计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2025,34(8):217−227 [doi: 10.15888/j.cnki.csa.009911] [CSTR: 32024.14.csa.009911]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于分层 Transformer 的相同时间戳错误修复①

徐　猛[1], 谢　凯[1,2]

[1](安徽理工大学 数学与大数据学院, 淮南 232001)
[2](安徽理工大学 安徽省煤矿安全大数据分析与预警技术工程实验室, 淮南 232001)
通信作者: 谢　凯, E-mail: kxie158@yahoo.com.cn

摘　要: 在流程挖掘领域, 众多流程操作高度依赖于事件日志中精确的时间戳信息. 因此, 与时间戳相关的质量问题影响尤为显著, 特别是相同时间戳错误, 这种错误会引发误导性的流程见解, 进而造成严重的流程偏差. 现有研究在处理此类错误时, 缺乏对事件间长期依赖关系以及属性间潜在关联性的充分考量, 在一定程度上限制了相同时间戳错误的修复精度. 针对这一问题, 本文提出了一种基于分层 Transformer 模型修复相同时间戳错误的方法. 该方法通过分层信息传递结合多视角交互, 捕获事件间的长距离行为依赖以及属性间的深层关联信息, 逐层完成对错误事件重排序以及对应时间戳的预测任务, 继而实现对相同时间戳错误事件日志的有效修复. 通过 4 个公开可用的数据集进行评估, 结果表明, 所提方法能够有效提高相同时间戳错误的修复精度.

关键词: 事件日志; 相同时间戳错误; 分层 Transformer; 多视角交互; 日志修复

引用格式: 徐猛,谢凯.基于分层 Transformer 的相同时间戳错误修复.计算机系统应用,2025,34(8):217–227. http://www.c-s-a.org.cn/1003-3254/9911.html

Identical Timestamp Error Correction Based on Hierarchical Transformer

XU Meng[1], XIE Kai[1,2]

[1](College of Mathematics & Big Data, Anhui University of Science & Technology, Huainan 232001, China)
[2](Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, Anhui University of Science & Technology, Huainan 232001, China)

Abstract: In the field of process mining, numerous process operations are highly dependent on accurate timestamp information in event logs. Consequently, quality issues related to timestamps have particularly significant impacts, especially identical timestamp errors, which can lead to misleading process insights and subsequently result in severe process deviations. Existing research addressing such errors lacks sufficient consideration of long-term dependencies between events and potential correlations between attributes, limiting the accuracy of repairing identical timestamp errors. To address this issue, a method for repairing identical timestamp errors based on a hierarchical Transformer model is proposed. This method captures long-distance behavioral dependencies between events and deep correlation information between attributes through hierarchical information propagation combined with multi-view interaction. The tasks of reordering erroneous events and predicting corresponding timestamps are accomplished layer by layer, thus effectively repairing event logs with identical timestamp errors. Evaluations conducted on four publicly available datasets demonstrate that the proposed method can effectively improve the accuracy of repairing identical timestamp errors.

Key words: event log; identical timestamp error; hierarchical Transformer; multi-view interaction; log repair

① 基金项目: 安徽省重点研究与开发计划 (2022a05020005)
收稿时间: 2024-12-18; 修改时间: 2025-01-21; 采用时间: 2025-02-18; csa 在线出版时间: 2025-06-24
CNKI 网络首发时间: 2025-06-25

流程挖掘技术旨在解析事件日志, 并从中获取有价值的流程洞察, 以深度剖析业务流程[1]. 事件日志记录了实际业务流程的执行过程, 流程挖掘技术利用事件日志能够检测实际流程与预设流程模型之间的偏差, 评估实际流程的执行效能, 并揭示流程优化的潜在机会. 为生成高质量流程模型, 众多研究致力于开发各种自动化流程发现算法, 例如 Alpha 挖掘[2]、归纳挖掘[2]、模糊挖掘[3]以及启发式挖掘[4]等. 鉴于流程挖掘技术的普遍适用性和强大功能, 其在医疗、物流、金融、信息技术等领域被广泛应用, 并取得了显著成效.

尽管流程挖掘技术正在蓬勃发展并趋于成熟, 但高质量的事件日志仍是使其充分发挥潜力的关键所在. 当前主流流程挖掘技术大多预设了完整且准确的事件日志作为输入, 然而这在现实业务环境中往往难以保证. 由于业务环境灵活多变且流程复杂多样, 事件日志中常夹杂着不准确或受损的数据记录. 尽管事件日志存在诸多质量问题, 但由时间戳错误引发的质量问题尤为严重, 因为众多流程操作 (例如流程发现、一致性检验和流程优化) 都高度依赖于精确的时间戳信息[5].

一种常见的时间戳问题是“相同时间戳错误”, 即在同一流程实例中, 多个事件被错误地标记为同一时间戳[6]. 这种错误可能源于事件日志记录系统过载导致的延迟, 或是系统记录时间的粒度不够精细. 相同时间戳错误丢失了事件原本的执行顺序和事件间的时间间隔, 使用包含此类错误的事件日志进行流程分析很可能会导致原本依次执行的事件被解释为并发执行, 造成业务流程模型失真和误导性的流程见解, 从而阻碍后续流程分析的进程[7].

对于相同时间戳错误, 已有方法基于直接跟随关系修复案例事件的执行顺序, 依据概率密度函数或深度学习技术去填补错误的时间戳. 然而, 这些方法忽略了事件间的长距离行为依赖以及属性间的复杂关联信息对时间戳修复的影响. 鉴于此, 本文提出了一种基于分层 Transformer 的相同时间戳修复方法, 该方法从两个维度入手: 一是事件级行为视角, 用于实现案例事件的重新排序; 二是属性级数据视角, 用于预测并修复错误的时间戳.

本文的主要贡献如下.

(1) 提出基于分层 Transformer 的相同时间戳错误修复方法, 通过分层信息传递结合多视角交互, 实现错误事件重排序以及对应时间戳修复.

(2) 使用公开可用数据集进行仿真实验, 并对实验结果进行分析, 结果表明本文所提方法可以有效提高相同时间戳错误的修复精度.

1 相关工作

事件日志作为业务流程执行过程的载体, 众多研究致力于精确界定和阐述其质量问题. 在“过程挖掘宣言”[8]中率先提出了事件日志质量问题的概念, 并将事件日志划分为 5 个成熟度级别, 这一分类体系成为评估事件日志质量的重要标准. 文献[9]指出缺失、错误、不精确以及不相关数据是造成事件日志质量低下的主要原因, 并针对这些问题, 进一步提供了检测异常值的方法.

