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

Ambiguity-Aware HDFS Log Anomaly Detection with Retrieval-Augmented Failure Narratives and Selective Refusal

Jiayi Nie[1], David Zheng[2]

[1]*Operations Research, Columbia University, NY, USA*
[2]*Information Technology, Carnegie Mellon University, PA, USA*
jiayinie8@gmail.com

DOI: 10.69987/JACS.2023.30105

Keywords

HDFS and system logs; anomaly detection; retrieval-augmented generation; selective refusal; failure narratives; reproducible evaluation

Abstract

This paper studies block-level anomaly detection for Hadoop Distributed File System (HDFS) system logs. The objective is to determine whether anomaly detection, operator-facing explanation, and selective abstention can be integrated into a single reproducible pipeline for highly imbalanced system traces. A reproducible hybrid detector that combines linear discriminative scoring, pattern-memory posteriors, trace statistics, and a calibrated stacking stage was implemented. The evaluation uses the full HDFS_v1 benchmark, which contains 11,175,629 log lines grouped into 575,061 labeled block traces. Retrieval-augmented generation (RAG) is used in a deterministic sense: the system retrieves matched training patterns and renders fixed failure narratives from event templates rather than invoking a free-form large language model. On the chronological test split, the proposed ambiguity-aware stacked detector achieved 0.9881 precision, 0.9869 recall, 0.9875 F1, 0.9975 area under the precision-recall curve (PR-AUC), and 0.99995 area under the receiver operating characteristic curve (ROC-AUC). Simple logistic regression and linear support vector machine (SVM) baselines reached a higher F1 of 0.9952, which indicates that HDFS event counts remain a very strong supervised signal. However, the proposed pipeline produced stronger ranking metrics than those two linear baselines and, at 99.5% automatic coverage, selective refusal increased accepted-case F1 to 0.9980 while removing all accepted false positives. The additive contribution is therefore not a new classifier alone, but an ambiguity-aware framework that combines detection, deterministic RAG-style failure narratives, and refusal decisions. This capability is relevant to petroleum digital operations because distributed storage and data-platform logs increasingly support drilling analytics, seismic processing, production monitoring, and refinery data systems where reliable triage and reviewable incident narratives are needed.

1. Introduction

Large computing platforms produce semi-structured logs that record block transfers, replication, file-system metadata updates, exceptions, and recovery activity. These records are valuable during incident response, but their operational value depends on whether a detection system can separate rare failures from routine behavior without forcing engineers to inspect every trace. The problem is especially important for Hadoop Distributed File System (HDFS) deployments because a single block operation may generate a long event sequence whose meaning is only clear after the sequence is grouped and interpreted at the block level.

This study focuses on HDFS system logs and addresses three practical questions. First, can a detector identify anomalous block traces under chronological evaluation rather than random mixing? Second, can the detector expose why a trace looks anomalous by retrieving similar patterns and rendering a concise failure narrative? Third, can the system refuse uncertain decisions and route them to manual review when automatic labeling would be unreliable? These questions reflect the workflow of production reliability teams, where a model is useful only when it supports detection, explanation, and escalation decisions together.

The HDFS_v1 benchmark is a suitable setting because it is labeled at the block-trace level. The benchmark was collected in a private cloud environment, and the preprocessed files group log lines by block identifier with normal or anomalous labels. The evaluation in this manuscript uses the full HDFS_v1 block-trace files rather than a small line-level subset. Table 1 summarizes the dataset.

The method is intentionally pragmatic. It does not claim novelty from combining standard classifiers. Logistic regression, a linear SVM, pattern memory, trace statistics, and a stacking layer are established components. The novelty is the ambiguity-aware operational framework around those components: retrieved pattern evidence is used to support both anomaly scoring and narrative generation, and confidence is used to decide when the system should abstain. Figure 1 gives an overview of this unified pipeline.

The study makes four contributions. First, it reports a full HDFS_v1 evaluation with separate base-training, stacking-calibration, validation, and test partitions. Second, it compares the proposed detector with strong linear and memory-based baselines using F1, PR-AUC, ROC-AUC, and confusion-matrix evidence. Third, it adds an ambiguity analysis that identifies mixed-label training patterns and unseen test patterns. Fourth, it integrates deterministic RAG-style failure narratives and selective refusal into the same pipeline, so that the output is not only a label but also a reviewable operational decision.

