计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
[doi: 10.15888/j.cnki.csa.009728] [CSTR: 32024.14.csa.009728]

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

基于改进 Transformer 的剩余时间预测①

刘海洲, 高俊涛

(东北石油大学 计算机与信息技术学院, 大庆 163318)
通信作者: 刘海洲, E-mail: 2509770056@qq.com

摘　要: 剩余时间预测能够帮助企业提升业务流程执行的质量和效率. 尽管现有的深度学习方法在剩余时间预测上有一定提升, 但在处理复杂业务流程时, 仍面临时间特征利用不足和局部特征挖掘能力有限的问题, 预测精度有待提高. 为此, 本研究提出了一种基于改进 Transformer 编码器模型的剩余时间预测方法. 针对已有方法忽略事件时间特征以及难以捕捉局部依赖的不足, 本研究在模型中引入了时间特征编码模块和局部依赖增强模块. 时间编码模块通过嵌入学习和多粒度拼接方式, 构建了富有语义且具判别力的事件时间表示. 局部依赖增强模块采用卷积神经网络, 在 Transformer 编码器之后提取轨迹前缀的局部细节特征. 实验表明, 融合时间特征和局部依赖增强可以提升复杂业务流程剩余时间的预测准确性.

关键词: 剩余时间预测; 过程挖掘; 深度学习; Transformer

引用格式: 刘海洲,高俊涛.基于改进 Transformer 的剩余时间预测.计算机系统应用. http://www.c-s-a.org.cn/1003-3254/9728.html

Remaining Time Prediction Based on Improved Transformer

LIU Hai-Zhou, GAO Jun-Tao

(School of Computer & Information Technology, Northeast Petroleum University, Daqing 163318, China)

Abstract: Remaining time prediction helps enterprises improve the quality and efficiency of business process execution. Although existing deep learning methods have shown improvement in remaining time prediction, they still face challenges when dealing with complex business processes. These challenges include insufficient utilization of time features and limited ability to extract local features, leaving room for improvement in prediction accuracy. This study proposes a remaining time prediction method based on the improved Transformer encoder model. Existing methods ignore event time features and struggle to capture local dependencies. To address these limitations, this study introduces a time feature encoding module and a local dependency enhancement module into the model. The time encoding module constructs a semantically rich and discriminative event time representation by embedding learning and multi-granularity concatenation. The local dependency enhancement module uses convolutional neural networks to extract fine-grained local features from the trajectory prefix after processing with the Transformer encoder. Experiments show that integrating time features and local dependency enhancement improves the prediction accuracy of the remaining time for complex business processes.

Key words: remaining time prediction; process mining; deep learning; Transformer

在业务流程挖掘研究中, 预测型业务流程监控尤为关键. 通过分析日志数据预测流程的未来特征, 如下一步活动、实例总时间以及剩余时间等, 可以实现业务流程的优化, 提升企业效率. 特别是利用日志数据预

① 基金项目: 黑龙江省省属本科高校基本科研业务费 (2022TSTD-03)
收稿时间: 2024-05-29; 修改时间: 2024-06-28; 采用时间: 2024-07-18; csa 在线出版时间: 2024-10-31

测流程实例的剩余执行时间, 被认为是业务流程监控的重要研究任务. 通过预测实例的剩余时间, 有助于发现流程中的瓶颈和低效环节, 为流程优化和改进提供依据. 近年来, 利用深度学习预测流程实例剩余时间的效果日趋显著. 它已在多个实际案例中证明了可行性与有效性. 这表明, 深度学习在业务流程挖掘与优化方面具有广阔的应用前景.

但现阶段的基于深度学习的剩余时间预测还存在着以下一些问题.

当前, 剩余时间预测中的轨迹前缀相关研究多集中于轨迹前缀序列本身的分析, 较少考虑事件的时间特征. 仅依据事件序列信息进行预测, 忽视了时间对流程执行的重要影响. 不同事件发生的时间间隔会对流程执行产生显著影响, 如果忽略时间特征, 很难准确捕捉流程的动态变化.

尽管当前一些方法已经采用了较为先进的 Transformer 模型, 在流程实例剩余时间预测效果上相较于以前的模型取得了显著提升, 但这些方法的预测精度仍未达到令人满意的水平.

从模型结构来看, Transformer 模型通过自注意力机制来学习轨迹前缀数据中的长距离依赖关系, 对轨迹前缀序列的全局模式建模效果较好. 但是, Transformer 对轨迹前缀序列数据的局部依赖关系学习能力较为薄弱. 自注意力机制过于关注全局的依赖关系, 而较少建模轨迹前缀序列的局部依赖关系. 这主要源于 Transformer 模型的编码器部分只包含自注意力层. 尽管自注意力虽然能够较好地建模全局依赖关系, 但也稀释了局部邻近位置的关联. 这一缺陷限制了 Transformer 在剩余时间预测上的效果, 也为进一步优化该模型提供了方向.