相同时间戳错误作为事件日志质量问题的一个方面, 其存在往往源于多种复杂因素. 文献[10]认为事件日志的低粒度是导致相同时间戳问题出现的一个关键因素. 文献[11]展示了一种基于表单的事件共享时间戳现象, 即由于系统采用电子表单记录活动执行过程, 导致所有事件错误地与表单提交时间的时间戳相关联. 此外, 文献[6]指出当日志系统面临过载时, 记录的事件可能会处于挂起状态. 一旦日志系统恢复可用, 这些挂起的事件会立即被处理, 从而引发多个事件共享同一时间戳的问题. 一旦相同时间戳错误发生, 日志中事件的正确顺序信息和事件间的时间间隔信息就会丢失. 正确的事件顺序信息对于流程发现等操作至关重要. 同时, 事件间的时间间隔也是分析流程效率、检测异常行为以及预测流程剩余时间不可或缺的数据. 相同时间戳错误不仅会导致这些关键信息的混乱与失真, 还可能进一步影响基于日志数据的决策支持的准确性和可靠性.

截至目前, 针对相同时间戳错误问题的修复研究相对匮乏, 主要基于统计学方法、机器学习技术以及深度学习算法来开展相关工作. 文献[6]基于直接跟随关系对错误事件进行重排序, 并依据概率密度函数采样填补错误时间戳提出了一种自动化修复方法. 然而, 这种方法未能充分捕捉业务流程中复杂的事件依赖关系, 同时也忽略了其他属性与时间戳之间的潜在关联. 文献[12]提出了一种使用自动编码器技术提高流程事件日志质量的方法, 修复错误的时间戳被作为该研究的重点子问题进行探讨. 该方法在人工事件日志上表现优异, 但在面对复杂事件日志时其适应性略显不足. 另外, 在修复具有相同时间戳的错误时, 该方法的效果

并不理想. 文献[7]通过引入 CGAN (条件生成对抗网络) 生成足以“以假乱真”的数据来修复相同时间戳错误, 但它仍然沿用了文献[10]中基于直接跟随关系进行事件重排序的方法. 此外, 该方法忽略了其他附加属性信息 (如执行资源) 对时间戳的影响. 文献[13]依据标注活动执行时间的流程参考模型, 设计了一种修复错误或缺失时间戳的方法. 但在实际业务环境中, 这种模型可能并不存在.

现有的时间戳修复方法存在两个方面的局限性: 首先, 这些方法未能充分捕获案例事件间的长距离行为依赖. 在真实业务场景中, 事件间的依赖关系往往不是简单、短期的, 而是复杂、长期的. 例如, 一个项目中的某个任务可能依赖于之前多个任务的完成情况, 而这些任务之间又可能存在相互依赖关系. 当前的修复策略普遍忽略了事件间错综复杂的行为联系, 仅考虑简单的直接跟随关系. 其次, 在处理属性间深层关联时也尚显不足. 在流程案例中, 每个事件都包含多个属性, 这些属性之间可能存在潜在的关联关系. 例如, 一个任务的完成时间可能与其优先级、所需资源以及执行者的能力等多个属性相关. 现有修复方法大多仅聚焦于单一属性对时间戳的影响, 而缺乏对多属性间深层关联性的洞察.

针对事件行为依赖捕获以及属性数据关联挖掘, 本文提出了一种基于自注意力机制的多视角分层 Transformer 模型来修复相同时间戳错误的方法.

2 基础知识

本节首先给出本文工作的相关定义, 然后介绍 Transformer 模型的基本知识.

2.1 基础定义

定义 1 (事件及其属性). 事件 E, 作为业务流程中活动 a 执行过程的记录, 可被形式化地表示为元组 E=(*CID*, *EID*, *Activity*, *Timestamp*, *Resource*, $P1,\cdots, Pm$), 其中, *CID* 是事件 E 所属业务流程实例的唯一标识; *EID* 为事件 E 在该流程实例中的唯一编号; *Activity* 表示事件 E 执行活动的具体名称; *Timestamp* 记录了事件 E 的执行时间戳; *Resource* 指执行活动 a 所需的资源; $P1$–Pm 是事件 E 包含的其他属性 (例如执行成本等)[1].

定义 2 (迹和事件日志). 迹σ是业务流程执行过程中一系列事件的有序排列, 可以被形式化的表示为元组σ=($E1$, $E2$, $E3,\cdots, En$), 其中 n 表示迹σ包含事件 E 的数量. 事件日志记录了业务流程的执行过程, 它包含所有观察到的迹. 在形式化表示中, 事件日志 L 可以定义为 L=($\sigma1$, $\sigma2$, $\sigma3,\cdots, \sigma|L|$), 其中, $|L|$表示事件日志 L 包含迹σ的数目[1].

表 1 呈现了一个事件日志的部分完整示例. 该示例涵盖了 2 个流程案例, 共包含 10 个事件, 以及流程案例、事件、活动、时间戳等 6 个属性标签.

表 1 事件日志部分示例

案例	事件	活动	时间戳	资源	成本 (元)
1	$E1$	Arrive at airport	1/9/2020 12:00:00	Tom	50
1	$E2$	Check in	1/9/2020 12:20:00	Jack	100
1	$E3$	Security check	1/9/2020 12:30:00	Thomas	150
1	$E4$	Boarding	1/9/2020 14:20:00	Lina	200
1	$E5$	Tack off	1/9/2020 15:00:00	James	300
2	$E6$	Arrive at airport	2/9/2020 20:00:00	Steven	70
2	$E7$	Check in	2/9/2020 20:20:00	Jack	120
2	$E8$	Security check	2/9/2020 20:30:00	Mark	210
2	$E9$	Priority boarding	2/9/2020 21:00:00	Lina	170
2	$E10$	Tack off	2/9/2020 21:30:00	Ethan	420

定义 3 (相同时间戳错误). 在给定事件日志 L 中, 若某一流程实例内部存在两个或多个事件, 它们被赋予完全相同的时间戳, 则视为发生了相同时间戳错误. 具体地, 若存在事件 EK 和 EL (其中, $K, L\in\{1, 2, 3,\cdots, n\}$ 且 $K\neq L$), 满足 CID_K=CID_L, 且 $Timestamp_K$=$Timestamp_L$, 则称在事件日志 L 中存在相同时间戳错误[6].

例如, 表 2 所示的相同时间戳错误事件日志示例中, 存在一组事件发生了相同时间戳错误. 具体而言, 事件 $E4$ 与 $E5$ 错误地共享了相同的时间戳. 同时表 2 增加了一列标识符, 存在相同时间戳错误时为 1, 否则为 0.

