LESR: LLM-Based Event Sequence Reconstruction for Repairing Identical Timestamp Errors in Logs

Zhengxu Huang[1, 2], Yishi Zhao[1, 2*], Jianga Shang[1, 2], and Shi Ying[3]

[1] The School of Computer Science, China University of Geosciences, Wuhan 430074, China.
[2] Engineering Research Center of Natural Resource Information Management and Digital Twin Engineering Software, Ministry of Education, Wuhan 430074, China.
[3] The School of Computer Science, Wuhan University, Wuhan 430072, China.

* Corresponding author. Email: zhaoyishi@cug.edu.cn
Manuscript submitted July 3, 2025; revised September 16, 2025; accepted October 29, 2025; published December 9, 2025
doi: 10.17706/jsw.20.2.95-109

Abstract: Event logs record key activity data for process mining, yet quality defects such as attribute and case errors undermine economic benefits by corrupting critical operational insights. Identical timestamp errors occur when multiple events of a process instance mistakenly share the same timestamp—distort execution sequences, misrepresent concurrency, and compromise analysis accuracy. These errors manifest as multi-layered problems, with case-level sequence errors being particularly impactful. Current solutions inadequately correct consecutive event ordering errors due to incomplete utilization of log sequence information. We propose LLM-based Event Sequence Reconstruction (LESR), a novel framework integrating probabilistic ranking with fine-tuned Large Language Model (LLM) to resolve sequence faults. Its two-phase LLM architecture first generates candidates then selects optimally. Evaluated on 5 real and synthetic logs from e-commerce, healthcare, and public administration domains, LESR significantly outperforms baselines in repairing sequence errors caused by identical timestamps.

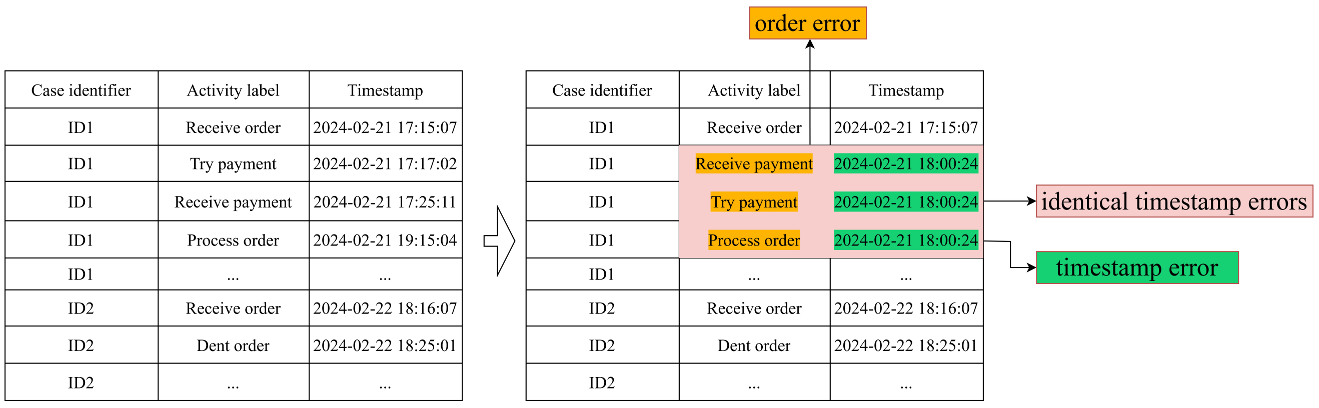
Keywords: event log repair, process mining, process data quality, large language model

1. Introduction

Process mining extracts knowledge from event logs—which contain historical records of business process execution—to provide insights into business operations [1]. This leads to economic benefits such as increased customer satisfaction or reduced costs. Realizing the full potential of process mining depends critically on using high-quality event logs as input. An event log consists of events representing the actual executions of activities, where each event belongs to a specific process instance and typically includes at least three fundamental core attributes: a case identifier, a timestamp, and an activity label [2]. Among timestamp-related quality problems, identical timestamp errors are described as a state where multiple events within a single process instance share the same timestamp [3]. Errors occur due to: (1) Coarse time units (e.g., daily timestamps), causing identical timestamps for same-day events; (2) System overloads queuing events, which then share a timestamp when processed after recovery. This error obscures the actual execution sequence and time intervals between events. Consequently, events that should be executed sequentially may be misinterpreted as concurrent, leading to distortion of process models and deviation in analysis results [4]. This directly undermines the reliability of subsequent process discovery and performance analysis, inevitably resulting in flawed operational decisions and misguided resource allocation that incur real economic costs.

Identical timestamp errors constitute not merely problems of incorrect timestamps, but rather complex problems involving event ordering faults at the case level and timestamp errors across multiple consecutive events. While identical timestamps could be correct in many logging contexts, the focus here is on event logs where the event sequence must strictly adhere to a total order. In such logs, two events sharing the same case identifier must have comparable and distinct timestamps. In such logs, identical timestamps are considered an error and require repair. Formally, an event log L can be defined as a sequence of events $e_i := (c_i, a_i, t_i)$, where c_i is the case identifier, a_i is the activity label, and t_i is the timestamp. Within each case c, events are assumed to follow a total order based on their timestamps (t_i). In this sense, the correct sequence of events in a case is given by the ascending order of timestamps, i.e., $t_1 < t_2 < \cdots < t_n$. An identical timestamp error occurs if two or more events within the same case share the same timestamp ($c_k = c_l, t_k = t_l, k \neq l$). In this situation, the total order assumption is violated and the true execution order is lost. This differs from legitimate concurrency scenarios, where parallel events may overlap in time but are explicitly modeled as such. Fig. 1 illustrates the distinction between a normal event log state and one affected by identical timestamp errors, using an e-commerce order processing event log [5] as an example. Consequently, generic event log repair methods often prove inadequate for resolving this specific problem. Repairing identical timestamp errors in software event logs can be viewed as a three-step [6]:

1. Detecting erroneous events.
2. Reordering these events to correct faulty sequences.
3. Rectifying incorrect timestamps.



Case identifier	Activity label	Timestamp
ID1	Receive order	2024-02-21 17:15:07
ID1	Try payment	2024-02-21 17:17:02
ID1	Receive payment	2024-02-21 17:25:11
ID1	Process order	2024-02-21 19:15:04
ID1	...	...
ID2	Receive order	2024-02-22 18:16:07
ID2	Dent order	2024-02-22 18:25:01
ID2	...	...

Case identifier	Activity label	Timestamp
ID1	Receive order	2024-02-21 17:15:07
ID1	Receive payment	2024-02-21 18:00:24
ID1	Try payment	2024-02-21 18:00:24
ID1	Process order	2024-02-21 18:00:24
ID1	...	...
ID2	Receive order	2024-02-22 18:16:07
ID2	Dent order	2024-02-22 18:25:01
ID2	...	...

Fig. 1. E-commerce order processing event log.

