# Semantic and Temporal Feature Augmentation for Log-Based System Failure Prediction

1st Yuan Tian
*School of Computer Science*
*Wuhan University*
Wuhan, China
tytian@whu.edu.cn

2nd Shi Ying
*School of Computer Science*
*Wuhan University*
Wuhan, China
yingshi@whu.edu.cn

3rd Li Junjun
*School of Computer Science*
*Wuhan University*
Wuhan, China
lijunjun@whu.edu.cn

***Abstract*—Early failure prediction from system logs is notoriously difficult because pre-failure signals are weak, heterogeneous, and easily overwhelmed by massive noisy logs. A key but under-exploited fact is that these early signals are structured across two levels: (i) rare semantic cues embedded in individual log messages, and (ii) temporal dynamics reflecting how the system state gradually drifts and occasionally transitions toward failure. Motivated by this observation, we propose a dual-level semantic and temporal feature augmentation approach for robust early failure prediction from noisy logs. At the semantic level, we introduce a random-walk-driven word reweighting mechanism that injects frequency-imbalance priors between normal and failure-related corpora, amplifying failure-indicative tokens without relying on log parsing or domain knowledge. At the temporal level, we design a complementary dynamics fusion predictor that couples an attention-based BiLSTM with multiple HMM components, where BiLSTM captures long-range nonlinear evolution trends while HMMs model local Markovian state transitions associated with abnormal shocks. The two representations are then fused for accurate sequence classification. Extensive experiments on four public log datasets show that our method consistently outperforms strong failure prediction baselines and substantially surpasses representative log anomaly detectors, yielding higher F1 scores and improved recall under noisy conditions with practical inference overhead. These results demonstrate that jointly augmenting semantic saliency and temporal dynamics is crucial for reliable early failure prediction in real-world logging environments.**

***Index Terms*—System log, Failure prediction, Random walk model, Feature augmentation**

## I. Introduction

Modern software and cyber-physical infrastructures, such as cloud platforms, datacenters, and HPC clusters, require high availability, where even brief downtimes can cause significant economic losses and service disruptions. System logs record detailed runtime behaviors, making them a key data source for proactive reliability management and failure prediction [1]. However, early failure prediction from logs is challenging. Pre-failure symptoms often appear as weak signals with low signal-to-noise ratios, easily overwhelmed by a large volume of normal, redundant logs, leading to high miss rates when applying conventional sequence modeling or anomaly detection methods [2]. Moreover, failure manifestations are heterogeneous. Some failures show subtle semantic deviations in individual log messages, while others emerge as short-term bursts, repetitions, or mild statistical irregularities in sequences.

A key observation is that early failure evidence in logs is structured across two complementary levels. At the message level, failure-indicative words or phrases are rare and show a clear frequency imbalance between normal and failure-related logs. Amplifying these semantic cues can significantly improve separability, even in noisy environments. At the sequence level, failure progression is reflected not only in long-term system state trends but also in local abnormal shocks that trigger state transitions and short-term Markovian irregularities. Existing approaches often focus on one aspect, making them fragile under noisy conditions and weak signals. These insights suggest that reliable early failure prediction requires enhancing both semantic saliency and temporal dynamics together, rather than relying on a single aspect.

Based on this observation, we propose a dual-level semantic and temporal feature augmentation approach for early failure prediction from noisy logs. For semantic-level augmentation, we introduce a random-walk-based word reweighting mechanism that leverages the frequency imbalance between normal and failure-related logs, automatically assigning higher weights to failure-indicative tokens. For temporal-level augmentation, we design a complementary dynamics fusion predictor that combines an attention-based BiLSTM with multiple HMM components. BiLSTM captures long-range non-linear dependencies and global evolution trends, while HMMs model local first-order state transitions and short-term statistical patterns associated with abnormal shocks.

We evaluate the proposed approach on four public log datasets, covering diverse system scales and noise conditions. Experimental results show consistent improvements over strong failure prediction baselines and clear advantages over log anomaly detectors. These findings confirm that jointly augmenting semantic saliency and temporal dynamics is essential for reliable early failure prediction in real-world noisy logging environments.Our contributions are as follows:

- **Dual-level viewpoint:** We propose a dual-level approach for early failure prediction from noisy logs, emphasizing

This work was supported in part by the National Natural Science Foundation of China under Grants 62472329 and 62072342, and in part by the National Key Research and Development Program of China under Grant 2022YFB3304300.

the importance of jointly modeling semantic saliency and temporal dynamics in low-SNR conditions.

- **Random-walk-driven semantic reweighting:** We introduce a random-walk-based semantic reweighting mechanism that amplifies failure-indicative tokens, improving message-level separability.
- **Complementary temporal fusion predictor:** We design a fusion predictor that combines an attention-based BiLSTM with multiple HMM components, capturing both long-term evolution trends and local Markovian transitions.