表 2 相同时间戳错误事件日志示例

案例	事件	活动	时间戳	资源	成本 (元)	标识符
1	$E1$	Arrive at airport	1/9/2020 12:00:20	Tom	50	0
1	$E2$	Check in	1/9/2020 12:20:40	Jack	100	0
1	$E3$	Security check	1/9/2020 12:30:00	Thomas	150	0
1	$E4$	Tack off	1/9/2020 14:20:05	James	300	1
1	$E5$	Boarding	1/9/2020 14:20:05	Linda	200	1
2	$E6$	Arrive at airport	2/9/2020 20:10:50	Steven	70	0
2	$E7$	Check in	2/9/2020 20:26:04	Jack	120	0
2	$E8$	Security check	2/9/2020 20:30:00	Mark	210	0
2	$E9$	Priority boarding	2/9/2020 21:00:00	Lina	170	0
2	$E10$	Tack off	2/9/2020 21:35:07	Ethan	420	0

2.2 Transformer 模型

Transformer[14]是一种基于自注意力机制的 Seq2Seq 模型, 因其高效准确处理序列数据的同时还能灵活应

对各种环境下的自然语言处理任务而备受关注.

自注意力机制作为 Transformer 模型的核心组件, 其强大的序列建模能力使得模型能够捕捉到序列中不同位置间的依赖关系, 从而更深入地理解上下文信息[15]. 自注意力机制还支持并行计算, 相较于传统的深度学习序列模型, 其更容易在 GPU 和 TPU 等硬件上进行训练. 此外, 由于自注意力机制能够并行地计算所有位置的权重得分, 因此它更能有效处理长距离依赖问题, 降低梯度消失或梯度爆炸的风险[16].

Transformer 模型结构如图 1 所示, 编码器和解码器是模型的基础构件. 编码器由多个相同层堆叠构建, 主要包含两个子层: 一是自注意力层, 用于捕捉序列中各单元间的复杂依赖关系; 二是前馈神经网络层, 用于增强模型的表达能力. 解码器结构与编码器相似, 但额外包含一个“掩蔽自注意力”层. 该层通过设置掩蔽矩阵, 确保在计算注意力权重时, 模型仅关注当前输出位置之前的序列信息, 从而保证生成的顺序性和自回归特性. 解码器中的第 2 个自注意力机制层则负责整合来自编码器的上下文信息, 以生成更精准的输出.

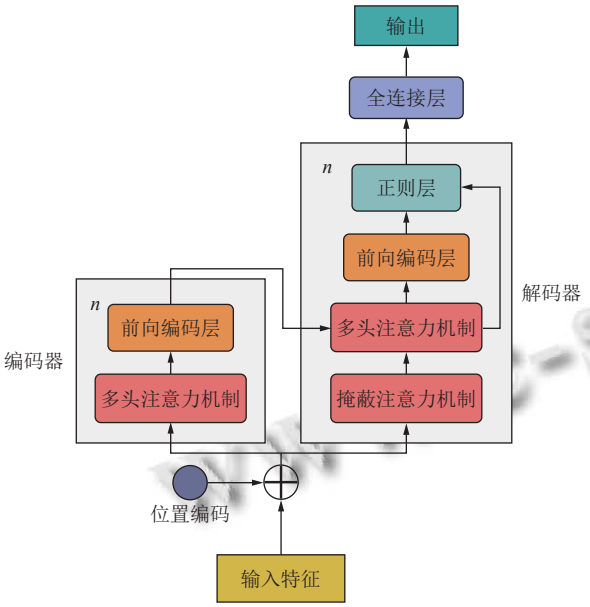


图 1 Transformer 模型架构

3 方法

本节介绍所提出的基于分层 Transformer 的相同时间戳错误修复方法. 首先, 鉴于相同时间戳错误可能引发时序错误等问题, 结合相关形式化定义, 本文方法将相同时间戳错误修复问题拆分为多目标规划. 而后, 基于多目标规划提出了分层 Transformer 的修复方法, 用于确定流程案例 (迹) 中事件之间的正确顺序并据此修复迹, 以及为重新排序的事件预测时间戳.

本文的相同时间戳修复方法, 主要包含以下几个步骤: (a) 将修复相同时间戳的问题拆分为多个目标设计. 通过预处理, 将事件日志划分为训练集和测试集, 完成事件属性的编码嵌入; (b) 构建行为视角 Transformer 模块, 学习事件间行为依赖关系及上下文语义信息; (c) 基于提取的事件行为关系和上下文语义信息, 对相同时间戳事件进行重排序处理; (d) 构建数据视角 Transformer 模块, 挖掘属性间数据的深层关联性; (e) 根据获取的行为特征以及数据关联, 实现对应时间戳的精准预测. 修复相同时间戳错误方法的框架如图 2 所示.

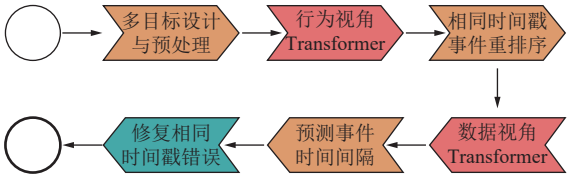


图 2 修复相同时间戳错误方法框架

3.1 多目标设计与预处理

● 多目标设计. 鉴于相同时间戳错误可能引发上的事件乱序以及多事件共享同一时间戳等问题, 本文结合 CGAN 方法[6], 将相同时间戳错误的修复任务拆分为 3 个设计目标.

目标 1: 相同时间戳错误事件检测;

目标 2: 相同时间戳错误事件重排序;

目标 3: 相同时间戳错误事件的时间戳预测.

本文将目标 1 视为数据预处理阶段中的一环, 并采用系统化的事件日志清理方法[17]检测相同时间戳. 鉴于错误检测方法相对较为直接, 且修复相同时间戳问题存在尚未解决的难题. 本文的研究聚焦于解决修复层面的关键问题, 即目标 2 和目标 3. 同时为了更加有效地应对相同时间戳错误修复, 本文根据定义 4 将事件时间戳转化为事件时间间隔, 并进行最大最小归一化处理.

定义 4 (事件间时间间隔). 在同一个流程实例 (迹) σ 中, 事件间的时间间隔 d_i 为当前事件的时间戳与上一个事件的时间戳之差. 即, 当 CID_i=CID_{i-1} 时, d_i=$Timestamps_i$–$Timestamps_{i-1}$ (当 $i>1$ 时). 对于某案例标识符的第 1 个事件, 事件间的时间间隔是未定义的. 在

这种情况下, 本文使用流程案例内事件间的平均时间间隔作为近似值[7].