Conforti *et al.* [3] and Schmid *et al.* [6] focus primarily on repairing identical timestamp errors and are, to the best of our knowledge, the only two methods we are aware of that specifically target this issue. Conforti *et al.* [3] first quantifies directly-follows relations as confidence scores. The event sequence with the highest confidence is identified by solving an Integer Linear Programming problem. Subsequently, kernel density estimation models activity durations to assign timestamps that accurately reflect the actual process execution. However, a key limitation of this method is its narrow focus on the direct successor relationships between activity names, failing to extract richer, more contextual information from the log sequences. This results in inadequate accuracy for sequence repair, especially when dealing with complex or sparse sequences. Schmid *et al.* [6] proposed a model based on Generative Adversarial Networks (GANs). The architecture consists of a generator and a discriminator, which are trained adversarially to capture contextual information from event logs, thereby achieving more accurate repair of identical timestamp errors. While this approach excels at timestamp correction, it performs poorly in sequence repair due to the inherent difficulties GANs face in handling discrete sequential data. Crucially, correcting event ordering errors is an essential step within identical timestamp error repair, significantly impacting process mining outcomes. Moreover, as an upstream task in error repair, it influences the subsequent task of timestamp correction. Consequently, sequence repair within identical timestamp error correction presents significant potential for improvement.

Process model-based methods were initially widely applied for sequence repair. Building on this, Rogge-Solti *et al.* [7] employed stochastic Petri nets to model process fragments, subsequently using cost-based alignment and selective alignment techniques to efficiently determine and repair the correct sequence of log events. Further extending this approach, Dixit *et al.* [8] integrated timestamp statistical analysis with stochastic Petri net modeling to broaden the scope of repairable anomaly types and enabled user interaction incorporating domain knowledge. Shahzadi *et al.* [9] adopted path probability analysis to insert out-of-sequence logs into the most probable positions, utilizing Monte Carlo simulation to handle stochastic events and enhance repair quality. However, these methods exhibit significant limitations: they heavily depend on the quality of the underlying process model, require specialized knowledge during model construction, and pose challenges for subsequent maintenance and updates. Sani *et al.* [10] implemented a probability-based method, repairing misordered logs through frequent context identification and the detection/replacement of infrequent context sub-patterns. Song *et al.* [11] clustered compliant event sequences into multiple trace clusters; when a log deviated, it was repaired using sub-processes or process fragments extracted from the nearest trace cluster. While these methods eliminate reliance on predefined process models, a common limitation persists in existing studies: they are primarily effective for repairing single events or short segments within logs. This is inadequate for identical timestamp errors, which often require reordering substantial segments of event logs. This leads to the core research question: How can log sequence information be fully leveraged to correct ordering errors in consecutive multiple events?

To address sequence errors within identical timestamp errors, this paper develops a method for repairing identical timestamp errors based on Large Language Model (LLM). Existing techniques are retained for the error detection and timestamp correction phases. The key innovation lies in the development of a novel LLM-based component for repairing event ordering errors, by uniquely integrating probabilistic ranking with LLM-based sequence plausibility assessment. This synergistic approach is further enhanced through specialized prompt engineering and domain-specific fine-tuning reinforced by log sequential patterns, significantly boosting the accuracy of event order scoring within targeted domains. Through software prototype implementation and validation, this study successfully repaired 5 real/artificial event logs (comprising 15 variants in total) and conducted accuracy comparisons with existing methods. Given that current approaches for improving event log quality remain inadequate, this study collectively advances progress in the field of process data quality.

The structure of the remainder of this paper is as follows: Section 2 presents the proposed solution. Section 3 evaluates the error repair capability of the method through comparisons with existing research findings. Section 4 discusses the evaluation results and outlines future research directions.

2. Research design

This study addresses sequence repair for identical timestamp errors in event logs. Fig. 2 illustrates the method overview and its functional interactions. During preprocessing, the mature detection method from Suriadi *et al.* [4] is first integrated to establish a reliable error foundation. Although this detection approach cannot distinguish correct event logs with identical timestamps, our experimental data avoids such scenarios, and error detection itself is not the core focus of our research. Our central contribution is the proposed LLM-based Event Sequence Reconstruction (LESR) framework, which leverages pre-trained language model to precisely order events by mining sequential dependencies, effectively solving the misordering challenge.

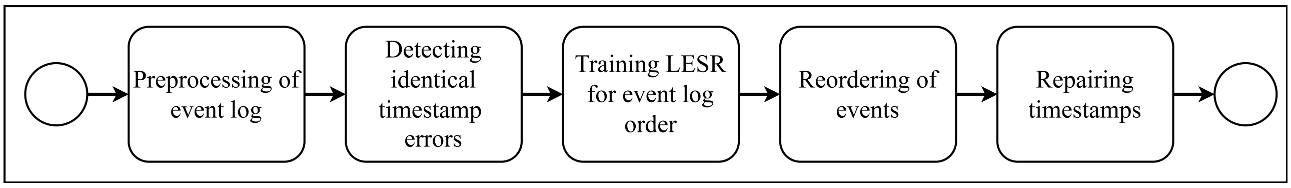

Preprocessing of event log
Detecting identical timestamp errors
Training LESR for event log order
Reordering of events
Repairing timestamps

Fig. 2. Overview of the proposed method for repairing identical timestamp errors.

2.1. Preprocessing and Error Detection

Preprocessing and error detection serve two critical functions: identifying identical timestamp errors and preparing data for training.

During preprocessing, a three-tier data cleaning strategy is implemented to enhance log quality: Rule-based noise elimination reduces interference in subsequent processing by filtering out noise (e.g., erroneous activity labels) and outliers (e.g., illogical activity sequences) [12]. Attribute simplification retains only activity names while discarding irrelevant attributes to improve LLM' event sequence learning efficacy. Regex-based normalization standardizes parameters by replacing structured variables with placeholders <> through predefined pattern-matching rules [13]. For instance, "Order #4792 processed for $189.50" becomes "Order #<> processed for <>"—preserving core semantics while eliminating distractions and avoiding semantic discontinuities that traditional log parsers may introduce with complex parameters.

Accurately distinguishing abnormal events from normal events during error detection presents a fundamental challenge for the entire repair framework. To identify identical timestamp errors, the dual-matching mechanism for case identifiers and timestamps proposed by Suriadi *et al.* [4] is adopted. Formally, any event $(\tilde{e}_i)_{i=1,\dots,n}$ is defined as a quadruple: $\tilde{e}_i := (c_i, a_i, t_i, f_i)$, where c_i denotes the unique case identifier, a_i represents the activity name, t_i records occurrence time, and f_i indicates error status (1 for erroneous, 0 otherwise). The algorithm operates as follows: Sort raw logs chronologically to generate ordered event sequences. Perform bidirectional adjacency checks, each event undergoes equivalence validation of case_id and timestamp against both predecessor and successor events. When events sharing identical case_id and timestamp are detected, they are marked as conflicting ($f_i = 1$). For middle-position events, the error flag is determined by logical OR of two independent detection results (predecessor-match and successor-match), eliminating missed detections common in single-check approaches.

