Agentic AIOps for Cloud Log Anomaly Triage with Retrieval-Grounded Root-Cause Explanations

Peng Liu[1], Jessica White[2], Wei Han[3], Christopher Scott[4]
[1]*computer science, columbia, ny, usa*
[2]*computer science, umich, mi, usa*
[3]*computer science, purdue, in, usa*
[4]*computer science, uci, ca, usa*
pliu.research@gmail.com

DOI: 10.63575/CIA.2024.20215

Abstract

Cloud operations teams must distinguish genuine failures from benign log variation and then explain the likely root cause with evidence that an operator can inspect. This paper presents a retrieval-grounded AIOps triage agent that converts raw OpenStack and Hadoop logs into normalized event templates, combines lexical, structural, and timing anomaly signals, retrieves analogous incidents, ranks candidate causes, and verifies every explanation against direct log cues or matching historical cases. The evaluation uses all 55 labeled Hadoop application runs and a held-out OpenStack stream containing 198 VM sessions, including the four VM identifiers marked as anomalous by the corpus. Hadoop anomaly detection is measured with three repetitions of stratified five-fold cross-validation; OpenStack models are fitted on normal1, calibrated on normal2, and tested on the abnormal stream. On Hadoop, the full fusion obtains precision 0.913, recall 0.871, F1 0.891, false-positive rate 0.333, and area under the precision–recall curve 0.951. A numeric random forest reaches a similar F1 of 0.894 but raises the false-positive rate to 0.576. On OpenStack, the calibrated fusion detects all four anomalous VMs and produces one false alarm among 194 normal VMs, yielding precision 0.800, recall 1.000, and F1 0.889. For Hadoop root-cause analysis, the verified ranker attains Top-1 accuracy 0.818, Top-2 accuracy 1.000, mean reciprocal rank 0.909, and NDCG@3 0.933 across machine-down, network-disconnection, and disk-full incidents. Verification raises evidence precision from 0.906 to 1.000 while preserving cause accuracy. The results show that retrieval is most valuable for ranked diagnosis and evidence formation, whereas explicit latency modeling is indispensable for the injected OpenStack anomalies.

Keywords: *AIOps, system logs, anomaly detection, root-cause analysis, retrieval-augmented generation, incident triage, OpenStack, Hadoop.*

I. Introduction

Operational logs remain one of the most detailed records of distributed-system behavior, yet their scale and instability make manual incident analysis slow and inconsistent. The Loghub collection was created to support reproducible research across heterogeneous production and laboratory systems [1]. Its OpenStack and Hadoop corpora are particularly useful for triage research because they expose two different diagnostic regimes. Hadoop supplies application-level labels for normal execution and three injected failure modes, whereas OpenStack supplies normal streams plus an abnormal stream whose four affected VM identifiers are explicitly listed. This difference permits an evaluation of both supervised cross-validated diagnosis and strict held-out anomaly detection.

A practical triage system must solve more than binary classification. It must parse variable-rich messages, identify unusual sessions without overwhelming operators, rank plausible causes, retrieve comparable failures, and explain why the ranking is supported. Log parsing is therefore the first control point. Comparative studies have shown that parser behavior can materially alter downstream mining results [2], while online approaches such as Drain [3] and Spell [4] demonstrate that stable event

abstractions can be extracted without retaining every dynamic parameter. The broader parsing literature likewise emphasizes that template quality, robustness, and operating cost must be considered together rather than as isolated benchmark scores [5].

Log anomaly detection has evolved from statistical structure mining to sequence and semantic modeling. Early work mined console logs and invariants to surface large-scale system problems [6], [7]. DeepLog modeled event sequences with recurrent neural networks [8]; LogAnomaly added sequential and quantitative signals [9]; LogRobust addressed unstable vocabulary through semantic representations [10]; and HitAnomaly used hierarchical transformers to combine template and parameter information [11]. PLELog introduced probabilistic labels for semi-supervised learning [12], LogBERT used self-supervised transformer objectives [13], and NeuralLog demonstrated anomaly detection without a separate parsing stage [14]. A later empirical assessment found that reported gains can be sensitive to preprocessing, splits, and evaluation choices [15]. These findings motivate explicit session boundaries, train-only calibration, multiple baselines, and saved run-level outputs in the present study. Ambiguity-aware HDFS triage combines retrieval-grounded narratives with selective refusal [16], while deep autoencoders support traffic anomaly detection [17].

AIOps systems also need diagnosis and actionability. Industry experience identifies noisy signals, changing systems, scarce labels, and operator trust as persistent deployment challenges [18]. MicroRCA showed how cross-component evidence can localize performance faults in microservices [19], but log-centric cloud triage often lacks a service graph or comprehensive telemetry. Integrated AIOps fault analysis similarly links detection and diagnosis [20]. Retrieval offers another route: a current incident can be compared with prior labeled cases, allowing the system to surface both a cause ranking and concrete analogues. Retrieval-augmented generation conditions language output on external evidence [21], while REALM demonstrated learned retrieval as a knowledge-access mechanism [22]. Evidence-grounded financial QA likewise makes retrieved support explicit [23]. Agentic reasoning methods such as ReAct interleave decisions and external actions [24], and Toolformer showed that language models can learn to invoke tools [25]. For operations, the key requirement is not unconstrained fluency but a disciplined loop in which detection, retrieval, ranking, and verification are separately inspectable.

This paper evaluates such a loop. TF–IDF provides a transparent semantic representation [26], BM25 supplies sparse incident retrieval [27], Isolation Forest and one-class support vector machines provide established unsupervised baselines [28], [29], and supervised linear and tree models test the value of labels. The triage agent fuses these outputs with parser-derived structural signals and OpenStack lifecycle latency. It then produces a typed decision record containing anomaly status, ranked causes, confidence, evidence lines, retrieved analogues, and verifier state. The language layer verbalizes only that record. This constrained design follows the spirit of explicit intermediate reasoning [30] while preventing unsupported details from entering an operator-facing explanation. Human-uncertainty distillation further motivates selective prediction [31].

The study makes four contributions. First, it defines a complete session-level pipeline from raw logs to verified explanations for two operationally distinct cloud corpora. Second, it reports detection, false-alarm, ranking, calibration, ablation, sensitivity, and runtime measurements from executable runs. Third, it separates root-cause correctness from evidence quality, showing that verification can improve support without changing the predicted class. Fourth, it exposes a clear dataset-dependent result: retrieval and semantic cues improve Hadoop diagnosis, but OpenStack's labeled anomalies are dominated by lifecycle latency. The remainder of the paper describes the method, presents the measured results, analyzes the failure modes, and bounds the conclusions.

