Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.085032

ARTICLE

SemBERT: Semantic BERT Embeddings and HDBSCAN Clustering for Unsupervised Log Parsing and Template Mining in Large-Scale Distributed Systems[#]

Gobinda Bhattacharjee[1], Joy Dey[1], Tanjim Mahmud[1,*], Mohammad Shahadat Hossain[2,3] and Karl Andersson[3]

[1]Department of Computer Science and Engineering, Rangamati Science and Technology University, Rangamati, Bangladesh
[2]Department of Computer Science and Engineering, University of Chittagong, Chittagong, Bangladesh
[3]Cybersecurity Laboratory, Luleå University of Technology, Skellefteå, Sweden

*Corresponding Author: Tanjim Mahmud. Email: tanjim_cse@yahoo.com or tanjim.cse@rmstu.ac.bd

[#]A preliminary version of this work was presented at the 9th International Conference on Mobile Internet Security (MobiSec 2025, Japan), organized by the KIISC Research Group on 6G Security at the Electronics and Telecommunications Research Institute (ETRI) and hosted by the Korea Institute of Information Security and Cryptology. This manuscript substantially extends the conference version by introducing statistical robustness analysis using three independent random seeds, scalability experiments on HDFS datasets ranging from 50K to 1M log entries, adaptive semantic-gap thresholding for automatic cluster merging, downstream validation using representative machine learning, deep learning, and transformer-based anomaly detection models, cross-dataset evaluation on the BGL benchmark, and expanded discussions on reproducibility, computational limitations, and future research directions. These additions provide significant methodological refinements and substantially broaden the experimental validation beyond the original conference publication

Received: 04 May 2026; Accepted: 07 August 2026; Published: 15 September 2026

ABSTRACT: Log parsing is a fundamental prerequisite for automated system monitoring, anomaly detection, and root cause analysis in large-scale distributed environments. However, existing parsing approaches often rely on heuristic rules, manually engineered features, or fixed similarity thresholds, limiting their adaptability to heterogeneous and evolving log structures. To address these challenges, this study presents SemBERT, a fully unsupervised log parsing framework that integrates semantic BERT embeddings, Incremental Principal Component Analysis (IPCA), HDBSCAN clustering, and adaptive centroid-based cluster merging for robust template mining. Unlike conventional methods that employ fixed merging criteria, SemBERT adaptively determines semantic merging thresholds according to the distribution of cluster centroids, reducing parameter sensitivity across datasets. Experiments on the LogHub HDFS benchmark demonstrated that SemBERT achieved perfect Grouping Accuracy (GA = 100.00%), Parsing Accuracy (PA = 95.82 ± 0.44%), and Template Accuracy (TA = 76.59 ± 4.49%), outperforming several classical parsers in semantic template discovery while maintaining competitive parsing performance. Repeated reproducibility experiments using three different random seed initializations (42, 123, and 999) produced identical adaptive thresholds and final template counts across all evaluated data scales, confirming the deterministic behavior and reproducibility of the proposed framework. Additional scalability experiments on randomly sampled subsets ranging from 50K to 1M log entries demonstrated stable threshold estimation and controlled template growth, highlighting the effectiveness of the adaptive semantic merging strategy. To validate the practical utility of the extracted templates, downstream anomaly detection experiments were conducted using representative machine learning, deep learning, and transformer-based models. The semantic templates generated by SemBERT provide informative representations suitable for downstream log analytics. Furthermore, experiments on the BGL dataset indicate that the framework generalizes effectively to heterogeneous logging environments through lightweight dataset-specific preprocessing. These findings suggest that SemBERT provides a scalable and semantically informed solution for unsupervised log parsing in modern distributed systems.

KEYWORDS: Log parsing; BERT semantic embeddings; unsupervised learning; HDBSCAN clustering; template mining; anomaly detection; distributed systems

1 Introduction

Modern large-scale distributed systems, such as the Hadoop Distributed File System (HDFS), continuously generate terabytes of unstructured log data that constitute the primary source for system monitoring, fault diagnosis, performance optimization, and anomaly detection [1–3]. Accurate and efficient analysis of these logs is critical for ensuring the reliability and security of cloud and enterprise environments. Nevertheless, the raw, semi-structured, and highly variable nature of system logs—marked by dynamic parameters (e.g., block IDs, IP addresses, and timestamps), evolving message formats, and massive volumes—poses significant challenges for automated processing [1,3,4].

Traditional log-parsing techniques predominantly rely on rule-based, heuristic, or clustering-based approaches, including fixed-depth trees (Drain) [5], unsupervised cluster-evolution analysis [6], self-supervised semantic template extraction [7], and template-correction-based parsing [8]. Although effective for small or homogeneous datasets, these methods frequently exhibit limited scalability, high sensitivity to parameter tuning, and inadequate semantic context capture, resulting in inaccurate templates and degraded performance in downstream anomaly detection tasks [1,3]. Recent research has advanced toward machine learning and deep learning solutions, such as n-gram dictionaries (Logram) [9], sentence embeddings with metadata (DeepSyslog) [2], self-supervised log parsing for heterogeneous log formats [10], and pre-trained language models for anomaly detection. More recently, large language model (LLM)-driven parsers (HELP [11], SemanticLog [12], LibreLog [13]) have demonstrated promising accuracy through few-shot and in-context learning methods. However, most existing approaches remain computationally intensive for million-scale logs, are heavily reliant on labeled data or prompt engineering, or fail to generate ready-to-use event sequences suitable for anomaly detection pipelines [1,3,12].

To address these limitations, this study proposes SemBERT, a fully unsupervised, semantically enriched, and scalable log parsing framework designed for large-scale distributed system logs. The proposed pipeline begins with lightweight semantic preprocessing and entity masking to normalize raw log messages while preserving their context. The normalized logs were then encoded using BERT-base-uncased with mean pooling to generate contextual semantic representations, leveraging GPU acceleration for efficient large-scale processing. To improve scalability and reduce computational overhead, Incremental Principal Component Analysis (IPCA) was employed for dimensionality reduction before density-based clustering was performed using HDBSCAN. Unlike conventional approaches that rely on manually selected similarity thresholds, SemBERT adopts an adaptive semantic threshold estimation strategy for centroid-based cluster merging, enabling robust template generation across varying log distribution. Finally, the extracted event templates are used to construct block-level event sequences that are aligned with the anomaly labels, producing structured datasets suitable for downstream anomaly detection tasks. This design facilitates accurate template extraction while mitigating template fragmentation, parameter sensitivity, and the limited semantic understanding commonly observed in heuristic parsers [5,14–16].

The main contributions of this study are summarized as follows:

1. We propose SemBERT, a fully unsupervised semantic log parsing framework that integrates entity-aware semantic masking, contextual BERT embeddings, Incremental PCA, HDBSCAN clustering, and adaptive centroid-based merging to generate compact and semantically coherent log templates from large-scale unstructured logs.

2. We introduce an adaptive semantic threshold estimation mechanism for cluster merging, reducing the dependence on manually selected similarity thresholds and improving robustness across different data distributions and sampling settings.
3. We conduct extensive experiments on large-scale HDFS datasets containing up to 11.2 million log entries and demonstrate strong scalability characteristics through multi-scale analyses from 50K to 1M logs, accompanied by statistical robustness evaluations and confidence interval analysis.
4. We evaluate the proposed framework using standard log parsing metrics, including Grouping Accuracy (GA), Parsing Accuracy (PA), and Template Accuracy (TA), providing both bootstrap-based confidence intervals and multiple-run statistical analyses to validate the reliability and reproducibility of the obtained results.
5. We further investigate the effectiveness of the generated templates for downstream anomaly detection using machine learning, deep learning, and transformer-based models, demonstrating that semantically meaningful parsing substantially benefits subsequent log intelligence tasks.

The remainder of this paper is organized as follows. Section 2 presents a detailed literature review of the relevant studies. Section 3 describes the proposed method. Section 4 reports the experimental results of this study. Section 5 discusses the findings of the study. Section 6 concludes the paper with future research directions, followed by the references.

2 Literature Review

Log parsing is a fundamental preprocessing step in system log analytics that transforms unstructured raw log messages into structured templates or event types that enable downstream tasks such as anomaly detection, fault diagnosis, and performance monitoring [3,4,16]. Over the past two decades, a wide spectrum of parsing techniques has been proposed, which can be broadly categorized into heuristic/rule-based, clustering-based, and deep learning/embedding-based approaches. Representative methods from these categories and their main strengths and limitations are summarized in Table 1.