## II. Related Work

### A. Log-based Failure Prediction and Early Warning

Early failure prediction from system logs is crucial for dependable large-scale systems, yet remains challenging because weak pre-failure signals are often buried in noisy and redundant logs. Classic log mining studies establish logs as a primary source for system diagnosis [3]–[5]. Unlike post-hoc diagnosis, early warning must detect failures in advance under strict false-alarm constraints while remaining effective across different lead times [6]. Existing methods can be broadly grouped into three categories: sequence forecasting, which predicts future log-event sequences for failure inference, e.g., Clairvoyant [7] and Time Machine [8]; rule-based prediction, which extracts executable failure patterns, e.g., Aarohi [9]; and feature-engineered prediction, which builds classical predictors on multivariate prefix features, e.g., Prefix [10]. Recent studies also explore combining logs with other telemetry such as traces and metrics, showing the value of cross-source evidence for complex failure analysis [11].

### B. Log Anomaly Detection and Representation Learning

Log anomaly detection mainly learns normal patterns and identifies deviations. Representative methods include DeepLog [12], LogBERT [13], and recent self-supervised methods such as LogSD [14] and SpikeLog [15]. Recent studies further improve robustness to unstable logs, unseen events, and parsing noise through enhanced log representation learning [16], [17]. Empirical analyses also show that anomaly detection performance is highly sensitive to representation choices and preprocessing strategies [18]. However, most anomaly detection methods are designed for post-hoc deviation discovery rather than early warning, and may therefore miss subtle pre-failure signals that still resemble normal behavior.

## III. Method

### A. Overview

We segment a log stream into overlapping windows with span $W$ and step $\Delta$. We denote the $j$-th window as $\mathbf{X}_j$ and its end time as $e_j$. Let $\{T_m\}_{m=1}^{M}$ be the failure timestamps and $\delta$ the lead time. We label $\mathbf{X}_j$ as positive if a failure occurs within $\delta$ after $e_j$, i.e.,

$$y_j = \mathbb{I}\left( \min_{m:\, T_m \geq e_j} (T_m - e_j) \leq \delta \right) \tag{1}$$

Then we learn a predictor $\hat{p}_j = f(\mathbf{X}_j) \in [0, 1]$ to estimate the failure risk for each window. Next, we describe the semantic-level and temporal-level augmentation modules.

### B. Semantic-level Augmentation

Each log message is encoded into contextualized token embeddings using a pretrained DeBERTa model, fine-tuned on the log corpus. Based on these embeddings, word weights are assigned to quantify their relative importance, thereby enhancing feature representation. This approach is inspired by the observation that word distributions differ significantly between normal and failure-related logs, with failure-indicative tokens occurring more frequently in failure-related logs. To leverage this information, words are categorized as failure-related or unrelated based on their frequency ratios across the two corpora. Prior knowledge is incorporated into representation learning by augmenting a generative model with an amplification term, and word weights are derived via score-based estimation.

To capture the relationship between log context and words, words are represented as nodes and word co-occurrence probabilities as edges, forming a random walk graph. Random walks on this graph simulate the log corpus generation process based on edge weights, modeling the structure of the logs. Building on this, the random walk model is modified to emphasize the importance of words in different contexts. The log corpus is divided into failure-related and failure-unrelated datasets, and prior knowledge is incorporated into the model.

For integrating this prior knowledge, an additive term, $\alpha \log \frac{p_N(w)}{p_F(w)}$, is introduced, where $p_N(w)$ and $p_F(w)$ represent the word frequencies in the failure-unrelated and failure-related datasets, respectively, and $\alpha$ is a scalar. By adding this term, the importance of failure-related words is amplified.

Building on the previous discussion, we define the score of a word $w$ appearing in the log $s$ with the discourse vector $c_s$ as follows:

$$\begin{aligned} \mathrm{Score}(w \mid c_s) &= (1-\alpha)\, P_{\mathrm{context}}(w \mid c_s) + \alpha\, S_{\mathrm{frequency}}(w) \\ \tilde{c}_s &= \beta c_0 + (1-\beta) c_s, \qquad c_0 \perp c_s \end{aligned} \tag{2}$$

The term $P_{\mathrm{context}}(w \mid c_s) = \exp(\langle \tilde{c}_s, v_w \rangle)/Z_{\tilde{c}_s}$ represents the probability of generating the word $w$ in context $c_s$, where $Z_{\tilde{c}_s} = \sum_{w \in \mathcal{V}} \exp(\langle \tilde{c}_s, v_w \rangle)$ is the normalizing constant. $S_{\mathrm{frequency}}(w) = \log \frac{p_N(w)}{p_F(w)}$ is a frequency-based prior score for adjusting word weights. $\beta$ is the mixing coefficient, and to prevent extreme values, we clip $\log \frac{p_N(w)}{p_F(w)}$ with a threshold $\epsilon$, ensuring it stays close to 0, as shown in Eq. (3):

$$\begin{aligned} \mathrm{Score}(w \mid c_s) = (1-\alpha)\, &\frac{\exp\big(\langle \tilde{c}_s, v_w \rangle\big)}{Z_{\tilde{c}_s}} \\ &+ \alpha\, \mathrm{clip}\left( \log \frac{p_N(w)}{p_F(w)},\, -\epsilon,\, \epsilon \right) \end{aligned} \tag{3}$$