● 数据预处理. 对于既定事件日志, 本文依据 80%与 20% 的比例将其划分为训练集和测试集, 分别用于模型训练和模型验证. 完成数据集划分后, 将数据集输入转换为向量形式嵌入模型训练, 采用 token embedding +segment embedding+position embedding 的编码方式. 分层 Transformer 模型编码结构如图 3 所示.

Token embedding 使用开始[CLS]和结束[SEP]标记将输入序列分割成文本段. Segment embedding 通过标准词嵌入方法为序列中的不同部分提供表示. Position embedding 则为每个位置初始化向量, 以确保输入文本编码序列的顺序性. 这 3 种编码嵌入的向量形式均为 (1, n, 128), 最终通过对应元素相加得到模型的嵌入编码.

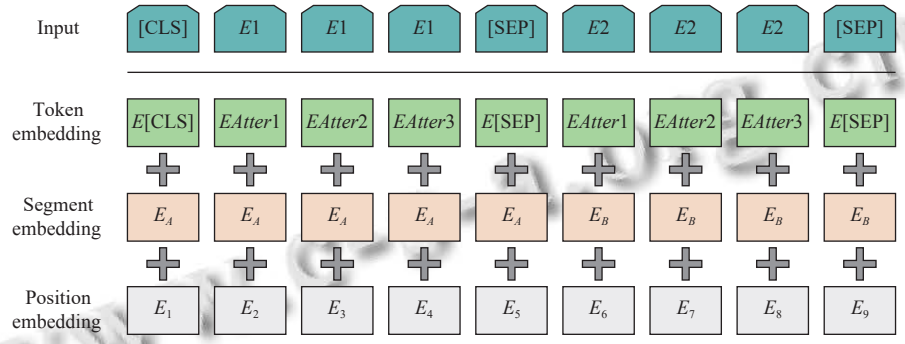


图 3 分层 Transformer 模型编码结构

3.2 修复相同时间戳

所提出的基于分层 Transformer 的相同时间戳错误修复方法, 其模型架构如图 4 所示.

该模型采用双层 Transformer 设计, 旨在从两个不同视角即事件行为与属性数据视角, 深入挖掘事件间的行为依赖关系以及属性间的潜在关联信息. 通过行为和数据视角的协同交互, 以实现对相同时间戳错误的有效修复.

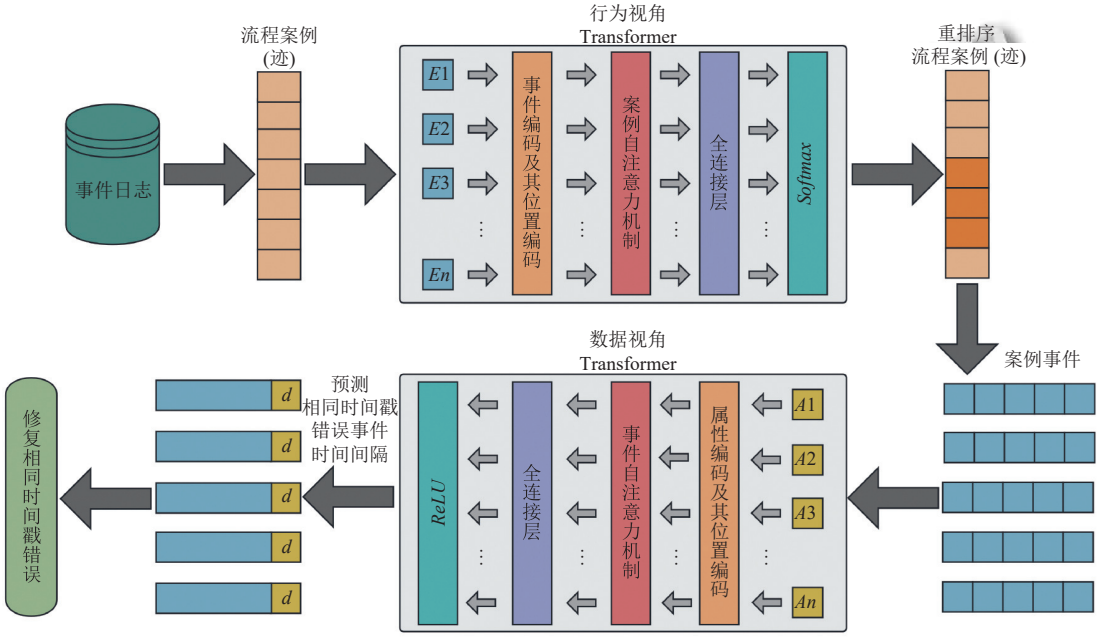


图 4 分层 Transformer 模型架构

3.2.1 相同时间戳错误事件重排序

解决因相同时间戳错误引发的事件乱序问题的关键在于精确预测事件在其所属迹中的正确位置. 文献[18]提出了一种基于 BERT 的多视角修复事件日志

中缺失值的方法, 该方法通过结合数据视角和行为视角并采用迁移学习的方式来预测缺失位置的事件. 然而这种方法更多是从整体或宏观层面分析事件日志中的缺失事件位置. 相比之下, 修复迹内事件乱序的问题, 则要求从更为微观的层面专注于流程案例内部对应位置的事件预测. 因此本文针对这一问题, 在第 1 层行为视角 Transformer 模型引入 MEM (masked event model) 预训练任务, 应用 Mask 掩码技术掩蔽同一案例中连续两个或多个事件, 然后通过预测这些被掩蔽的连续位置所对应的事件, 从而达到预测正确事件顺序的目的.

在数据输入层, 模型融合了 3 个嵌入特征 (文本信息、语义信息、位置信息) 对流程案例中的事件进行编码, 生成相应的向量表示. 同时, 采用一个双向 Transformer 架构, 考虑信息的正向与反向流动. 前向视角洞察事件演变脉络, 后向视角校验逻辑一致性, 捕获全局依赖关系.

作为行为视角 Transformer 模型的核心, 案例自注意力机制 (如图 5 所示) 通过多头注意力 (multi-head attention) 来同时关注输入案例事件间的不同子空间, 从而捕捉更为丰富的行为特征. 每个注意力头通过缩放点积 (scaled dot-product) 来计算输入的注意力权重, 并将这些计算结果拼接起来, 形成最终的注意力输出. 计算公式如式 (1) 所示:

$$MultiHead(Q,K,V)=Concat(head_1,\cdots,head_h)W_0$$
$$head_i=Attention(Q_i,K_i,V_i)=Softmax\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V \tag{1}$$

其中, $head_i$代表一个注意力头, h 为注意力头的个数, 通过缩放点积求得. d_k是Q和K向量的维度, Q为输入数据经过嵌入层后得到的向量, K和V是编码器的输出向量.