Table 1: Comparison of representative log parsing approaches and their characteristics.

Method	Category	Strengths	Limitations
Drain [5]	Heuristic	High efficiency and online processing capability	Sensitive to tree depth and predefined rules
Spell [14]	LCS-based	Supports streaming log parsing	Threshold sensitivity and template fragmentation
Dynamic cluster evolution [6]	Unsupervised clustering	Incremental cluster analysis and anomaly detection under evolving logs	Dependent on time-window and clustering parameters
LogStamp [15]	Sequence labelling	Online parsing using pretrained language representations	Requires model training and labeled or partially labeled data
LogSL [10]	Self-supervised parsing	Robust template extraction across heterogeneous log formats	Higher model complexity and computational requirements

(Continued)

Table 1 (continued)

Method	Category	Strengths	Limitations
HELP [11]	Embedding based	Hierarchical semantic representations	Higher computational requirements
SemanticLog [12]	Semantic parsing	Large-scale semantic feature extraction and parameter interpretation	Depends on language-model-based representations and substantial computational resources
InferLog [19]	LLM-based	Efficient online inference through prefix caching	Dependent on prompt engineering and LLM resources

Early research has focused on heuristic and rule-based methods. Landauer et al. [6] proposed an unsupervised cluster-evolution approach for dynamic log-file analysis. Their method incrementally groups log lines within time windows and analyzes cluster splits, merges, and other transitions to identify anomalous temporal behavior. Yu et al. [7] introduced a self-supervised log parsing method that models the semantic contribution of words and uses these semantic differences to distinguish template words from variable words during log-template extraction. He et al. [5] developed Drain, an online log-parsing approach using a fixed-depth parse tree, which has become one of the most widely adopted methods owing to its efficiency and accuracy on structured logs. Other notable parsing approaches include template-correction-based parsing (Cognition) [8], online streaming parsing (Spell) [14], and n-gram dictionary-based parsing (Logram) [9], the latter of which uses frequent n-gram patterns to support efficient log-template extraction. Although these methods perform well on small or moderately sized datasets, they often suffer from poor scalability, high sensitivity to parameter tuning, and limited semantic understanding when applied to large-scale, heterogeneous logs [3,17].

To address the limitations of purely syntactic approaches, researchers have incorporated machine learning and embedding techniques. DeepSyslog [2] combines sentence embeddings with metadata for syslog anomaly detection. Cao et al. [10] proposed a self-supervised log-parsing method that models template extraction as a multi-token prediction task and learns template-token distributions from unlabeled log messages. LogStamp [15] formulated online log parsing as a sequence-labelling problem using a pretrained language model to distinguish template tokens from dynamic variables. Recent advances have introduced large language model (LLM)-based log parsers that utilize prompt engineering, in-context learning, and adaptive caching mechanisms to improve generalization across heterogeneous formats. Prompt-based approaches leverage few-shot examples to infer event templates without explicit rule construction [18], whereas semantic approaches such as SemanticLog [12] extract informative representations and semantic information from log messages. More recent LLM-based methods such as InferLog [19] and self-correcting frameworks such as SCULP [20] further improve parsing efficiency and quality through inference optimization and template correction. LibreLog [13] demonstrates the potential of open-source large language models for unsupervised log parsing tasks.

Despite these advances, LLM-based methods often require substantial computational resources, prompt engineering, or carefully designed in-context examples, which may limit their applicability in large-scale production environments and resource-constrained systems. In contrast, SemBERT adopts a fully unsupervised

paradigm that combines lightweight semantic preprocessing, contextual BERT embeddings, density-based clustering, and adaptive centroid merging, eliminating the need for labeled data, prompt construction, or proprietary foundation models while maintaining high scalability and parsing effectiveness.

Although existing heuristic, embedding-based, and LLM-driven approaches have achieved considerable progress in automated log parsing, several challenges remain to be addressed. Rule-based methods frequently suffer from parameter sensitivity and template fragmentation, whereas semantic parsers often require manually selected similarity thresholds that may not generalize across datasets. Despite their strong representational capabilities, LLM-based solutions introduce additional computational costs and dependencies on prompt engineering or proprietary models. These limitations motivate the development of a lightweight, fully unsupervised, and semantically aware framework that combines contextual embeddings with adaptive clustering mechanisms to achieve a scalable and robust log template extraction. The proposed SemBERT framework addresses these challenges through semantic masking, BERT-based representations, Incremental PCA, HDBSCAN clustering, and adaptive centroid-based merging.

3 Methodology

3.1 Overview of the Proposed Framework

The proposed methodology introduces a fully unsupervised, semantically aware, and highly scalable end-to-end framework for log parsing and anomaly detection in large-scale distributed systems. The pipeline transforms raw, unstructured log messages into compact and semantically meaningful event templates and subsequently constructs block-level event sequences that can be directly used for downstream anomaly detection tasks. By leveraging contextual representations from a pre-trained BERT model, incremental dimensionality reduction, density-based clustering, and adaptive centroid-level refinement, the proposed framework preserves the semantic relationships among log events while maintaining computational efficiency on datasets of millions of scale.

As illustrated in Fig. 1, the overall workflow consists of nine sequential stages. The process begins with large-scale log sampling and data acquisition, followed by semantic preprocessing, which includes message extraction, entity masking, and textual normalization. The cleaned messages were then encoded using BERT-base-uncased, where mean pooling was applied to obtain fixed-dimensional contextual embeddings. To support scalable processing, Incremental Principal Component Analysis (IPCA) is employed to reduce the dimensionality of the embedding space while preserving the semantic structure.

The reduced representations were subsequently grouped using HDBSCAN, followed by an adaptive centroid-based semantic merging strategy that consolidated highly similar clusters according to the underlying similarity distribution of each dataset. The refined clusters were then transformed into concise log templates and assigned unique event identifiers. Using HDFS block identifiers as session keys, the extracted templates were organized into ordered event sequences and integrated with official anomaly labels to facilitate downstream machine learning, deep learning, and transformer-based anomaly detection experiments.

The framework adopts a modular design in which each stage can operate independently while maintaining end-to-end reproducibility and scalability of the model. GPU acceleration is used for semantic embedding generation, whereas memory-efficient techniques, such as Incremental PCA, enable the processing of large-scale log collections. Furthermore, reproducibility analyses across multiple random samples were conducted to verify the stability of the adaptive clustering mechanism and the consistency of the generated templates. The overall workflow of the proposed SemBERT framework is summarized in Algorithm 1. The following subsections describe each component of the framework, including the underlying algorithms, parameter configurations, and implementation details of the framework.

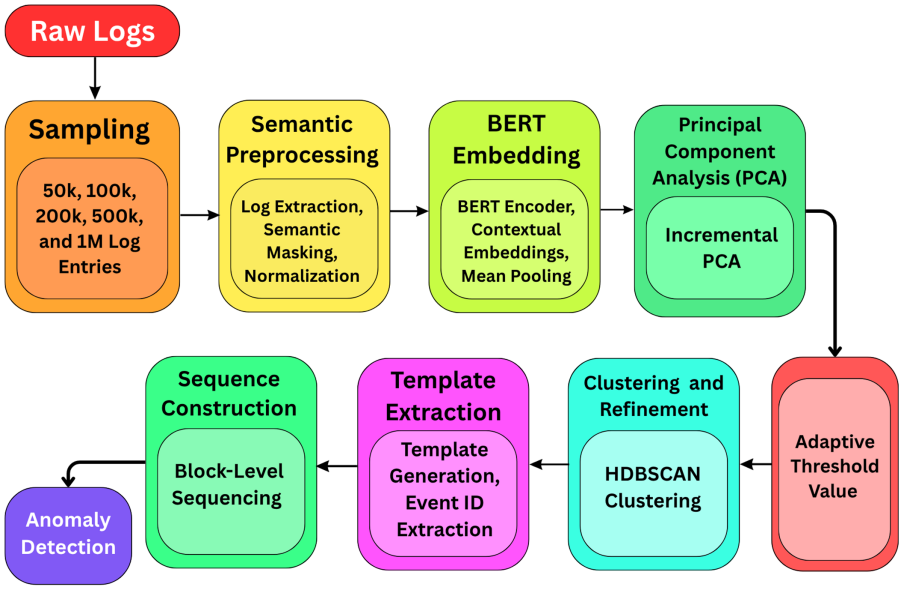

Figure 1: Overview of the proposed log parsing and anomaly detection framework.