In this adjusted random walk model, log embeddings are defined by the maximum-score estimation of the discourse vector $c_s$ that generates them. We borrow the key modeling

**Algorithm 1** Log Embedding

**Require:** Subword word embeddings $\{e_w : w \in \mathcal{V}\}$, segmented log sequence $S$, estimated probability rates $\left\{\frac{p_N(w)}{p_F(w)} : w \in \mathcal{V}\right\}$ of words, parameter $a$

**Ensure:** Log embeddings $\{y_s : s \in S\}$

1: **for** each log $s$ in $S$ **do**

2: $\quad y_s \leftarrow \frac{1}{|s|}\sum_{w\in s} \frac{a}{a+\log\left(\frac{p_N(w)}{p_F(w)}\right)} e_w$

3: **end for**

4: From a matrix $X$ whose columns are $\{y_s : s \in S\}$, and let $\mu$ be its first singular vector

5: **for** each log $s$ in $S$ **do**

6: $\quad y_s \leftarrow y_s - \mu\mu^\top y_s$

7: **end for**

assumption of that $Z_{\tilde{c}_s}$ is roughly the same, say $Z$ for all $\tilde{c}_s$. According to Eq. (2), the score of the log is:

$$\text{Score}[s \mid c_s] = \prod_{w\in s}\left[(1-\alpha)\frac{\exp(\langle \tilde{c}_s, v_w\rangle)}{Z} + \alpha \log \frac{p_N(w)}{p_F(w)}\right] \tag{4}$$

Let $f_w(\tilde{c}_s)$ represent the contribution score of word $w$ in log $s$:

$$f_w(\tilde{c}_s) = \log\left[(1-\alpha)\frac{\exp(\langle \tilde{c}_s, v_w\rangle)}{Z} + \alpha \log \frac{p_N(w)}{p_F(w)}\right] \tag{5}$$

The maximum-score estimator for $\tilde{c}_s$ (ignoring normalization) is approximately:

$$\arg\max \sum_{w\in s} f_w(\tilde{c}_s) \propto \sum_{w\in s} h(w)\cdot e_w \tag{6}$$

where

$$h(w) = \frac{a}{a + \log\frac{p_N(w)}{p_F(w)}}, \qquad a = \frac{1-\alpha}{\alpha} \tag{7}$$

From the above derivation, it is clear that the maximum-score estimate can be viewed as a weighted average of the word vectors in the log. The design of the weight function $h(w)$ carries a clear meaning. When a word $w$ is related to failure, $\log\frac{p_N(w)}{p_F(w)}$ is small, which causes $h(w)$ to increase. In contrast, for words unrelated to failure, $h(w)$ decreases. This mechanism enables the model to focus more on failure-related signals and suppresses the impact of irrelevant information, thereby improving the accuracy of log message prediction.

Based on the above derivation, we obtain the maximum-score estimate $\tilde{c}_s$. The next step is to extract the final log embedding $c_s$. This step first calculates the primary component of the log messages $\tilde{c}_s$ that reflects the first principal direction of the log message. Then, by reducing the influence of $\tilde{c}_s$ on this principal component, we obtain the final log embedding $c_s$. The complete process is as shown in Algorithm 1.

### C. Temporal-level Augmentation

Building on the weighted message-embedding sequence from the semantic-level module, temporal evidence is further modeled for early failure prediction. Since log sequences rarely contain explicit failure markers, the predictor must capture subtle evolving dynamics from weak temporal signals. To this end, an ABiLSTM-HMM predictor is designed to exploit the complementary strengths of HMMs and attention-based BiLSTM. Specifically, HMMs model local Markovian transitions and short-term statistical fluctuations, while BiLSTM captures long-range nonlinear temporal dependencies. Fig. 1 shows the overall architecture.

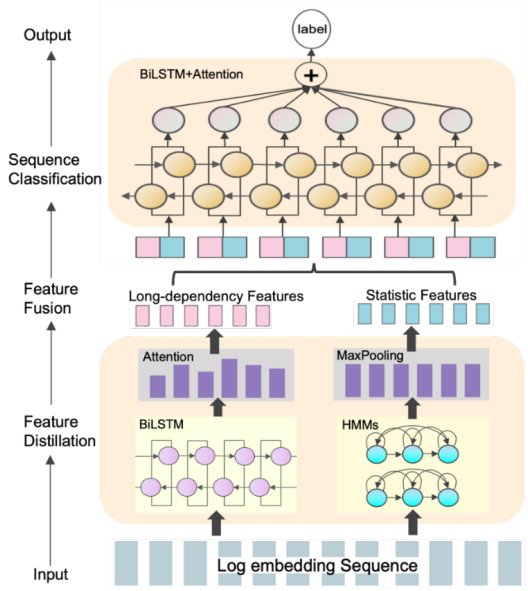


Fig. 1. The architecture of ABiLSTM-HMM.