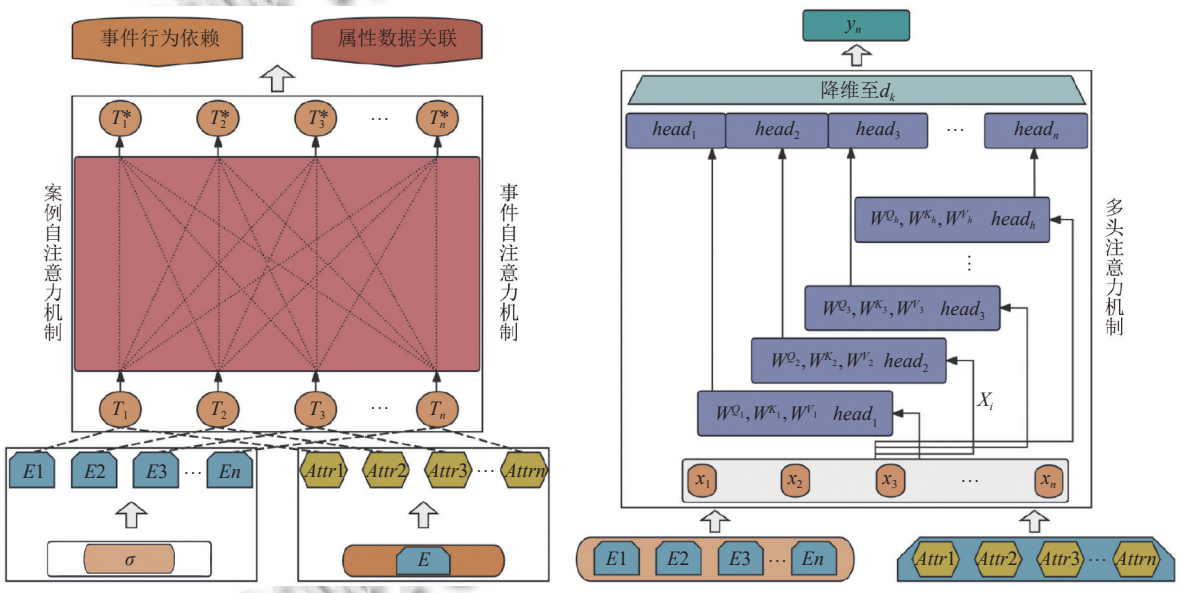


图 5 自注意力机制模块

本文将预测正确的事件顺序视为多分类问题, 作为第 1 层 Transformer 模型的最终任务. 为了将模型输出转化为可操作的预测结果, 输出层首先将获取到的所有向量表示输入到全连接层中, 全连接层输出向量包含输入序列的事件的文本信息、完整上下文语义以及事件间行为依赖信息. 全连接层再将信息传递给使用 *Softmax* 激活函数的密集层, 用于计算当前位置事件执行的概率分布. *Softmax* 激活函数定义如式 (2):

$$Softmax(x_i)=\frac{\exp(x_i)}{\sum_j \exp(x_i)} \tag{2}$$

其中, x_i代表当前位置的执行事件, j 表示事件数.

依据 *Softmax* 密集层输出的当前位置执行事件的概率分布, 选择概率最高的事件执行顺序来实现相同时间戳错误事件重排序. 此外, 行为视角 Transformer 模型采用反向传播算法来优化其可训练参数, 并选用交叉熵作为损失函数. 第 1 层 Transformer 编码器和解码器层数 N 均为 6, 头的个数 h 为 8.

3.2.2 修复相同时间戳

第 2 层数据视角 Transformer 模型针对相同时间戳错误事件的时间戳修正问题设计, 与第 1 层行为视

角 Transformer 不同的是, 其采用 MTM (masked time-stamp model) 预训练任务. 模型利用 Mask 掩码技术, 掩蔽了重排序事件中, 根据定义 4 所转换的、以秒为单位的归一化事件时间间隔. 通过预测以秒为单位的事件时间间隔 *di*, 将累积预测的事件时间间隔, 逐一添加到其前驱事件的时间戳上, 继而实现对相同时间戳的有效修复.

在数据嵌入层, 第 2 层 Transformer 不仅对流程案例中的事件属性文本、语义以及位置信息编码, 还融合了来自第 1 层行为视角 Transformer 的事件行为特征编码信息. 同样采用双向 Transformer 架构, 考虑信息的正反向流动, 进一步增强模型处理复杂数据结构和挖掘深层次语义关系的能力.

事件自注意力机制构成了数据视角 Transformer 模型的核心组件 (如图 5 所示), 其同样借助多头注意力机制, 能够并行聚焦输入事件属性间的多元子空间, 进而深度挖掘属性数据间的潜在关联. 每个注意力头运用缩放点积来计算输入的注意力权重, 并将这些独立的计算结果进行拼接融合, 生成最终的注意力输出. 计算公式如式 (1) 所示.

本文将事件时间间隔的预测视为时间序列回归问题, 并将其作为分层 Transformer 架构中第 2 层模型的最终处理目标. 在输出层级, 所获取的向量表征被输入至一个全连接层网络中, 该网络结构由两个线性变换模块构成. 利用事件自注意力机制的输出结果, 通过非线性变换深度挖掘输入属性特征之间的复杂交互关系. 同时, 线性变换的引入使得该层能够学习输入序列中不同位置特征的不同权重分配. 全连接层再将信息传递给使用 *ReLU* (rectified linear unit) 激活函数的密集层, 用于计算事件时间间隔的数值. 第 2 层 Transformer 编码器和解码器层数 N 均为 6, 头的个数 h 为 8. *ReLU* 激活函数定义如下, 其中, x_t 为事件时间间隔 (s).

$$ReLU(x_t) = \max(0, x_t) \tag{3}$$

依据 *ReLU* 密集层输出的事件时间间隔数值, 添加到前驱事件的时间戳上, 进而实现对相同时间戳的修复. 此外, 数据视角 Transformer 模型采用反向传播算法来优化其可训练参数, 并选用均方误差 (*MSE*) 作为损失函数. 均方误差公式如下:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(d_i - \hat{d}_i\right) \tag{4}$$

其中, n 为预测事件的总数, d_i 为真实的事件间的时间间隔, $\hat{d}_i$ 为预测的事件间的时间间隔.

4 实验评估

4.1 实验设置与数据集

为验证本文方法对于相同时间戳错误修复的适用性和有效性, 以及其在不同场景下的通用性, 本节设计了两组实验. 第 1 组实验旨在对比分析本方法与现有方法在修复相同时间戳错误方面的性能表现; 第 2 组实验则侧重于检验本方法在不同业务流程领域事件日志上的具体应用效果.