The log data requires partitioning and training set pre-repair for subsequent experiments involving large model fine-tuning and log repair. First, the dataset was partitioned into 80% for training and 20% for testing. Next, identical timestamp errors were introduced into the training set, accounting for 50% of all event groups. These errors were subsequently repaired using Conforti *et al.*'s method [3], which leverages the probability distribution of successor events. This method—detailed in Section 2.2 as part of the sequence repair process—ensures that the fine-tuned large model becomes more sensitive to sequencing errors unresolved by existing techniques.

2.2. Sequence Repair Architecture

To address sequence repair for events with identical timestamps, the architecture shown in Fig. 3 was designed. This solution employs a two-stage collaborative mechanism: The ranking component generates multiple candidate sequence arrangements. The selection component—implemented using a fine-tuned large language model—identifies the optimal sequence. The final output is an event sequence that aligns with business logic.

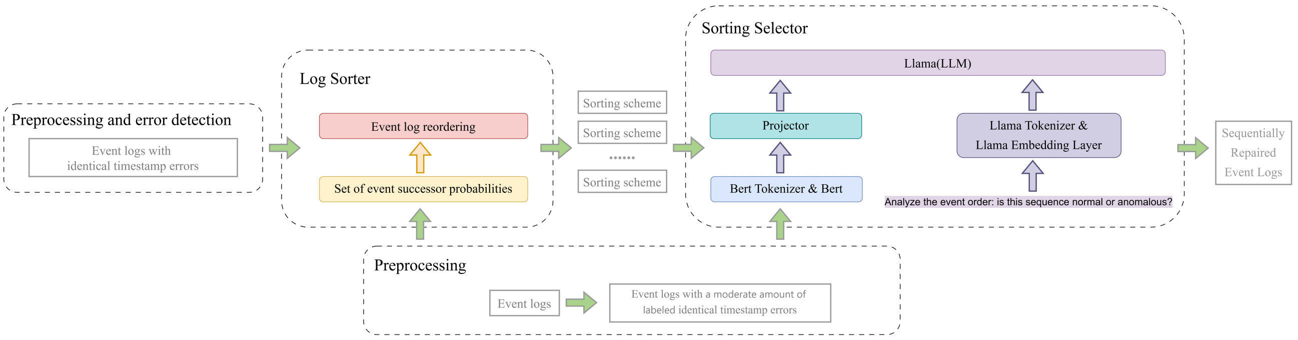

Preprocessing and error detection
Event logs with identical timestamp errors
Log Sorter
Event log reordering
Set of event successor probabilities
Sorting scheme
Sorting scheme
......
Sorting scheme
Sorting Selector
Llama(LLM)
Projector
Llama Tokenizer & Llama Embedding Layer
Bert Tokenizer & Bert
Analyze the event order: is this sequence normal or anomalous?
Sequentially Repaired Event Logs
Preprocessing
Event logs
Event logs with a moderate amount of labeled identical timestamp errors

Fig. 3. Model architecture diagram.

The ranking component leverages the successor event probability distribution concept proposed by Conforti *et al.* [3] to generate multiple candidate sequence arrangements. As established in Section 2.1, successor event confidence scores are derived based on direct follow frequencies of activities within the training set, where $\#(x, y)$ denotes the frequency of y directly following x, and Γ is the set of all activities. The confidence function $\Phi(x, y)$ computes the probability that y directly follows x:

$$\Phi(x, y) \triangleq \frac{\#(x,y)}{\sum_{z\in\Gamma} \#(x,z)} \tag{1}$$

Events $e_1, e_2, \ldots, e_n$ within the same process case $C(e_i) = C(e_j)$ and sharing an identical timestamp $T(e_x) = T(e_y)$ for any pair e_i and e_j are termed temporally equivalent. Such events within temporally equivalent groups ω. For each group ω, our selection component outputs a ranked sequence along with its corresponding confidence score. This involves enumerating all possible permutations of the events within ω and calculating the average confidence score S for each permutation:

$$S = \frac{\sum_{1\le i<j\le n} \Phi(e_i, e_j)}{n} \tag{2}$$

The selection component employs a large language model-based log analysis framework to score candidate sequences and identify the optimal ordering. Building upon the three-tier architecture from Ref. [14], the output was modified from binary (normal/abnormal) classification to sequence scoring, with sequential analysis capabilities enhanced through prompt engineering and data structure optimization. It comprises three core modules: Bidirectional Encoder Representations from Transformers (BERT) extracts semantic features from log sequences; the Projector aligns feature representations across models; and Llama generates sequence validity scores, forming an end-to-end processing pipeline. Crucially, the architecture uses only a single BERT instance and single projector.

The semantic feature extraction layer employs a BERT-base model as its core encoder. This 12-layer Transformer architecture with 768-dimensional hidden units converts raw log messages into structured semantic vectors. Specifically, input logs are tokenized and processed through multi-head self-attention mechanisms to generate context-aware representations. The 768-dimensional vector corresponding to the [CLS] token, which is a special token used for classification tasks, is extracted as the global feature, undergoing linear transformation and Tanh activation to output normalized semantic vectors—creating a unique "semantic fingerprint" for each event log.

The cross-model adaptation layer implements feature space transformation via a single linear projection layer. This component functions as a semantic translator through its 768×4096 weight matrix, linearly mapping BERT's 768-dimensional vectors to Llama's required 4096-dimensional input format while preserving semantic integrity.

The multimodal reasoning layer builds a prompt-driven classifier using Llama-3-8B. Its input template follows a three-segment structure: (1) instructional preamble ("Analyze the event order:"), (2) projected feature vectors, and (3) decision prompt ("Is this sequence abnormal?"). Through autoregressive generation, the model captures dependencies between log events and computes sequence normality scores based on token generation probabilities.

LESR operates on an event log $L = \{C_1, C_2, \ldots, C_m\}$, where each case C_j is a timestamped sequence of events. The framework adopts a two-stage design: in the first stage, for each subset of events $E' \subseteq C$ with identical timestamps, all possible permutations $P(E')$ are enumerated and ranked by a probabilistic model to obtain scores $S(p)$, and the top-K candidates are retained. In the second stage, a prompt-engineered and fine-tuned large language model (Llama-3-8B) computes semantic plausibility scores $P(p)$ for each candidate, which are combined with probabilistic scores into a unified criterion $F(p)=\alpha \cdot S(p)+\beta \cdot P(p)$. The highest-scoring candidate sequence is used to repair E', followed by timestamp correction to produce the final repaired log L^*. This design integrates candidate generation and semantic evaluation into a cohesive framework for fine-grained repair of complex ordering errors. The overall repair workflow of LESR is summarized in Algorithm 1.