为提高剩余时间预测的精度, 本研究对 Transformer 模型进行了优化. 首先, 在模型中加入时间特征, 以增强对事件间的时间依赖关系的建模. 过去更多依赖事件的顺序特征, 而忽略了时间相关性, 这降低了预测的准确性和可靠性. 其次, 改进了 Transformer 模型的结构, 在 Transformer 编码器输出后加入了局部依赖增强模块. 局部依赖增强模块通过卷积层通过卷积核提取局部特征, 最大池化层进行下采样. 卷积操作能够学习特征之间的局部相关性, 而池化通过聚合邻近特征实现信息整合. 局部依赖增强模块的加入增强了模型对轨迹前缀的局部模式的建模能力. 卷积层的多层堆叠结构使得模块可以学习复杂的局部依赖关系. 与自注意力机制不同, 卷积专注于局部相邻特征, 可以提供互补的局部依赖建模.

通过引入时间特征和局部依赖捕捉模块, 改进后的模型兼具自注意力的全局建模能力和卷积的局部建模能力. 全局信息和局部特征的融合提升了模型对轨迹前缀的表示能力, 增强了流程实例剩余时间预测的准确性.

1 研究现状

剩余时间预测按照其发展的不同阶段, 主要可以分为 3 个不同的研究方向, 分别为: 基于统计学的剩余时间预测方法、基于机器学习的剩余时间预测方法、基于深度学习的剩余时间预测方法.

早期的剩余时间预测方法主要为基于统计学的剩余时间预测方法, 主要通过运用统计学方法对收集到的事件日志数据进行整理、统计和分析, 从而确定每项活动的平均执行时间和概率分布, 以此来实现对剩余时间的预测. van der Aalst 等[1]从业务流程的日志当中挖掘出标注的变迁系统进行预测, Folino 等[2]在 van der Aalst 等[1]的研究基础上提出了分布式变迁随机 Petri 网进行预测, 基于当前执行状态, 生成 n 个可能的剩余时间, 对这 n 个结果求平均值来预测剩余处理时间. 高俊涛等[3]在传统的变迁系统的基础上提出了在线预测模型的构建及剩余时间预测方法, 设计出了能够及时反映系统运行规律的增量式模型构建计算算法, 降低了传统方法的预测波动性, 提高了预测模型的可信度. 基于统计学的剩余时间预测方法虽然对流程有结构化的理解, 便于分析和解释. 但是其预测精度较低, 并且其在大规模的事件日志当中的预测表现并不理想.

基于机器学习的剩余时间预测方法主要是通过一些机器学习方法辅助预测剩余时间. Polato 等[4]将朴素贝叶斯和支持向量机应用到基于变迁的预测系统当中, 以辅助预测的方式参与到了剩余时间预测当中. Pandey 等[5]将隐马尔可夫模型应用于标注的变迁系统. 尽管机器学习在处理复杂数据集、发现潜在模式和关系方面更为强大, 但是机器学习的算法在处理非线性、复杂的关系时可能会受限.

近几年深度学习的火爆带动了研究者利用深度学习技术进行剩余时间预测. Navarin 等[6]提出了一种 LSTM 结构来预测轨迹的完成时间. Tax 等[7]首先利用 onehot

编码技术将事件数据转化为特征向量输入 LSTM 模型进行预测. Taymouri 等[8]首次将生成式对抗网络应用到业务过程领域中用于剩余事件预测. 随着深度学习的不断发展, 一些新兴出现的神经网络模型和神经网络机制也得到了应用. 倪维健等[9]通过引入双向循环神经网络[10], 并且对不同轨迹长度的分别训练剩余时间预测模型, 提高了业务流程剩余时间预测的准确度. Bukhsh 等[11]通过结合了长程记忆, 以及自注意力机制, 建立了大量的事件序列和相应的输出之间的依赖关系, 并且使用 Transformer 模型[12]实现了对业务流程的剩余时间的准确预测. 徐兴荣等[13]将 Word2Vec[14]应用于事件的表示, 为预测模型提供了高质量的输入事件向量. 并且提出了带有注意力机制的双向准循环神经网络, 以及迁移学习的方法对数据量较少的长轨迹前缀进行训练并获得了较好的预测结果. 郭娜等[15]提出了一种可解释特征分层的方法, 通过 LightGBM[16]对事件日志进行特征组合选择, 在保证了准确率的前提下减少了特征选取的数量, 提升了预测的效果.

2 预备知识

本节将对业务流程剩余时间预测任务的相关知识进行详细介绍, 并给出相关定义.

定义 1. 轨迹. 轨迹是一种有序的事件序列, 表示为 $\sigma = \langle e_1, e_2, e_3, \cdots, e_n \rangle$. e_i代表第i个执行的事件.