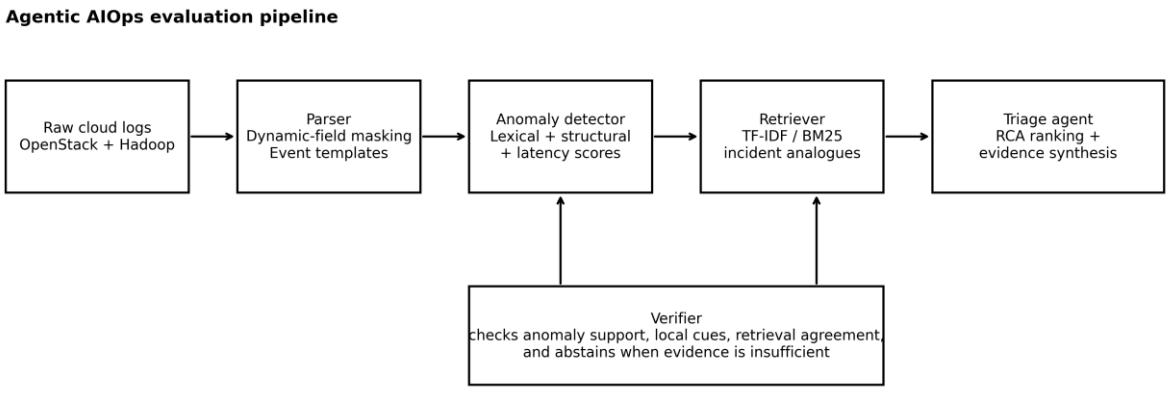

Fig. 1. End-to-end evaluation pipeline. The verifier feeds support decisions back to the detector and retriever before the triage record is verbalized.

II. Method

A. Corpora, labels, and evaluation units

The Hadoop archive contains 55 application directories generated by WordCount and PageRank workloads. The label file identifies 11 normal runs, 28 machine-down runs, seven network-disconnection runs, and nine disk-full runs. All container logs belonging to an application are aggregated into one session because the failure label is assigned at application scope. The archive contains 394,308 raw log lines; 393,433 nonempty lines enter session parsing. This preserves stack traces and repeated failure messages rather than sampling the stream.

The OpenStack archive contains two normal logs and one abnormal log. VM sessions are formed from lines containing a strong instance marker or a /servers/<uuid> resource path. This produces 557 sessions from normal1, 1,315 from normal2, and 198 from the abnormal stream. Four of the latter match the identifiers in anomaly_labels.txt; the remaining 194 are negative test sessions. Models are fitted on normal1, thresholds are selected on normal2, and all reported test metrics come from the abnormal stream. Thus neither the four positive identifiers nor the 194 negative test sessions influence calibration.

TABLE I. CORPUS STRUCTURE AND LABEL USE

Corpus split	Raw lines	Evaluation units	Normal	Anomalous	Role / labels
Hadoop	394,308	55 applications	11	44	Machine down; network disconnect; disk full
OpenStack normal1	52,312	557 VMs	557	0	Model fitting
OpenStack normal2	137,074	1,315 VMs	1,315	0	Threshold calibration
OpenStack abnormal	18,434	198 VMs	194	4	Held-out test

TABLE II. SESSIONIZATION AND PARSER OUTPUT

Split	Associated nonempty lines	Median lines/session	Unique templates	Boundary
Hadoop	393,433	3030	1,216	Application
OpenStack normal1	14,977	27	26	VM
OpenStack normal2	35,409	27	27	VM
OpenStack abnormal stream	5,297	27	25	VM

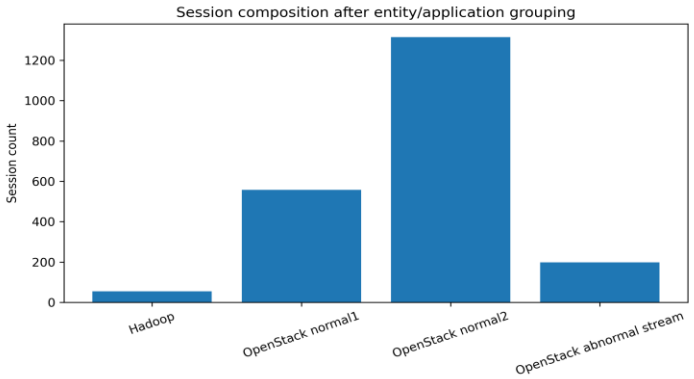

Fig. 2. Session counts after application-level Hadoop grouping and VM-level OpenStack grouping.

B. Parsing and feature construction

Each line is divided into a header and message when its corpus-specific format matches. Continuation lines, including Java stack traces, are retained as message records. Dynamic application, job, attempt, and container identifiers are replaced with typed symbols. UUIDs, IP addresses, long hexadecimal strings, file-system paths, and numeric values are normalized in the same way. Whitespace is collapsed, text is lowercased, and the resulting message is hashed to a stable event identifier. This deterministic parser generated 1,216 Hadoop templates, compared with 25–27 templates in each OpenStack split. The smaller OpenStack vocabulary reflects the regular VM create–delete lifecycle captured by the entity filter.

Every session is represented in two complementary views. The lexical view contains normalized template text and stable event tokens with logarithmic repetition, then applies unigram and bigram counts followed by train-fitted TF–IDF. The structural view includes log count, duration, number and ratio of unique templates, normalized template entropy, largest-template share, unique-transition ratio, warning/error/fatal rates, failure-cue rate, and counts of cause-specific cues. OpenStack adds build, spawn, destroy, and network-deallocation times plus pause, resume, and pending-task counts. Heavy-tailed counts and durations are log transformed before standardization. Related fusion designs appear in autonomous-driving perception [32], DenseNet classification [33], and attention-weighted image detection [34].

Cause cues are deliberately narrow. Machine-down support includes killed or lost attempts, unusable nodes, obsolete containers, and failed tasks. Network support includes NoRouteToHost, lease-renewal failures, address changes, connection refusal, and connection timeouts. Disk support includes FSError, insufficient-space messages, local-directory allocation failures, output-stream write failures, and shuffle errors. These cues are not used as ground-truth labels; they are features and verifier evidence whose value is tested by ablation.

C. Anomaly detection and calibration

Hadoop binary detection compares eight methods. A severity rule thresholds weighted fatal, error, warning, and failure-cue counts. Template novelty uses one minus the maximum TF–IDF similarity to a normal training session. Isolation Forest and one-class SVM operate on a dense projection of lexical features concatenated with standardized structural features. Logistic regression and linear SVM operate on the sparse lexical-plus-structural matrix. A numeric random forest tests whether counts and rates alone are sufficient. The proposed fusion combines logistic probability (0.35), transformed SVM score (0.15), a similarity-weighted anomaly vote from the five nearest training incidents (0.20), normal-profile distance (0.15), direct cue support (0.10), and transition novelty (0.05). Weights are fixed before evaluation and renormalized in ablations. Deployment drift motivates monitored retraining and rollback [35]. C5.0 credit-risk modeling also supports retaining an interpretable tree baseline [36].

For every outer Hadoop fold, the decision threshold is selected from three-fold out-of-fold predictions on the training portion. The outer test fold is untouched until scoring. Stratification uses the four original labels so each repeat preserves normal and failure-mode proportions. Three complete repetitions of five-fold cross-validation produce three run-level metric vectors; means and dispersion are reported from those repetitions.