Algorithm 1. LESR Framework for Identical Timestamp Error Repair

```
1:  Input: Event log $L = \{C_1, C_2, \ldots, C_m\}$, each case $C_j = \{E_1, E_2, \ldots, E_n\}$ with timestamps $\{t_1, t_2, \ldots, t_n\}$.
2:  Initialize repaired log $L^* \leftarrow \emptyset$.
3:  for each case C in L:
4:      Identify subsets $E' \subseteq C$ with identical timestamp errors.
5:      # Stage 1: Candidate Generation (probabilistic ranking).
6:      Generate permutations P(E') of events in E'.
7:      for each $p \in P(E')$:
8:          Compute ranking score S(p) using probabilistic model.
9:      end for
10:     Select top-K candidates $R = \{(p_1, s_1), \ldots, (p_1, s_k)\}$.
11:     # Stage 2: Candidate Selection (semantic evaluation with LLM).
12:     for each (p, s) in R:
13:         Construct prompt Q(p, C) with three segments (instructional preamble, feature vectors,
    decision prompt).
14:         Query fine-tuned LLM (Llama-3-8B) with Q(p, C) to obtain autoregressive token
    probabilities.
15:         Convert token probabilities into continuous semantic plausibility score P(p).
16:         Compute final score $F(p) = \alpha \cdot s + \beta \cdot P(p)$.
17:     end for
18:     Choose best candidate p* = argmax_{p $\in$ R} F(p).
19:     Replace E' in C with repaired sequence p*.
20:     Add repaired case C into L*.
21: end for
22: Output: Final repaired event log L*.
```

2.3. Training

The training objective is to enable the selection component to output accurate normality scores for event sequences. This requires fine-tuning the model to accurately distinguish between normal and abnormal log orderings.

Supervised learning was employed, requiring annotated normal and abnormal samples for training. During data preprocessing, a preliminary processing of the training set was performed for fine-tuning. To ensure method generalizability and prevent class imbalance from biasing model training, minority classes were oversampled to maintain a minimum proportion of β. Given current minority proportion α and total Sample_num, the minority class must be expanded to a target size of N_{targe} :

$$N_{target} = \frac{\beta(1-\alpha)}{1-\beta} \times \text{Sample\_num} \tag{3}$$

Three-stage training was implemented for the selection component's tripartite architecture. First, Llama was optimized for instruction responsiveness via few-shot learning to adapt it to discriminative queries: "Analyze the event order: is this sequence normal or anomalous?"—specifically designed to enhance sequence sensitivity. Next, the embedding modules (BERT and projector) for log messages were trained to project each log into Llama's optimal token embedding space, enabling anomaly detection. Finally, end-to-end fine-tuning ensured synchronized operation across all three tiers. Quantized Low-Rank Adaptation (QLoRA) reduced memory overhead by backpropagating gradients through frozen 4-bit quantized models while maintaining 16-bit full-precision performance levels.

2.4. Sequence Repair

After the ranking component generates confidence scores for successor events and the selection component completes fine-tuning, the repair can be performed on event logs containing identical timestamp errors. Fig. 4 introduces the entire workflow of LESR solving event log sequence repair.

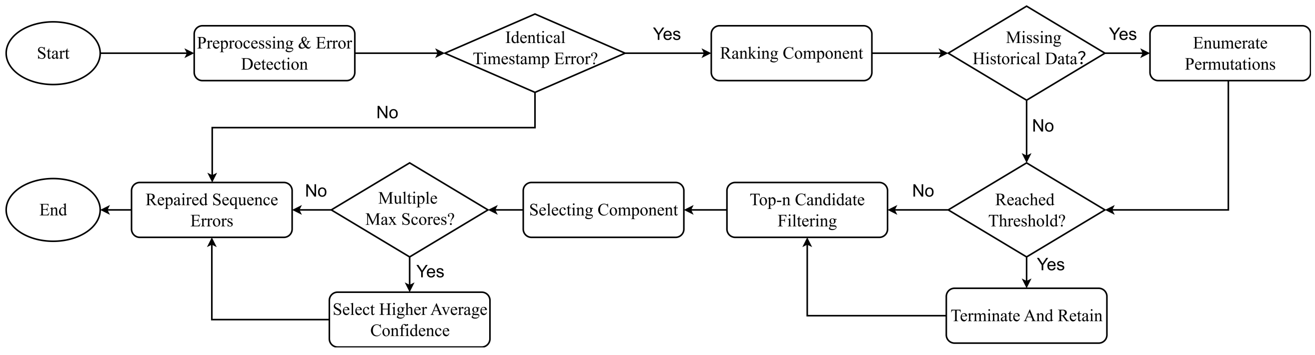


Fig. 4. Event log sequence repair process.

The event logs first undergo preprocessing and error detection to identify identical timestamp errors. They are then processed by the ranking component to generate sequencing schemes with their corresponding average confidence scores. To address the limitations of this approach, the following strategies are implemented: For scenarios with insufficient historical data: Retain the original sequence and enumerate permutations, ensuring the candidate set includes the baseline reference. For combinatorial explosion scenarios: Set a maximum enumeration threshold m = 50 (based on experimental equipment). When the count of generated permutations reaches m, activate a truncation strategy to terminate further generation and retain all existing sequences.

Finally, the selection component identifies the optimal sequencing scheme from the candidates. To optimize processing time, this stage employs a Top-n candidate filtering strategy, where only the top n sequences by average confidence proceed to the selection component. We set parameter $n = 5$, which achieves satisfactory repair results while significantly reducing repair time. The top n sequencing schemes (ranked by average confidence) are input into the selection component. The fine-tuned LLM scores the event orderings, and the scheme with the highest score is selected. If multiple schemes share the highest score, the result with the higher average confidence is chosen.

3. Experiments

3.1. Dataset

To evaluate these criteria, five real-world and artificially generated event logs (comprising 15 variants in total) were repaired using a prototype tool: E-commerce Order Processing Log (EcommOrder) [5]: Generated from a Colored Petri Net performance simulation model, this log captures the business process lifecycle of order handling. Environmental Permit Application Log (EnvPermit) [15]: Empirical data from the Dutch municipal service standardization study, recording operational traces of building permit application intake procedures at a city council. Workshop Scheduling Log (WorkshopSched) [16]: Operational records from a multi-agent modeled workshop scheduling simulation, detailing event sequences (e.g., task initiation, completion, agent interactions) generated by agents representing machines, orders, etc. Business Process Intelligence Challenge 2020 Log (BPIC20) [17]: Documents travel expense reimbursement processes at a university, covering domestic/international claim workflows. Healthcare Process Log (HealthProc) [18]: Generated via Colored Petri Net simulation of medical workflows, tracing therapeutic activity sequences. Collectively representing e-commerce, public administration, manufacturing, education, and healthcare domains, these logs ensure the method's practical applicability across diverse sectors.

3.2. Experimental Setup

This section details the fine-tuning configuration and hardware environment. The selection component was fine-tuned over 5 epochs across four stages: Llama warm-up, projector training, joint projector-BERT training, and full architecture fine-tuning (1, 1, 1, and 2 epochs, respectively). Training used AdamW with learning rates from 5×10^{-4} to 5×10^{-5}, an effective batch size of 16, and QLoRA for efficiency. Fine-tuning proceeded at a rate of ~1 hour per 10,000 sequences. All experiments were conducted on a server with the configuration specified in Table 1.