*a) HMM-based statistical modeling:* For temporal statistical modeling, $F$ failure-specific HMMs are constructed, each corresponding to one failure type, as shown in Fig. 2. Given a log sequence $Y = [y_1, y_2, \ldots, y_T]$, the posterior probability of state $i$ in the $f$-th HMM at time step $t$ is computed as

$$p_t^{fi} = \frac{\alpha_t(s_{fi})\,\beta_t(s_{fi})}{\sum_{j=1}^{M}\alpha_t(s_{fj})\,\beta_t(s_{fj})} \tag{8}$$

The posterior features are then distilled through a feed-forward layer and max pooling:

$$\mathbf{HMMsF}_t = \text{Max}(\text{FFN}(p_t)) \tag{9}$$

*b) BiLSTM-based contextual modeling and fusion:* In parallel, an attention-based BiLSTM is used to model long-range contextual dependencies from the log sequence. The HMM-based statistical representation and the BiLSTM-based contextual representation are fused by concatenation:

$$\mathbf{HAB}_t = [\mathbf{HMMsF}_t; \mathbf{ABiL}_t] \tag{10}$$

The fused representation is further encoded into a sequence-level representation through BiLSTM and attention pooling, and then fed into a softmax classifier for prediction:

$$\hat{p}(l \mid Y) = \text{softmax}\left(W^{(Y)}\mathbf{h}^* + b^{(Y)}\right) \tag{11}$$

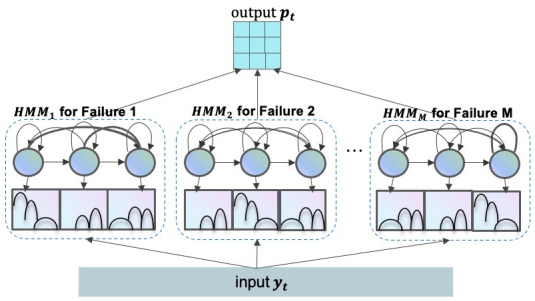


Fig. 2. The architecture of HMMs.

Model training minimizes the cross-entropy loss with $L_2$ regularization:

$$J(\theta) = -\frac{1}{E}\sum_{i=1}^{E} q_i \log(l_i) + \lambda\|\theta\|_F^2 \tag{12}$$

Here, $q \in \mathbb{R}^E$ denotes the ground-truth label distribution, $l \in \mathbb{R}^E$ denotes the predicted probability distribution, and $\lambda$ is the regularization coefficient.

### D. Training and Inference

Algorithm 2 summarizes a two-stage training pipeline that first trains the HMMs and then trains the ABiLSTM on the original sequences augmented with HMM-derived features.

Stage I: HMMs training. Given the training set $\mathcal{D}$, sequences are partitioned into $\{\mathcal{D}_f\}_{f=1}^F$ by failure type. For each type $f$, an HMM $\lambda_f^{(0)}$ is initialized and trained with the EM algorithm until the convergence criterion $\epsilon$ is met. The resulting HMMs $\{\lambda_f\}_{f=1}^F$ are then fixed and used to provide auxiliary statistical cues for downstream training.

Stage II: ABiLSTM-HMMs training. With the HMM parameters frozen, the network parameters $\theta$ are optimized for $N_{\text{ep}}$ epochs. For each mini-batch $\mathcal{B}$, HMM posterior features $P$ are first computed using the trained HMMs (Eq. (8)). These posterior features are treated as additional inputs and combined with the original sequences inside the loss computation, i.e., $L = \text{Loss}(\mathcal{B}, P; \theta)$ (Eq. (12)). The parameters $\theta$ are then updated by gradient-based optimization with learning rate $\eta$, and epoch-wise learning-rate decay $\eta \leftarrow \zeta\eta$ is applied to stabilize training.

**Algorithm 2** Two-stage Training for ABiLSTM–HMMs

**Input:** training sequences $\mathcal{D}$; type partitions $\{\mathcal{D}_f\}_{f=1}^F$; initial HMMs $\{\lambda_f^{(0)}\}_{f=1}^F$; EM tolerance $\epsilon$; initial learning rate $\eta$; decay factor $\zeta$; epochs $N_{\text{ep}}$.
**Output:** trained HMMs $\{\lambda_f\}_{f=1}^F$ and network parameters $\theta$.