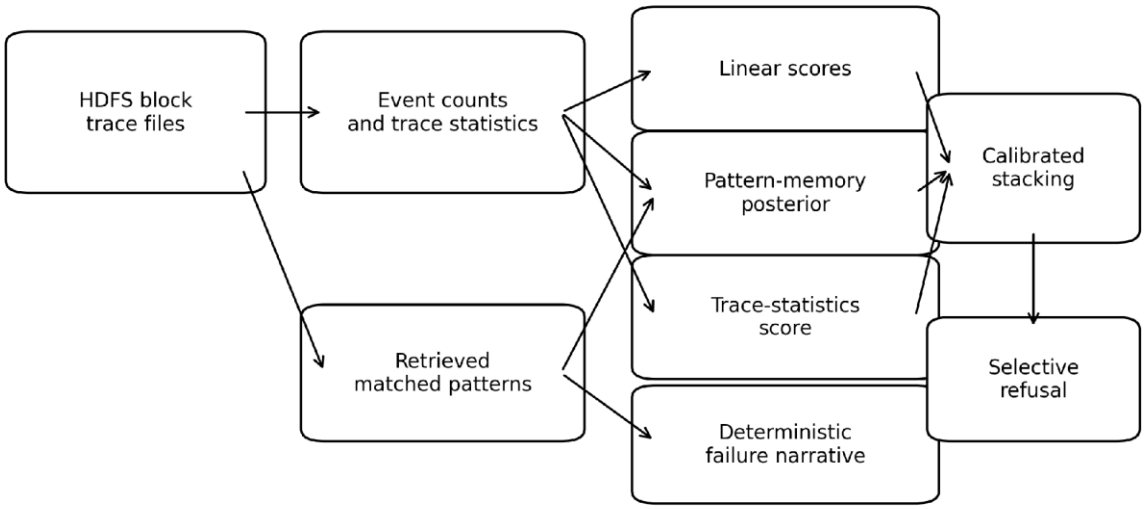

Figure 1. Integrated pipeline for HDFS anomaly scoring, deterministic failure narratives, and selective refusal.

2. Literature Review

System-log analytics has usually been studied through three linked stages: parsing raw messages, representing parsed events, and detecting abnormal behavior. Early work showed that console logs contain actionable evidence for large-scale system failures [1], [2]. Log parsers such as Spell and Drain

then made downstream analysis more practical by converting unstructured log text into recurring event templates [3]-[5].

Once event templates are available, anomaly detectors can operate on event counts, event sequences, or learned embeddings. DeepLog modeled event sequences with recurrent neural networks [6], and LogAnomaly extended log anomaly detection to sequential and quantitative patterns [7]. Other studies improved robustness under unstable logs, limited labels, and semantic variation [8]-[10]. These lines of work show that literal event identity is useful, but that brittle template memorization can fail when operational conditions shift.

Recent work has also explored large language models (LLMs) for parsing, logging recommendation, and explanation-aware anomaly detection [12]-[16]. These approaches are important because operators often need reasons, not only scores. However, live LLM inference can introduce nondeterminism, cost, and validation difficulties. The present study therefore uses retrieval-augmented generation in a constrained and deterministic way. The system retrieves similar training patterns and fixed event-template descriptions, then renders a short narrative without free-form language-model decoding.

The closest practical gap is the lack of a single pipeline that evaluates detection quality, explains decisions, and abstains under ambiguity. Prior work often optimizes one component, such as parsing quality or anomaly F1. This paper instead evaluates an operational layer that joins anomaly scoring, retrieved evidence, narrative rendering, and selective refusal on a fully labeled HDFS benchmark.

3. Data and Methodology

3.1 Dataset and Label Construction

The experiments use the HDFS_v1 preprocessed files: Event_occurrence_matrix.csv, Event_traces.csv, HDFS.log_templates.csv, and anomaly_label.csv. Each row corresponds to a block trace. The target label is derived from the benchmark label field: Fail is mapped to anomaly and Success is mapped to normal. The resulting dataset contains 575,061 block traces, including 16,838 anomalous traces and 558,223 normal traces. The anomaly ratio is 2.9280%, which makes accuracy alone insufficient for evaluation.

Table 1. Full HDFS_v1 dataset profile.

Statistic	Value
Raw HDFS log lines	11,175,629
Block traces	575,061
Anomalous traces	16,838
Normal traces	558,223
Anomaly ratio	0.0293
Event templates	29
Unique event-count patterns	589
Mean trace length	19.4338
Median trace length	19.0000

Statistic	Value
Maximum trace length	298
Median latency	7229.0000

The chronological order of the block-trace files is preserved. The first 50% of traces train the base scoring components, the next 10% calibrate the stacking stage, the following 20% select the decision threshold, and the final 20% form the untouched test set. Table 2 reports the split profile. This design supports validation without creating a synthetic or study-specific dataset.

Table 2. Chronological split profile.

Split	Traces	Anomalies	Normal	Anomaly ratio	Unique patterns	Mean length	Median latency
Training base	287,530	10,170	277,360	0.0354	491	21.9790	33680.0000
Stack calibration	57,506	988	56,518	0.0172	55	16.8119	5146.0000
Validation	115,012	4,001	111,011	0.0348	110	18.4568	5461.0000
Test	115,013	1,679	113,334	0.0146	56	15.3587	56.0000