Algorithm 1: SemBERT: adaptive semantic log parsing framework

Require: Raw log dataset L
Ensure: Final template set T and event sequences S

$L_s \leftarrow \textsc{SampleLogs}(L)$
$M \leftarrow \varnothing$
for all $l \in L_s$ **do**
    $m \leftarrow \textsc{ExtractMessage}(l)$
    $m \leftarrow \textsc{SemanticMask}(m)$
    $m \leftarrow \textsc{Normalize}(m)$
    Append m to M
end for
$E \leftarrow \text{BERTEncode}(M)$
$E_r \leftarrow \textsc{IncrementalPCA}(E)$
$C \leftarrow \text{HDBSCAN}(E_r)$
$P \leftarrow \textsc{ComputeCentroids}(C)$
$\tau \leftarrow \textsc{AdaptiveThreshold}(P)$
$C_m \leftarrow \textsc{MergeClusters}(C, \tau)$
$T \leftarrow \textsc{ExtractTemplates}(C_m)$
$S \leftarrow \textsc{ConstructSequences}(T)$
optionallabelalignment (S)
return T, S

3.2 Dataset Description

The experiments in this study utilized the well-known Hadoop Distributed File System (HDFS) log dataset obtained from the LogHub repository [3]. LogHub is a large-scale, publicly available collection of real-world system log datasets specifically designed for research on automated log parsing, anomaly detection, and AI-driven log analytics. To further evaluate the scalability and generalization capability of the

proposed framework, we employed the BGL (BlueGene/L) dataset from the same LogHub repository. The main characteristics of the datasets are summarized in Table 2.

Table 2: Dataset description.

Dataset	Original Log Lines	Sampled Log Lines
HDFS	11,175,629	50K–1M
BGL	4,631,261	50K–1M

3.3 Log Sampling Strategy

A deterministic streaming-based sampling approach was adopted to ensure scalability evaluation while maintaining computational feasibility in a standard GPU environment. The completely sorted HDFS log file was processed line by line without loading the entire dataset into memory. Five sample sizes were generated: 50,000, 100,000, 200,000, 500,000, and 1,000,000 log entries. Similarly, the BGL dataset was sampled using the identical streaming method for cross-dataset validation. This approach preserves the original temporal sequence of events. The official block-level anomaly labels from LogHub were aligned with the resulting sequences. This strategy enables a direct comparison of the parsing efficiency across different data scales while ensuring full reproducibility.

3.4 Semantic Preprocessing

Semantic preprocessing constitutes a crucial stage in the proposed framework, transforming raw and noisy log messages into semantically consistent representations that are suitable for contextual embedding and clustering. This process aims to reduce structural variability while preserving the underlying semantic information contained in the system events. The preprocessing pipeline consisted of three sequential operations: message extraction, semantic entity masking, and text normalization.

3.4.1 Log Message Extraction

The raw log entries in both the HDFS and BGL datasets contain metadata, including timestamps, severity levels, and component identifiers, followed by the actual event message. The `extract_message()` function separates the semantic content from the metadata by splitting each log line at the first occurrence of the colon (`:`) delimiter and retaining only the message.

3.4.2 Semantic Masking

To handle the highly variable nature of logs, a set of regular expressions is applied to replace the dynamic entities with fixed semantic tokens. The patterns mask IP addresses (<IP>), IP:Port combinations (<IP_PORT>), block identifiers (<BLOCK_ID>), numeric values (<NUM>), and file paths (<PATH>). This masking is performed by the semantic_mask() function, which produces standardized log messages while preserving their structural and semantic meanings. The regular-expression rules used for semantic masking are presented in Table 3.

3.4.3 Log Normalization

After semantic masking, a normalization step was applied to eliminate minor formatting inconsistencies and improve token uniformity. The normalization procedure removes redundant whitespace, collapses consecutive occurrences of identical semantic tokens (e.g., `<IP> <IP>` to `<IP>`), and standardizes textual

representations across log messages. These operations reduce unnecessary lexical variations while preserving the semantic structures of the original events.

The three-stage preprocessing pipeline produces compact and semantically enriched log representations that serve as high-quality inputs for the subsequent BERT embedding stage, thereby improving the clustering effectiveness and template generation quality.

Table 3: Regular-expression rules used for semantic masking.

Entity Type	Regular Expression	Replacement Token
IP Address with Port	/\d+\.\d+\.\d+\.\d+:\d+	<IP_PORT>
IP Address	/\d+\.\d+\.\d+\.\d+	<IP>
Block Identifier	blk_-?\d+	<BLOCK_ID>
Numeric Value	\b\d+\b	<NUM>
File Path	(/[\w\.-]+)+	<PATH>

3.5 Log Representation Using BERT

After semantic preprocessing, the normalized log messages were transformed into dense contextual vector representations using a pretrained BERT model. The 'bert-base-uncased' model was loaded and moved to the GPU (Tesla T4) for accelerated inference. Each cleaned log message was tokenized with a maximum sequence length of 64, padded within batches, and passed through the BERT encoder. Contextual embeddings were obtained from the last hidden state and aggregated using mean pooling weighted by the attention mask, producing a fixed 768-dimensional vector for every log entry.

The resulting embeddings capture contextual semantic relationships among log events beyond the lexical similarities exploited by traditional rule- and frequency-based approaches. Depending on the experimental scale, the generated embedding matrices ranged from 50,000 × 768 to 1,000,000 × 768 dimensions and served as inputs for the subsequent dimensionality reduction and clustering stages. By leveraging contextual representations learned from large-scale natural language corpora, the proposed framework achieves a more semantically coherent grouping of structurally diverse log messages.

3.6 Dimensionality Reduction

Although BERT embeddings provide rich contextual representations of log messages, their high dimensionality (768 dimensions) increases the computational cost of large-scale clustering, particularly when processing hundreds of thousands or millions log entries. To improve scalability while preserving semantic information, dimensionality reduction was performed using Incremental Principal Component Analysis (IncrementalPCA).

Unlike conventional PCA, IncrementalPCA processes data in mini-batches, making it well-suited for the memory-efficient analysis of large log datasets. In our implementation, the algorithm was configured with $n_{\text{components}} = 50$ and a batch size of 5,000 samples. Consequently, the original embedding matrices were transformed from dimensions ranging from 50,000 × 768 to 1,000,000 × 768 into corresponding low-dimensional representations with 50 principal components.

Given an embedding matrix $X \in \mathbb{R}^{N\times 768}$, IncrementalPCA projects the data onto a lower-dimensional subspace according to

$$Z = XW, \tag{1}$$

where $W \in \mathbb{R}^{768\times50}$ contains the principal component vectors corresponding to the largest eigenvalues of the covariance matrix, and $Z \in \mathbb{R}^{N\times50}$ is the reduced representation used for subsequent clustering.

The incremental processing strategy substantially reduces memory consumption and computational overhead while retaining the dominant semantic variations that are encoded by BERT. These low-dimensional embeddings serve as direct inputs to the HDBSCAN clustering stage, enabling the efficient semantic grouping of log events across both the HDFS and BGL datasets at scales of up to one million log entries.

3.7 Log Clustering

Following dimensionality reduction, the 50-dimensional semantic embeddings were grouped into event types using a two-stage clustering framework consisting of density-based clustering and adaptive semantic refinement. This design enables the robust identification of log templates while maintaining scalability across datasets ranging from 50K to 1M log entries.

3.7.1 HDBSCAN Clustering

The reduced embeddings were initially clustered using the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm. HDBSCAN was selected because of its ability to discover clusters of varying densities while explicitly identifying outliers and noise points, which are common characteristics of large-scale system log data.

The clustering process employs the following parameters:

- `min_cluster_size = 100`,
- `min_samples = 10`,
- `metric = Euclidean`,
- `cluster_selection_method = eom`.

Unlike traditional clustering methods, HDBSCAN does not require prior knowledge of the number of event types present. Instead, it automatically determines the cluster structure based on local density variations and assigns ambiguous samples to the noise class (−1). This property makes the method particularly suitable for heterogeneous log environments, such as HDFS and BGL, where event frequencies are highly imbalanced.

3.7.2 Adaptive Semantic Cluster Refinement

Although HDBSCAN effectively groups semantically similar log messages, closely related event types may still be separated into multiple clusters because of subtle lexical differences. To address this issue, an adaptive centroid-based merging strategy was employed.

For each valid cluster C_i, a centroid vector is computed as

$$\mathbf{c}_i = \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j, \tag{2}$$

where $\mathbf{x}_j$ denotes a reduced embedding belonging to cluster C_i.