**Stage I: HMMs training**
**for** $f = 1$ **to** $F$ **do**
  $\lambda_f \leftarrow$ EM algorithm$(\mathcal{D}_f, \lambda_f^{(0)}, \epsilon)$
**end for**

**Stage II: ABiLSTM–HMMs training**
Initialize $\theta$
**for** epoch $= 1$ **to** $N_{\text{ep}}$ **do**
  **for each** mini-batch $\mathcal{B} \subset \mathcal{D}$ **do**
    $P \leftarrow$ posterior features$(\mathcal{B}, \{\lambda_f\}_{f=1}^F)$ (Eq. (8))
    $L \leftarrow$ loss$(\mathcal{B}, P; \theta)$ (Eq. (12))
    Update $\theta$ using learning rate $\eta$
  **end for**
  $\eta \leftarrow \zeta\eta$
**end for**

## IV. Experiments

### A. Experiment Setup

*1) Datasets:* Four public log datasets are used for evaluation (Table I). Spirit and Thunderbird are supercomputer syslogs from Sandia National Laboratories [19], [20], and the first 50 million messages are used for each. LID-DS is a public Linux dataset with controlled injections and fine-grained ground truth [21], whereas AIT is an enterprise testbed dataset released by the Austrian Institute of Technology via Zenodo [22]. Spirit and Thunderbird provide anomaly labels without failure types, while LID-DS and AIT include failure times and types. Across all datasets, samples are constructed using sliding windows and split into training, validation, and test sets in a 7:1:2 ratio.

TABLE I
Datasets used in experiments

| Dataset | Total | Days | Failures | Used Messages |
|---|---|---|---|---|
| Spirit | 30.28GB | 558 | 5266 | 50,000,000 |
| Thunderbird | 29.60GB | 244 | 3730 | 50,000,000 |
| LID-DS | 14.50GB | 3 | 1022 | 3,538,923 |
| AIT | 96.00GB | 6 | 4019 | 13,606,477 |

*2) Baselines:* ABiLSTM-HMMs is compared with four representative failure prediction methods, including Time Machine [8], Aarohi [9], Bi-LSTM [6], and Prefix [10]. Three representative log anomaly detection methods, namely LogSD [14], SpikeLog [15], and LogBERT [13], are also included to highlight the gap between anomaly detection and early failure prediction.

*3) Evaluation Metrics:* Precision ($P$), Recall ($R$), and F1 score are reported:

$$P = \frac{\text{TP}}{\text{TP}+\text{FP}}, \quad R = \frac{\text{TP}}{\text{TP}+\text{FN}}, \quad \text{F1} = \frac{2PR}{P+R} \tag{13}$$

Here, TP, FP, and FN denote true positives, false positives, and false negatives, respectively.

*4) Implementation Details:* ABiLSTM-HMMs is implemented in PyTorch 1.7.0 with hmmlearn 0.2.6. Each log message is encoded into an 80-d semantic embedding. Unless otherwise specified, ABiLSTM-HMMs uses a BiLSTM hidden size of 128, 8 hidden states per HMM, and a word-weight hyperparameter of $\alpha$=3. Training is performed with a batch size of 64 and early stopping. AdamW is adopted with an initial learning rate of 0.01 and a decay factor of 0.5 per epoch. All results are averaged over three runs with different random seeds. Experiments are conducted on an Intel Core i9-14900HX CPU and an NVIDIA RTX 4080 GPU.

TABLE II
FAILURE PREDICTION RESULTS.

| Method | Spirit | | | Thunderbird | | |
|---|---|---|---|---|---|---|
| | Pre | Rec | F1 | Pre | Rec | F1 |
| **Failure Prediction Methods** | | | | | | |
| TimeMachine | **0.864** | 0.497 | 0.631 | **0.869** | 0.551 | 0.674 |
| Aarohi | 0.734 | 0.443 | 0.552 | 0.833 | 0.601 | 0.698 |
| Bi-LSTM | 0.715 | 0.403 | 0.515 | 0.715 | 0.500 | 0.588 |
| Prefix | 0.728 | 0.490 | 0.586 | 0.861 | 0.621 | 0.721 |
| **Anomaly Detection Methods** | | | | | | |
| LogSD | 0.717 | 0.221 | 0.338 | 0.849 | 0.287 | 0.429 |
| SpikeLog | 0.520 | 0.247 | 0.335 | 0.763 | 0.319 | 0.450 |
| LogBERT | 0.549 | 0.160 | 0.247 | 0.670 | 0.177 | 0.280 |
| **Ours** | 0.795 | **0.540** | **0.643** | 0.868 | **0.660** | **0.749** |

| Method | LID-DS | | | AIT | | |
|---|---|---|---|---|---|---|
| | Pre | Rec | F1 | Pre | Rec | F1 |
| **Failure Prediction Methods** | | | | | | |
| TimeMachine | 0.929 | **0.961** | **0.945** | 0.910 | 0.933 | 0.921 |
| Aarohi | 0.931 | 0.799 | 0.860 | 0.861 | 0.760 | 0.807 |
| Bi-LSTM | 0.808 | 0.598 | 0.687 | 0.761 | 0.641 | 0.696 |
| Prefix | 0.791 | 0.740 | 0.765 | 0.780 | 0.700 | 0.738 |
| **Anomaly Detection Methods** | | | | | | |
| LogSD | 0.655 | 0.353 | 0.459 | 0.628 | 0.470 | 0.538 |
| SpikeLog | 0.451 | 0.314 | 0.370 | 0.549 | 0.522 | 0.535 |
| LogBERT | 0.479 | 0.226 | 0.307 | 0.545 | 0.373 | 0.443 |
| **Ours** | **0.939** | 0.902 | 0.920 | 0.910 | **0.960** | **0.935** |

