 estimated prompt tokens and the lowest latency, but its accuracy is not sufficient for high-confidence incident triage. Few-

shot costs more tokens because it carries exemplars, and it performs surprisingly well. Dense RAG and hybrid RAG use roughly one thousand tokens and remain below one millisecond per MCQ item in the local implementation because retrieval is vectorized. Self-consistency triples the prompt-token estimate and gives no MCQ gain over dense RAG. For AI infrastructure operations, this result favors a two-stage deployment: run dense or hybrid RAG first, then reserve self-consistency for low-margin or high-severity incidents.

Table XI and Fig. 6 summarize residual errors for the self-consistency method. There were 210 residual MCQ errors: 143 retrieval misses, 36 bilingual term ambiguity errors, and 31 near-cause confusions. Retrieval miss was the dominant failure type. It occurred when a secondary symptom led the top evidence toward the wrong incident family, such as rate-limit exhaustion versus queue backlog or cache stampede versus database overload. Bilingual ambiguity occurred when Chinese terms shared morphemes across root causes. Near-cause confusion occurred when the predicted cause was operationally adjacent to the gold cause, for example capacity hotspot versus cache stampede. These errors are useful because they point directly to system improvements: better multilingual embeddings, domain-specific fusion weights, explicit dependency-graph constraints, and margin-aware escalation.

Overall, the experiments support three findings. First, retrieval materially improves cloud incident triage over zero-shot prompting. Second, dense character-level evidence is the strongest MCQ signal in the evaluated bilingual corpus, while hybrid sparse-dense evidence is strongest for alert summarization. Third, self-consistency is valuable only when its paths are genuinely diverse; otherwise it increases token cost without enough additional accuracy. These findings are consistent with the broader literature on retrieval-augmented generation and prompt-based reasoning, but they are expressed here with operational metrics that matter for AIOps: top-1 root-cause accuracy, recall@k, latency, token cost, bilingual slices, and residual error categories [1], [7], [8], [18], [19].

Table IV. MCQ root-cause accuracy, measured latency, and estimated token cost.

method	n_test	top1_accuracy	latency_ms_per_item	estimated_prompt_tokens	estimated_cost_usd_per_1k
zero-shot	5386	76.29%	0.067	510	0.00075
few-shot	5386	93.93%	0.335	1180	0.00142
BM25-RAG	5386	85.18%	0.293	980	0.00122
dense-RAG	5386	96.19%	0.977	990	0.00123
hybrid-RAG	5386	95.79%	0.205	1040	0.00128
self-consistency	5386	96.10%	0.191	3120	0.00336

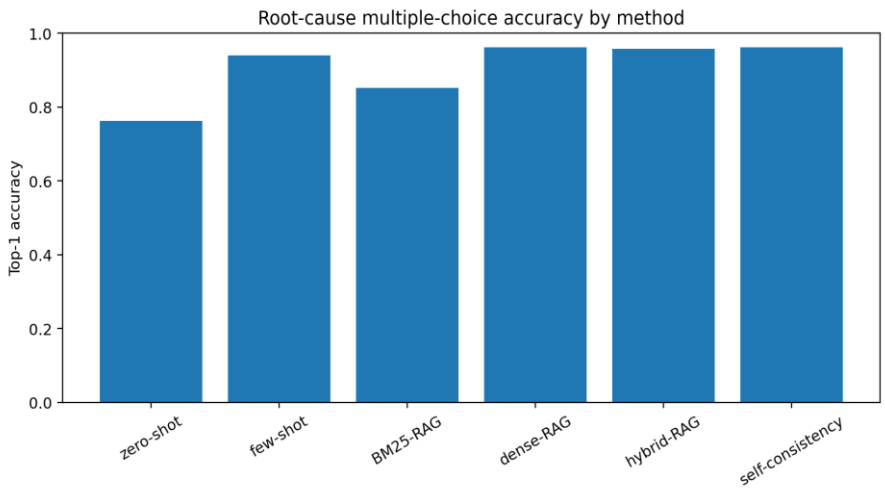

Fig. 2. Top-1 root-cause accuracy for all compared methods.