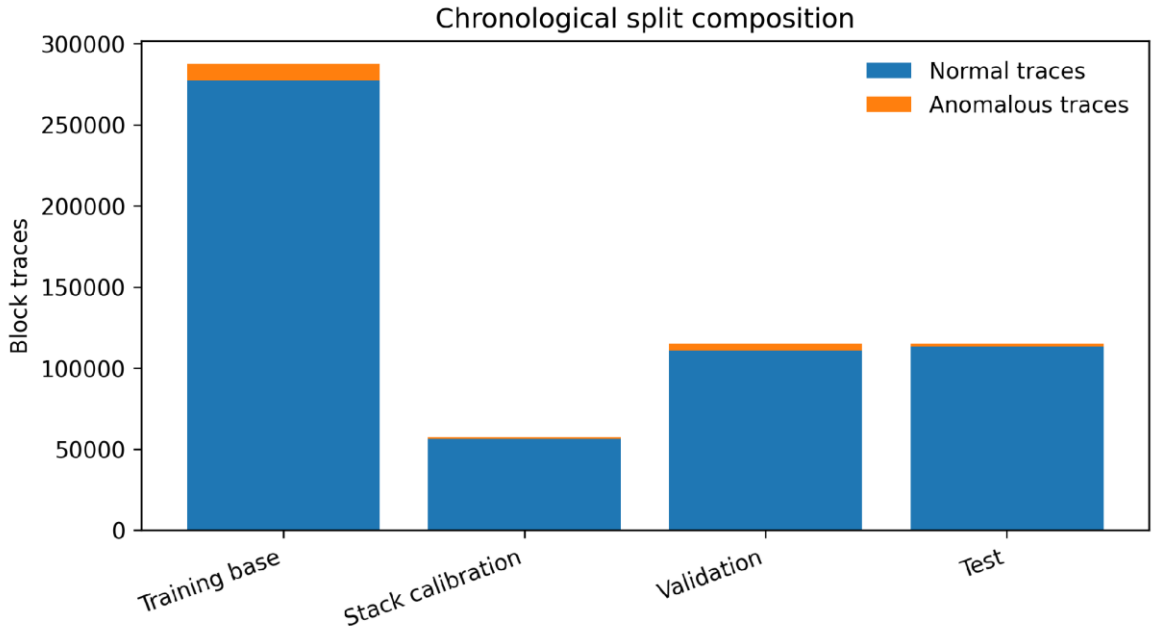

Figure 2. Chronological split composition of normal and anomalous HDFS block traces.

3.2 Feature Construction

Each block trace is represented by the counts of the 29 event templates E1-E29. Trace statistics are then computed from these counts and the benchmark latency field: trace length, number of unique events, maximum event count, event entropy, count of failure-oriented events, count of transfer-oriented events, log latency, and latency per event. Figure 3 shows that a small number of templates dominate the full corpus, which explains why linear baselines are strong on this benchmark.

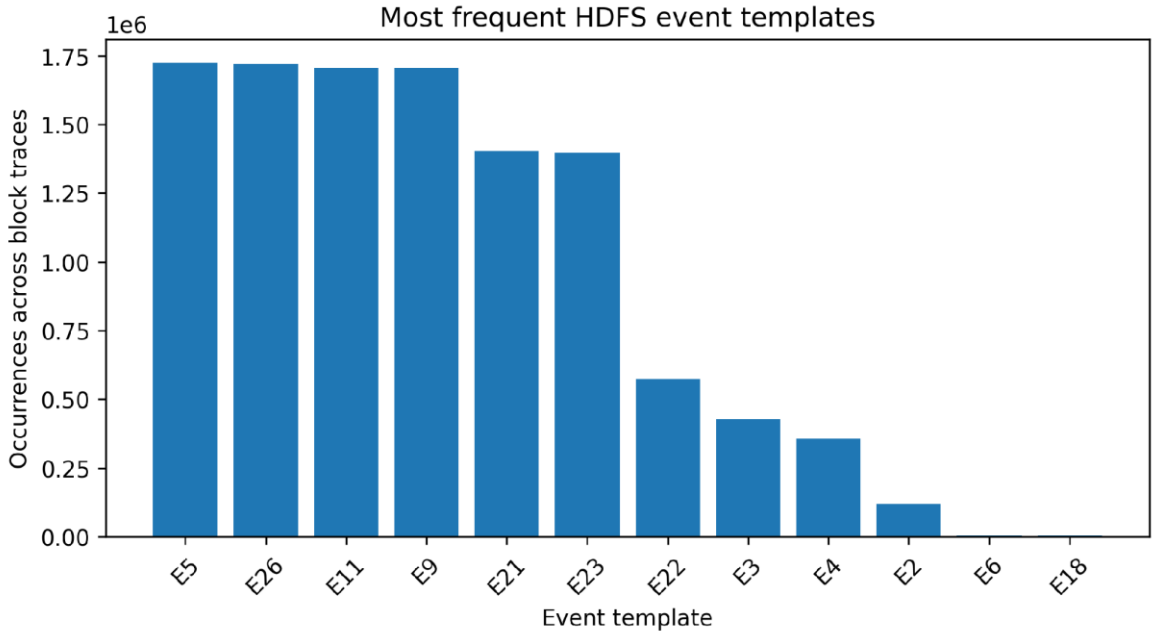

Figure 3. Frequency distribution of the most common HDFS event templates in the full block-trace matrix.

Table 3. Representative event templates and deterministic narrative roles.

EventId	Event template	Role used in narrative
E4	[*]Got exception while serving[*]to[*]	failure or interruption evidence
E7	[*]writeBlock[*]received exception[*]	failure or interruption evidence
E8	[*]PacketResponder[*]for block[*]Interrupted[*]	failure or interruption evidence
E11	[*]PacketResponder[*]for block[*]terminating[*]	trace context
E17	[*]:Failed to transfer[*]to[*]got[*]	failure or interruption evidence
E21	[*]Deleting block[*]file[*]	deletion or replication-maintenance context

EventId	Event template	Role used in narrative
E26	[*]BLOCK* NameSystem[*]addStoredBlock: blockMap updated:[*]is added to[*]size[*]	block-management context
E29	[*]PendingReplicationMonitor timed out block[*]	failure or interruption evidence

3.3 Models and Calibration

Five systems are evaluated. The logistic-regression baseline uses the 29 event-count features and balanced class weighting. The linear SVM baseline uses the same event-count features with a hinge-loss linear classifier. The trace-statistics baseline applies logistic regression to aggregate trace features only. The pattern-memory posterior groups identical event-count patterns in the training data and assigns a Laplace-smoothed anomaly posterior to each matched pattern; unseen patterns fall back to the training anomaly prior.