定义 2. 事件. 事件是业务系统中某个活动被实际执行的具体实例, 也是业务流程的构成的基本元素. 表示为元组$e = \{CaseID, activity, time\}$, $CaseID$为当前事件所属的流程的实例的ID, $activity$为当前事件的活动名称, $time$为事件的发生时间. 定义 2 阐述了进行流程挖掘所必需的基础事件属性模型. 虽然实际的事件日志可能包含更丰富的信息, 如资源、经办人或资源使用量等, 为了研究的普遍适用性, 本研究将使用基本属性来探讨剩余时间预测的方法.

定义 3. 轨迹前缀. 轨迹前缀是通过截取轨迹σ的前k个事件得到的, 其中轨迹σ_k代表截取了前k个事件, 记作$\sigma_k = \langle e_1, \cdots, e_k \rangle$. 轨迹前缀的剩余时间可以表示为:

$$RT(\sigma, k) = e_{|\sigma|}.Time_{\text{end}} - e_k.Time_{\text{end}} \tag{1}$$

定义 4. 流程实例. 流程实例是一个流程从开始到结束的完整执行, 表示为$z = \{Caseid, \sigma, P_{\{1,2,\cdots,n\}}\}$, $CaseID$为流程实例的ID, σ为流程实例中的轨迹, $P_{\{1,2,\cdots,n\}}$为该流程的其他属性.

定义 5. 事件日志. 事件日志是流程实例的集合, 是业务系统的具体的执行情况的记录, 表示为$L = \{z_1, z_2, \cdots, z_{|l|}\}$.

3 剩余时间预测方法

本节将详细介绍本研究中提出的剩余时间预测方法, 本研究通过 Transformer 编码器、局部依赖增强模块和时间特征的融合, 实现了对轨迹前缀序列的全局和局部特征的学习和挖掘, 以精确预测流程实例的剩余时间.

3.1 事件日志数据处理

3.1.1 前缀序列处理

轨迹前缀序列获取的主要处理步骤为: 初始化一个字典来存储每条轨迹的前缀序列, 初始化一个集合来存储所有不同的前缀. 遍历轨迹日志中的每一条轨迹, 对于每条轨迹, 逐步生成从长度为 1 到全长的所有前缀子序列. 在生成每个前缀时, 需要判断它是否已经存在于前缀集合中, 如不存在, 就将其添加到集合中, 同时也添加到该条轨迹的前缀序列中. 最终, 返回存储每条轨迹前缀序列的字典以及包含所有前缀的集合. 具体见算法 1.

算法 1. 获取轨迹前缀序列

输入: 轨迹日志 traces.
输出: 前缀轨迹序列 prefix_seqs, 前缀轨迹集合 prefixes.

算法过程:

```
prefix_seqs = {} # 存储每个 trace 的前缀序列
prefixes = set() # 前缀集合, 使用 set 避免重复
for trace in traces:
    prefix_seq = [] # 当前 trace 的前缀序列
    for i in range(len(trace)):
        prefix = trace[1:i+1] # 生成前缀轨迹
        if prefix not in prefixes:
            prefixes.add(prefix) # 如果前缀轨迹不在集合中, 添加
        prefix_seq.append(prefix) # 将前缀添加到当前 trace 的前缀序列中
    prefix_seqs[trace] = prefix_seq # 将当前 trace 的前缀序列存储到字典中
return prefix_seqs, prefixes
```

基于上述算法获得了轨迹前缀向量. 这为后续的时间特征融合和模型训练提供了高质量的输入数据, 有助于提高模型的预测准确性和泛化能力.

3.1.2 时间-轨迹前缀序列处理

在处理的轨迹前缀的序列的基础上, 对每个序列

的时间进行处理, 以便将时间转化为特征输入到模型中.

在获取轨迹的前缀序列后, 需要进一步处理时间特征, 以便将其转换为模型的输入. 具体步骤如下: 从日志数据中提取时间相关字段, 将提取的时间日期转换为 datetime 格式, 从中分别提取年、月、日作为新的特征. 对前缀序列进行填充 (padding), 使其达到统一长度. 最后, 将年、月、日特征拼接到前缀序列的前部. 最终, 包含时间特征的前缀序列将作为模型的输入.

算法 2. 时间-轨迹前缀序列处理

输入: 经过算法 1 处理的轨迹数据 trace_data.
输出: 融合时间特征的轨迹编码序列 encoded_data.

算法过程:

```
time_features = {} # 存储时间特征的字典
encoded_sequences = [] # 存储编码后的轨迹序列列表
max_length = 0 # 最大序列长度
# 步骤 1: 提取时间相关特征
for data_point in trace_data:
    datetime_feature = extract_datetime_features(data_point['datetime'])
    if datetime_feature not in time_features:
        time_features[data_point['datetime']] = datetime_feature
# 步骤 2: 对轨迹序列进行编码
for trajectory in trace_data:
    encoded_trajectory = []
    for data_point in trajectory:
        transformed_point = transform_time_feature(time_features
[data_point['datetime']])
        encoded_trajectory. append(transformed_point)
    encoded_sequences. append(encoded_trajectory)
    if len(encoded_trajectory) > max_length:
        max_length = len(encoded_trajectory)
# 步骤 3: 序列等长 padding 和特征融合
for i, sequence in enumerate(encoded_sequences):
    while len(sequence) < max_length:
        sequence. append(padding_element) # padding_element 是预定义
的填充元素
    encoded_sequences[i] = concatenate_features(sequence, trajectory_
features[i])
# 输出处理后的数据
encoded_data = encoded_sequences
return encoded_data
```

将时间特征和轨迹前缀序列进行了编码和融合之后, 确保了每个轨迹都被等长表示, 同时含有丰富的时间信息. 这样的处理不仅标准化了输入数据格式, 使其适合于大多数监督学习算法.