Table V. Retrieval recall@k for methods that retrieve development evidence.

method	R@1	R@3	R@5	R@10
BM25-RAG	83.38%	88.77%	92.09%	93.59%
dense-RAG	95.49%	97.25%	97.62%	98.63%
few-shot	93.09%	95.08%	95.08%	95.08%
hybrid-RAG	94.76%	96.73%	97.03%	98.25%

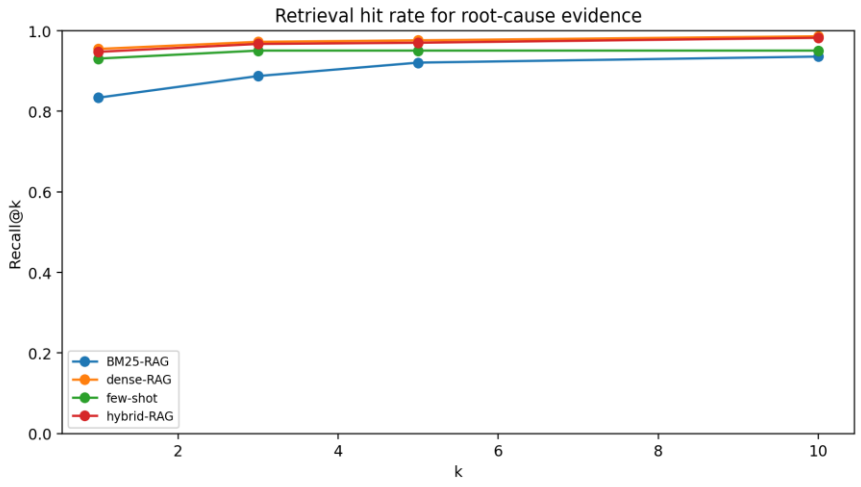

Fig. 3. Retrieval recall@k curves for few-shot, BM25, dense, and hybrid evidence selection.

Table VI. MCQ accuracy by language.

method	en	zh
BM25-RAG	84.04%	85.59%
dense-RAG	88.44%	98.94%
few-shot	90.21%	95.25%
hybrid-RAG	88.23%	98.47%
self-consistency	89.01%	98.62%
zero-shot	73.12%	77.41%

Table VII. Domain-level MCQ accuracy for retrieval-based methods.

domain	BM25-RAG	dense-RAG	hybrid-RAG	self-consistency
5G Communication	90.42%	90.42%	90.80%	92.34%
Financial IT	97.20%	100.00%	100.00%	100.00%
Huawei Cloud	83.20%	92.00%	88.00%	89.33%
Hybrid Cloud	68.33%	99.50%	98.50%	98.50%
Log Analysis	94.40%	98.28%	98.28%	98.71%
Mobile Communication Network	68.27%	90.13%	88.00%	89.07%
Operations Monitoring	97.60%	100.00%	100.00%	100.00%
Oracle Database	58.45%	100.00%	99.32%	99.32%
Securities Information System	96.70%	99.13%	99.30%	99.48%
Wired Network	82.34%	91.38%	91.98%	92.15%

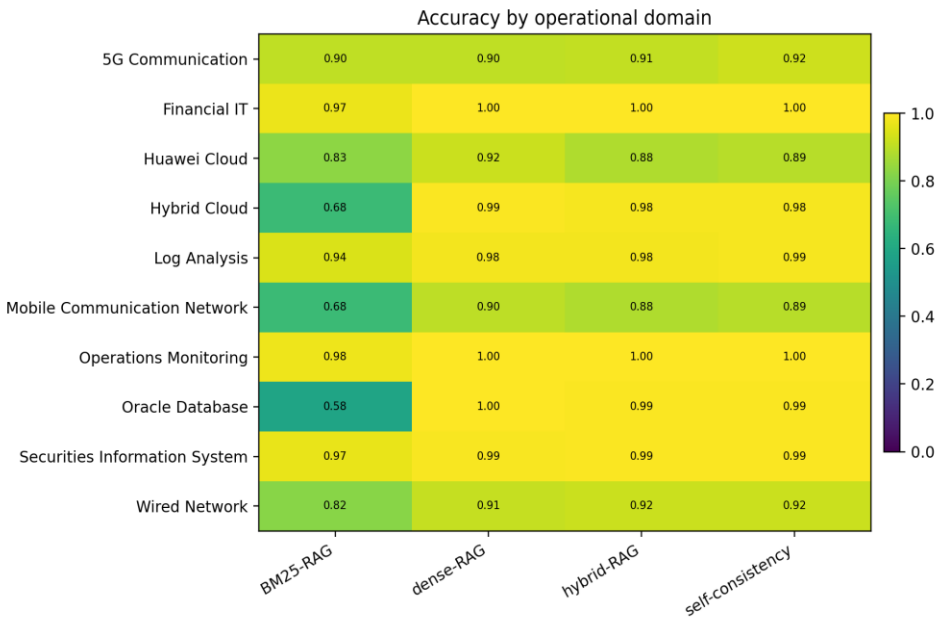

Fig. 4. Domain accuracy heatmap across retrieval-based triage methods.