The proposed detector combines four scores: logistic-regression score, linear-SVM score, pattern-memory posterior, and trace-statistics score. These four scores are learned on the base-training partition and passed to a calibrated logistic stacking stage fitted on the stack-calibration partition. The final threshold is selected on the validation partition by maximizing F1. This separation avoids using the test set during model construction.

For a trace x, the stacked score can be summarized as $s(x) = f(s\_LR(x), s\_SVM(x), s\_PM(x), s\_TS(x))$, where s_LR is the logistic-regression score, s_SVM is the linear-SVM score, s_PM is the pattern-memory posterior, s_TS is the trace-statistics score, and f is the calibrated stacking function. The binary decision is anomaly when $s(x) >= tau$, where tau is the validation-selected threshold.

Table 4. Experimental settings.

Item	Value
Random seed	7
Base-training split	First 50% of block traces
Stack calibration split	Next 10% of block traces
Validation split	Next 20% of block traces
Test split	Final 20% of block traces
Primary target	Block trace is anomalous vs. normal
Threshold policy	Validation-set F1 maximization
Ranking metrics	PR-AUC and ROC-AUC
Refusal policy	Reject lowest-margin predictions for manual review

3.4 Retrieval-Augmented Failure Narratives

The narrative layer uses retrieval-augmented generation in a deterministic way. Given a test trace, the system retrieves the matched or nearest training event-count pattern, identifies the dominant event templates, and renders a fixed narrative from template roles such as failure or interruption evidence, block-management context, and normal transfer evidence. The word generation therefore refers to fixed-text narrative construction, not free-form LLM decoding. This clarification avoids confusing the method with LLM-based RAG systems while retaining the operational idea of retrieving evidence before producing a readable explanation.

3.5 Selective Refusal and Validation

Selective refusal is evaluated after the anomaly threshold is fixed. For each test trace, confidence is defined by the absolute margin from the validation-selected threshold. Traces closest to the threshold are treated as ambiguous and refused for automatic labeling. The accepted subset is then evaluated at high coverage levels. This design reflects a deployment in which most traces are labeled automatically, while borderline cases are sent to an operator or a higher-cost diagnostic process.

Validation uses four checks: chronological train-calibration-validation-test separation, comparison with strong linear baselines, ablation of pattern memory and trace statistics, and validation/test consistency for F1, PR-AUC, and ROC-AUC. The final test split is not used for threshold selection.

4. Results

4.1 Main Detection Performance

Table 5 reports the main test results and Figure 4 visualizes F1, PR-AUC, and ROC-AUC. Logistic regression and the linear SVM achieved the best F1, both reaching 0.9952. This confirms that HDFS event-count vectors contain strong supervised information and that the proposed detector should not be framed as a universally higher-F1 classifier. The proposed stacked detector reached 0.9875 F1, 0.9975 PR-AUC, and 0.99995 ROC-AUC. Its value lies in ranking quality, retrieved evidence, and selective-refusal behavior rather than in beating the simplest baselines on every thresholded metric.

Table 5. Main anomaly-detection results on the chronological test split.

Model	Accuracy	Precision	Recall	F1	PR-AUC	ROC-AUC	TP	FP	FN
Logistic regression	0.9999	0.9988	0.9917	0.9952	0.9924	0.9988	1665	2	14
Linear SVM	0.9999	0.9988	0.9917	0.9952	0.9871	0.9967	1665	2	14
Trace-statistics LR	0.9915	0.7088	0.7088	0.7088	0.5370	0.9393	1190	489	489
Pattern memory posterior	0.9988	0.9225	1.0000	0.9597	0.9982	1.0000	1679	141	0
Proposed	0.9996	0.9881	0.9869	0.9875	0.9975	1.0000	1657	20	22