Table 1. System Configuration

Category	Component	Specification
Hardware	CPU	Intel Xeon Platinum 8481C (25 vCPU)
	GPU	1 × vGPU (32GB)
Software	OS	Ubuntu 20.04
	Python	3.8
	PyTorch	2.0.0

3.3. Benchmark Methods

To validate the superiority of our method over existing techniques, we compare its accuracy with relevant baseline methods. Multiple existing log repair solutions address timestamp quality problems.

The probability distribution-based method by Conforti *et al.* [3] and the GANs approach by Schmid *et al.* [6] demonstrate effectiveness in sequence repair and represent, to the best of our knowledge, the only two existing methods targeting identical timestamp error correction, as such errors typically involve multiple consecutive ordering faults and are thus better addressed by these approaches than by general ordering repair methods. These are thus adopted as comparable solutions. Additionally, two methods are introduced: Random-Sort: Randomly reorders erroneous activity sequences. No-Sort: Retains original faulty event ordering as a accuracy evaluation baseline.

Additionally, to provide comprehensive insights into methodological development, we incorporate intermediate results from the design phase into benchmark comparisons. Within our framework, the LLM undergoes fine-tuning despite its inherent advanced Natural Language Processing capabilities. To validate the effectiveness of this fine-tuning, results from the non-fine-tuned LLM variant are included as a comparative baseline. Furthermore, a version where the Llama-3B model directly performs sequencing without the ranking component is benchmarked, demonstrating the necessity of the proposed ranking architecture.

For timestamp correction, our objective is to demonstrate that sequence repair improves subsequent timestamp repair. Consequently, the comparison here focuses not on methodological superiority but on evaluating how the upstream-repaired event logs (with corrected sequences) influence timestamp repair outcomes. Thus, we exclusively employ the timestamp correction method by Schmid *et al.* [6]—the current state-of-the-art approach for this task—to perform timestamp repair across all compared sequence-repaired logs.

3.4. Evaluation Results

Table 2 presents the evaluation results of LESR against baseline methods across 15 variants of five public datasets, where identical timestamp errors were injected at rates of 10%, 30%, and 50%. The evaluation metrics include: Position Recovery Accuracy for Repaired Events (PREP), which measures the proportion of events correctly placed in their order after repair (higher is better); Perfectly Repaired Sequence Accuracy (PRSA), which evaluates the percentage of cases whose entire sequence is repaired without any error (higher is better); and Average Edit Distance (AED), which quantifies the average number of edits required to transform a repaired sequence into the ground truth (lower is better).

Table 2. Results of the Experimental Evaluation

Event log	Error rate	Metrics	LESR-Sort	Conforti-Sort	GAN-Sort	Random-Sort	No-Sort
EcommOrder	10%	PREP	**85.49%**	81.70%	63.93%	26.90%	27.31%
		PRSA	**80.33%**	72.67%	54.33%	17.33%	15.67%
		AED	**0.48**	0.64	1.30	3.07	3.08
	30%	PREP	**82.02%**	76.11%	61.88%	23.99%	23.15%
		PRSA	**75.67%**	65.00%	50.25%	6.33%	8.00%
		AED	**0.58**	0.83	1.31	3.66	3.71
	50%	PREP	**80.52%**	74.39%	57.83%	20.71%	20.97%
		PRSA	**72.67%**	59.67%	48.33%	6.67%	6.67%
		AED	**0.67**	0.99	1.39	4.56	4.42
EnvPermit	10%	PREP	**74.66%**	57.17%	46.54%	24.81%	27.90%
		PRSA	**74.62%**	63.08%	55.39%	14.62%	13.08%
		AED	**0.75**	1.25	1.51	2.62	2.53
	30%	PREP	**75.98%**	54.22%	43.52%	23.64%	23.08%
		PRSA	**74.74%**	60.77%	42.69%	12.31%	11.54%
		AED	**0.70**	1.33	1.62	2.98	2.64
	50%	PREP	**71.12%**	51.59%	42.95%	19.73%	21.20%
		PRSA	**65.38%**	49.23%	41.75%	3.85%	4.62%
		AED	**0.98**	1.58	1.78	3.34	3.15
WorkshopSched	10%	PREP	**70.39%**	69.09%	68.64%	35.89%	34.85%
		PRSA	**40.77%**	40.77%	40.00%	13.08%	11.54%
		AED	**1.32**	1.39	1.41	2.65	2.78
	30%	PREP	**71.24%**	67.81%	65.30%	36.26%	35.09%
		PRSA	**42.31%**	31.54%	29.23%	7.54%	6.85%
		AED	**1.36**	1.60	1.68	3.01	3.08

	50%	PREP	**70.00%**	65.80%	64.66%	28.66%	33.98%
		PRSA	**26.15%**	19.23%	18.46%	3.05%	3.42%
		AED	**1.75**	2.05	2.12	3.82	3.73
BPIC20	10%	PREP	**97.04%**	66.24%	61.16%	27.95%	28.06%
		PRSA	**97.97%**	74.37%	70.95%	16.83%	19.05%
		AED	**0.07**	0.81	0.92	2.34	2.26
	30%	PREP	**96.88%**	63.23%	59.50%	28.12%	27.86%
		PRSA	**97.58%**	70.99%	65.22%	17.02%	17.02%
		AED	**0.10**	0.91	1.10	2.36	2.38
	50%	PREP	**95.87%**	61.85%	58.21%	24.14%	23.65%
		PRSA	**97.10%**	67.38%	65.20%	8.13%	8.13%
		AED	**0.12**	1.08	1.12	2.76	2.73
HealthProc	10%	PREP	**51.05%**	49.25%	40.23%	20.66%	20.41%
		PRSA	**27.10%**	25.80%	7.84%	8.20%	8.10%
		AED	**2.11**	2.16	3.30	3.62	3.66
	30%	PREP	**49.05%**	47.64%	38.76%	16.53%	15.64%
		PRSA	**10.60%**	9.71%	13.70%	0.10%	0.31%
		AED	**3.03**	3.08	3.62	5.09	5.08
	50%	PREP	**45.11%**	43.26%	37.14%	12.72%	12.8%
		PRSA	**9.67%**	9.16%	11.70%	0.00%	0.00%
		AED	**3.03**	3.09	3.67	5.82	5.92

Table 2 demonstrate that our method outperforms all baseline solutions across every metric. The results reveal two key observations: Inverse Relationship: Repair effectiveness for misordered events exhibits an inverse correlation with the identical timestamp error rate, regardless of the method employed. LESR Advantage: LESR delivers a more pronounced improvement in Average Edit Distance (AED) (averaging 34.84% over the next best method) compared to Position Recovery Accuracy (PREP) and Perfectly Repaired Sequence Accuracy (PRSA). This indicates that while the LESR-repaired sequence may not always position every event perfectly, it consistently produces a more logically coherent overall event order. Compared with Conforti *et al.* [3], which relies purely on probabilistic timing distributions and thus struggles when identical timestamp errors disrupt longer activity contexts, LESR benefits from semantic reasoning and maintains more consistent ordering even when multiple consecutive events are misaligned. Likewise, unlike the GAN-based approach by Schmid *et al.* [6], which tends to underperform on discrete activity sequences due to its focus on continuous timestamp generation, LESR effectively captures the logical dependencies between activities. This explains why LESR achieves larger gains in AED—reflecting global sequence coherence—than in PREP or PRSA, as it excels at reconstructing a broad sequence order that aligns more closely with the ground truth.