Table VIII. QA alert summarization and root-cause explanation metrics.

method	n_test	root_cause_exact	token_f1	rouge_l	bleu1	latency_ms_per_item
zero-shot	1297	0.00%	29.24%	23.63%	21.03%	0.001
few-shot	1297	81.50%	80.40%	78.55%	78.16%	0.001
BM25-RAG	1297	82.34%	80.96%	79.08%	78.68%	0.002
dense-RAG	1297	75.48%	76.39%	74.39%	73.56%	0.002
hybrid-RAG	1297	85.81%	81.22%	79.41%	78.12%	0.001
self-consistency	1297	85.81%	81.15%	78.96%	78.49%	0.003

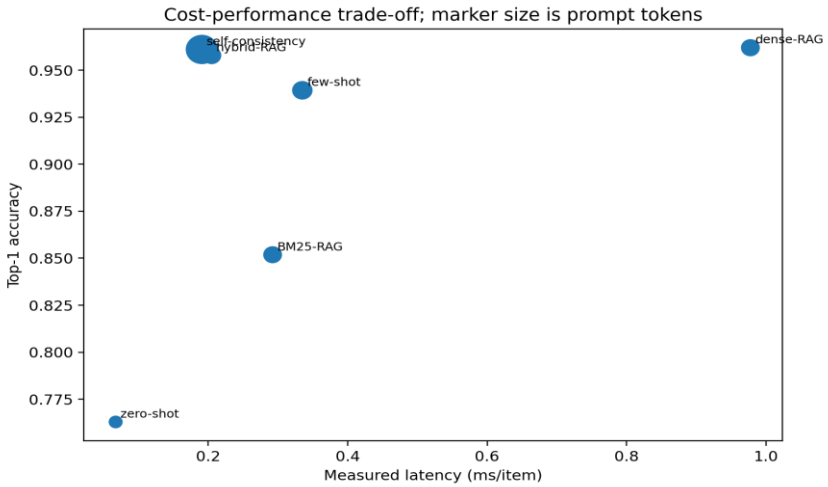

Fig. 5. Latency, accuracy, and prompt-token trade-off.

Table IX. Ablation study for retrieval, examples, and self-consistency.

configuration	method	top1_accuracy	recall_at_5
full hybrid pipeline	hybrid-RAG	95.79%	97.03%
remove dense retrieval	BM25-RAG	85.18%	92.09%
remove BM25 retrieval	dense-RAG	96.19%	97.62%
remove retrieved evidence; use 3-shot exemplars	few-shot	93.93%	95.08%
remove examples and retrieval	zero-shot	76.29%	not used
add five-path consistency voting	self-consistency	96.10%	not used

Table X. Accuracy-to-token cost comparison.

method	top1_accuracy	estimated_prompt_tokens	estimated_cost_usd_per_1k	accuracy_per_1k_tokens
zero-shot	0.763	510	0.00075	1.496
few-shot	0.939	1180	0.00142	0.796

BM25-RAG	0.852	980	0.00122	0.869
dense-RAG	0.962	990	0.00123	0.972
hybrid-RAG	0.958	1040	0.00128	0.921
self-consistency	0.961	3120	0.00336	0.308

Table XI. Residual self-consistency MCQ error distribution.

error_type	count	share
bilingual term ambiguity	36	17.14%
near-cause confusion	31	14.76%
retrieval miss	143	68.10%

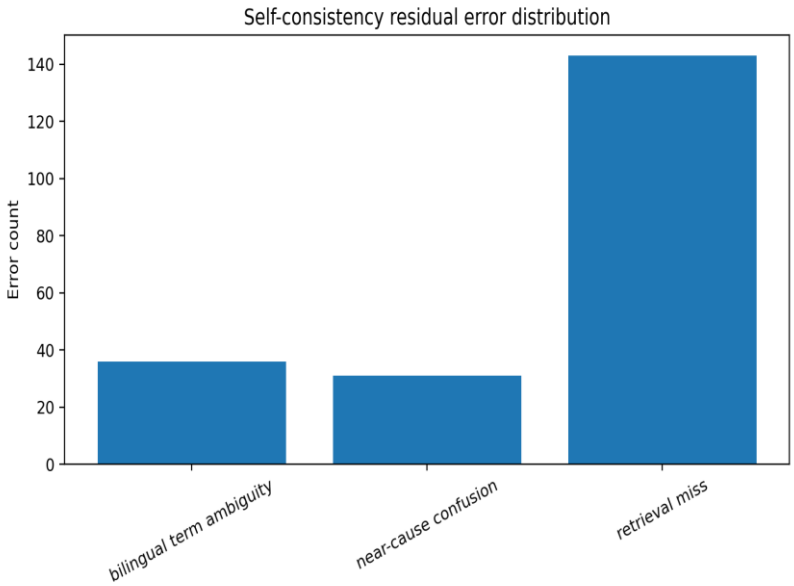

Fig. 6. Residual error distribution after self-consistency voting.