本文方法核心目标在于解决事件日志中的相同时间戳问题, 具体涉及受影响事件的重新排序及相应时间戳的预测. 因此需要对这两项任务进行对比评估.

针对相同时间戳事件重排序任务, 实验采用序列级准确率 *SLA* (sequence label accuracy) 作为评估指标. *SLA* 定义如式 (5) 所示, 其中, *ES* 代表所有包含相同时间戳错误的流程案例总数, *PRS* 代表经过重新排序后完全正确的流程案例数量.

$$SLA = \frac{PRS}{ES} \tag{5}$$

对于事件间时间间隔预测, 实验遵循标准的回归指标, 使用平均绝对误差 (*MAE*) 和均方根误差 (*RMSE*) 作为评估指标. *MAE*、*RMSE* 定义如下:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|d_i - \hat{d}_i\right| \tag{6}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(d_i - \hat{d}_i\right)^2} \tag{7}$$

其中, n 为预测事件的总数, d_i 为真实的事件间的时间间隔, $\hat{d}_i$ 为预测的事件间的时间间隔.

实验使用了 4 个公开可用的数据集, 如表 3 所示. Hospital billing 源自一家医院的企业资源规划系统, 详细记录了医疗服务的计费流程; BPIC 2017 是一家金融机构的贷款申请流程; BPIC 2019 为一家跨国公司在涂料与油漆领域的采购订单处理流程; Sepsis 来自一家医院中脓毒症患者的处理流程. 这些事件日志源自不同的业务流程场景, 并且所记录的流程执行时间跨度也各不相同.

此外, 针对本文所提出的方法, 实验选取了上述事件日志中的 4 个通用事件属性, 即案例 (case)、事件

(event)、资源 (resource) 以及时间戳 (timestamp). 鉴于不同事件日志所处的业务环境差异, 从中额外选取了一个与时间戳相关的附加属性 (variant).

表 3 实验数据集信息

数据集	流程案例数	事件数	活动数
Hospital billing	92 748	405 794	18
BPIC 2017	42 995	193 849	8
BPIC 2019	229 813	1 204 639	29
Sepsis	1 050	15 214	16

4.2 实验结果

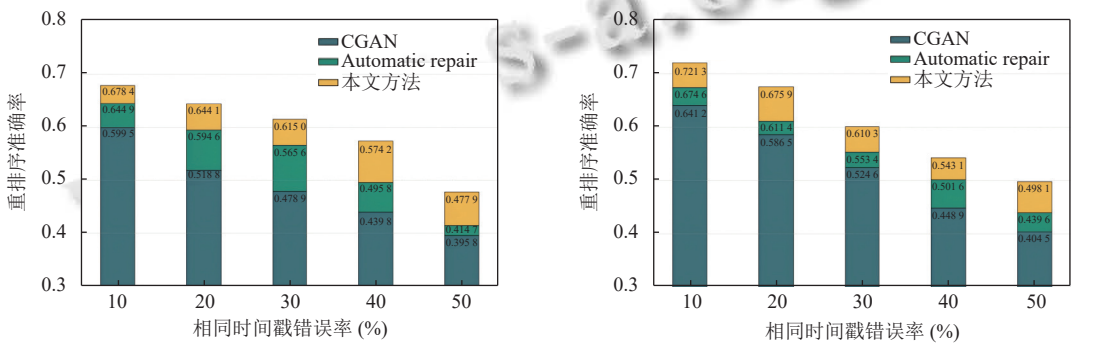
4.2.1 有效性评估

为验证本文方法在修复相同时间戳错误方面的有效性, 实验考量相同时间戳事件案例重排序和错误时间戳的修复 (即事件时间间隔的预测) 两个方面. 依据序列级准确率 *SLA*, 将真实序列与修复后的序列进行比较, 使用平均绝对误差 *MAE* 和均方根误差 *RMSE* 对比了真实时间戳与修复后时间戳之间的差异.

实验在公开的“Hospital billing”和“BPIC 2019”数据集上, 对本文提出的修复方法进行了评估, 并将其与 Automatic repair[6]和条件生成对抗网络 CGAN[7]这两种修复方法进行了对比. 为了确保实验的一致性和可重复性, 实验遵循文献中的实验设置. 具体而言, 按照 10%–50% 的比例, 以 10% 为间隔, 随机生成发生相同时间戳错误的事件, 每次修复进行 100 次迭代. 有效性评估结果如表 4、图 6–图 8 所示.

表 4 有效性评估结果

数据集	方法	错误率 (%)	重排序准确率	*MAE* (s)	*RMSE* (s)	数据集	方法	错误率 (%)	重排序准确率	*MAE* (s)	*RMSE* (s)
Hospital billing	Automatic	10	0.644 9	2 476 723	5 604 632	BPIC 2019	Automatic	10	0.674 6	650 974	1 287 022
	CGAN		0.599 5	1 724 704	3 651 212		CGAN		0.641 2	482 166	921 203
	本文方法		0.678 4	1 353 821	2 574 616		本文方法		0.721 3	430 634	632 402
	Automatic	20	0.594 6	2 533 668	6 253 468		Automatic	20	0.611 4	680 412	1 300 981
	CGAN		0.518 8	1 993 404	4 148 983		CGAN		0.586 5	499 832	994 862
	本文方法		0.644 1	1 618 811	3 099 677		本文方法		0.657 9	448 621	703 735
	Automatic	30	0.565 6	2 646 568	6 584 315		Automatic	30	0.553 4	709 713	1 454 722
	CGAN		0.478 9	2 246 568	5 270 584		CGAN		0.524 6	524 593	1 088 229
	本文方法		0.615 0	1 832 487	3 445 522		本文方法		0.601 3	489 710	722 163
	Automatic	40	0.495 8	2 758 163	7 016 896		Automatic	40	0.501 6	736 004	1 469 337
	CGAN		0.439 8	2 458 163	6 043 843		CGAN		0.448 9	559 648	1 147 804
	本文方法		0.574 2	2 012 182	4 043 271		本文方法		0.543 1	518 226	792 459
	Automatic	50	0.414 7	2 762 676	7 154 863		Automatic	50	0.439 6	764 985	1 573 252
	CGAN		0.395 8	2 552 676	6 706 835		CGAN		0.404 5	589 110	1 221 485
	本文方法		0.477 9	2 242 219	4 509 173		本文方法		0.498 1	558 729	812 476