3.2 时间特征融合

在流程实例的剩余时间预测任务中, 充分利用事件过程中的时间信息至关重要, 因为时间特征对于准确预测剩余执行时间起着关键作用. 事件日志数据中往往蕴含着许多复杂的时间模式, 如周期性、趋势性等, 这些对于准确预测剩余时间至关重要. 但如果方法无法有效利用时间信息, 就难以从数据中学习到这些有价值的时间模式知识, 从而导致预测性能的下降. 当前的剩余时间预测方法中一般只会考虑相对时间特征, 相对时间信息只反映了事件之间的先后顺序和时间间隔, 但无法体现事件发生的绝对时间信息. 然而, 不同的时间可能对应着完全不同的时间环境, 如工作日/节假日, 这些时间环境的动态变化会极大影响流程的执行效率和剩余时间. 缺乏绝对时间信息, 模型将难以适应这种时间环境的动态变化. 绝对时间信息可以帮助发现流程事件发生的时间规律, 如工作日/节假日的周期性规律、特定时间段的高峰/低峰等. 同时也有助于发现异常事件, 如在非工作时间突然发生的事件. 缺少绝对时间信息, 模型将无法高效发现和利用这些规律性和异常情况.

为获取语义合理且富有区分度的时间表示, 本研究构建了一个可学习的时间编码模块来捕捉时间戳的内在时间信息, 通过融合相对时间和绝对时间这两类特征, 来提高模型对时间特征的利用.

3.2.1 绝对时间信息的嵌入表示

具体而言, 将原始时间戳拆解为“年-月-日”的粒度, 并为每个粒度定义可学习的嵌入映射, 以学习时间点的语义表示:

$$E_{\text{year}}, E_{\text{month}}, E_{\text{day}} = Embedding(y, m, d) \tag{2}$$

其中, $E_{\text{year}}, E_{\text{month}}, E_{\text{day}}$ 分别表示对年、月、日进行嵌入映射学习所得到的嵌入表示.

随后采用特征级拼接, 在特征维度上堆叠各时间粒度的 *Embedding* 向量, 以获得一个固定长度的时间特征表达:

$$E_{\text{absolute}} = [E_{\text{year}}; E_{\text{month}}; E_{\text{day}}] \tag{3}$$

其中, [;]表示特征拼接操作.

3.2.2 相对时间信息的嵌入表示

相对时间信息能有效捕捉轨迹前缀序列中时间的动态变化. 本研究主要考虑以下两种相对时间差值.

(1) 流程首事件的时间戳差值

这个时间差值表示当前事件与整个流程第 1 个事件之间的时间间隔, 反映了事件发生的相对时间位置. 用公式表示为:

$$\Delta t_{\text{start}} = t_{\text{current}} - t_{\text{start}} \tag{4}$$

其中, t_{current} 是当前事件的时间戳, t_{start} 是当前实例下第 1 个事件的时间戳.

(2) 与相邻事件的时间戳差值

这个时间差值表示当前事件与前一个事件之间的时间间隔, 捕捉了事件之间的局部时间关系. 用公式表示为:

$$\Delta t_{\text{prev}} = t_{\text{current}} - t_{\text{prev}} \tag{5}$$

其中, t_{prev} 是相邻事件的时间戳.

将得到的时间差特征进行嵌入表示:

$$E_{\text{start}} = Embedding(\Delta t_{\text{start}}) \tag{6}$$

$$E_{\text{prev}} = Embedding(\Delta t_{\text{prev}}) \tag{7}$$

最后, 将这两个嵌入表示在特征维度上进行拼接, 得到最终的相对时间嵌入表示.

$$E_{\text{relative}} = [E_{\text{start}}; E_{\text{prev}}] \tag{8}$$

为了将绝对时间信息和相对时间信息融合, 首先将绝对时间嵌入表示和相对时间嵌入表示在特征维度上拼接:

$$E_{\text{combined}} = [E_{\text{absolute}}; E_{\text{relative}}] \tag{9}$$

为使该时间表示适配后续模块, 将一个线性层将拼接后的时间 *Embedding* 投影到与位置编码相同的嵌入空间:

$$E_{\text{proj}} = W_f(E_{\text{combined}}) + b_f \tag{10}$$

其中, E_{proj} 投影后的时间嵌入表示. W_f 和 b_f 分别为线性层的权重和偏置参数.

这种基于时间粒度拆解与重构的时间建模方式综合了下述优点.