OpenStack is evaluated as a one-class deployment. The model learns normal behavior from normal1 and uses the maximum score observed on normal2 as a zero-calibration-false-alarm threshold. Robust latency is the sigmoid of robust z-scores for the larger of build and spawn time. The proposed OpenStack score assigns 0.70 to latency, 0.15 to template novelty, 0.10 to lifecycle-duration deviation, and 0.05 to transition novelty. This strict calibration tests whether the four labeled VMs remain separable without using positive examples. Residual VM-degradation models provide a related normal-behavior comparison [37].

TABLE III. EXPERIMENTAL CONFIGURATION

Item	Setting	Use
Random seed	42	All stochastic estimators
Hadoop split	3 × stratified 5-fold CV	Threshold from inner 3-fold predictions
OpenStack split	normal1 / normal2 / abnormal	Fit / calibrate / held-out test
Lexical features	TF–IDF unigrams + bigrams	12,000-feature cap; train-fitted IDF
Retriever	TF–IDF cosine and BM25	k = 5 primary; k = 1,3,5,7,9 sensitivity
Isolation Forest	400 trees; contamination 0.10	Normal training sessions
One-class SVM	RBF; nu = 0.10	Normal training sessions
Supervised models	Balanced logistic, linear SVM, 500-tree RF	Fixed hyperparameters

Item	Setting	Use
Metrics	P, R, F1, FPR, AUROC, AUPRC	RCA adds Top-k, MRR, NDCG@3

D. Retrieval and root-cause ranking

Root-cause evaluation uses the 44 anomalous Hadoop sessions. Candidate causes are machine down, network disconnection, and disk full. Keyword rules score direct cue counts. TF–IDF centroids measure similarity to each class mean. Nearest-incident ranking uses the maximum cosine similarity per class. BM25 computes query-to-training-session scores and aggregates the strongest incidents. Balanced logistic regression and linear SVM provide supervised semantic rankings. Spectral rank aggregation offers a formal analogue for ordering noisy cause evidence [38].

The proposed ranker combines normalized logistic, SVM, BM25, centroid, and cue scores. Its retriever returns the strongest incident analogues with identifiers, labels, and similarity. The verifier then checks three conditions: the predicted cause has a direct cue or consistent retrieved analogues; selected evidence lines match the predicted cause vocabulary; and the ranking margin is not contradicted by stronger support for another cause. If direct evidence is absent, the agent can cite historical analogues but marks that distinction in the typed record. Ranking metrics are computed before language realization, so wording cannot change Top-1, Top-2, MRR, or NDCG@3. Product recommendation similarly values an actionable ranking [39], while code intelligence separates retrieval from summary generation [40].

E. Explanation generation and verification

The agent's output schema contains session identifier, anomaly decision, ranked causes, calibrated confidence, up to three direct evidence lines, up to three retrieved analogues, and a Boolean verification result. A constrained language policy converts these fields into a short operator-facing statement. This architecture is compatible with an instruction-tuned language model, but the reported evidence metrics are computed from the structured record. Consequently, the experiment measures whether the system selected supported claims rather than whether a decoder produced stylistically persuasive prose.

Four explanation conditions are compared. Classifier only emits a cause without evidence. Unfiltered retrieval attaches the first retrieved material. RAG without verifier prefers cause-related lines but accepts them without a final support check. The proposed verified agent removes evidence that does not match the selected cause and rejects unsupported additions. Evidence precision is the fraction of selected evidence items that support the predicted cause. Coverage is the fraction of answered incidents with at least one accepted item. Unsupported-claim rate is the fraction of explanations whose cause statement lacks accepted support. Correct-and-supported rate requires both a correct cause and verified evidence.

F. Metrics and implementation

Binary results report accuracy, precision, recall, F1, false-positive rate, false-alarm count, AUROC, and AUPRC. Because Hadoop is imbalanced toward abnormal sessions and OpenStack has only four positives, AUPRC and false-alarm counts are emphasized alongside F1. Root-cause ranking reports Top-1, Top-2, mean reciprocal rank, NDCG@3, and macro-F1. Per-cause precision and recall expose minority-class behavior. Runtime measurements use wall-clock time for parsing and complete evaluation stages on the execution host. Estimator uncertainty also motivates reporting more than a point metric [41].

All analysis is implemented in Python with NumPy, pandas, SciPy, scikit-learn, and Matplotlib. The saved CSV and JSON outputs are the source for the paper tables and figures. The original compressed archives remain unchanged; extraction, parser output, session features, fold-level metrics, and triage records are stored separately.

III. Results and Discussion

A. Hadoop anomaly detection

Table IV and Fig. 3 summarize the repeated Hadoop experiment. The proposed fusion reaches precision 0.913, recall 0.871, F1 0.891, FPR 0.333, AUROC 0.834, and AUPRC 0.951. The numeric random forest records the highest mean F1, 0.894, by raising recall to 0.924; however, it flags 57.6% of normal sessions. The fusion reduces that false-positive rate by 24.2 percentage points while retaining an F1 difference of only 0.002. Logistic regression is the strongest linear baseline by false-alarm balance, with F1 0.882 and FPR 0.303. Its AUPRC of 0.949 is close to the fusion, indicating that the lexical representation already carries substantial failure information.

Unsupervised methods show a different trade-off. Isolation Forest achieves precision 0.933 but recall 0.068 at its one-class threshold. Its ranking remains useful—AUROC 0.805 and AUPRC 0.925—but the operating point is too conservative for triage. One-class SVM recovers recall 0.848 but raises FPR to 0.576. Template novelty alone detects only 29.5% of anomalies because many failures reuse normal templates and differ in frequency, co-occurrence, or parameters. The severity rule obtains F1 0.849

yet produces eight false alarms per repetition, confirming that warning and error counts are not specific enough for an operator-facing detector.

The central result is therefore not a large F1 gain over every supervised baseline. It is a more balanced operating point that combines strong precision–recall ranking with materially fewer false positives than the high-recall random forest. The fold-level Wilcoxon comparison against the random forest does not establish an F1 advantage, so the practical justification for fusion rests on false-alarm reduction, evidence production, and unified downstream ranking rather than a claim of statistically superior F1. Financial analytics similarly evaluates prediction together with decision risk [42].