### B. Overall Comparison

Table II reports the comparison with four failure prediction baselines and three anomaly detectors. ABiLSTM-HMMs achieves the best F1 on Spirit (0.643), Thunderbird (0.749), and AIT (0.935), and ranks second on LID-DS (0.920), only 2.5% below the best method while exceeding the third-best by 6%. In contrast, anomaly detectors consistently underperform: their best F1 is at least 11.7% lower than the lowest failure-prediction score, suggesting that anomaly detection is not a drop-in replacement for early failure prediction.

Among failure predictors, deep recurrent/generative models generally outperform classic statistical approaches. Prefix remains competitive due to richer feature construction (sequence/frequency/seasonal/burst cues), which helps under low SNR and blurred class boundaries. Time Machine performs strongly on the small-scale LID-DS (F1=0.945) but degrades on Spirit and Thunderbird, likely due to noisy repetitions and the difficulty of capturing shallow, high-parallelism call patterns. Aarohi benefits from rule extraction based on annotated failure chains but depends on domain knowledge, while Bi-LSTM is limited in modeling complex nonlinear interactions. Overall, recall is consistently lower than precision—especially on large-scale systems—indicating a higher miss rate. ABiLSTM-HMMs boosts recall while preserving strong precision across both small and large datasets, supporting the effectiveness of the proposed dual-level feature augmentation.

### C. Robustness to Noisy Logs

Fig. 3 shows the F1 score as the redundancy removal ratio varies. As the retained-duplicate ratio increases, all methods experience performance degradation, whereas ABiLSTM-HMMs remains consistently more stable. Specifically, when the ratio increases from 30% to 70%, most baselines drop by more than 30% in F1, while our method decreases by less than 10%. We also observe a mild non-monotonic trend at low noise levels: F1 increases from 0.851 to 0.910 when the ratio rises from 10% to 30%. A plausible explanation is that early failures may trigger routine retries (e.g., due to transient latency), which introduce informative repetitions and thus act as weak but useful pre-failure cues. Overall, appropriate duplicate filtering improves reliability, and ABiLSTM-HMMs demonstrates strong robustness across a wide range of noise settings.

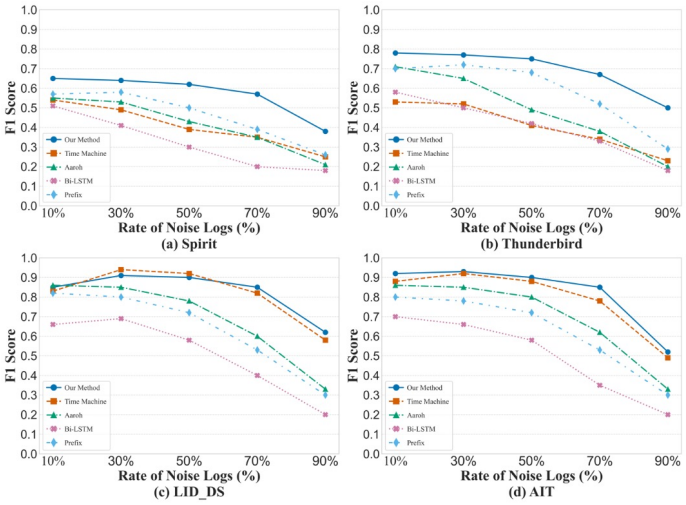


Fig. 3. F1 scores under different noise levels.

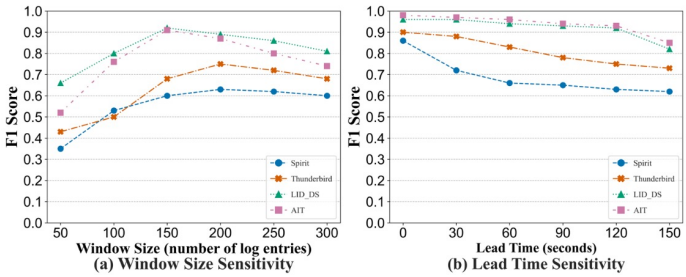


Fig. 4. Sensitivity analysis of (a) window size and (b) lead time.