Fig. 5 further analyzes accuracy across logs with varying sequence lengths. The left panel reports AED, while the right panel shows PRE. It confirms that LESR consistently outperforms alternatives, with particularly strong effectiveness on shorter sequences. This explains the notably high repair success rate observed for the BPIC20 dataset in Table 2. The accuracy arises partly from our Top-n candidate selection strategy (n = 5) implemented in Section 2.4.

The repair of k events with identical timestamps theoretically requires O(k!) time for full permutation evaluation, which becomes computationally infeasible for large k. To address this, a truncation strategy (m = 50) is introduced, reducing the candidate generation complexity to O(1). Given that LLM inference constitutes the primary computational bottleneck, the Top-N strategy is employed to restrict the number of candidate sequences for evaluation to n. This design substantially reduces runtime while preserving repair accuracy. As shown in Fig. 6, which illustrates the influence of the Top-N strategy on the repair accuracy of the BPIC20 dataset under a 50% error rate, performance reaches a threshold at n = 5. Beyond this point, additional computational investment yields diminishing returns, thereby validating the effectiveness of the proposed design.

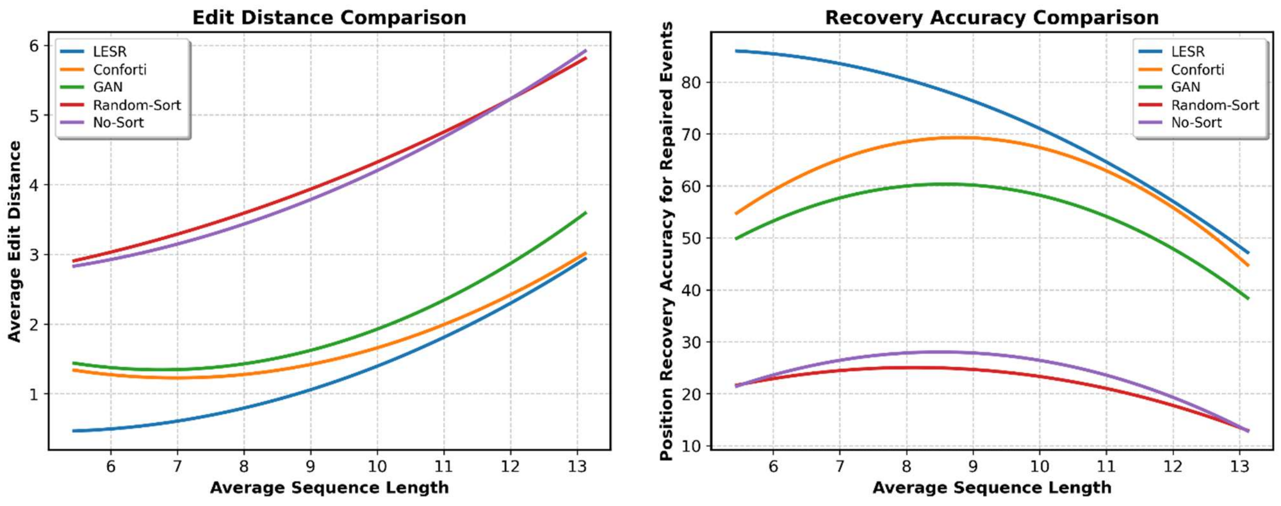


Fig. 5. Comparison of log repair method accuracies by sequence length.

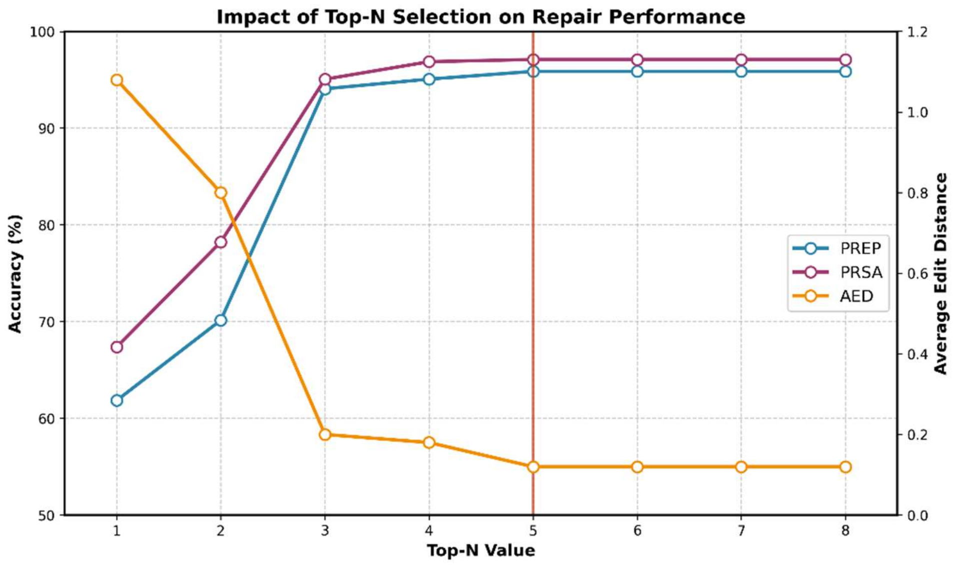


Fig. 6. Impact of Top-N candidate selection on repair performance.

Fig. 7 compares LESR with two scenarios—lacking fine-tuning and using only the large model for repair sequencing—using the Precision of Repair Event Position (PREP) metric across different error rates. The results confirm that every functional element in sequential repair module is indispensable. Without fine-tuning, the large model neglects sequence constraints during anomaly judgment and lacks domain-specific knowledge. Using only the large model for repair sequencing yields inferior results, particularly on more complex datasets.

Fig. 8 presents the results of applying existing timestamp repair methods to event logs after they have been repaired using different sequential repair methods. Accuracy is evaluated using Mean Absolute Error in Days, which measures the average absolute deviation between repaired and ground-truth timestamp, and Root Mean Squared Error in Days, which emphasizes larger deviations by averaging squared difference. This approach completes the closed loop for timestamp repair within the same sequence framework. The results confirm that high-quality sequential repair positively impacts timestamp repair accuracy, and our method provides a more favorable context for timestamp restoration.