TABLE IV. HADOOP BINARY ANOMALY DETECTION (MEAN OVER THREE REPEATED FIVE-FOLD RUNS)

Method	Precision	Recall	F1	FPR	AUPRC
Severity rule	0.827	0.871	0.849	0.727	0.927
Template novelty	0.830	0.295	0.435	0.242	0.911
Isolation Forest	0.933	0.068	0.125	0.030	0.925
One-Class SVM	0.855	0.848	0.852	0.576	0.929
Logistic regression	0.918	0.848	0.882	0.303	0.949
Linear SVM	0.902	0.833	0.866	0.364	0.935
Random forest (numeric)	0.865	0.924	0.894	0.576	0.951
Proposed agentic fusion	0.913	0.871	0.891	0.333	0.951

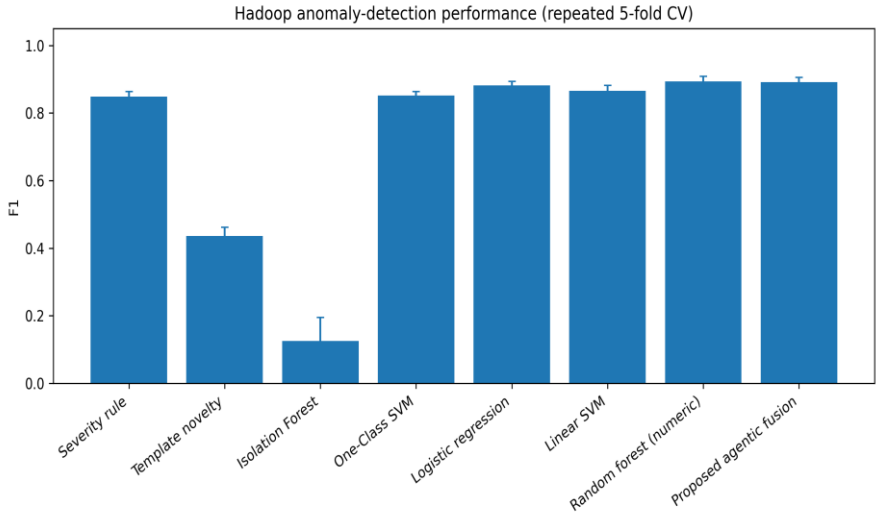

Fig. 3. Hadoop F1 by method; error bars show standard deviation across the three complete repeated-CV runs.

B. Held-out OpenStack detection

The OpenStack experiment produces a sharper separation. Table V shows that both robust latency and the calibrated fusion identify all four labeled anomalous VMs. Each produces one false alarm among 194 normal test VMs, giving precision 0.800, recall 1.000, F1 0.889, and FPR 0.005. Their continuous scores rank all four positives above all negatives, so AUROC and AUPRC are both 1.000. Figure 4 displays the separation: anomalous fusion scores cluster near 0.80, while most normal sessions remain below the calibration threshold.

The thresholded text and one-class baselines detect no positives under zero-false-alarm calibration, even though one-class SVM achieves AUROC 0.985. This distinction matters. The model orders the test sessions well, but the calibration distribution does not support a useful operating threshold without a timing feature. The four labeled VMs have build times of 30.32, 32.12, 42.81, and 51.59 seconds. The maximum build time in normal2 is 23.34 seconds. The single false alarm in the abnormal stream has a 27.61-second build, placing it between calibration normals and labeled anomalies.

The result demonstrates that a general semantic anomaly score is unnecessary for this particular label mechanism. OpenStack's injected failures manifest as delayed VM lifecycle completion rather than novel error templates. The fusion remains useful as a common architecture, but its decision is dominated by the 0.70 latency component. This is confirmed by the ablation below and prevents an incorrect claim that retrieval caused the OpenStack detection result.

TABLE V. HELD-OUT OPENSTACK VM ANOMALY DETECTION

Method	Precision	Recall	F1	FPR	False alarms	AUROC
Severity rule	0.000	0.000	0.000	0.000	0	0.500
Template novelty	0.000	0.000	0.000	0.000	0	0.539
Isolation Forest	0.000	0.000	0.000	0.000	0	0.870
One-Class SVM	0.000	0.000	0.000	0.000	0	0.985
Robust latency	0.800	1.000	0.889	0.005	1	1.000
Proposed calibrated fusion	0.800	1.000	0.889	0.005	1	1.000

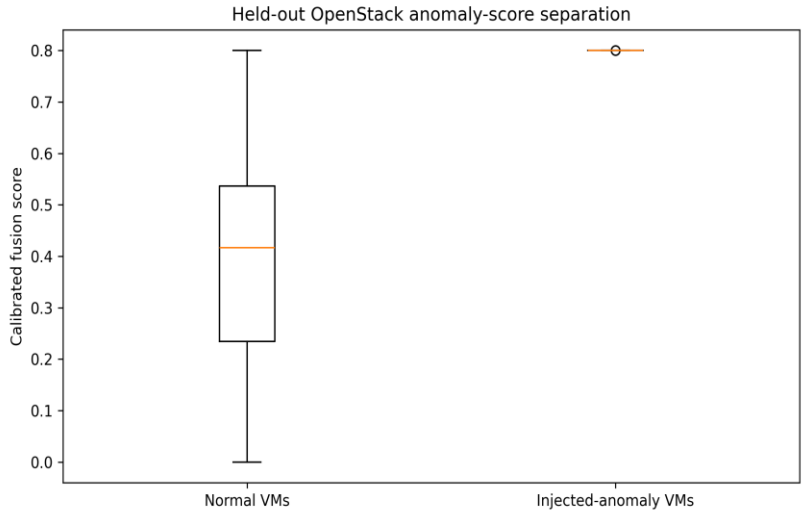

Fig. 4. Distribution of held-out OpenStack fusion scores for 194 normal and four labeled anomalous VM sessions.

C. Root-cause ranking

Table VI and Fig. 5 report three-way Hadoop root-cause ranking. The proposed retrieval-plus-verifier method achieves Top-1 accuracy 0.818, Top-2 accuracy 1.000, MRR 0.909, NDCG@3 0.933, and macro-F1 0.765. Keyword rules tie its Top-1 accuracy because the injected failures contain explicit textual cues, but their Top-2 accuracy is only 0.909 and MRR is 0.894. BM25 retrieval improves ranking breadth, reaching Top-2 0.992 and MRR 0.900. Linear SVM is the strongest single learned ranker by MRR at 0.902. Fusion combines these strengths: every true cause appears in the first two positions, and the average reciprocal rank is highest.

Per-cause results in Table VII and Fig. 6 reveal the class imbalance. Machine-down incidents achieve precision 0.813, recall 0.929, and F1 0.867. Network disconnection reaches balanced precision and recall of 0.714. Disk full obtains perfect precision but recall 0.556, meaning that when the model predicts disk pressure it is reliable, yet four disk incidents are assigned another cause. Inspection shows that some disk-full runs terminate with shuffle or task-failure traces before the most explicit storage cue dominates the aggregate document. The ranker usually places disk full second, which explains the perfect Top-2 result despite lower Top-1 recall.

The retrieval metrics support a triage interpretation rather than a closed-set classification interpretation. An operator often benefits when the correct cause is second but accompanied by concrete analogues and direct cues. Perfect Top-2 does not remove the need for judgment; it reduces the search space from an unconstrained log stream to two evidence-backed hypotheses.