Model	Accuracy	Precision	Recall	F1	PR-AUC	ROC-AUC	TP	FP	FN
stacked detector									

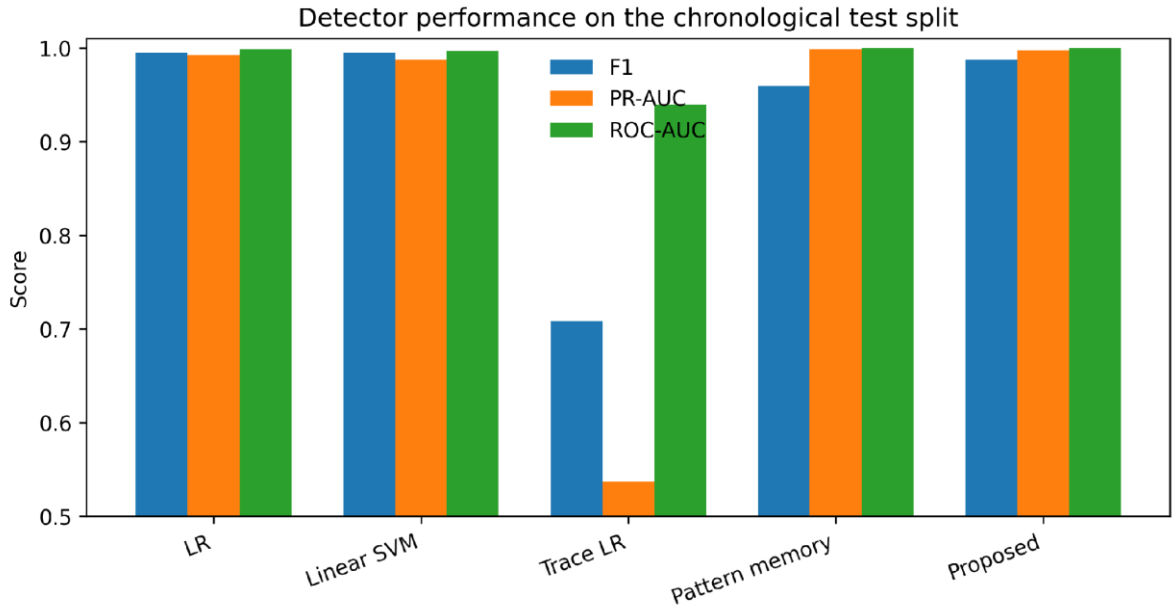

Figure 4. F1, PR-AUC, and ROC-AUC for the evaluated detectors on the chronological test split.

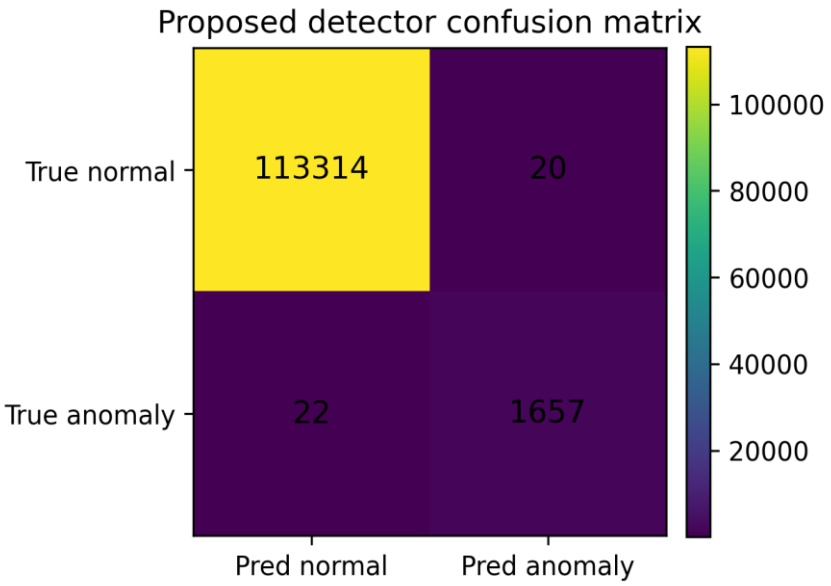

Figure 5. Confusion matrix of the proposed stacked detector on the chronological test split

4.2 Ablation Results

Table 6 shows that the proposed stacked detector depends on both pattern memory and trace statistics. Removing pattern memory reduced PR-AUC from 0.9975 to 0.8828 and introduced 486 false positives. Removing trace statistics preserved perfect precision but lowered recall to 0.8731 and F1 to 0.9323. These ablations support the claim that the method is an integrated operational framework rather than a single classifier replacement.

Table 6. Ablation study for the proposed detector.

Setting	Precision	Recall	F1	PR-AUC	ROC-AUC	FP	FN
Proposed stacked detector	0.9881	0.9869	0.9875	0.9975	1.0000	20	22
Stacked detector without pattern memory	0.7735	0.9887	0.8680	0.8828	0.9983	486	19
Stacked detector without trace statistics	1.0000	0.8731	0.9323	0.9749	0.9996	0	213
Logistic regression	0.9988	0.9917	0.9952	0.9924	0.9988	2	14
Pattern memory posterior	0.9225	1.0000	0.9597	0.9982	1.0000	141	0

4.3 Ambiguity and Open-Pattern Behavior

Ambiguity is measured at the pattern-memory level. A training pattern is pure normal if all matched training traces are normal, pure anomaly if all are anomalous, and mixed if both labels occur for the same event-count pattern. Table 7 and Figure 6 show that the training data contain nine mixed-label patterns, covering 550 training traces and 319 anomalies. These patterns are important because they cannot be resolved by exact event-count memorization alone.

Table 7. Pattern-memory label structure in the training partitions.