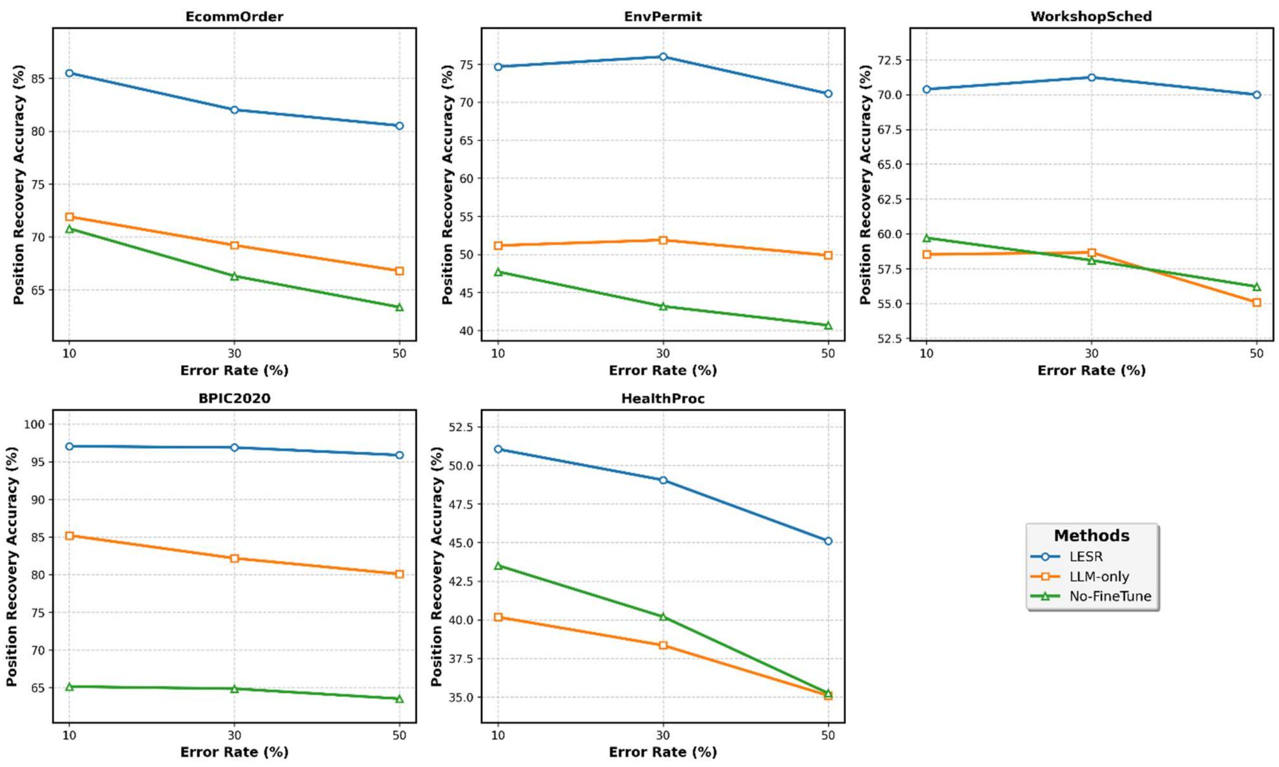


Fig. 7. Comparison of effectiveness in reordering sequences of events.

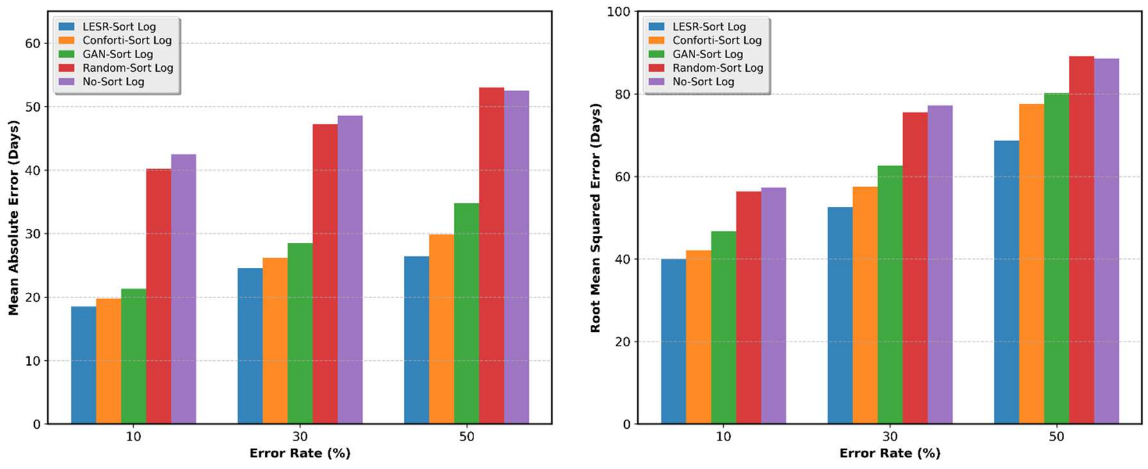


Fig. 8. Effectiveness comparison of timestamp repair under different repaired dataset.

3.5. Generality Analysis

To assess the comprehensiveness of our approach, we conducted repairs on a total of five real-world and synthetic event logs, comprising 15 distinct variants. These logs span multiple domains such as e-commerce, healthcare, and public administration, with error rates varying from 10% to 50%. Across all datasets and error configurations, LESR consistently outperforms baseline methods. This robust performance is attributed to its capacity to leverage semantic reasoning for capturing contextual dependencies among activities, resulting in more coherent and reliable sequence reconstruction.

It should be noted that our method is constrained by its reliance on the presence of three essential attributes in the event log: a case identifier, an activity label, and a timestamp. This design is aligned with conventional eXtensible Event Stream (XES)-based logging practices. However, a number of emerging standards, such as Object-Centric Event Logs (OCEL), eXtensible Object-Centric event logs, or Parquet, support multiple case notions or object-centric representations [19], which do not conform to the single-case-per-event assumption. Since these formats lack a singular case identifier, which prevents the direct application of our sequence repair mechanism, they currently lie beyond the scope of our approach. Nonetheless, this limitation does not significantly affect the general applicability of our method. Firstly, the XES format remains the de facto standard in both academic and industrial process mining, a status further reinforced by its IEEE standardization since 2014 [19]. Secondly, our method focuses primarily on reconstructing event sequences rather than detecting errors. It therefore remains broadly effective for logs with non-standard structures, provided they can be transformed into a compatible format through attribute alignment or semantic enrichment, as the sole prerequisite is the establishment of a coherent event sequence to repair. This approach to conversion is directly applicable to emerging formats. Consider an OCEL log: a pragmatic adaptation strategy involves converting it into standard XES format by treating each object instance as a separate case. While this simplification discards the inter-object relationships, it successfully reconstructs the linear sequences for individual objects, to which our repair technique can then be effectively applied. Thus, LESR maintains broad applicability for the vast majority of conventional event logs, ensuring its practical utility across a wide range of real-world scenarios.