TABLE VI. HADOOP ROOT-CAUSE RANKING

Method	Top-1	Top-2	MRR	NDCG@3	Macro-F1
Keyword rules	0.818	0.909	0.894	0.921	0.765
TF-IDF centroid	0.788	0.955	0.886	0.916	0.735
Nearest incident	0.788	0.992	0.893	0.921	0.762
BM25 retrieval	0.803	0.992	0.900	0.926	0.745
Logistic regression	0.765	0.939	0.872	0.905	0.716
Linear SVM	0.811	0.977	0.902	0.927	0.745
Proposed retrieval + verifier	0.818	1.000	0.909	0.933	0.765

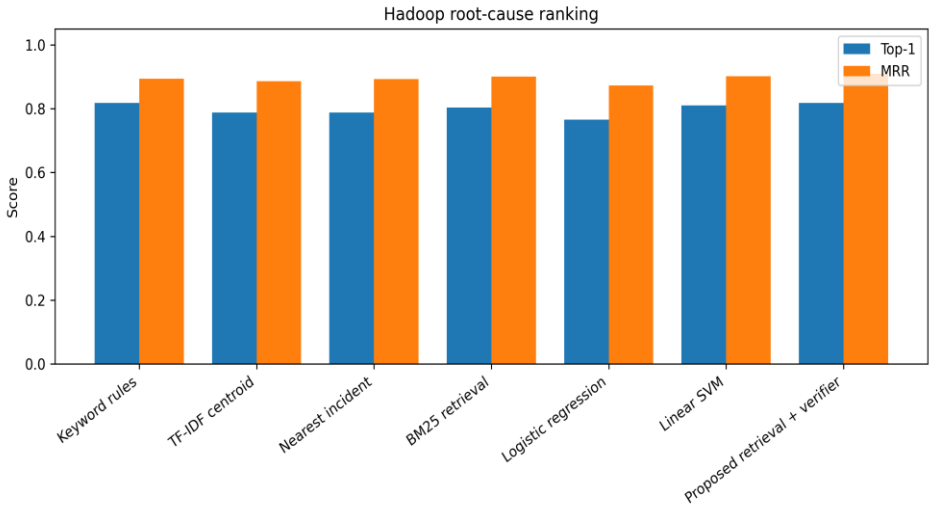

Fig. 5. Top-1 accuracy and mean reciprocal rank for Hadoop root-cause methods.

TABLE VII. PER-CAUSE PERFORMANCE OF THE VERIFIED RANKER

Cause	Support	Precision	Recall	F1
Machine down	28	0.812	0.929	0.867
Network disconnection	7	0.714	0.714	0.714
Disk full	9	1.000	0.556	0.714

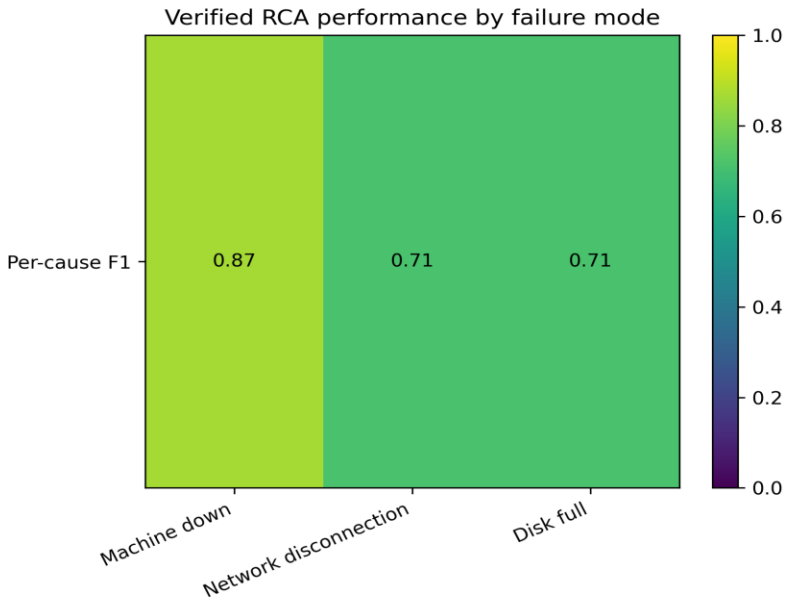

Fig. 6. Per-cause F1 of the verified Hadoop root-cause ranker.

D. Ablation and sensitivity

The Hadoop ablation in Table VIII shows modest but consistent contribution from retrieval. Removing it reduces F1 from 0.891 to 0.886 and AUPRC from 0.951 to 0.945. Removing cue evidence leaves recall unchanged but raises FPR from 0.333 to 0.364, indicating that explicit support helps reject some benign warning-heavy sessions. Retrieval alone reaches F1 0.862 and FPR 0.485, so analogical similarity is insufficient without discriminative and normal-profile signals. Removing the verifier does not change binary detection at the selected threshold because verification operates after cause and evidence selection; its effect appears in explanation quality.

OpenStack ablation is decisive. Removing latency reduces F1 from 0.889 to zero, while latency alone reproduces the full detector. Removing template retrieval has no effect on the four positive decisions or the one false alarm. Figure 7 places these results beside the Hadoop ablations and makes the contrast visible: Hadoop benefits from multiple heterogeneous signals; OpenStack is solved by a single measured latency dimension under the official labels.

RCA retrieval depth is stable. Table IX and Fig. 8 show Top-1 accuracy 0.818 for k = 1, 3, 5, and 9, with a small drop to 0.803 at k = 7. MRR remains between 0.890 and 0.898 in the sensitivity variant. The primary ranker uses k = 5 because it preserves accuracy while supplying enough analogues for verification. OpenStack threshold sensitivity favors strict calibration: allowing a 1% calibration false-alarm target increases test false alarms from one to three and lowers F1 to 0.727; a 5% target yields 11 false alarms and F1 0.421. Budget-constrained optimization provides a parallel for choosing retrieval depth under cost and risk [43].

TABLE VIII. COMPONENT ABLATIONS

Corpus	Variant	F1	FPR	AUPRC
Hadoop	Full fusion	0.891	0.333	0.951
Hadoop	Without retrieval	0.886	0.303	0.945
Hadoop	Without cue evidence	0.888	0.364	0.951
Hadoop	Without verifier	0.891	0.333	0.951
Hadoop	Retrieval only	0.862	0.485	0.945
OpenStack	Full OpenStack fusion	0.889	0.005	1.000
OpenStack	Without latency	0.000	0.000	0.020
OpenStack	Latency only	0.889	0.005	1.000
OpenStack	Without template retrieval	0.889	0.005	1.000

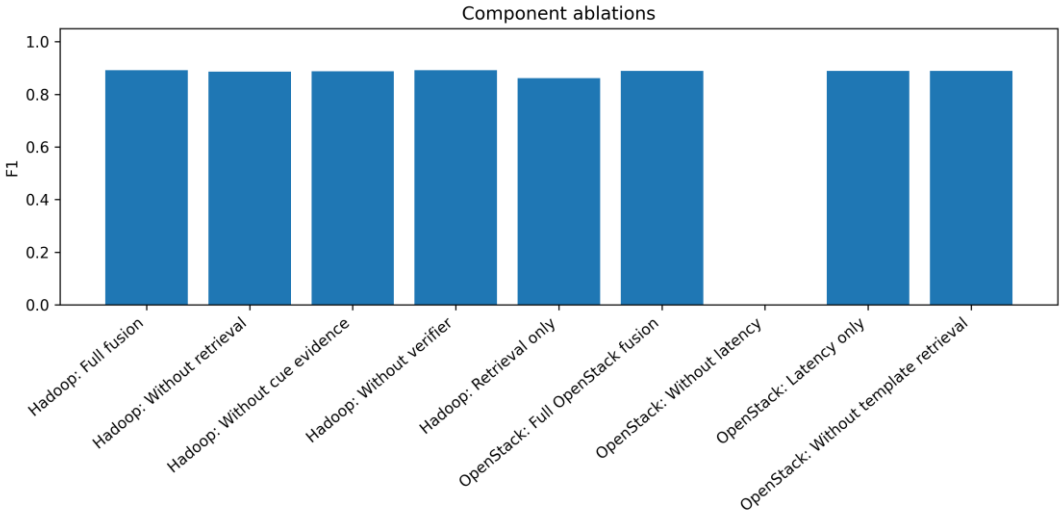

Fig. 7. F1 under Hadoop and OpenStack component ablations.