(1) 借助 *Embedding* 学习各时间粒度的语义信息, 增强时间表示的语义性.

(2) 拼接不同粒度的表示并映射到固定长度, 保证时间表达的判别性与可融合性.

通过增强对时间特征的建模, 该时间编码机制提供了绝对时间和相对时间信息的多粒度融合, 增强了模型学习时间依赖的能力. 这为预测业务流程的剩余时间提供了支持.

3.3 基于局部依赖增强的 Transformer 剩余时间预测模型

处理剩余时间预测的问题上, 当前深度学习模型已取得长足进展. 许多研究采用循环神经网络或注意力机制来挖掘轨迹前缀序列的内在依赖关系. 例如, Transformer 模型通过 multi-head attention 机制捕捉轨迹前缀中的全局信息. 然而, 仅依赖全局上下文的方法在学习局部特征时存在弱点. 而卷积神经网络具有直接对局部特征进行建模的优势.

本节基于 Transformer 模型, 结合局部依赖增强, 提出了一种剩余时间预测模型, 命名为 LEN-Transformer 模型. 该模型在 Transformer 编码器的基础上, 通过引入局部依赖增强层, 增强了模型对局部特征的提取能力, 同时保留了 Transformer 的全局信息捕捉能力. LEN-Transformer 模型的整体结构如图 1 所示. 模型主要由嵌入层 (Embedding)、位置编码 (positional encoding)、时间编码 (time encoding)、Transformer 编码器模块 (Transformer block)、局部依赖增强模块以及全连接层组成.

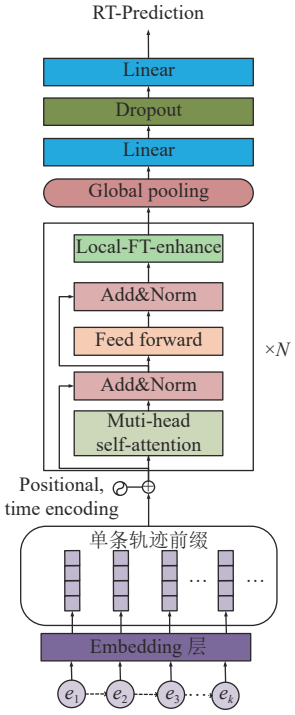

图 1 LEN-Transformer 结构图

具体如下, 模型输入为业务流程前缀轨迹特征. 嵌入层将其映射为固定维度的嵌入向量, 位置编码捕捉前缀轨迹中的位置信息, 时间编码捕捉时间信息. 嵌入向量通过 Transformer 编码器处理, 自注意力机制学习前缀轨迹中的全局依赖关系. Transformer 的输出经过

局部依赖增强, 学习轨迹前缀中的局部依赖关系. 最后, 通过线性层预测业务流程的剩余时间.

3.3.1 局部依赖增强

尽管自注意力机制能够有效建模长期依赖, 但由于自注意力计算基于整个序列, 因此难以捕捉局部依赖关系, 影响了剩余时间预测的准确性. 为解决这一问题, 设计了局部增强网络 (local enhancement network, LEN). LEN 是基于卷积神经网络 (CNN) 设计的模块, 通过卷积滤波器对输入序列进行卷积操作, 从而增强局部依赖关系的捕捉能力.

首先将输入数据 X 传递给卷积层 ($Conv1d$), 它对特征进行加权求和, 并添加偏置.

$$C = Conv1d(X; W_{\text{conv}}, b_{\text{conv}}) \tag{11}$$

其中, W_{conv} 代表卷积层的权重, b_{conv} 而是相应的偏置. 这一步的作用是提取局部特征, 以补充 Transformer 编码器输出的全局信息.

为了加强模型的泛化能力, 在卷积操作之后应用批量归一化 ($BatchNorm1d$). 加速训练过程, 减少模型对初始权重的依赖.

$$C' = BatchNorm1d(C) \tag{12}$$

使用 ELU (指数线性单元) 作为非线性激活函数, ELU 能够提供平滑的非线性变换, 有助于缓解梯度消失问题, 并能捕捉到更加复杂的数据特征.

$$E = ELU(C') \tag{13}$$

最后, 在卷积激活后引入最大池化层 ($MaxPool1d$), 以实现特征的下采样.

$$M = MaxPool1d(E) \tag{14}$$

池化层通过汇总邻近数据点的信息, 提取出代表局部区域的重要特征, 同时显著减少数据维度, 降低了模型的计算复杂度. 这一操作不仅减轻了模型对局部位置的敏感性, 而且加强了特征的平移不变性, 这对于剩余时间预测具有重要意义.

通过该模块有效地从 Transformer 编码器输出中提取出局部和全局信息, 为剩余时间预测提供了更加丰富和精细化的特征表示.

3.3.2 Transformer 编码器

Transformer 编码器是 LEN-Transformer 模型的核心组件, 负责捕捉业务流程前缀轨迹中的全局依赖关系. Transformer 编码器由多个相同的层堆叠而成, 每一层包括两个子层: 多头自注意力机制 (multi-head attention) 和前馈神经网络 (feed forward network).