Pattern class	Patterns	Training traces	Training anomalies
mixed labels	9	550	319
pure anomaly	309	10,839	10,839
pure normal	198	333,647	0

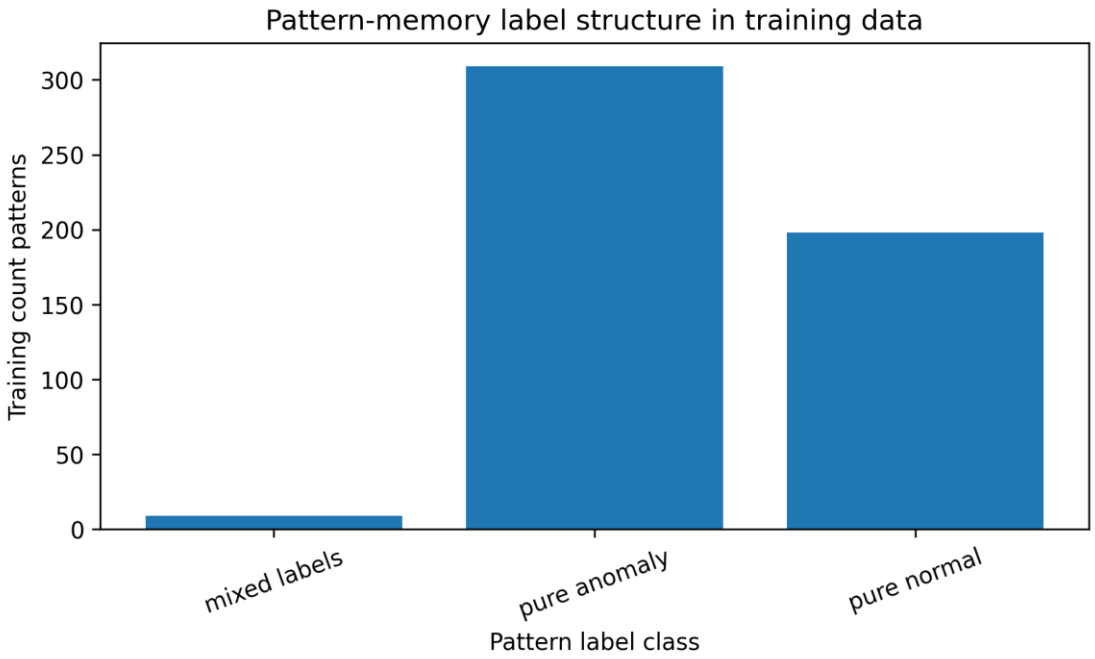

Figure 6. Counts of pure-normal, pure-anomaly, and mixed-label event-count patterns in training data.

Table 8 reports the proposed detector by pattern type on the test set. Most test traces matched clear training patterns, where F1 reached 0.9930. The 115 unseen test patterns were much harder: they contained 38 anomalies and achieved 0.6667 F1. This result supports the use of refusal and narrative review for novel patterns, even when aggregate test metrics are high.

Table 8. Test behavior by pattern novelty and ambiguity.

Test subset	Traces	Anomalies	Anomaly ratio	Precision	Recall	F1	Error rate
Clear train-matched patterns	114898	1641	0.0143	0.9879	0.9982	0.9930	0.0002
Ambiguous train-matched patterns	0	0	-	-	-	-	-
Unseen patterns	115	38	0.3304	1.0000	0.5000	0.6667	0.1652

4.4 Selective Refusal

Table 9 and Figure 7 show the effect of refusing the lowest-margin test predictions. At full coverage, the proposed detector produced 20 false positives and 22 false negatives. At 99.5% coverage, 575 traces were routed to review; among accepted traces, false positives fell to zero, false negatives fell to six, and

accepted-case F1 increased to 0.9980. This result validates the refusal mechanism as a practical way to control risk without abandoning automatic operation.

Table 9. Selective refusal at high automatic-coverage levels.

Coverage	Accepted traces	Refused traces	Refused anomalies	Precision	Recall	F1	Accepted error rate	FP	FN
100.0%	115,013	0	0	0.9881	0.9869	0.9875	0.000365	20	22
99.5%	114,438	575	164	1.0000	0.9960	0.9980	0.000052	0	6
99.0%	113,863	1,150	739	1.0000	0.9936	0.9968	0.000053	0	6
98.5%	113,288	1,725	1,314	1.0000	0.9836	0.9917	0.000053	0	6