The semantic similarity between two clusters is then measured using cosine similarity:

$$\mathrm{Sim}(C_i, C_j) = \frac{\mathbf{c}_i \cdot \mathbf{c}_j}{\|\mathbf{c}_i\| \|\mathbf{c}_j\|}. \tag{3}$$

Instead of employing a fixed similarity threshold, the proposed framework determines an adaptive threshold based on the distribution of centroid similarities within each dataset and sampling scale. This adaptive mechanism improves the robustness across heterogeneous log collections and prevents both over-merging and template fragmentation.

Clusters whose similarity exceeds the adaptive threshold are merged using a disjoint-set (union-find) structure to produce the final set of semantic templates. The experimental results demonstrate that this refinement stage substantially reduces redundant clusters while preserving meaningful event distinctions, thereby improving template compactness and consistency across large-scale datasets.

The combination of HDBSCAN and adaptive semantic merging provides an effective balance between clustering accuracy, robustness to noise, and computational scalability, enabling the proposed framework to process up to one million log entries while maintaining a high template quality.

3.8 Template Extraction

Following adaptive semantic cluster refinement, each merged cluster was transformed into a single canonical log template that represented a unique event type. The objective of this stage was to generate a compact, semantically coherent, and human-interpretable event vocabulary that could be directly utilized for downstream sequence construction and anomaly detection tasks.

For every refined cluster c, the representative template is determined by selecting the most frequent normalized log message within that cluster.

$$T_c = \{t_i \mid y_i = c\},$$

where y_i denotes the refined cluster label assigned to the i-th normalized log message. Formally, given the set of normalized messages, the template for cluster c is obtained as:

$$\text{template}_c = \arg\max_{t \in T_c} \text{count}(t).$$

where $\text{count}(t)$ denotes the occurrence frequency of the normalized message t within cluster c. This majority-vote strategy ensures that the selected template corresponds to the most representative event pattern while minimizing intra-cluster lexical variations.

To remove any residual redundancy arising from the merging process (i.e., cases in which distinct merged clusters converge to identical normalized strings), an exact-string deduplication step is applied. Only the first occurrence of each unique template was retained, ensuring a minimal and non-redundant event vocabulary.

The extracted templates are stored in a structured CSV file in which the `cluster_id` corresponds to the identifier of the final merged cluster and serves as the event ID for subsequent analyses. In addition, an augmented dataset is generated that contains the original log message, its normalized representation, the initial HDBSCAN label, the refined cluster assignment, and the resulting template. This dual representation facilitates manual validation, statistical analysis, and seamless integration into the block-level sequence construction process described in the following sections.

3.9 Sequence Construction

Following template extraction in Section 3.8, the final stage of the HDFS parsing pipeline constructs ordered event sequences at the block-level. This transformation converts individually labeled log entries into

structured execution traces that capture the temporal evolution of each data block, thereby providing the input representation required for downstream anomaly detection models.

Notably, sequence construction is performed exclusively on the HDFS dataset, as the official anomaly labels provided by LogHub are defined at the block level. In contrast, the BGL dataset was utilized solely to evaluate the scalability and generalization capability of the proposed log-parsing framework.

For each log entry, the corresponding refined cluster identifier (i.e., the event ID obtained after HDBSCAN clustering and adaptive semantic merging) is associated with the original log message. Block identifiers are subsequently extracted using a lightweight regular expression matching the pattern `blk_-?\d+`. Only log entries containing valid block identifiers are retained, thereby ensuring that each generated sequence corresponds to a unique HDFS data block.

The extracted event IDs were then grouped according to their block identifiers while preserving their original chronological order. Formally, for a block b, the resulting event sequence is represented as

$$S_b = [e_1, e_2, \ldots, e_n],$$

where e_i denotes the event ID assigned to the i-th log message associated with block b, and n is the length of the sequence. Preserving temporal ordering enables anomaly detection models to learn both the occurrence patterns and execution dynamics of normal and abnormal system behaviors.

Finally, the constructed sequences were aligned with the official anomaly labels provided by LogHub, producing a fully structured dataset for supervised learning experiments. Each sequence is assigned a binary label indicating whether the corresponding block is normal or anomalous in nature. The resulting event-sequence dataset served as the foundation for the machine learning, deep learning, and transformer-based anomaly detection models evaluated in this study.

3.10 Anomaly Label Integration and Final Dataset Preparation

The primary objective of this study was to develop and evaluate an unsupervised semantic log-parsing framework. Subsequent anomaly detection experiments were conducted solely as a downstream validation task to assess whether the event templates and sequences generated by SemBERT preserved sufficient semantic information for practical log analytics applications. No new anomaly detection algorithms or learning architectures were proposed in this study.

Following the construction of block-level event sequences in Section 3.9, the generated sequences were aligned with the official HDFS anomaly labels provided by LogHub. These labels are defined at the block level and indicate whether a particular HDFS block exhibits normal or anomalous behaviors.

The label annotations were converted into a binary representation, where anomalous blocks were encoded as 1 and normal blocks as 0. A relational join operation is subsequently performed using the common key `block_id`, ensuring that each event sequence is associated with its corresponding ground truth label. Only entries present in both datasets were retained in the final corpus.

Each resulting instance therefore consists of:

- A unique HDFS block identifier;
- An ordered sequence of event template IDs generated by SemBERT;
- A binary label indicating normal or anomalous behavior.

The resulting labeled sequence dataset served as an evaluation benchmark for assessing the utility of the proposed parser in downstream anomaly detection scenarios. To this end, a diverse set of conventional machine learning, deep learning, and transformer-based classifiers were employed in Section 4. The purpose

of these experiments was not to introduce a novel anomaly detection framework but rather to demonstrate that the semantic representations produced by SemBERT can effectively support existing detection methods.

3.11 Evaluation Metrics

To comprehensively evaluate the effectiveness of the proposed SemBERT framework, three widely adopted log-parsing metrics were employed: Parsing Accuracy (PA), Template Accuracy (TA), and Grouping Accuracy (GA). These metrics assess different aspects of parsing quality, including instance-level correctness, template extraction capability, and clustering consistency [4,17,21].

3.11.1 Parsing Accuracy (PA)

Parsing Accuracy measures the proportion of log messages that are assigned to their correct event templates. Let N denote the total number of log entries and N_c the number of correctly parsed entries. PA is defined as

$$PA = \frac{N_c}{N}.$$

A higher PA indicates that the parser successfully preserves the semantic structure of the original log events and minimizes incorrect template assignments.

3.11.2 Template Accuracy (TA)

Template Accuracy evaluates the ability of the parser to discover the correct set of event templates. Let T_p represent the number of correctly identified templates and T_g denote the total number of ground-truth templates. TA is computed as

$$TA = \frac{T_p}{T_g}.$$

This metric reflects how effectively the parser captures the underlying event vocabulary while avoiding template fragmentation or excessive merging.

3.11.3 Grouping Accuracy (GA)

Grouping Accuracy assesses whether log messages belonging to the same event type are assigned to a common cluster. Following prior studies [21], GA is defined as

$$GA = \frac{2N_{pair}}{N(N-1)},$$

where N_{pair} denotes the number of correctly grouped log-message pairs and N is the total number of log entries. GA provides a direct measure of clustering consistency and semantic coherence among parsed events.

3.11.4 Robustness and Scalability Evaluation

Beyond the standard parsing metrics, the robustness of SemBERT was assessed through repeated experiments using multiple random seeds (42, 123, and 999) at each sampling scale. For each experimental setting, the mean and standard deviation of the adaptive similarity threshold, number of extracted templates, and noise ratio were reported to quantify the statistical stability.

Scalability was evaluated on progressively larger subsets ranging from 50K to 1M log entries for both the HDFS and BGL datasets. The analysis focuses on the evolution of template discovery, adaptive threshold behavior, and parsing consistency as the data volume increases, thereby demonstrating the applicability of the proposed framework to million-scale industrial log repositories.

3.12 Implementation Details

The proposed framework was implemented in Python 3.11 using PyTorch, Transformers, Scikit-learn, HDBSCAN, NumPy, and Pandas. All experiments were conducted on Google Colaboratory using an NVIDIA Tesla T4 GPU with 16 GB of memory.

The BERT encoder employed the `bert-base-uncased` model from the Hugging Face Transformers library. The log messages were processed using a maximum sequence length of 64 tokens and a batch size of 128. Dimensionality reduction was performed using Incremental PCA with 50 principal components and a batch size of 5000 samples.