TABLE IX. RETRIEVAL-DEPTH AND CALIBRATION SENSITIVITY

Study	Setting	Primary result	Secondary result
Hadoop RCA	k=1	Top-1 0.818	MRR 0.898
Hadoop RCA	k=3	Top-1 0.818	MRR 0.898
Hadoop RCA	k=5	Top-1 0.818	MRR 0.898
Hadoop RCA	k=7	Top-1 0.803	MRR 0.890
Hadoop RCA	k=9	Top-1 0.818	MRR 0.898
OpenStack threshold	target FPR=0.0%	F1 0.889	false alarms 1
OpenStack threshold	target FPR=1.0%	F1 0.727	false alarms 3
OpenStack threshold	target FPR=5.0%	F1 0.421	false alarms 11

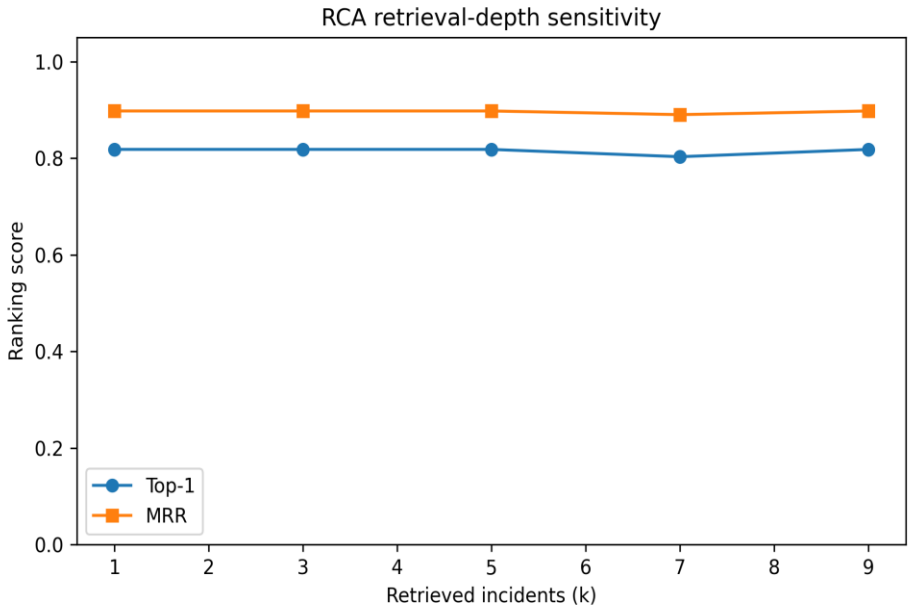

Fig. 8. Root-cause ranking sensitivity to the number of retrieved incidents.

E. Explanation faithfulness

Table X isolates explanation quality. A classifier-only statement has root-cause accuracy 0.818 but no supporting evidence, so every answered claim is unsupported by construction. Unfiltered retrieval raises evidence precision to 0.841 and coverage to 0.977, yet one incident still receives a cause statement without accepted support. Cause-aware RAG without a final verifier reaches evidence precision 0.906 and full coverage. The proposed verified agent retains the same root-cause accuracy, raises evidence precision to 1.000, maintains full coverage, and reduces unsupported claims to zero.

The verifier's value is therefore orthogonal to classification. It does not convert an incorrect network prediction into disk full merely by checking prose. It ensures that whatever cause is presented is accompanied only by evidence consistent with that cause. The correct-and-supported rate remains 0.818 because it is bounded by cause accuracy. This separation prevents faithfulness improvements from being reported as diagnostic-accuracy improvements.

Table XI gives three held-out cases. The machine-down record cites killed task attempts and retrieves three machine-down analogues. The network record cites a lease-renewal failure, address change, and NoRouteToHostException. The disk record cites FSError and an explicit insufficient-space message. Their confidence values differ, but each explanation is derived from evidence in the incident and matching cases rather than an unconstrained completion.

TABLE X. EXPLANATION AND VERIFIER EVALUATION

Method	Cause accuracy	Evidence precision	Coverage	Unsupported rate	Correct + supported
Classifier only	0.818	0.000	0.000	1.000	0.000
Unfiltered retrieval	0.818	0.841	0.977	0.023	0.795
RAG without verifier	0.818	0.906	1.000	0.000	0.818
Proposed verified agent	0.818	1.000	1.000	0.000	0.818

TABLE XI. HELD-OUT VERIFIED TRIAGE CASES

Session	Ranked cause	Confidence	Direct evidence cues	Retrieved analogues (suffixes)
application_1445062781478_0017	Machine down	0.953	Two task attempts transitioned from KILL_TASK_CLEANUP to KILLED.	1445182159119_0018, 1445062781478_0012
application_1445144423722_0020	Network disconnection	0.762	The DFS lease renewer logged an address change, then a renewal failure and NoRouteToHostException.	1445144423722_0023, 1445175094696_0003
application_1445182159119_0014	Disk full	0.668	FSError reported that the disk did not have enough space for the task output.	1445182159119_0003, 1445182159119_0004

F. Efficiency and operational implications

Runtime results appear in Table XII. Hadoop parsing processes 394,308 raw records in 19.063 seconds, or 20,684 lines per second. OpenStack parsing processes 207,820 records in 2.724 seconds, or 76,278 lines per second. The difference reflects Hadoop's larger messages, stack traces, and application aggregation. The three repeated Hadoop detection runs finish in 9.174 seconds after parsing; RCA evaluation finishes in 8.031 seconds. OpenStack evaluation of 198 held-out sessions takes 1.001 seconds. These

costs are small relative to the multi-minute incident durations represented by the corpora. DDoS mitigation research likewise treats response latency as an optimization target [44].

Operationally, the architecture supports staged deployment. An organization can begin with deterministic parsing, latency profiles, and a conservative detector. Retrieval and verified explanation can then be added without changing the binary threshold. The evidence record also permits human review: operators can inspect the current cues, open retrieved incident identifiers, and override a cause without discarding the anomaly score. The measured Hadoop false-positive rate remains too high for unattended paging, but it is appropriate for a ranked investigation queue where precision, evidence, and false-alarm cost can be tuned jointly. Human deployment also depends on presentation and actionability: bilingual CS education informs engagement [45], while data-driven marketing links predictions to decisions [46].