### D. Window and Lead Time Sensitivity

We study two key factors for practical failure prediction: the window size (sequence length) and the lead time (reaction time after an alarm). A larger window increases overhead, while too short a window may truncate failure-related evidence; longer lead times typically make omens sparser. Fig. 4(*a*) varies the window size with lead time fixed to 120 s. Performance peaks at a window size of 200, achieving an average F1 of 0.785 across datasets. Reducing the window harms performance (average F1 drops to 0.49 at window size 50), since critical pre-failure contexts are cut off. Conversely, overly large windows ($> 200$) introduce excessive normal logs, diluting weak failure signals and increasing confusing patterns. Fig. 4(*b*) shows that longer lead times consistently reduce F1, as pre-failure omens

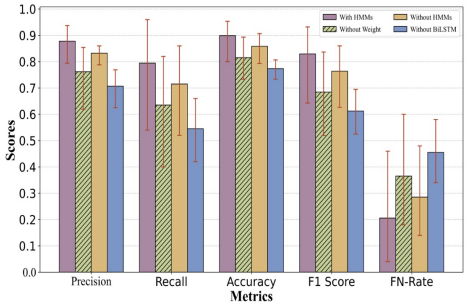


Fig. 5. Ablation results of ABiLSTM-HMMs.

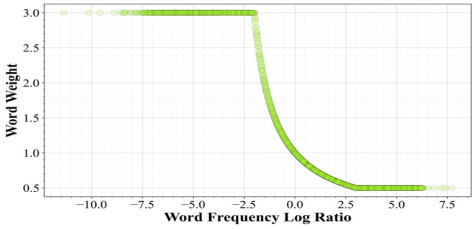


Fig. 6. Random-walk-based word weight distribution.

become less frequent farther from the failure point. This effect is stronger for failures whose omens concentrate shortly before breakdown (e.g., memory-leak-driven crashes), while failures with long-horizon degradation (e.g., disk aging or network degradation) are less sensitive.

### *E. Ablation Study*

Fig. 5 summarizes ablations of ABiLSTM-HMMs across four datasets. Removing the word-weighting module causes the largest drop: on average, precision decreases by 11.6% and F1 by 12.5%. This module emphasizes failure-indicative tokens (e.g., "disconnect", "not", "again"), strengthening semantic cues in weak-signal sequences and reducing missed alarms (FN) by 13.4% on average. The HMM module is also critical, improving average precision by 4.6% and F1 by 6.6% by capturing short-term statistics (first-order Markov patterns) complementary to BiLSTM's long-range modeling. Pre-training HMMs per failure type further helps detect subtle structural deviations, especially under repetitive routine logs, reducing FN by 7.9%. Overall, BiLSTM provides essential temporal dependencies, while word weighting and HMMs jointly enhance robustness to noisy and low-signal pre-failure patterns.

### *F. Case Study*

We conduct a case study on LID-DS to illustrate the effect of our dual feature augmentation under low signal-to-noise ratios. LID-DS contains multiple injected failure/attack categories whose logs often interleave with normal operation, making early failure cues subtle and hard to separate. Table III compares alternative feature augmentation choices. Replacing our random-walk-based word weighting with TF-IDF substantially degrades performance (F1 drops from 92.0% to 66.6%), since

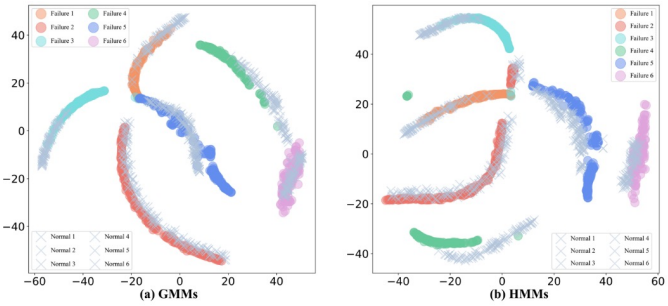


Fig. 7. Visualization of log sequence.

TABLE III
FAILURE PREDICTION RESULTS FOR DIFFERENT FEATURE AUGMENTATION METHODS

| Method | Precision | Recall | F1 Score |
|---|---|---|---|
| ABiLSTM-HMMs | 93.9 | 90.2 | 92.0 |
| with TF-IDF | 72.7(↓21.2) | 61.5(↓28.7) | 66.6(↓25.4) |
| with GMM | 79.2(↓14.7) | 73.0(↓17.2) | 75.9(↓16.1) |

rare tokens in balanced corpora are not necessarily failure-indicative, whereas our weighting emphasizes failure-specific terms (Fig. 6).

We also replace HMM-based statistical augmentation with a GMM-based variant. GMM provides weaker gains (F1=75.9%), likely due to its independence assumption over log messages. In contrast, HMM models first-order Markov transitions, capturing sequential dynamics more faithfully. As visualized by t-SNE (Fig. 7), HMM-extracted features form clearer boundaries between failure and normal samples than GMM, highlighting the benefit of modeling state transitions for failure prediction.