4. Conclusion

High-quality input data is critical for successful process mining. This study proposes an LLM-based method for repairing identical timestamp errors in event logs to enable effective process mining analysis. An innovative sequence repair method is introduced as the core contribution and completes the end-to-end repair framework for reparing identical timestamp errors better. The solution was validated through a software prototype on 5 real-world and simulated event logs (comprising 15 variants in total). Experiment results demonstrate that our method outperforms baseline methods in effectiveness and generalizability, while also enhancing subsequent timestamp correction accuracy. The integration of LESR's ranking and selection components achieves high-quality results while avoiding the computational overhead of full permutation enumeration. Critically, fine-tuning enables the LLM to adapt to domain-specific characteristics across diverse event logs. Our approach demonstrates significant advantages in repairing identical timestamp errors, establishing a novel pathway for addressing this challenge.

Nevertheless, this work has several limitations. First, identical timestamps are inherently assumed to indicate errors, implying that timestamp equivalence cannot occur naturally. This assumption constrains the applicability of the detection method. Second, the LESR framework specifically targets sequence repair within identical-timestamp errors, delegating timestamp correction to existing methods. Finally, despite near-realistic simulations, real-world deployment validation remains pending. These limitations define critical avenues for future research.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Zhengxu Huang: Conceptualization, Investigation, Methodology, Investigation, Software, Writing-original draft preparation. Yishi Zhao: Conceptualization, Methodology, Investigation, Writing—review and editing, Funding acquisition. Jianga Shang: Validation, Writing—review and editing, Supervision. Ying Shi: Writing—review and editing, Funding acquisition. All authors had approved the final version.

Funding

This research was supported by the National Natural Science Foundation of China under grant no. 62072342.

Acknowledgment

We would like to thank our colleagues who participated in our research, as well as the editors for their valuable feedback.

References

[1] Van der Aalst, W. M. P. (2022). Process mining: A 360 degree overview. In W. M. P. van der Aalst & J. Carmona (Eds.), *Process Mining Handbook* (pp. 3–34). Cham: Springer International Publishing.

[2] Van der Aalst, W. (2016). Getting the data. In W. van der Aalst (Ed.), *Process Mining: Data Science in Action* (pp. 125–162). Berlin, Heidelberg: Springer.

[3] Conforti, R., La Rosa, M., ter Hofstede, A. H. M., & Augusto, A. (2020). Automatic repair of same-timestamp errors in business process event logs. In D. Fahland, C. Ghidini, J. Becker, & M. Dumas (Eds.), *Proceedings of the International Conference on Business Process Management* (pp. 327–345). Cham: Springer International Publishing.

[4] Suriadi, S., Andrews, R., ter Hofstede, A. H. M., & Wynn, M. T. (2017). Event log imper-fection patterns for process mining: towards a systematic approach to cleaning event logs. *Information Systems*, *64*: 132–150.

[5] Hompes, B. (2024). Synthetic event log with specific process performance characteristics. *4TU.ResearchData*. Retrieved July 2, 2025, from https://doi.org/10.4121/670bf51b-cb96-4c80-9c20-caff31b2f568.v1

[6] Schmid, S. J., Moder, L., Hofmann, P., & Röglinger, M. (2023). Everything at the proper time: Repairing identical timestamp errors in event logs with generative adversarial networks. *Information Systems*, *118*: 102246.

[7] Rogge-Solti, A., Mans, R. S., van der Aalst, W. M. P., & Weske, M. (2013). Improving documentation by repairing event logs. In J. Grabis, M. Kirikova, J. Zdravkovic, & J. Stirna (Eds.), *Proceedings of Ifip Working Conference on the Practice of Enterprise Modeling* (pp. 129–144). Berlin, Heidelberg: Springer.

[8] Dixit, P. M., Suriadi, S., Andrews, R., Wynn, M. T., ter Hofstede, A. H. M., Buijs, J. C. A. M., & v-an der Aalst, W. M. P. (2018). Detection and interactive repair of event ordering imperfection in process logs. In J. Krogstie & H. A. Reijers (Eds.), *Advanced Information Systems Engineering* (pp. 274–290). Cham: Springer International Publishing.

[9] Shahzadi, S., Fang, X., Shahzad, U., Ahmad, I., & Benedict, T. (2022). Repairing event logs to enhance the performance of a process mining model. *Mathematical Problems in Engineering*, *2022(1)*: 4741232.

[10] Sani, M. F., van Zelst, S. J., & van der Aalst, W. M. P. (2018). Repairing outlier behaviour in event logs. In W. Abramowicz and A. Paschke (Eds.), *Business Information Systems* (pp. 115–131). Cham: Springer International Publishing.

[11] Song, W., Jacobsen, H.-A., & Zhang, P. (2021). Self-healing event logs. *IEEE Transactions on Knowledge and Data Engineering*, *33(6)*, 2750–2763.

[12] Leemans, S. J. J., Fahland, D., & van der Aalst, W. M. P. (2013). Discovering block-structured process models from event logs—A constructive approach. In J.-M. Colom and J. Desel (Eds.), *Processing of International Conference on Application and Theory of Petri Nets and Concurrency* (pp. 311–329). Berlin, Heidelberg: Springer.

[13] Le, V.-H., & Zhang, H. (2023). Log parsing with prompt-based few-shot learning. *Proceedings of 2023*

IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 2438–2449).

[14] Guan, W., Cao, J., Qian, S., Gao, J., & Ouyang, C. (2024). LogLLM: Log-based anomaly detection using large language models. arXiv Preprint, arXiv:2411.08561.

[15] Buijs, J. (2022). Receipt phase of an environmental permit application process (WABO), CoSeLoG project. *Eindhoven University of Technology.*

[16] Bemthuis, R. H., Koot, M., Mes, M. R. K., Bukhsh, F. A., Iacob, M.-E., & Meratnia, N. (2019). An agent-based process mining architecture for emergent behavior analysis. *Proceedings of 2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)* (pp. 54–64).

[17] van Dongen, B. (2020). BPI Challenge 2020. *4TU.Centre for Research Data*. Retrieved July 2, 2025, from https://doi.org/10.4121/uuid:52fb97d4-4588-43c9-9d04-3604d4613b51

[18] Mehr, A. M. (2023). Healthcare treatment process (logs and CPN model): Data underlying the publication "Identifying the context of data usage to diagnose privacy issues through Process Mining". *4TU.ResearchData*. Retrieved July 2, 2025, from https://doi.org/10.4121/8683fc1a-aca1-447b-aba4-8d7806a9977f.v1

[19] Ghahfarokhi, A. F., Park, G., Berti, A., & van der Aalst, W. M. P. (2021). OCEL: A standard for object-centric event logs. In L. Bellatreche, M. Dumas, P. Karras, R. Matulevičius, A. Awad, M. Weidlich, M. Ivanović, & O. Hartig (Eds.), *Proceedings of the European Conference on Advances in Database and Information Systems* (pp. 169–175). Cham: Springer International Publishing.