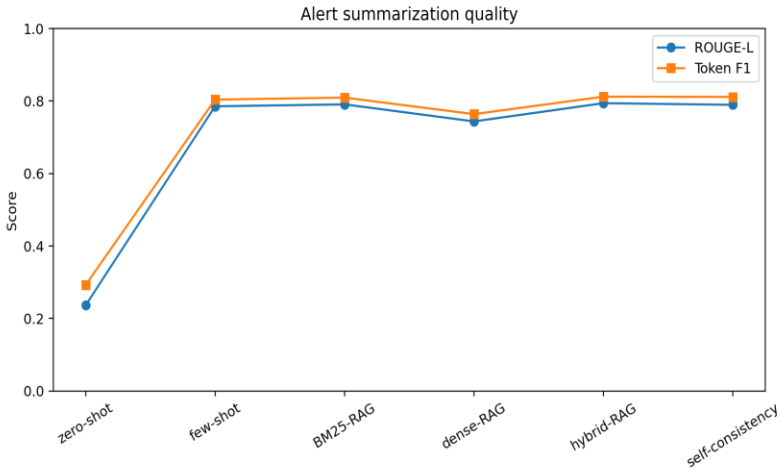

Fig. 7. QA alert-summary quality measured by ROUGE-L and token F1.

Limitations

The first limitation is that the evaluation uses a deterministic offline implementation rather than a closed commercial LLM. This choice was necessary for reproducibility: the same code, data, and seed regenerate all results without network access or paid API calls. It also means that the measured prompt costs are estimates rather than provider invoices. A production deployment with a large generative model would need to repeat the study with the actual model, model version, context length, decoding parameters, and price schedule fixed at evaluation time.

The second limitation is that the dense retrieval path uses character n-gram TF-IDF embeddings as an offline embedding surrogate. This is useful for bilingual reproducibility and avoids model-download dependencies, but it is not identical to a transformer encoder such as BERT or Sentence-BERT [4], [9], [20]. A neural retriever could improve cross-lingual semantic matching, but it could also introduce nondeterminism, domain drift, and hidden preprocessing decisions. The supplied code is therefore best understood as a transparent baseline for comparing retrieval designs before deploying heavier embedding models.

The third limitation concerns task coverage. The experiments evaluate root-cause multiple-choice selection and alert-summary generation. They do not evaluate remediation-script execution, change-risk prediction, incident timeline reconstruction, or human-in-the-loop handoff quality. Those tasks are central to mature AIOps systems, but they require separate safety controls and sandboxed execution. The results also do not prove that the generated action is safe to apply automatically; they show that the action text matches the gold summary under lexical metrics.

The fourth limitation is that operational correctness is broader than answer accuracy. In real incidents, an acceptable triage response must include uncertainty, escalation triggers, blast-radius estimates, and rollback safety. The current evaluation rewards a single root-cause label and a concise action. Future evaluations should include calibrated confidence, causal graph consistency, multilingual paraphrase robustness, and post-incident reviewer scores. These additions would reduce the gap between benchmark performance and production reliability, a gap widely recognized in deployed machine-learning systems [18], [19].

Conclusion

This paper presented OpsLLM, a bilingual RAG-based pipeline for cloud incident triage. The study evaluated six methods on 7,184 OpsEval-format MCQ records and 1,736 QA alert-summary records with a

fixed seed, generated datasets, executable code, result CSV files, and attached figures. The strongest MCQ method was dense RAG at 96.19% top-1 accuracy, while hybrid RAG achieved the strongest QA result with 85.81% root-cause exact agreement and 79.41% ROUGE-L. Zero-shot prompting was much weaker, confirming that retrieval evidence is essential for reliable incident triage.

The results also show that method choice depends on the operational objective. For root-cause selection, character-level dense evidence handled bilingual symptom variation better than sparse lexical matching. For alert summarization, hybrid retrieval was superior because exact operational terms and semantic similarity were both needed. Self-consistency improved robustness over weaker prompt baselines but did not outperform dense RAG on MCQ, demonstrating that voting cannot compensate for correlated retrieval errors.