## V. CONCLUSION

This paper presented a dual-level feature augmentation approach for log-based system failure prediction under low signal-to-noise ratios and weak pre-failure signals. The proposed method enhances representations at both the semantic and temporal levels. On the semantic side, a random-walk-based word-weighting scheme automatically amplifies failure-related tokens without relying on domain-specific semantic knowledge, improving the separability of log representations. On the temporal side, a fusion-based encoder couples BiLSTM with HMMs to jointly model long-range dependencies and short-horizon transition dynamics, enabling a more comprehensive characterization of pre-failure patterns. Extensive experiments on four public datasets demonstrate consistent gains over competitive baselines. Further analyses on noise robustness, window/lead-time sensitivity, and ablations verify that both word weighting and HMM-based augmentation are critical for reliable early warning in noisy logs. In future work, we will explore additional augmentation techniques tailored to log data and develop a flexible incremental update mechanism to better adapt to emerging failure modes.

## References


[1] M. De la Cruz Cabello, T. P. Sales, and M. R. Machado, "Aiops for log anomaly detection in the era of llms: A systematic literature review," *Intelligent Systems with Applications*, vol. 28, p. 200608, 2025.

[2] F. Hadadi, J. H. Dawes, D. Shin, D. Bianculli, and L. Briand, "Systematic evaluation of deep learning models for log-based failure prediction," *Empirical Software Engineering*, vol. 29, no. 105, pp. 1–12, 2024. [Online]. Available: https://doi.org/10.1007/s10664-024-10501-4

[3] A. J. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Communications of the ACM*, vol. 55, no. 2, pp. 55–61, 2012.

[4] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP)*, 2009, pp. 117–132.

[5] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining invariants from console logs for system problem detection," in *Proceedings of the 2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.

[6] J. Gao, H. Wang, and H. Shen, "Task failure prediction in cloud data centers using deep learning," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1411–1422, may 2022.

[7] K. A. Alharthi, A. Jhumka, S. Di, and F. Cappello, "Clairvoyant: A log-based transformer-decoder for failure prediction in large-scale systems," in *Proceedings of the 36th ACM International Conference on Supercomputing (ICS)*, 2022, pp. 35:1–35:14.

[8] K. A. Alharthi, A. Jhumka, S. Di, L. Gui, F. Cappello, and S. McIntosh-Smith, "Time machine: Generative real-time model for failure (and lead time) prediction in hpc systems," in *Proceedings of the 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2023.

[9] A. Das, F. Mueller, and B. Rountree, "Aarohi: Making real-time node failure prediction feasible," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020.

[10] S. Zhang, Y. Liu, W. Meng, Z. Luo, J. Bu, S. Yang, P. Liang, D. Pei, and J. Xu, "Prefix: Switch failure prediction in datacenter networks," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, pp. 1–29, 2018.

[11] W. Li, J. Zhang, H. Zhang, H. Zhang, D. Hao, and L. Zhang, "DeepTraLog: Trace-log combined microservice anomaly detection through graph-based deep learning," in *Proceedings of the 44th International Conference on Software Engineering (ICSE)*. ACM, 2022, pp. 431–442. [Online]. Available: https://doi.org/10.1145/3510003.3510089

[12] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 1285–1298.

[13] Z. Yuan, P. Liu, S. Zhao, and Z. Liu, "Logbert: Log anomaly detection via BERT," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[14] Y. Xie, H. Zhang, and M. A. Babar, "Logsd: Detecting anomalies from system logs through self-supervised learning and frequency-based masking," *Proceedings of the ACM on Software Engineering*, vol. 1, pp. 2098–2120, 2024, (FSE).

[15] J. Qi, Z. Luan, S. Huang, C. Fung, H. Yang, and D. Qian, "Spikelog: Log-based anomaly detection via potential-assisted spiking neuron network," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 12, pp. 9322–9335, 2024.

[16] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, jul 2019, pp. 4739–4745.

[17] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, J. Chen, X. He, R. Yao, J.-G. Lou, M. Chintalapati, F. Shen, and D. Zhang, "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19)*, 2019.

[18] X. Wu, H. Li, and F. Khomh, "On the effectiveness of log representation for log-based anomaly detection," *Empirical Software Engineering*, vol. 28, no. 6, p. 137, 2023. [Online]. Available: https://doi.org/10.1007/s10664-023-10364-1

[19] A. J. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2007, pp. 575–584.

[20] J. Stearley and A. J. Oliner, "Bad words: Finding faults in spirit's syslogs," in *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, 2008.

[21] M. Grimmer, T. Kaelble, F. Nirsberger, E. Schulze, T. Rucks, J. Hoffmann, and E. Rahm, "Dataset report: LID-DS 2021," in *Critical Information Infrastructures Security (CRITIS 2022), Revised Selected Papers*, ser. Lecture Notes in Computer Science, 2022.

[22] A. I. of Technology and collaborators, "AIT Log Data Set V2.0," Zenodo, 2021, record: 5789063.