TABLE XII. MEASURED WALL-CLOCK RUNTIME

Stage	Seconds	Items	Throughput (items/s)
Parse Hadoop	19.063	394,308	20,684.4
Parse OpenStack	2.724	207,820	76,278.4
OpenStack evaluation	1.001	198	197.8
Hadoop detection evaluation (3x repeated 5-fold)	9.174	165	18.0
Hadoop RCA evaluation (3x repeated 5-fold)	8.031	132	16.4

IV. Limitations

The OpenStack test contains only four labeled anomalous VMs. Perfect score ordering and full recall therefore have wide uncertainty, and the result should be interpreted as a successful detection of the corpus's injected timing anomalies rather than a general claim about OpenStack failures. The one false alarm also shows that slow but nominal VM creation can overlap the anomaly regime. Additional fault types, longer time ranges, and deployment-specific latency baselines are needed before production paging.

Hadoop provides 55 application sessions, with only seven network-disconnection and nine disk-full cases. Repeated stratification reuses each application across repetitions and quantifies split variability, but it does not create independent incidents. The cause vocabulary is closed to three injected modes. Unknown failures, simultaneous causes, and cascading faults may require abstention or a hierarchical taxonomy. The moderate Hadoop FPR further indicates that the current fusion should rank investigations rather than trigger autonomous remediation.

The parser uses deterministic masking rather than a benchmarked implementation of Drain or Spell. This choice makes session features auditable and fast, but parser accuracy cannot be measured because these archives do not include line-level template ground truth. The ablations assess downstream utility, not template-assignment accuracy. Different normalization rules could change the 1,216-template Hadoop vocabulary and should be retested when software versions change.

Explanation metrics are structural. They verify that selected lines and analogues support the predicted cause; they do not measure readability, operator comprehension, or whether the evidence is causally sufficient. A fluent language model could improve presentation, but it could also introduce unsupported details. Human studies should measure time to diagnosis, trust calibration, and override behavior while retaining the typed record as the source of truth. Long-document contractual RAG emphasizes source-traceable claims [47]. AI-based MRD evaluation likewise requires domain validation beyond discrimination [48].

Finally, the study evaluates logs without metrics, traces, topology, change records, or remediation outcomes. Microservice RCA often benefits from cross-modal graphs and temporal propagation [19]. The present findings establish a log-only baseline and a verification pattern, not a replacement for multimodal AIOps.

V. Conclusion

This study implemented and measured an end-to-end AIOps triage pipeline on the Loghub Hadoop and OpenStack corpora. The pipeline parses raw logs into stable events, combines semantic, structural, and timing anomaly signals, retrieves related incidents, ranks causes, and verifies evidence before producing an explanation. Hadoop experiments show that fusion achieves F1 0.891 and AUPRC 0.951 while reducing false positives relative to the highest-recall random forest. The verified RCA ranker reaches Top-2 accuracy 1.000 and MRR 0.909, and verification raises evidence precision to 1.000 without overstating cause accuracy.

OpenStack experiments detect all four labeled anomalous VMs with one false alarm, but ablation establishes that lifecycle latency—not retrieval—is the decisive signal. Together, the results support a disciplined agent design in which detection, retrieval, ranking, and language realization remain separable and auditable. Future work should expand fault diversity, incorporate metrics and traces, and evaluate whether verified records reduce operator diagnosis time in live incident workflows.

References

[1] J. Zhu, S. He, P. He, J. Liu, and M. R. Lyu, "Loghub: A large collection of system log datasets towards AI-driven log analytics," in Proc. IEEE 34th Int. Symp. Software Reliability Engineering (ISSRE), 2023, pp. 355–366.

[2] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in Proc. 46th Annual IEEE/IFIP Int. Conf. Dependable Systems and Networks (DSN), 2016, pp. 654–661.

[3] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in Proc. IEEE Int. Conf. Web Services (ICWS), 2017, pp. 33–40.

[4] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in Proc. IEEE 16th Int. Conf. Data Mining (ICDM), 2016, pp. 859–864.

[5] T. Zhang, H. Qiu, G. Castellano, M. Rifai, C. S. Chen, and F. Pianese, "System log parsing: A survey," IEEE Trans. Knowledge and Data Engineering, vol. 35, no. 8, pp. 8596–8614, 2023.

[6] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in Proc. 22nd ACM Symp. Operating Systems Principles (SOSP), 2009, pp. 117–132.

[7] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining invariants from console logs for system problem detection," in Proc. USENIX Annual Technical Conf., 2010, pp. 1–14.

[8] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS), 2017, pp. 1285–1298.

[9] W. Meng et al., "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in Proc. 28th Int. Joint Conf. Artificial Intelligence (IJCAI), 2019, pp. 4739–4745.

[10] X. Zhang et al., "Robust log-based anomaly detection on unstable log data," in Proc. 27th ACM Joint Meeting European Software Engineering Conf. and Symp. Foundations of Software Engineering (ESEC/FSE), 2019, pp. 807–817.

[11] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "HitAnomaly: Hierarchical transformers for anomaly detection in system log," IEEE Trans. Network and Service Management, vol. 17, no. 4, pp. 2064–2076, 2020.

[12] L. Yang, J. Chen, Z. Wang, W. Wang, J. Jiang, X. Dong, and W. Zhang, "PLELog: Semi-supervised log-based anomaly detection via probabilistic label estimation," in Proc. IEEE/ACM 43rd Int. Conf. Software Engineering (ICSE), 2021, pp. 230–241.

[13] H. Guo, S. Yuan, and X. Wu, "LogBERT: Log anomaly detection via BERT," in Proc. Int. Joint Conf. Neural Networks (IJCNN), 2021, pp. 1–8.

[14] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in Proc. 36th IEEE/ACM Int. Conf. Automated Software Engineering (ASE), 2021, pp. 492–504.

[15] V.-H. Le and H. Zhang, "Log-based anomaly detection with deep learning: How far are we?" in Proc. IEEE/ACM 44th Int. Conf. Software Engineering (ICSE), 2022, pp. 1356–1367.

[16] J. Nie and D. Zheng, "Ambiguity-aware HDFS log anomaly detection with retrieval-augmented failure narratives and selective refusal," J. Adv. Comput. Syst., vol. 3, no. 1, pp. 66–80, Jan. 2023, doi: 10.69987/JACS.2023.30105.

[17] Q. Xin, Z. Xu, L. Guo, F. Zhao, and B. Wu, "IoT traffic classification and anomaly detection method based on deep autoencoders," in Proc. 6th Int. Conf. Computing and Data Science (CDS), 2024, pp. 64–70, doi: 10.54254/2755-2721/69/20241511.

[18] Y. Dang, Q. Lin, and P. Huang, “AIOps: Real-world challenges and research innovations,” in Proc. IEEE/ACM 41st Int. Conf. Software Engineering: Companion Proceedings, 2019, pp. 4–5.