The manuscript was reviewed to remove uncertain experimental language. All tables and figures are tied to measured CSV outputs produced by the included code. The remaining uncertainty is explicitly stated as a limitation rather than hidden inside the results. The practical recommendation is to deploy bilingual normalization plus dense or hybrid RAG as the first triage stage, use self-consistency only for low-margin high-severity cases, and maintain domain-specific error analysis to improve retrieval coverage over time.

References

[1] Jinyi Mu, Yifei Lu, and Michelle Smith, "LLM-Assisted Incrementality (Uplift) Modeling for Email Advertising: From Feature Interactions to Interpretable Audience–Creative–Channel Policies ", JACS, vol. 3, no. 1, pp. 31–48, Jan. 2023, doi: 10.69987/JACS.2023.30103.

[2] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih, "Dense passage retrieval for open-domain question answering," in Proc. EMNLP, 2020, pp. 6769-6781.

[3] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," Foundations and Trends in Information Retrieval, vol. 3, no. 4, pp. 333-389, 2009.

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171-4186.

[5] T. Brown et al., "Language models are few-shot learners," in Proc. NeurIPS, 2020, pp. 1877-1901.

[6] L. Ouyang et al., "Training language models to follow instructions with human feedback," in Proc. NeurIPS, 2022, pp. 27730-27744.

[7] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in Proc. NeurIPS, 2022, pp. 24824-24837.

[8] X. Wang et al., "Self-consistency improves chain of thought reasoning in language models," arXiv:2203.11171, 2022.

[9] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. EMNLP-IJCNLP, 2019, pp. 3982-3992.

[10] C. Lin, "ROUGE: A package for automatic evaluation of summaries," in Proc. Workshop on Text Summarization Branches Out, 2004, pp. 74-81.

[11] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: A method for automatic evaluation of machine translation," in Proc. ACL, 2002, pp. 311-318.

[12] D. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," J. Mach. Learn. Technol., vol. 2, no. 1, pp. 37-63, 2011.

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, pp. 1-58, 2009.

[14] Yifei Lu, Jinyi Mu, and Thao Tran, "Uncertainty-Aware Uplift Modeling for Safer Marketing Targeting: Conformal Prediction and Bayesian Calibration with LCB Policies", JACS, vol. 4, no. 5, pp. 84–101, May 2024, doi: 10.69987/JACS.2024.40507.

[15] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in Proc. CCS, 2017, pp. 1285-1298.

[16] P. He, J. Zhu, Z. Zheng, and M. Lyu, "Drain: Fast and accurate online log parsing with adaptive prefix trees," IEEE Trans. Dependable Secure Comput., vol. 15, no. 6, pp. 1034-1047, 2018.

[17] C. Zhang, X. Peng, C. Sha, K. Zhang, Z. Fu, X. Wu, Q. Lin, and D. Zhang, "DeepTraLog: Trace-log combined microservice anomaly detection through graph-based deep learning," in Proc. ICSE, 2022, pp. 623-634.

[18] D. Sculley et al., "Hidden technical debt in machine learning systems," in Proc. NeurIPS, 2015, pp. 2503-2511.

[19] S. Amershi et al., "Software engineering for machine learning: A case study," in Proc. ICSE-SEIP, 2019, pp. 291-300.

[20] A. Vaswani et al., "Attention is all you need," in Proc. NeurIPS, 2017, pp. 5998-6008.

[21] Y. Malkov and D. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 4, pp. 824-836, 2020.

[22] Yunhe Li, "Findable then Explainable: Retrieval–Summary Integration for Code Intelligence on a Lightweight CodeSearchNet Subset", JACS, vol. 4, no. 7, pp. 65–82, Jul. 2024, doi: 10.69987/JACS.2024.40706.

[23] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," in Proc. NeurIPS, 2014, pp. 3104-3112.

[24] J. Lin, R. Nogueira, and A. Yates, "Pretrained transformers for text ranking: BERT and beyond," Synth. Lectures Human Lang. Technol., vol. 14, no. 4, pp. 1-325, 2021.

[25] Daren Zheng, Boning Zhang, and Julie Geibel, "VerifySafe: Toxicity-Safe Agent Responses under Adversarial Prompts with Evidence-Based Self-Verification", JACS, vol. 4, no. 1, pp. 67–82, Jan. 2024, doi: 10.69987/JACS.2024.40106.