多头自注意力机制 (multi-head attention) 允许模型在不同的表示子空间中并行地学习注意力权重. 给定输入矩阵 , multi-head attention 的计算过程如下:

通过线性变换得到 Query 矩阵、Key 矩阵和 Value 矩阵:

$$Q = XW^Q, K = XW^K, V = XW^V \tag{15}$$

其中, Q, K, V 是查询 (Query) 矩阵、键 (Key) 矩阵和值 (Value) 矩阵. X 是输入矩阵, W^Q, W^K, W^V 是对应的权重矩阵.

将 Q, K, V 划分为 h 个头, 每个头的维度为 d_k, 再对每个头并行计算注意力:

$$Attention(Q_i, K_i, V_i) = Softmax\left(\frac{Q_i K_i^{\mathrm{T}}}{\sqrt{d_k}}\right) \cdot V_i \tag{16}$$

其中, d_k 为缩放因子.

最后将所有头的输出拼接起来, 经过另一个线性变换得到最终的多头注意力输出:

$$MultiHead(Q, K, V) = Concat(h_1, \cdots, h_h)W^O \tag{17}$$

其中, W^O 为另一个可学习的权重矩阵.

多头自注意力的输出经过残差连接和层归一化后, 再输入到前馈神经网络中. 前馈神经网络包含两个线性变换, 中间用 $ReLU$ 激活函数:

$$FFN(x) = ReLU(x \cdot W_1 + b_1) \cdot W_2 + b_2 \tag{18}$$

其中, W_1、W_2 为可学习的权重矩阵, 为偏置向量.

前馈神经网络的输出同样经过残差连接和层归一化, 作为当前 Transformer 编码器层的最终输出. 通过堆叠多个这样的编码器层, 模型能够学习到输入序列中的深层次全局依赖关系.

Transformer 编码器通过自注意力机制有效地捕捉了业务流程前缀轨迹的长期依赖, 但对局部依赖的建模能力有限. 因此, 本文引入了局部增强模块来增强模型对局部依赖的提取能力, 与 Transformer 编码器形成互补, 共同构建了 LEN-Transformer 模型用于业务流程剩余时间预测任务.

4 实验结果与分析

本节将根据提出的基于 Transformer 和局部依赖增强模块以及时间特征的剩余时间预测方法, 进行实验和研究.

4.1 实验数据

本研究使用了来自 4TU Center for Research 平台公开的 7 个事件日志数据集, 分别为 Production、BPIC_2015_1、BPIC_2015_2、BPIC_2015_3、BPIC_2015_4、BPIC_2015_5、Helpdesk. 其中, Helpdesk 事件日志记录了某公司技术支持系统的问题处理流程. Production 事件日志包含了某生产车间 2012 年 1–3 月部分产品的生产流程数据. BPIC2015 事件日志来自 2015 年 BPI 挑战赛, 共包含以上 5 个事件日志, 事件日志数据集的详细统计信息如表 1 所示.

表 1 数据集的基本信息

数据集	轨迹数量	事件数量	活动数量	最长轨迹长度	轨迹平均	最短轨迹长度
BPIC_2015_1	1 199	52 217	398	101	455	2
BPIC_2015_2	832	44 354	410	132	531	1
BPIC_2015_3	1 409	59 681	383	124	42.35	3
BPIC_2015_4	1 053	47 293	356	116	44.91	1
BPIC_2015_5	1 156	59 083	389	154	51.1	5
Production	225	4 543	55	175	20.19	1
Helpdesk	4 580	21 348	14	15	4.66	2

4.2 实验评价标准

本研究选用平均绝对误差 *MAE* (mean absolute error) 作为评价模型预测精度的指标. *MAE* 是一种广泛使用的统计误差度量方法, 通过计算预测结果与真实值之间差异的平均绝对值, 来反映模型的预测准确程度. 相比之下, 平均平方误差 (MSE) 由于计算误差的平方, 容易受到个别大误差的影响. 而 *MAE* 则能够更好地反映平均误差的整体情况. *MAE* 的定义如下:

$$MAE = \frac{1}{|D|}\sum_{\sigma_k \in D} |f(\sigma_k) - RT(\sigma_k)| \tag{19}$$

其中, 函数 $f(\sigma_k)$ 表示轨迹前缀 σ_k 的剩余时间的预测值, $RT(\sigma_k)$ 表示相应的真实剩余时间. D 代表测试数据集中的轨迹前缀集合, σ_k 表示截取自完整轨迹 σ 的前 k 个事件的子序列. *MAE* 的值越低, 说明预测的效果越好.

4.3 实验结果

4.3.1 不同方法对比实验

本研究在 7 个事件日志数据集上, 与 LSTM、Process-Transformer、Auto-encoded、EFH 等方法进行了对比实验. 表 2 展示了不同方法在各数据集上的平均绝对误差 (*MAE*) 结果.