To evaluate robustness, each experiment was repeated using three random seeds (42, 123, and 999), and the reported results corresponded to the mean and standard deviation across runs. All preprocessing, clustering, and template extraction procedures were implemented in a fully unsupervised manner, without requiring any manually annotated templates.

3.13 Use of AI-Assisted Tools

The authors used AI-assisted tools during the implementation and preparation of this study for two limited purposes. First, the tool was used to assist with Python-code debugging, identify possible implementation errors, explain programming-related issues, and suggest alternative programming approaches. The tool did not autonomously design, implement, execute, or validate the SemBERT framework. All code used for the experiments was written, inspected, modified, executed, and validated by the authors.

Second, the tool was used to improve author-written manuscript text through grammar correction, linguistic refinement, sentence restructuring, and paraphrasing. It was not used to generate the research objectives, methodology, experimental protocol, datasets, results, statistical analyses, figures, interpretations, or scientific conclusions. AI-generated suggestions were treated only as editorial or programming assistance and were critically reviewed by the authors

4 Results and Discussion

This section presents a comprehensive evaluation of the proposed SemBERT framework on the HDFS and BGL benchmark datasets obtained from Loghub. Experiments on the HDFS dataset investigated the scalability, robustness, and semantic parsing quality of the proposed approach across multiple data volumes ranging from 50,000 to 1 million log entries. The BGL dataset was further employed to assess the cross-dataset generalization capability of SemBERT under substantially different logging characteristics and system environments. All experiments were conducted according to the methodology described in Section 3 and were repeated under three different random seed initializations to verify the reproducibility of the proposed deterministic pipeline.

The proposed approach was compared with representative state-of-the-art log parsers, including Drain, AEL, Spell, and LogMine, using the standard evaluation metrics of Grouping Accuracy (GA), Parsing Accuracy (PA), and Template Accuracy (TA). In addition, block-level event sequences generated from the parsed HDFS logs were utilized in downstream anomaly detection experiments using machine learning, deep learning, and transformer-based models. These experiments were performed solely to demonstrate the

practical utility and semantic quality of the extracted templates, rather than to propose a novel anomaly detection framework.

4.1 Parsing Accuracy and Robustness on the HDFS Dataset

The proposed SemBERT framework achieved the highest Parsing Accuracy (PA) among all evaluated methods, attaining a bootstrap mean PA of 95.82 ± 0.44% with a 95% confidence interval of [94.95%, 96.70%]. This represents a substantial improvement over Drain (88.92%), AEL (62.11%), Spell (62.11%), and LogMine (60.01%). SemBERT also maintained perfect Grouping Accuracy (GA = 100.00%), demonstrating its ability to consistently cluster semantically identical log events into coherent event groups without introducing fragmentation.

Fig. 2 presents the comparative performance of SemBERT and four representative log parsing baselines on the HDFS dataset and Table 4 summarizes the quantitative results of all methods.

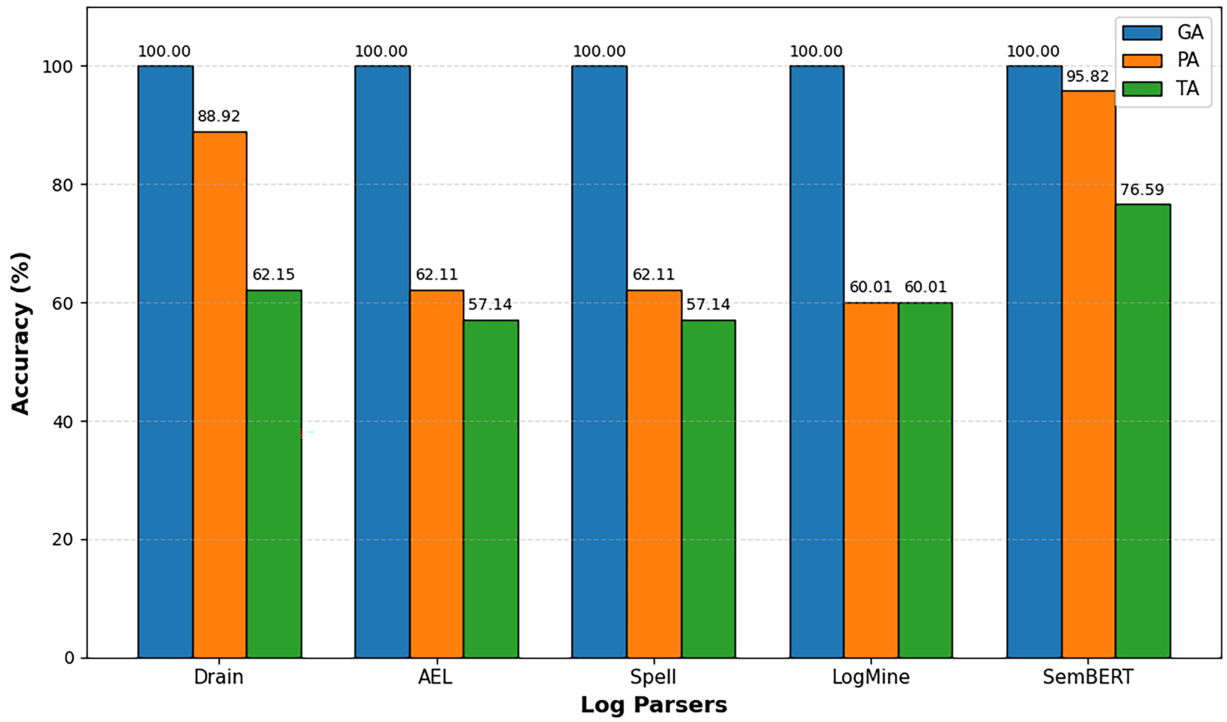

Figure 2: Comparison of log parsing performance on the HDFS dataset using Grouping Accuracy (GA), Parsing Accuracy (PA), and Template Accuracy (TA).

Table 4: Performance comparison of SemBERT and representative log parsers on the HDFS dataset.

Method	GA (%)	PA (%)	TA (%)
Drain	100.00	88.92	62.15
AEL	100.00	62.11	57.14
Spell	100.00	62.11	57.14
LogMine	100.00	60.01	60.01
SemBERT	100.00	95.82 ± 0.44	76.59 ± 4.49

For Template Accuracy (TA), SemBERT achieved a bootstrap mean of 76.59 ± 4.49% with a 95% confidence interval of [71.43%, 83.33%], outperforming all baseline methods while preserving meaningful semantic distinctions among structurally similar log messages. The higher variability observed in TA reflects the inherent diversity of valid semantic template representations rather than instability in the clustering process, as the final number of extracted templates remained identical across multiple experimental runs.

Unlike heuristic parsers that primarily rely on handcrafted rules, fixed tree structures, or lexical matching, SemBERT combines semantic masking, contextual BERT embeddings, Incremental PCA, HDBSCAN clustering, and adaptive centroid-based merging to capture both structural and contextual relationships among log events. This semantic representation enables the framework to group conceptually similar messages, even when their surface forms differ substantially.

Furthermore, the adaptive semantic threshold mechanism eliminates the need for manually selecting similarity parameters by automatically identifying the largest semantic gap among the cluster centroids. This dynamic merging strategy improves the robustness across datasets of different scales and characteristics, reducing template fragmentation while preserving semantic fidelity. Overall, the results demonstrate that SemBERT provides a scalable, statistically robust, and semantically aware alternative to conventional log-parsing approaches for large-scale distributed systems.

4.2 Efficiency and Scalability on the HDFS Dataset

The computational efficiency of SemBERT was evaluated on the HDFS dataset using an NVIDIA Tesla T4 GPU. The complete end-to-end pipeline, including semantic preprocessing, BERT embedding generation, Incremental PCA, HDBSCAN clustering, adaptive threshold estimation, and template extraction, required a total execution time of 1623 s (approximately 27.1 min) for 1 million log entries. Table 5 presents the runtime contribution of each processing stage.

Table 5: Runtime breakdown of the SemBERT pipeline for 1 million HDFS log entries.

Processing Stage	Time (s)
Semantic Preprocessing	42
BERT Embedding Generation	497
Incremental PCA	420
HDBSCAN Clustering	660
Adaptive Threshold Estimation	2
Template Extraction	2
Total	1623

The experimental results indicate that most of the computational cost originates from semantic representation learning and density-based clustering, whereas the adaptive threshold estimation and template extraction stages introduce negligible overhead. Despite relying on contextual embeddings, the framework remains computationally feasible for million-scale log datasets and benefits significantly from GPU acceleration during embedding generation.