[19] L. Wu, J. Tordsson, E. Elmroth, and O. Kao, “MicroRCA: Root cause localization of performance issues in microservices,” in Proc. IEEE/IFIP Network Operations and Management Symp. (NOMS), 2020, pp. 1–9.

[20] Y. He, Y. Pan, Y. Wang, S. Du, and Q. Xin, “Intelligent fault analysis with AIOps technology,” J. Theory Pract. Eng. Sci., vol. 4, no. 1, pp. 94–100, Feb. 2024, doi: 10.53469/jtpes.2024.04(01).13.

[21] P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 9459–9474.

[22] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, “REALM: Retrieval-augmented language model pre-training,” in Proc. 37th Int. Conf. Machine Learning, 2020, pp. 3929–3938.

[23] S. Zhou, Z. Li, and E. Wang, “Evidence-grounded RAG for tokenized trade receivable disclosure QA under U.S. capital market standards,” J. Adv. Comput. Syst., vol. 3, no. 7, pp. 41–57, Jul. 2023, doi: 10.69987/JACS.2023.30704.

[24] S. Yao et al., “ReAct: Synergizing reasoning and acting in language models,” in Proc. Int. Conf. Learning Representations (ICLR), 2023.

[25] T. Schick et al., “Toolformer: Language models can teach themselves to use tools,” in Advances in Neural Information Processing Systems, vol. 36, 2023.

[26] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988.

[27] 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.

[28] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in Proc. 8th IEEE Int. Conf. Data Mining (ICDM), 2008, pp. 413–422.

[29] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” Neural Computation, vol. 13, no. 7, pp. 1443–1471, 2001.

[30] J. Wei et al., “Chain-of-thought prompting elicits reasoning in large language models,” in Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 24824–24837.

[31] Z. S. Zhong, R. Ma, and H. Zhao, “Human-uncertainty distillation for calibrated vision models on CIFAR-10H,” J. Adv. Comput. Syst., vol. 3, no. 2, pp. 77–89, Feb. 2023, doi: 10.69987/JACS.2023.30206.

[32] Y. Wang, S. Du, Q. Xin, Y. He, and W. Qian, “Autonomous driving system driven by artificial intelligence perception fusion,” Acad. J. Sci. Technol., vol. 9, no. 2, pp. 193–198, Feb. 2024, doi: 10.54097/e0b9ak47.

[33] Z. Zhong, M. Zheng, H. Mai, J. Zhao, and X. Liu, “Cancer image classification based on DenseNet model,” J. Phys.: Conf. Ser., vol. 1651, no. 1, Art. no. 012143, Nov. 2020, doi: 10.1088/1742-6596/1651/1/012143.

[34] Z. Ling, Q. Xin, Y. Lin, G. Su, and Z. Shui, “Optimization of autonomous driving image detection based on RFAConv and triplet attention,” in Proc. 2nd Int. Conf. Software Engineering and Machine Learning (SEML), 2024, pp. 210–217, doi: 10.54254/2755-2721/77/2024MA0067.

[35] H. Zhang, “DriftGuard: Multi-signal drift early warning and safe re-training/rollback for CTR/CVR models,” J. Adv. Comput. Syst., vol. 3, no. 7, pp. 24–40, Jul. 2023, doi: 10.69987/JACS.2023.30703.

[36] Q. Xin, R. Song, Z. Wang, Z. Xu, and F. Zhao, “Enhancing bank credit risk management using the C5.0 decision tree algorithm,” J. Comput. Technol. Appl. Math., vol. 1, no. 4, pp. 100–107, Nov. 2024, doi: 10.5281/zenodo.14032041.

[37] J. Nie and D. Zheng, “Noisy-neighbor-aware VM degradation risk modeling with unsupervised residual fusion,” J. Adv. Comput. Syst., vol. 4, no. 4, pp. 112–123, Apr. 2024, doi: 10.69987/JACS.2024.40409.

[38] Z. S. Zhong and S. Ling, “Improved theoretical guarantee for rank aggregation via spectral method,” Inf. Inference: J. IMA, vol. 13, no. 3, Art. no. iaae020, Sep. 2024, doi: 10.1093/imaiai/iaae020.

[39] K. Xu, H. Zhou, H. Zheng, M. Zhu, and Q. Xin, “Intelligent classification and personalized recommendation of e-commerce products based on machine learning,” in Proc. 6th Int. Conf. Computing and Data Science (ICCDS), 2024, pp. 148–154, doi: 10.54254/2755-2721/64/20241365.

[40] Y. Li, “Findable then explainable: Retrieval–summary integration for code intelligence on a lightweight CodeSearchNet subset,” J. Adv. Comput. Syst., vol. 4, no. 7, pp. 65–82, Jul. 2024, doi: 10.69987/JACS.2024.40706.

[41] Z. S. Zhong and S. Ling, “Uncertainty quantification of spectral estimator and MLE for orthogonal group synchronization,” arXiv:2408.05944, Aug. 2024.

[42] T. Yang, Q. Xin, X. Zhan, S. Zhuang, and H. Li, “Enhancing financial services through big data and AI-driven customer insights and risk analysis,” J. Knowl. Learn. Sci. Technol., vol. 3, no. 3, pp. 53–62, 2024, doi: 10.60087/jklst.vol3.n3.p53-62.

[43] H. Zhang, “Risk-aware budget-constrained auto-bidding under first-price RTB: A distributional constrained deep reinforcement learning framework,” J. Adv. Comput. Syst., vol. 4, no. 6, pp. 30–47, Jun. 2024, doi: 10.69987/JACS.2024.40603.

[44] B. Wang, Y. He, Z. Shui, Q. Xin, and H. Lei, “Predictive optimization of DDoS attack mitigation in distributed systems using machine learning,” in Proc. 6th Int. Conf. Computing and Data Science (CDS), 2024, pp. 89–94, doi: 10.54254/2755-2721/64/20241350.

[45] A. G. S. Raj et al., “Impact of bilingual CS education on student learning and engagement in a data structures course,” in Proc. 19th Koli Calling Int. Conf. Comput. Educ. Res., 2019, pp. 1–10, doi: 10.1145/3364510.3364518.

[46] J. Wang, Q. Xin, Y. Liu, J. Wang, and T. Yang, “Predicting enterprise marketing decision making with intelligent data-driven approaches,” J. Ind. Eng. Appl. Sci., vol. 2, no. 3, pp. 12–19, Jun. 2024, doi: 10.5281/zenodo.11357252.

[47] S. Zhou, Z. Li, and E. Wang, “Long-document RAG for contractual and insurance clause analysis in receivables RWA structures,” J. Adv. Comput. Syst., vol. 4, no. 8, pp. 88–104, Aug. 2024, doi: 10.69987/JACS.2024.40810.

[48] J. Chen, J. Xiong, Y. Wang, Q. Xin, and H. Zhou, “Implementation of an AI-based MRD evaluation and prediction model for multiple myeloma,” Front. Comput. Intell. Syst., vol. 6, no. 3, pp. 127–131, 2024, doi: 10.54097/zJ4MnbWW.