本研究提出的方法与 LSTM、ProcessTransformer、Auto-encoded 和 EFH 方法在 7 个公开数据集上的对比结果表明, 本方法总体上优于其他方法, 能够达到更高的预测精度.

表 2 业务流程剩余时间预测方法结果对比 (*MAE*)

数据集	LSTM[6]	Process-Transformer[10]	Auto-encoded[17]	EFH[16]	本研究方法
BPIC_2015_1	49.648	39.472	38.324	25.493	**15.87**
BPIC_2015_2	127.834	87.622	88.895	66.993	**20.876**
BPIC_2015_3	61.190	17.544	17.588	17.323	**10.91**
BPIC_2015_4	151.565	47.77	39.619	50.227	**21.22**
BPIC_2015_5	120.750	51.847	38.270	34.64	**20.98**
Production	521.296	14.992	11.775	11.510	**4.287**
Helpdesk	62.08	5.419	8.733	**4.270**	6.501

具体来看, 在 BPIC2015 系列的 5 个数据集上, 本方法的平均绝对误差 (*MAE*) 在 11.91–29.99 之间, 而其他方法的 *MAE* 普遍在 17.544–151.565 之间, 本方法较之降低了约 10%–80%. 结果表明本研究方法可以更好地建模这类较大规模的日志数据.

在 Production 生产流程数据集上, 本方法 *MAE* 仅为 4.23, 比其他方法减小了约 40%, 展现了显著的优势.

Helpdesk 技术支持日志上, 本方法 *MAE* 虽高于 EFH 方法, 但仍低于 LSTM 等其他方法. 由于 Transformer 擅长捕捉长距离依赖, 而 LSTM 更擅长处理短期依赖. Helpdesk 的平均轨迹长度仅为 4.66, 极短的平均轨迹长度意味着模型在预测时可参考的历史信息极为有限, 不足以发挥 Transformer 的优势.

本方法集全局依赖和局部依赖关系于一体的技术设计, 使其可以适应规模不同、复杂度不同的业务流程日志, 在多个公开数据集上展现出较强的泛化性能, 这也验证了方法的有效性.

4.3.2 模块验证实验

为验证所提出方法中时间特征模块和局部依赖增强模块的有效性, 设计了消融实验对模型进行模块析构, 分析每个组件的贡献.

具体实验设计如下.

以 Transformer 为基础模型, 分别将带有时间特征的轨迹前缀特征的局部依赖增强模块的 Transformer 编码器 (命名为 TF-LEN-Transformer), 带有局部依赖增强模块的 Transformer 原编码器 (命名为 LEN-Transformer), 以及不带有时间特征的轨迹前缀特征的原编码器 (命名为 EN-Transformer), 在中长轨迹前缀上的数据集上对比实验, 以验证时间特征对剩余时间预测的影响, 以及改进后的局部依赖增强机制能否提升预测的实际效果. 具体的实验结果如表 3 所示.

从表 3 中可以看出, 相比 EN-Transformer, 无论是引入局部依赖增强模块的 LEN-Transformer 模型, 还是在此基础上进一步加入时间特征的 TF-LEN-Transformer, 在所有数据集上的预测误差都有明显的降低. 这验证了本研究提出的两个模块的改进的有效性.

表 3 不同模块对预测效果的影响对比

数据集	EN-Transformer	LEN-Transformer	TF-LEN-Transformer
BPIC_2015_1	39.472	18.38	**15.87**
BPIC_2015_2	87.622	28.27	**20.876**
BPIC_2015_3	17.544	14.67	**10.91**
BPIC_2015_4	47.77	32.99	**21.22**
BPIC_2015_5	51.847	27.47	**20.98**
Production	14.992	7.07	**4.287**
Helpdesk	**5.419**	6.8	6.501

具体来看, LEN-Transformer 相比 EN-Transformer 减小了 50% 以上的误差, 如在 BPIC_2015_1 数据集的误差由 39.472 降至 18.38. 这显示出局部依赖增强模块的引入增强了模型对轨迹前缀序列的局部特征的建模和利用能力. 卷积操作可以捕捉局部元素之间的依赖关系. 这种局部特征学习减小了预测误差.

而 TF-LEN-Transformer 在 LEN-Transformer 的基础上加入时间特征后, 进一步减小了预测误差. 如在 Production 数据集上, 误差由 7.07 降至 4.287. 这显示时间特征的加入增强了模型对时间相关性的建模. 绝对时间编码和相对时间编码提供了多粒度的时间信息, 使模型更准确地把握轨迹前缀序列中的时间依赖关系, 提升了预测性能.

本研究提出的 LEN-Transformer 和 TF-LEN-Transformer 作为通用的剩余时间预测建模框架, 在多个公开业务流程挖掘的数据集上都取得了较好的提升. 这充分验证了局部依赖增强模块和时间特征对剩余时间预测具有一定的优化效果. 未来工作将在此基础上继续优化模块设计和特征提取, 以获得对时间相关性和局部特征建模更优的模型, 从而提高复杂业务流程的剩余时间预测的准确性.