To evaluate scalability, the proposed framework was executed on HDFS subsets containing 50,000, 100,000, 200,000, 500,000, and 1,000,000 log entries. The detailed scalability characteristics across different HDFS log volumes are presented in Table 6 in the following subsection. Fig. 3 illustrates the relationship between input size and the number of extracted templates. The number of templates remains remarkably

compact, increasing from only seven templates at 50K logs to eighteen templates at one million logs. This near-linear growth demonstrates that the semantic clustering and adaptive centroid-merging strategy effectively mitigate template fragmentation, preventing the combinatorial explosion commonly observed in large-scale clustering-based parsers.

Table 6: Scalability characteristics of SemBERT on the HDFS dataset.

Log Volume	Initial Clusters	Merged Clusters	Adaptive Threshold	Final Templates
50K	14	7	0.6509	7
100K	14	7	0.6488	7
200K	16	10	0.8179	10
500K	38	17	0.8365	17
1M	43	18	0.8362	18

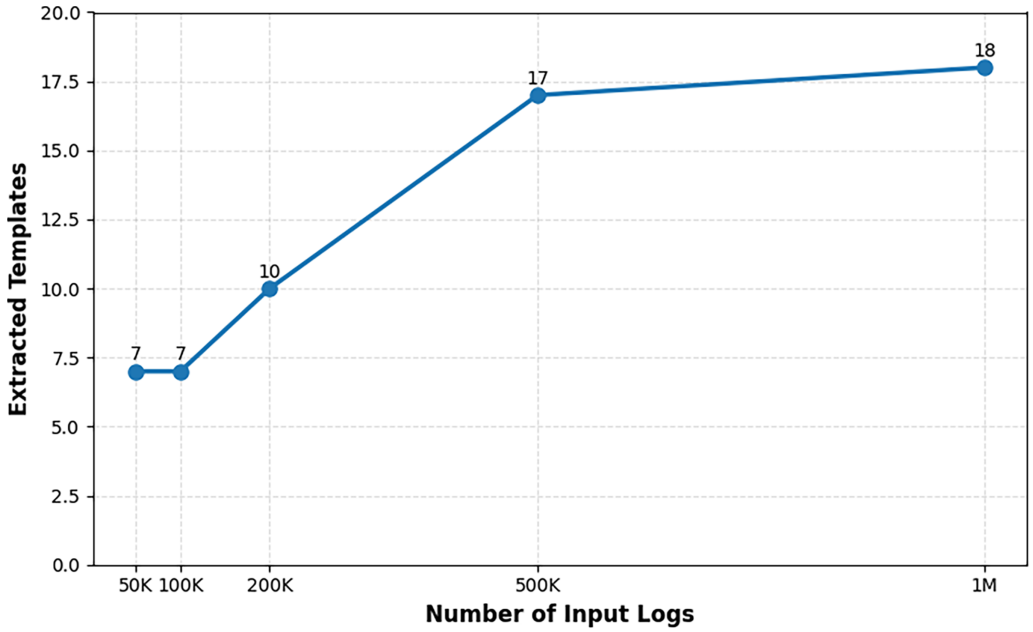

Figure 3: Scalability analysis of SemBERT on the HDFS dataset: number of input logs vs. extracted templates.

Furthermore, the adaptive similarity threshold remained relatively stable as the dataset size increased, converging to approximately 0.84 for larger samples. This behavior suggests that the semantic structure learned by the framework becomes increasingly consistent with more training data, enabling robust cluster merging without requiring manually tuned similarity parameters. Overall, these findings demonstrate that SemBERT scales effectively to million-scale log collections while maintaining a compact and semantically meaningful event vocabulary.

4.3 Scalability and Reproducibility on the HDFS Dataset

To evaluate the scalability and reproducibility of the proposed framework, SemBERT was applied to HDFS subsets containing 50K, 100K, 200K, 500K, and 1M log entries. Each experiment was repeated under three different random seed initializations (42, 123, and 999) as a reproducibility check. Since the proposed

pipeline is deterministic, identical parsing outputs were expected across repeated executions. Therefore, these repeated executions were performed to verify the reproducibility and implementation stability of the proposed framework, rather than to evaluate the stochastic variability. Table 6 summarizes the scalability characteristics of SemBERT, including the numbers of initial clusters, merged clusters, adaptive thresholds, and final extracted templates across different dataset sizes.

The results demonstrate a highly stable scaling behavior as the number of input logs increases. Although the number of initial HDBSCAN clusters grows from 14 at 50K logs to 43 at 1M logs, the adaptive semantic merging strategy consistently consolidates these clusters into a compact set of meaningful templates. The final template vocabulary increases gradually from seven templates at 50K and 100K logs to only 18 templates at 1 million log entries, indicating that the proposed framework effectively prevents template explosion while preserving semantic distinctions among log events.

The adaptive similarity threshold also exhibited consistent behavior across larger datasets. The threshold values converged to approximately 0.84 for the 500K and 1M experiments, suggesting that the semantic relationships learned by the BERT embedding space remained stable as additional log instances were introduced. This automatic threshold estimation mechanism eliminates the need to manually tune the similarity parameters and improves the robustness of the clustering process across different data volumes.

Repeated executions with three different random seed initializations confirmed the reproducibility of the proposed framework. For every dataset size, identical numbers of initial clusters, merged clusters, adaptive thresholds, final templates, and noise logs were obtained, demonstrating that the proposed parsing pipeline produced deterministic outputs for identical inputs. Minor differences were observed only in the execution time, which were attributable to hardware scheduling and GPU resource allocation rather than algorithmic randomness.

These findings demonstrate that SemBERT maintains both scalability and reproducibility when processing large-scale log repositories, making it suitable for deployment in real-world distributed systems, where log volumes may vary substantially over time.

4.4 Qualitative Analysis of Extracted Templates

To further examine the semantic quality of the generated templates, Table 7 presents a comparison between representative HDFS templates provided by LogHub and the corresponding templates produced by SemBERT. Five frequently occurring events were selected to illustrate how semantic preprocessing and contextual clustering preserve meaningful system information while maintaining template generalization.

Table 7: Comparison between LogHub reference templates and SemBERT-generated templates on the HDFS dataset.

LogHub	LogHub Template	SemBERT	SemBERT Template
E2	< * > Verification succeeded for < * >	Cluster 4	Verification succeeded for <BLOCK_ID >
E3	< * > Served block < * > to < * >	Cluster 19	Served block <BLOCK_ID > to <IP >
E9	< * > Received block < * > of size < * > from < * >	Cluster 18	Received block <BLOCK_ID > of size <NUM > from <IP >
E21	< * > Deleting block < * > file < * >	Cluster 1	Deleting block <BLOCK_ID > file <PATH ><PATH >

(Continued)

Table 7 (continued)

LogHub	LogHub Template	SemBERT	SemBERT Template
E27	< * > BLOCK* NameSystem < * > addStoredBlock: Redundant addStoredBlock request received for < * > on < * > size < * >	Cluster 10	BLOCK* NameSystem.addStoredBlock: Redundant addStoredBlock request received for <BLOCK_ID > on <IP_PORT > size <NUM >

The comparison reveals that SemBERT preserves substantially richer semantic information than the generic LogHub templates. While the reference templates replace most variable components using anonymous wildcards (< * >), SemBERT employs semantic masking to distinguish between different entity types, such as block identifiers, IP addresses, ports, numeric values, and file paths. Consequently, the resulting templates remain both generalized and semantically interpretable.

For example, the LogHub template for block transmission events (E3) abstracts the destination host as a generic wildcard, whereas SemBERT explicitly represents it as an <IP > token. Similarly, redundant block storage requests (E27) retain meaningful distinctions between block identifiers, network endpoints, and numerical attributes through the use of <BLOCK_ID >, <IP_PORT >, and <NUM > placeholders. Such semantic preservation improves the interpretability of the extracted templates and provides more informative event sequences for downstream log analysis tasks, including anomaly detection.

These observations further support the quantitative results reported in the previous sections, demonstrating that SemBERT achieves high parsing accuracy while maintaining semantically meaningful template representations across large-scale distributed system logs.

4.5 Cross-Dataset Generalization on the BGL Dataset

To evaluate the generalization capability of SemBERT beyond HDFS, experiments were conducted on the BGL dataset using log volumes ranging from 50K to 1M entries. Table 8 summarizes the clustering behavior, adaptive semantic thresholds, and final template counts across different scales.