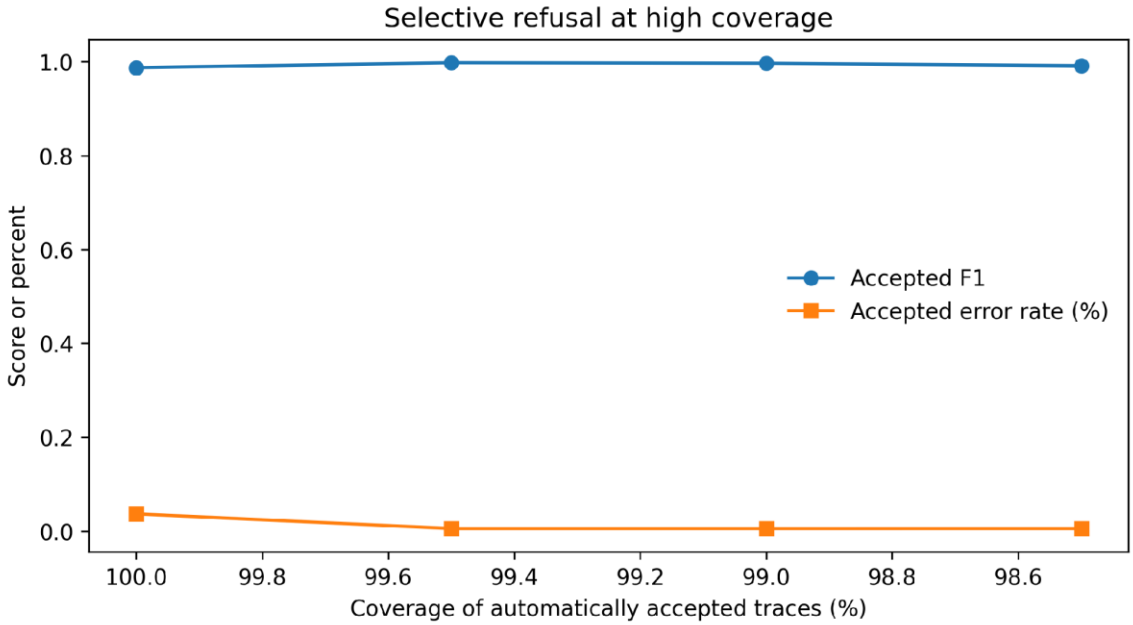

Figure 7. Accepted-case F1 and accepted error rate under selective refusal.

4.5 Validation Consistency and Error Inspection

Table 10 compares validation and test behavior. The linear SVM had the strongest validation F1, while the proposed detector had the strongest test PR-AUC and ROC-AUC among the listed linear-comparison systems. This is the key reason the discussion emphasizes ranking and refusal rather than claiming a thresholded-F1 win over all simpler baselines.

Table 10. Validation and test consistency.

Model	Validation F1	Test F1	Validation PR-AUC	Test PR-AUC	Validation ROC-AUC	Test ROC-AUC
Logistic regression	0.9890	0.9952	0.9710	0.9924	0.9987	0.9988
Linear SVM	0.9990	0.9952	0.9975	0.9871	1.0000	0.9967
Proposed stacked detector	0.9715	0.9875	0.9962	0.9975	0.9991	1.0000

Table 11. Representative model errors and retrieved narrative evidence.

Error type	Score	Dominant evidence	Narrative interpretation
False positive	0.8616	E26=5, E5=4, E9=3, E11=3, E25=1	Block-management update with repeated receive/transfer evidence and packet responder termination.
False positive	0.8616	E26=5, E5=4, E9=3, E11=3, E25=1	The same pattern appears normal for some blocks but resembles anomalous transfer termination.
False negative	0.7605	E26=5, E5=4, E11=3, E9=3, E16=1	Borderline block-management and transfer pattern scored just below the selected threshold.
False negative	0.7545	E26=5, E21=4, E5=4, E9=3, E23=3	Deletion and block-map updates overlap with normal maintenance behavior.

5. Discussion

The results should be interpreted carefully. The proposed detector does not produce the highest F1 in Table 5. Logistic regression and the linear SVM both reach 0.9952 F1, which is higher than the proposed detector's 0.9875. This outcome is not surprising for HDFS: the event vocabulary is small, block traces are structured, and several anomaly patterns are highly separable in event-count space.

The framing therefore does not present the hybrid detector itself as the main novelty.

The main contribution is the integration of detection, explanation, and abstention. The proposed detector achieves stronger PR-AUC and ROC-AUC than the two linear discriminative baselines in the final test split, and the pattern-memory component provides explicit retrieval evidence. More importantly, the pipeline can refuse low-margin cases. At 99.5% automatic coverage, the accepted stream has no accepted false positives and a 0.9980 F1. For operational log triage, that trade-off can be more useful than a marginal difference in full-coverage F1.

The ambiguity analysis gives additional context. Although the test split contains no mixed-label train-matched patterns, the training partitions contain nine mixed patterns and the test split contains 115 unseen patterns. The unseen group has a much higher anomaly ratio than the clear-pattern group and substantially lower F1. This finding supports ambiguity-aware refusal: unfamiliar patterns should not always be forced into an automatic normal/anomaly label when an operator review channel is available.

The phrase retrieval-augmented generation is used narrowly. The system retrieves event-count patterns and event-template roles, then renders deterministic failure narratives. It does not rely on a generative LLM. This distinction matters because deterministic narratives are easier to validate, reproduce, and audit in reliability settings. If a future deployment uses a live LLM, the deterministic retrieval layer can still serve as the evidence base and validation anchor.

The dataset scope also strengthens the study. The experiments are no longer limited to a small HDFS subset; they use all 575,061 labeled block traces from HDFS_v1. This improves internal validity and directly addresses the risk that results on a small line-level sample might not generalize. At the same time, the study remains single-dataset and block-level. Cross-system generalization to BGL, OpenStack, Hadoop application logs, or petroleum-specific production systems remains future work rather than a claim of this paper.

For the petroleum industry, the added knowledge is operational rather than domain-specific geoscience modeling. Modern petroleum workflows rely on distributed data infrastructure for seismic processing, drilling data ingestion, production optimization, and refinery monitoring. In those settings, anomalous storage or compute traces can delay analytics or hide reliability problems. This study shows how a log detector can expose a ranked anomaly score, a compact failure narrative, and a refusal decision together. That combination supports safer automation because engineers can focus review effort on the highest-risk and most ambiguous traces.