5 结论与展望

本研究通过建立带时间特征的 Transformer 模型和局部依赖增强模块, 来预测业务流程的剩余时间. 该方法在 Transformer 模块中加入时间特征, 增强对事件时间依赖关系的建模. 另外, 方法在 Transformer 编码器后接入了局部增强模块, 用于学习轨迹前缀序列的局部依赖, 与 Transformer 的全局模式建模能力形成互补. 实验结果显示, 在多个公开日志数据上, 本研究方法相比 LSTM 等算法提高了 10%–80% 的预测精度, 这验证了模型的融合有效提升了预测能力.

虽然本研究所提的方法提高了预测的效果, 但是如何在轨迹流程相对较短的数据集中提高预测精度还需要进一步的研究. 未来可继续改进卷积模块的结构, 以获得更好的短轨迹流程的剩余时间预测模型. 这也是未来的进一步研究的方向.

参考文献

1 Van der Aalst WMP, Schonenberg MH, Song M. Time prediction based on process mining. Information Systems, 2011, 36(2): 450–475. [doi: 10.1016/j.is.2010.09.001]

2 Folino F, Guarascio M, Pontieri L. Discovering context-aware models for predicting business process performances. Meersman R, Panetto H, Dillon T, *et al*. On the Move to Meaningful Internet Systems: OTM 2012. Berlin: Springer, 2012. 287–304. [doi: 10.1007/978-3-642-33606-5_18]

3 高俊涛, 陈珂, 刘云峰, 等. 基于在线模型的业务过程剩余时间预测. 计算机集成制造系统, 2022, 28(10): 3090–3099. [doi: 10.13196/j.cims.2022.10.006]

4 Polato M, Sperduti A, Burattin A, *et al*. Time and activity sequence prediction of business process instances. Computing, 2018, 100(9): 1005–1031. [doi: 10.1007/s00607-018-0593-x]

5 Pandey S, Nepal S, Chen SP. A test-bed for the evaluation of business process prediction techniques. Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing. Orlando: IEEE, 2012. 382–391. [doi: 10.4108/icst.collaboratecom.2011.247129]

6 Navarin N, Vincenzi B, Polato M, *et al*. LSTM networks for data-aware remaining time prediction of business process instances. Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence. Honolulu: IEEE, 2017. 1–7. [doi: 10.1109/SSCI.2017.8285184]

7 Tax N, Verenich I, La Rosa M, *et al*. Predictive business process monitoring with LSTM neural networks. Proceedings of the 29th International Conference on Advanced Information Systems Engineering. Essen: Springer, 2017. 477–492. [doi: 10.1007/978-3-319-59536-8_30]

8 Taymouri F, La Rosa M, Erfani S, *et al*. Predictive business process monitoring via generative adversarial nets: The case

of next event prediction. Proceedings of the 18th International Conference on Business Process Management. Seville: Springer, 2020. 237–256. [doi: 10.1007/978-3-030-58666-9_14]

9 倪维健, 孙宇健, 刘彤, 等. 基于注意力双向循环神经网络的业务流程剩余时间预测方法. 计算机集成制造系统, 2020, 26(6): 1564–1572. [doi: 10.13196/j.cims.2020.06.013]

10 Graves A. Long short-term memory. Supervised Sequence Labelling with Recurrent Neural Networks. Berlin: Springer, 2012. 37–45. [doi: 10.1007/978-3-642-24797-2_4]

11 Bukhsh ZA, Saeed A, Dijkman RM. ProcessTransformer: Predictive business process monitoring with transformer network. arXiv:2104.00721, 2021.

12 Vaswani A, Shazeer N, Parmar N, *et al*. Attention is all you need. Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach: Curran Associates Inc., 2017. 6000–6010.

13 徐兴荣, 刘聪, 李婷, 等. 基于双向准循环神经网络和注意力机制的业务流程剩余时间预测方法. 电子学报, 2022, 50(8): 1975–1984. [doi: 10.12263/DZXB.20211477]

14 Mikolov T, Chen K, Corrado G, *et al*. Efficient estimation of word representations in vector space. Proceedings of the 1st International Conference on Learning Representations. Scottsdale: ICLR, 2013.

15 郭娜, 刘聪, 李彩虹, 等. 一种预测流程剩余时间的可解释特征分层方法. 软件学报, 2024, 35(3): 1341–1356. [doi: 10.13328/j.cnki.jos.006824]

16 Ke G, Meng Q, Finley T, *et al*. LightGBM: A highly efficient gradient boosting decision tree. Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach: Curran Associates Inc., 2017. 3146–3154.

17 Ni WJ, Yan M, Liu T, *et al*. Predicting remaining execution time of business process instances via auto-encoded transition system. Intelligent Data Analysis, 2022, 26(2): 543–562. [doi: 10.3233/IDA-215755]

(校对责编: 孙君艳)