Table 8: Scalability analysis of SemBERT on the BGL dataset.

Log Volume	Initial Clusters	Merged Clusters	Adaptive Threshold	Final Templates
50K	24	15	0.9181	14
100K	57	17	0.7751	16
200K	74	25	0.8374	24
500K	91	25	0.7626	24
1M	122	50	0.8752	53

The results demonstrate that SemBERT maintains strong scalability and semantic consistency when applied to fundamentally different log environments. While the HDFS dataset primarily consists of storage and replication events, the BGL dataset contains heterogeneous system-level messages, hardware faults, kernel exceptions, and application errors. Despite this increased complexity, the proposed framework successfully controlled template growth, producing only 53 semantic templates from one million log entries.

The adaptive semantic threshold mechanism exhibited dataset-dependent behavior, varying from 0.7751 to 0.9181 for different input sizes. Unlike fixed similarity thresholds, the proposed approach automatically adjusts the merging criterion according to the semantic distribution of the data, enabling robust clustering without manual parameter tuning. At the 1M scale, the mean adaptive threshold of 0.8752 yielded 53 final templates after semantic merging, indicating the effective consolidation of structurally diverse yet semantically related log events.

The representative templates extracted from the BGL dataset are presented in Table 9. The results show that SemBERT preserves meaningful operational semantics by generating interpretable templates for kernel failures, application errors, hardware exceptions, and system-level information messages while abstracting variable entities such as file paths, memory addresses, numerical identifiers, and IP addresses.

Table 9: Representative semantic templates extracted by SemBERT from the BGL dataset.

Template ID	Semantic Template
T1	NULL DISCOVERY ERROR: <MESSAGE >
T2	RAS APP FATAL ciod: Error creating node map from file <PATH >: <ERROR >
T3	RAS KERNEL FATAL data storage interrupt
T4	RAS KERNEL INFO <NUM > DDR errors detected and corrected on rank <NUM >, symbol <NUM >, bit <NUM >
T5	RAS LINKCARD INFO MidplaneSwitchController performing bit sparing on R27-M1-L3-U18-C bit <NUM >

These findings demonstrate that the proposed semantic parsing framework generalizes effectively across heterogeneous distributed systems without requiring dataset-specific heuristics or handcrafted parsing rules. The ability to maintain compact, interpretable, and semantically meaningful templates across both HDFS and BGL suggests that SemBERT provides a robust foundation for large-scale log understanding and downstream analytics tasks in diverse computing environments.

4.6 Downstream Validation through Anomaly Detection (HDFS Dataset)

To assess the practical utility of the proposed semantic parser, the extracted event templates were converted into block-level event sequences and aligned with the official HDFS anomaly labels. The resulting sequences were subsequently evaluated using representative machine learning, deep learning, and transformer-based models. This experiment serves solely as a downstream validation of template quality and does not constitute a new anomaly detection methodology.

Fig. 4 presents the confusion matrices of the best-performing model from each category, namely Random Forest (machine learning), CNN-BiGRU (deep learning), and Transformer-CNN (transformer-based). These models were selected to illustrate the effectiveness of the semantic templates in supporting diverse learning paradigms.

Table 10 summarizes the performance of the representative machine learning models. Random Forest achieved the strongest overall performance, attaining an accuracy of 98.01%, an F1-score of 71.63%, and a precision of 98.52%. XGBoost produced comparable results with the highest precision (99.39%), while Logistic Regression provided a competitive baseline despite its simpler linear formulation. These results

indicate that the semantic event sequences generated by SemBERT preserve sufficient contextual information to support effective anomaly discrimination using conventional learning algorithms.

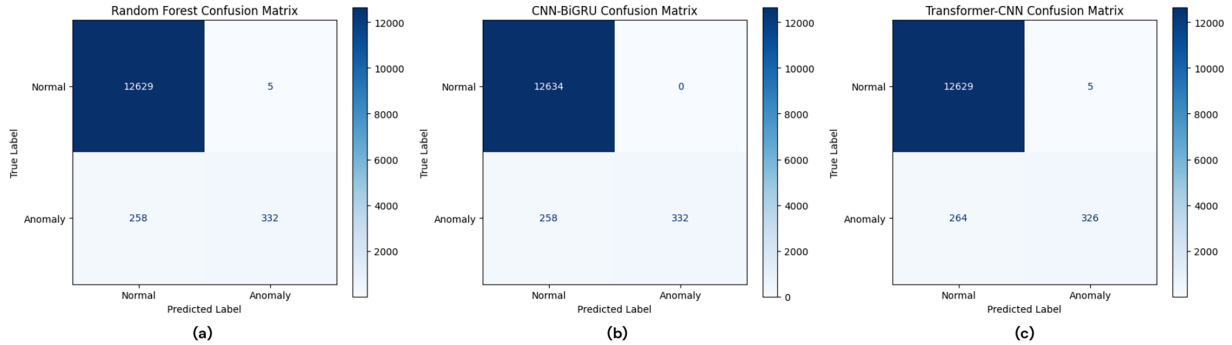

Figure 4: Confusion matrices of representative downstream models on the HDFS dataset: (**a**) Random Forest, (**b**) CNN-BiGRU, and (**c**) Transformer-CNN.

Table 10: Representative machine learning models for downstream anomaly detection on HDFS.

Model	Accuracy	Precision	Recall	F1
Logistic Regression	0.9731	0.8611	0.4729	0.6105
Random Forest	0.9801	0.9852	0.5627	0.7163
XGBoost	0.9797	0.9939	0.5492	0.7074

The deep learning models further confirmed the effectiveness of the extracted semantic templates. As shown in Table 11, the CNN-BiGRU architecture achieved the highest overall performance, obtaining an accuracy of 98.05%, perfect precision (100%), and an F1-score of 72.02%. Standard CNN and LSTM models also demonstrated competitive performance, suggesting that the generated event sequences provide robust representations that generalize across different sequence-learning architectures without requiring additional feature engineering.

Table 11: Representative deep learning models for downstream anomaly detection on HDFS.

Model	Accuracy	Precision	Recall	F1
LSTM	0.9795	0.9819	0.5508	0.7058
CNN	0.9802	0.9852	0.5644	0.7177
CNN-BiGRU	0.9805	1.0000	0.5627	0.7202

Table 12 presents the results obtained using transformer-based architectures. The Transformer-CNN hybrid achieved the highest accuracy (97.97%) and F1-score (70.79%), while the Deep Transformer model produced slightly higher recall values. The standard Transformer Encoder yielded comparable performance, demonstrating that the semantic structures produced by SemBERT remain informative across fundamentally different neural modeling approaches.

Although anomaly detection is not the primary contribution of this work, the consistently strong performance across machine learning, deep learning, and transformer-based models provides additional evidence that the semantic templates generated by SemBERT capture meaningful execution patterns and

preserve sufficient contextual information for downstream system analytics. These findings demonstrate that the proposed parsing framework can serve as an effective preprocessing component for log-based monitoring and anomaly analysis in large-scale distributed systems.

Table 12: Representative transformer-based models for downstream anomaly detection on HDFS.

Model	Accuracy	Precision	Recall	F1
Transformer Encoder	0.9784	0.9810	0.5254	0.6843
Deep Transformer	0.9778	0.9157	0.5525	0.6892
Transformer-CNN	0.9797	0.9849	0.5525	0.7079

5 Discussion

The experimental findings demonstrate that SemBERT provides an effective balance between semantic parsing quality, scalability, and practical applicability in large-scale distributed systems. By leveraging contextual BERT embeddings and density-based clustering, the framework captures semantic relationships that conventional heuristic and rule-based parsers often overlook, achieving a Parsing Accuracy of 95.82 ± 0.44%, a Template Accuracy of 76.59 ± 4.49%, and perfect Grouping Accuracy on the HDFS benchmark. The adaptive semantic gap thresholding mechanism further improves robustness by automatically determining suitable merging thresholds according to the semantic characteristics of each dataset, eliminating the need for manually selecting similarity parameters. Across HDFS and BGL, the method consistently produced compact template vocabularies and demonstrated identical results across multiple random seeds, confirming the reproducibility of the proposed pipeline. Moreover, the scalability analysis showed controlled template growth, increasing from seven templates at 50K HDFS logs to only 18 templates at one million logs, suggesting that the combination of Incremental PCA and HDBSCAN effectively mitigates template fragmentation and cluster explosion in large-scale environments. The generated templates were also useful for downstream analytics, where representative machine learning, deep learning, and transformer-based models achieved competitive performance without additional feature engineering, supporting the practical value of the parsed event sequences while not constituting a separate anomaly detection contribution.