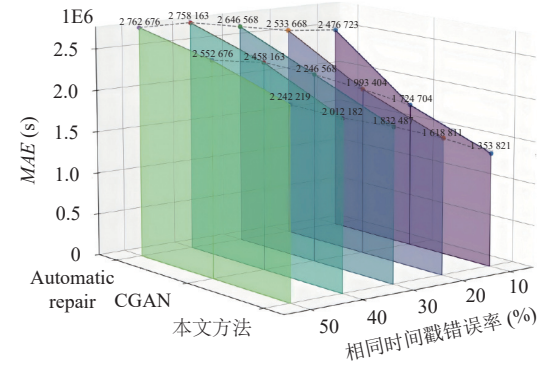


(a) Hospital billing 相同时间戳事件重排序准确率方法对比 (b) BPIC 2019 相同时间戳事件重排序准确率方法对比

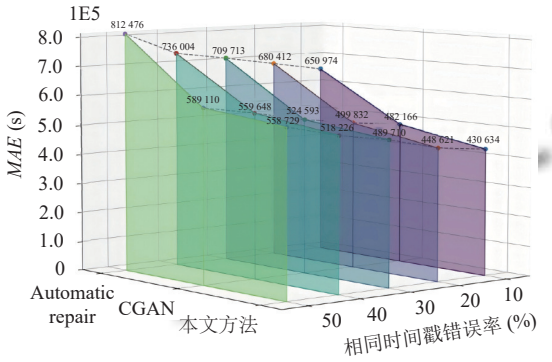
图 6 重排序准确率 (*SLA*) 方法对比

图 6 展示了本文的修复方法、Automatic repair 方法以及条件生成对抗网络 CGAN 方法, 在“Hospital billing”和“BPIC 2019”事件日志上, 对相同时间戳事件案例的重排序准确率对比.

图 7 与图 8 展示了这 3 种方法在“Hospital billing”与“BPIC 2019”事件日志上对于错误时间戳修复 (即事件间时间间隔预测) 的平均绝对误差 *MAE* 和均方根误差 *RMSE* 对比.

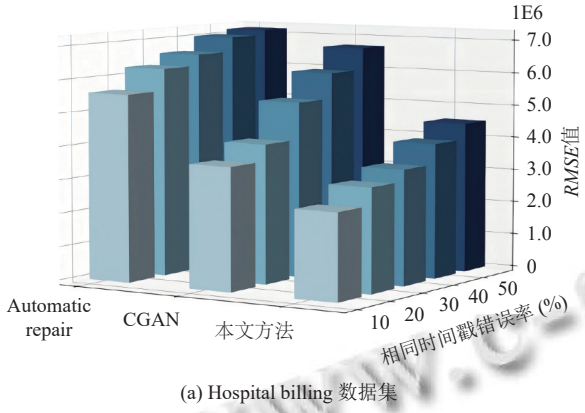


(a) Hospital billing 数据集

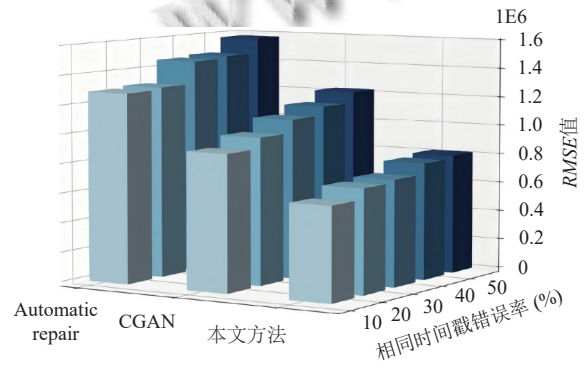


(b) BPIC 2019 数据集

图 7 平均绝对误差 (*MAE*) 方法对比



(a) Hospital billing 数据集



(b) BPIC 2019 数据集

图 8 均方根误差 (*RMSE*) 方法对比

评估结果显示, 相较于 Automatic repair 和条件生成对抗网络 CGAN 方法, 本文方法在案例事件重排序准确率、时间戳预测精度以及预测稳定性上都取得了更为出色的表现. 结果表明所提方法可以有效提高相同时间戳错误的修复精度.

4.2.2 通用性评估

为评估本文方法的通用性, 实验选取了涵盖 3 个业务领域 (医疗、金融以及制造业) 的 4 个事件日志, 这些事件日志包含了多样化的流程事件和流程案例时间跨度. 具体而言, 所选数据集对应着 4 个不同的流程类型包括计费流程、交易流程、贷款申请流程以及病例处理流程, 这些数据集包含了 366 606 个流程案例、1 819 496 个事件以及 71 个活动 (数据集信息如表 3 所示). 其中, Hospital billing 医疗计费流程案例事件时间间隔长达数月至数年, BPIC 2019 采购订单处理流程和 BPIC 2017 贷款申请流程案例事件时间间隔在几周至数月之间, Sepsis 脓毒症患者的处理流程案例事件时间间隔在几天至一周左右.

通过所提出方法在各种类型、流程案例 (迹)、案例时间跨度数据集上, 不同相同时间戳错误比例上与 Automatic 方法的修复性能表现对比, 探究方法在面对不同业务领域和流程类型时的泛化能力与适用性. 通用性评估结果如表 5 所示.

表 5 通用性评估结果

数据集	方法	错误率 (%)	重排序准确率	*MAE* (s)	*RMSE* (s)
Hospital billing	Automatic	10	0.6449	2476723	5604632
	本文方法		0.6784	1353821	2574616
	Automatic	20	0.5946	2533668	6253468
	本文方法		0.6441	1618811	3099677
	Automatic	30	0.5656	2646568	6584315
	本文方法		0.6150	1832487	3445522
	Automatic	40	0.4958	2758163	7016896
	本文方法		0.5742	2012182	4043271
	Automatic	50	0.4147	2762676	7154863
	本文方法		0.4779	2242219	4509173
BPIC 2019	Automatic	10	0.6746	650974	1287022
	本文方法		0.7213	430634	632402
	Automatic	20	0.6114	680412	1300981
	本文方法		0.6579	448621	703735
	Automatic	30	0.5534	709713	1454722
	本文方法		0.6013	489710	722163
	Automatic	40	0.5016	736004	1469337
	本文方法		0.5431	518226	792459
	Automatic	50	0.4396	764985	1573252
	本文方法		0.4981	558729	812476

表 5 通用性评估结果 (续)