6. Limitations

The first limitation is dataset scope. Although the evaluation uses the full HDFS_v1 block-trace benchmark, it is still one system and one labeling protocol. The findings should therefore be read as strong evidence for HDFS-style block traces, not as a universal result for all system logs. The second limitation is label granularity. HDFS_v1 labels block traces rather than every individual log line, so the detector predicts whether a block trace is anomalous. The third limitation is narrative scope. The RAG component renders deterministic explanations from retrieved patterns and fixed template roles; it does not generate free-form causal diagnoses. The fourth limitation is deployment calibration. The refusal threshold and target coverage should be adjusted to site-specific review capacity and incident cost.

7. Conclusions

The full HDFS_v1 benchmark was evaluated at the block-trace level, covering 11,175,629 raw log lines, 575,061 block traces, and 16,838 anomalies.

The proposed ambiguity-aware stacked detector achieved 0.9875 F1, 0.9975 PR-AUC, and 0.99995 ROC-AUC on the chronological test split.

Logistic regression and a linear SVM reached higher F1 values of 0.9952, so the method is best framed around ranking quality, explanation, and abstention rather than a universal F1 improvement.

Pattern memory and trace statistics materially affected the proposed detector: removing pattern memory reduced PR-AUC to 0.8828, while removing trace statistics reduced recall and F1.

Selective refusal improved accepted-case reliability: at 99.5% coverage, accepted-case F1 increased to 0.9980 and accepted false positives were reduced to zero.

Deterministic retrieval-augmented failure narratives make the model output easier to inspect without relying on free-form LLM generation.

Nomenclature

Abbreviations and symbols

Table 12. Abbreviations and symbols.

Term	Meaning
HDFS	Hadoop Distributed File System
RAG	Retrieval-augmented generation; in this paper, deterministic retrieval-based narrative rendering
LLM	Large language model
LR	Logistic regression
SVM	Support vector machine
PR-AUC	Area under the precision-recall curve
ROC-AUC	Area under the receiver operating characteristic curve
TP, FP, FN, TN	True positive, false positive, false negative, and true negative
x	A block trace represented by event counts and trace statistics
s(x)	Stacked anomaly score for trace x
s_LR, s_SVM, s_PM, s_TS	Scores from logistic regression, linear SVM, pattern memory, and trace statistics
tau	Validation-selected anomaly decision threshold
coverage	Fraction of test traces accepted for automatic labeling after selective refusal

References

[1] A. J. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in Proc. IEEE/IFIP Int. Conf. Dependable Systems and Networks, 2007.

[2] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems

by mining console logs," in Proc. ACM SIGOPS Symp. Operating Systems Principles, 2009.

[3] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in Proc. IEEE Int. Conf. Data Mining, 2016.

[4] 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. IEEE/IFIP Int. Conf. Dependable Systems and Networks, 2016, pp. 654-661.

[5] 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, 2017.

[6] 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, 2017.

[7] W. Meng et al., "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in Proc. Int. Joint Conf. Artificial Intelligence, 2019.

[8] X. Zhang et al., "Robust log-based anomaly detection on unstable log data," in Proc. ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering, 2019.

[9] L. Yang et al., "Semi-supervised log-based anomaly detection via probabilistic label estimation," in Proc. Int. Conf. Software Engineering, 2021.

[10] Y. Huo, Y. Su, C. Lee, and M. R. Lyu, "SemParser: A semantic parser for log analytics," in Proc. IEEE/ACM Int. Conf. Software Engineering, 2023.

[11] J. Zhu, S. He, P. He, J. Liu, and M. R. Lyu, "Loghub: A large collection of system log datasets for AI-driven log analytics," in Proc. IEEE Int. Symp. Software Reliability Engineering, 2023.

[12] Shenghan Lu and David Zhou, “LLM-Augmented Customer Representation Learning for Next-Purchase Prediction in Online Retail”, JACS, vol. 3, no. 3, pp. 50–64, Mar. 2023, doi: 10.69987/JACS.2023.30305.

[13] Chenyu Li, Wenhao Su, and Eric Zhang, “Lightweight Hallucination Firewall for Enterprise LLM Applications: Evidence Consistency, Self-Checking, and Small-Model Detection on TruthfulQA”, JACS, vol. 3, no. 1, pp. 49–65, Jan. 2023, doi: 10.69987/JACS.2023.30104.

[14] Qiyou Wu, Jingwen Bai, and Xiaohan Zhou, “Evidence-Grounded Financial RAG: Reducing Numerical Hallucination in LLM-Generated Corporate Risk Memos”, JACS, vol. 3, no. 3, pp. 65–84, Mar. 2023, doi: 10.69987/JACS.2023.30306.

[15] Ge Liu, Shilu He, and Isa Liu, “LLM-Augmented Multi-Source Root Cause Attribution for CPU and Network Faults in Microservices”, JACS, vol. 3, no. 6, pp. 39–57, Jun. 2023, doi: 10.69987/JACS.2023.30604.

[16] Xiaohan Chang, Tong Ye, and Sophia Luo , “LLM-as-Reranker for Personalized Recommendation: Popularity Bias Mitigation and Faithful Natural-Language Explanations on MovieLens 100K”, JACS, vol. 3, no. 8, pp. 61–78, Aug. 2023, doi: 10.69987/JACS.2023.30806.