Experiments on the BGL dataset further indicated that the semantic embedding and clustering framework generalizes across heterogeneous logging environments with only minor adjustments to dataset-specific message extraction and masking rules. Nevertheless, this study has several limitations. The current evaluation focuses on HDFS and BGL benchmarks, and broader validation on enterprise-scale systems, such as Spark, OpenStack, and cloud-native infrastructures, would strengthen the generalizability of the findings. In addition, the use of transformer embeddings benefits substantially from GPU acceleration, which may limit deployment in resource-constrained environments and motivate future investigations into lightweight or distilled language models. The present implementation is also designed for offline batch processing and does not support online or streaming log analysis for real-time monitoring scenarios. Finally, although recent LLM-based parsers offer promising few-shot capabilities, they typically depend on prompt engineering and external supervision, whereas SemBERT remains fully unsupervised and deterministic, requiring neither annotated data nor hand-crafted prompts. These observations suggest that semantic representations combined with adaptive density-based clustering provide a promising foundation for future research on scalable automated log analytics and intelligent system monitoring.

6 Conclusion

This study presents SemBERT, a fully unsupervised semantic log parsing framework that integrates entity-aware preprocessing, contextual BERT embeddings, Incremental PCA, HDBSCAN clustering, and adaptive semantic-gap thresholding to transform large-scale unstructured logs into compact and meaningful event templates. Experimental results on the HDFS benchmark demonstrated strong parsing performance, achieving a Parsing Accuracy of 95.82 ± 0.44%, a Template Accuracy of 76.59 ± 4.49%, and perfect Grouping Accuracy while producing identical parsing results across multiple random seeds, confirming the reproducibility of the proposed framework. The generated templates also produced informative event sequences that supported downstream analytics tasks using representative machine learning, deep learning, and transformer-based models, without additional feature engineering.

Experiments on the BGL dataset further showed that the proposed framework generalizes to heterogeneous logging environments with only minor dataset-specific preprocessing adjustments, highlighting the practicality of semantic representations for automated log analysis. Although the current implementation focuses on offline batch processing and relies on GPU-accelerated transformer embeddings, the results suggest that adaptive density-based semantic clustering offers a promising direction for scalable and reproducible log parsing in the future. Future work will explore broader evaluations of enterprise-scale datasets, lightweight language models for resource-constrained environments, and streaming extensions for real-time system monitoring and fault diagnosis.

Acknowledgement: The authors acknowledge the use of AI for assistance with Python code debugging and programming-related suggestions during implementation and AI for language editing, grammar correction, and paraphrasing during manuscript preparation. All research activities, experimental validation, data analysis, and scientific conclusions are the original work of the authors and were independently verified before submission.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Gobinda Bhattacharjee and Joy Dey; methodology, Gobinda Bhattacharjee, Tanjim Mahmud; software, Gobinda Bhattacharjee; validation, Gobinda Bhattacharjee, Joy Dey and Tanjim Mahmud; formal analysis, Gobinda Bhattacharjee; investigation, Gobinda Bhattacharjee; data curation, Gobinda Bhattacharjee; writing—original draft preparation, Gobinda Bhattacharjee; writing—review and editing, Gobinda Bhattacharjee, Tanjim Mahmud, Mohammad Shahadat Hossain and Karl Andersson; visualization, Gobinda Bhattacharjee; supervision, Tanjim Mahmud. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly available in the LogHub repository at https://github.com/logpai/loghub.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Khan ZA, Shin D, Bianculli D, Briand LC. Impact of log parsing on deep learning-based anomaly detection. Empir Softw Eng. 2024;29(6):139. doi:10.1007/s10664-024-10533-w.
2. Zhou J, Qian Y, Zou Q, Liu P, Xiang J. DeepSyslog: deep anomaly detection on syslog using sentence embedding and metadata. IEEE Trans Inf Forensics Secur. 2022;17(6):3051–61. doi:10.1109/TIFS.2022.3201379.
3. Zhu J, He S, Liu J, He P, Xie Q, Zheng Z, et al. Tools and benchmarks for automated log parsing. In: Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP); 2019 May 25–31; Montreal, QC, Canada. p. 121–30.

4. He P, Zhu J, He S, Li J, Lyu MR. Towards automated log parsing for large-scale log data analysis. IEEE Trans Dependable Secur Comput. 2018;15(6):931–44. doi:10.1109/TDSC.2017.2762673.
5. He P, Zhu J, Zheng Z, Lyu MR. Drain: an online log parsing approach with fixed depth tree. In: Proceedings of the 2017 IEEE International Conference on Web Services (ICWS); Honolulu, HI, USA. p. 33–40. doi:10.1109/ICWS.2017.13.
6. Landauer M, Wurzenberger M, Skopik F, Settanni G, Filzmoser P. Dynamic log file analysis: an unsupervised cluster evolution approach for anomaly detection. Comput Secur. 2018;79(4):94–116. doi:10.1016/j.cose.2018.08.009.
7. Yu S, Chen N, Wu Y, Dou W. Self-supervised log parsing using semantic contribution difference. J Syst Softw. 2023;200(1):111646. doi:10.1016/j.jss.2023.111646.
8. Tian R, Diao Z-L, Jiang H-Y, Xie G-G. Cognition: accurate and consistent linear log parsing using template correction. J Comput Sci Technol. 2023;38(5):1036–50. doi:10.1007/s11390-021-1691-3.
9. Dai H, Li H, Chen C-S, Shang W, Chen T-H. Logram: efficient log parsing using n-gram dictionaries. IEEE Trans Softw Eng. 2022;48(3):879–92. doi:10.1109/TSE.2020.3007554.
10. Cao J, Di X, Liu X, Xu R, Li J, Ren W, et al. Towards robust log parsing using self-supervised learning for system security analysis. Intell Data Anal. 2024;28(4):1093–1113. doi:10.3233/IDA-230133.
11. Xu A, Gau A. HELP: hierarchical embeddings-based log parsing. arXiv:2408.08300. 2024.
12. Zhang C, Xu W, Liu J, Zhang L, Liu G, Guan J, et al. SemanticLog: towards effective and efficient large-scale semantic log parsing. IEEE Trans Softw Eng. 2026;52(1):155–70. doi:10.1109/TSE.2025.3625121.
13. Ma Z, Kim DJ, Chen T-HP. LibreLog: accurate and efficient unsupervised log parsing using open-source large language models. In: Proceedings of the 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE); 2025 Apr 27–May 3; Ottawa, ON, Canada. p. 924–36. doi:10.1109/ICSE55347.2025.00103.
14. Du M, Li F. Spell: online streaming parsing of large unstructured system logs. IEEE Trans Knowl Data Eng. 2019;31(11):2213–27. doi:10.1109/TKDE.2018.2875442.
15. Tao S, Meng W, Chen Y, Zhu Y, Liu Y, Du C, et al. LogStamp: automatic online log parsing based on sequence labelling. ACM SIGMETRICS Perform Eval Rev. 2022;49(4):93–8. doi:10.1145/3543146.3543168.
16. Zhang T, Qiu H, Castellano G, Rifai M, Chen CS, Pianese F. System log parsing: a survey. IEEE Trans Knowl Data Eng. 2023;35(8):8596–614. doi:10.1109/TKDE.2022.3222417.
17. Vervaet A, Callau-Zori M, Chabchoub Y, Chiky R. Online log parsing using evolving research tree. Knowl Inf Syst. 2024;66(2):1231–55. doi:10.1007/s10115-023-01953-z.
18. Le V-H, Zhang H. PreLog: a pre-trained model for log analytics. Proc ACM Manag Data. 2024;2(3):163. doi:10.1145/3654966.
19. Wang Y, Chen P, Huang H, He Z, Tan G, Zhang C, et al. InferLog: accelerating LLM inference for online log parsing via ICL-oriented prefix caching. arXiv:2507.08523. 2025.
20. Zhang L, Meng X, Chen J, Chi Y. SCULP: an unsupervised LLM-based log parser with self-correcting capabilities. J Syst Softw. 2026;240(8):112935. doi:10.1016/j.jss.2026.112935.
21. He P, Zhu J, He S, Li J, Lyu MR. An evaluation study on log parsing and its use in log mining. In: Proceedings of the 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN); 2016 Jun 28–Jul 1; Toulouse, France. p. 654–61. doi:10.1109/DSN.2016.66.