数据集	方法	错误率 (%)	重排序准确率	*MAE* (s)	*RMSE* (s)
BPIC 2017	Automatic	10	0, 7086	167654	458435
	本文方法		0.7543	116725	302894
	Automatic	20	0.6365	169922	47536
	本文方法		0.6920	130585	330158
	Automatic	30	0.5765	183947	509106
	本文方法		0.6215	141512	351916
	Automatic	40	0.5011	190217	528453
	本文方法		0.5109	167989	368147
	Automatic	50	0.4637	193234	541835
	本文方法		0.4732	179367	377916
Sepsis	Automatic	10	0.6625	69238	691782
	本文方法		0.7041	57458	638788
	Automatic	20	0.6023	74562	752012
	本文方法		0.6584	64960	639838
	Automatic	30	0.5523	85523	882933
	本文方法		0.6335	74361	641256
	Automatic	40	0.5193	100234	934586
	本文方法		0.5922	92811	644365
	Automatic	50	0.4657	116732	1023459
	本文方法		0.5107	96970	649226

评估结果显示, 本文方法在各数据集上修复相同时间戳错误与 Automatic 方法表现出一致性且效果明显优于 Automatic 方法. 具体而言, 在不同时间戳错误比例下, 处理流程较为复杂的案例, 时间跨度最大的 Hospital billing 数据集上的时间戳误差范围在 15–25 天之间; 而在流程最为复杂且时间跨度也较大的 BPIC 2019 数据集上, 时间戳误差则通常在 5–7 天; 对于流程相对复杂且事件跨度较大的 BPIC 2017 数据集, 时间戳误差介于 30–48 h 之间; 相比之下, 在处理流程最为简单且时间跨度最小的 Sepsis 数据集时, 时间戳误差则相对较小, 在 14–26 h 之间. 不论是来自不同领域、类型或时间跨度的事件日志, 本文方法均能稳定有效修复相同时间戳错误, 验证了其在应对多样化应用场景时的通用性及泛化能力.

5 结束语

本文提出了一种基于分层 Transformer 架构的方法, 旨在修复事件日志中的相同时间戳错误. 该方法通过构建分层 Transformer 模块结合事件行为与属性数据视角交互, 捕获流程案例事件间行为依赖关系, 挖掘事件属性间潜在数据关联. 通过逐层完成错误事件重排序和对应的时间戳预测任务, 进而实现对相同时间戳错误的修复. 实验结果显示, 所提方法可以有效地提高相同时间戳错误的修复精度, 且该方法在所选各种业务场景及不同时间跨度下的数据集中均表现出了出色的适用性和泛化能力.

本文主要聚焦于相同时间戳错误的修复层面, 使用的错误检测方法在事件级别而不是活动级别上执行操作, 因此忽略了事件并发的可能性. 未来工作应深入探索相同时间戳错误的精确检测技术. 此外, 流程模型蕴含着丰富的结构化信息, 研究计划将分层 Transformer 架构同多尺度卷积神经网络相结合, 通过多尺度特征表示协同多通道注意力权重, 有效整合并解析多维度特征信息, 进一步提高相同时间戳错误的修复精度.

参考文献

1 van der Aalst WMP. Process mining: A 360 degree overview. In: van der Aalst WMP, Carmona J, eds. Process Mining Handbook. Cham: Springer, 2022. 3–34.

2 Günther CW, van der Aalst WMP. Fuzzy mining—Adaptive process simplification based on multi-perspective metrics. Proceedings of the 5th International Conference on Business Process Management. Brisbane: Springer, 2007. 328–343.

3 Weijters AJMM, van der Aalst WMP, de Medeiros AKA. Process mining with the heuristics miner algorithm. https://www.researchgate.net/publication/229124308. (2014-05-19).

4 Leemans SJJ, Fahland D, van der Aalst WMP. Process and deviation exploration with inductive visual miner. Proceedings of the 12th International Conference on Business Process Management. Eindhoven: CEUR-WS, 2014. 46–50.

5 Dixit PM, Suriadi S, Andrews R, *et al*. Detection and interactive repair of event ordering imperfection in process logs. Proceedings of the 30th International Conference on Advanced Information Systems Engineering. Tallinn: Springer, 2018. 274–290.

6 Conforti R, La Rosa M, ter Hofstede AHM, *et al*. Automatic repair of same-timestamp errors in business process event logs. Proceedings of the 18th International Conference on Business Process Management. Seville: Springer, 2020. 327–345.

7 Schmid SJ, Moder L, Hofmann P, *et al*. Everything at the proper time: Repairing identical timestamp errors in event logs with generative adversarial networks. Information Systems, 2023, 118: 102246.

8 van der Aalst W, Adriansyah A, de Medeiros AKA, *et al*. Process mining manifesto. Proceedings of the 2011 International Workshops on Business Process Management

Workshops. Clermont-Ferrand: Springer, 2012. 169–194.
9 Bose RPJC, Mans RS, van der Aalst WMP. Wanna improve process mining results? Proceedings of the 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM). Singapore: IEEE, 2013. 127–134.
10 Mans RS, van der Aalst WMP, Vanwersch RJB, *et al*. Process mining in healthcare: Data challenges when answering frequently posed questions. Proceedings of the 2012 BPM Joint Workshop on Process Support and Knowledge Representation in Health Care. Tallinn: Springer, 2013. 140–153.
11 Jiang P, Obi T, Nakajima Y. Integrating prior knowledge to build Transformer models. International Journal of Information Technology, 2024, 16(3): 1279–1292.
12 Nguyen HTC, Lee S, Kim J, *et al*. Autoencoders for improving quality of process event logs. Expert Systems with Applications, 2019, 131: 132–147. [doi: 10.1016/j.eswa.2019.04.052]
13 Song SX, Cao Y, Wang JM. Cleaning timestamps with temporal constraints. Proceedings of the VLDB Endowment, 2016, 9(10): 708–719. [doi: 10.14778/2977797.2977798]
14 Ni WJ, Zhao G, Liu T, *et al*. Predictive business process monitoring approach based on hierarchical Transformer. Electronics, 2023, 12(6): 1273.
15 陈奉贤. 基于 NR-Transformer 的集群作业运行时间预测. 计算机工程与科学, 2022, 44(7): 1181–1190. [doi: 10.3969/j.issn.1007-130X.2022.07.005]
16 Lu F, Li W, Sun YF, *et al*. CGS-Mask: Making time series predictions intuitive for all. Proceedings of the 2024 AAAI Conference on Artificial Intelligence. Vancouver: AAAI, 2024. 14149–14157.
17 Suriadi S, Andrews R, ter Hofstede AHM, *et al*. Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Information Systems, 2017, 64: 132–150. [doi: 10.1016/j.is.2016.07.011]
18 张振虎, 王丽丽, 袁永旺. 基于 BERT 的多视角事件日志修复. 计算机应用研究, 2024, 41(2): 515–520.

(校对责编: 王欣欣)